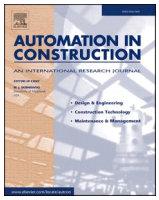
Automation in Construction 163 (2024) 105405

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon



High-resolution infrastructure defect detection dataset sourced by unmanned systems and validated with deep learning

Benyun Zhao [a], Xunkuai Zhou [b,a,*], Guidong Yang [a], Junjie Wen [a], Jihan Zhang [a], Jia Dou [a], Guang Li [a], Xi Chen [a,*], Ben M. Chen [a]

[a] Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, N.T, Hong Kong, China
[b] School of Electronics and Information Engineering, Tongji University, Jiading, Shanghai, China

ARTICLE INFO

Keywords:
Defect detection
Object detection
High-resolution dataset
Deep learning
Unmanned system
Automated robotic platform
Infrastructure inspection

ABSTRACT

Visual inspection of civil infrastructures has traditionally been a crucial yet labor-intensive task. In contrast, unmanned robots equipped with deep learning-based visual defect detection methods offer a more comprehensive and efficient solution compared to conventional manual inspection techniques. However, the full potential of deep learning in defect detection has yet to be fully realized, primarily due to the scarcity of annotated, high-quality defect datasets. In this study, we introduce CUBIT-Det, a high-resolution defect detection dataset that includes over 5500 images captured under various scenarios using professional-grade equipment. Distinguishing itself from existing datasets, CUBIT-Det encompasses a wide array of practical situations, backgrounds, and defect categories. We perform extensive benchmarking experiments on the dataset with nearly 30 cutting-edge real-time detection methods, and analyze both the impact of the dataset's annotation methods and zero-shot transfer ability of it. This effort lays a robust foundation for future advancements in defect detection solutions. Additionally, the practicality and effectiveness of CUBIT-Det are confirmed through thorough inspections of real-world buildings. Finally, we detail the features and acknowledge the limitations of our dataset, thereby highlighting significant opportunities for future research.

1. Introduction

Civil infrastructure is vulnerable to damage caused by a multitude of factors such as weather impacts, external loads, structural deterioration, and poor design. Periodic infrastructure inspections are crucial for remaining safe and functional infrastructures. Currently, non-destructive testing (NDT) devices like optical cameras [1], laser scanners [2], impact echo [3], and ground-penetrating radar [4] are used for manual defect detections in civil infrastructure. Although human visual inspection is the most flexible and feasible method for preliminary diagnosis, it is subjective, time-consuming, laborious, and error-prone. It can also pose significant health and safety risks to human inspectors, especially when inspecting high-rise buildings and large spaces. To overcome these challenges, robotic platforms like unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) [5,6] have been developed to achieve more accurate and efficient infrastructure inspections, from data collection and defect detection. These unmanned platforms integrating computer vision techniques help achieve better inspection results.

Recent advancements in automatic image processing, driven by deep learning methods [7–10], have marked a significant breakthrough, demonstrating substantial advantages in efficiency and effectiveness over traditional image processing techniques [11,12]. Object detection, a task widely applied across various scenarios and domains, exemplifies the exceptional performance of deep learning in speed and accuracy. Consequently, an increasing number of researchers in the construction field [13,14] are turning to deep learning-based object detection methods for inspecting and managing infrastructure defects. However, deep learning algorithms are notoriously data-hungry, necessitating specialized image datasets tailored for object detection in the construction domain. Most existing object detection models are trained on high-quality, open-source datasets featuring common objects [15–17], characterized by *high image resolutions*, *a large volume of images*, *a diversity of object types*, and *varied target backgrounds*. Yet, the collection and annotation of images related to infrastructure defects present unique challenges, given the complex and dynamic nature of

* Corresponding authors at: The Chinese University of Hong Kong, China
E-mail addresses: xunkuaizhou@cuhk.edu.hk (X. Zhou), xichen002@cuhk.edu.hk (X. Chen).

https://doi.org/10.1016/j.autcon.2024.105405
Received 22 October 2023; Received in revised form 14 March 2024; Accepted 22 March 2024
Available online 18 April 2024

construction activities. This leads to a scarcity of quality-assured, object-level datasets specifically designed for the construction industry.

Building upon our previous work [18], we observed that the majority of publicly available defect datasets [19–25] are primarily focused on defect classification or segmentation tasks. Defect classification, while useful, falls short in pinpointing the precise location of defects. Defect segmentation, being a pixel-level classification task, often lacks the speed necessary for rapid inspections. In contrast, the defect object detection task not only swiftly classifies defect types but also provides crucial coordinate information of the defects, which is instrumental for subsequent defect registration processes. Furthermore, these existing datasets exhibit two significant limitations: firstly, they generally consist of a constrained collection of images with considerably low resolution; secondly, they tend to either focus on a single type of infrastructure scene (such as pavements, buildings, or bridges) or solely address the common defect of cracking. However, in practical infrastructure scenarios, especially concerning buildings, there are additional predominant defect types like spalling and moisture that need to be accounted for.

Overall, establishing a high-quality defect detection dataset annotated at the bounding-box level is crucial and urgent to facilitate automated infrastructure defect detection. To address the aforementioned challenges, we present a large-scale defect detection dataset, namely CUBIT-Det[1], consisting of more than 5500 high-resolution (4624 × 3472) images with bounding-box level defect annotation. With the aim of detecting critical defect types including the crack, spalling, and moisture, our proposed dataset covers various infrastructure scenarios: buildings, pavements, and bridges. To demonstrate the feasibility of the proposed dataset, we conduct a comprehensive evaluation of state-of-the-art object detection algorithms on our dataset for detecting infrastructure defects. The sample of detection result on the test set of our CUIBT-Det dataset are shown in Fig. 1, where the rectangles indicate the output prediction box containing the defect position, category, and confidence score from YOLOv6-l [26] trained on the training set of our proposed dataset. The inspection results demonstrate the feasibility of the CUBIT-Det dataset.

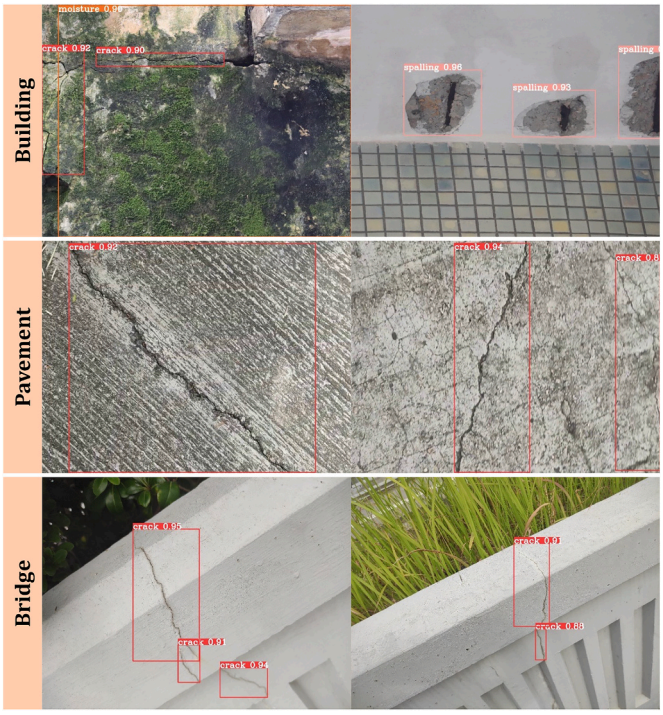


Fig. 1. Prediction results on the test set of the proposed CUBIT-Det defect dataset. The CUBIT-Det dataset covers three infrastructure types: Building, Pavement, and Bridge, and aims for three types of defect: Crack, Spalling, and Moisture. Rectangles indicate the output prediction box (Red for Crack, Pink for Spalling, and Orange for Moisture) with inferred defect type and confidence score from YOLOv6-l trained on the training set of our proposed dataset.

The contributions of our work is as follows:

- We release an open-source[2] dataset, CUBIT-Det, that features high resolution, multiple defect types, and is applicable to defect detection in various infrastructure scenarios. For data from different scenarios, we extensively use our self-developed unmanned system platform combined with our custom-designed algorithms for collection, greatly reducing manual labor.
- To verify the feasibility of our CUBIT-Det dataset, we comparatively conduct extensive benchmarking experiments on the CUBIT-Det dataset with more than 25 state-of-the-art models, where the influence of a particular defect category on the detection accuracy is analyzed. And the influence of annotation methods and the transferability of CUBIT-Det are explored.
- Moreover, a real-world infrastructure defect detection has been conducted by utilizing the model trained on CUBIT-Det dataset to further demonstrate the reliability of our dataset and the effectiveness and convenience of unmanned system platform.

The remainder of this paper is organized as follows. In Section 2, we provide a detailed overview of previous defect detection datasets and the deep learning algorithms that have been utilized to assess their feasibility and usability. In Section 3, we present the production process of CUBIT-Det, which includes the selection of defect types; the use of our specially designed automated unmanned system to collect data from different scenarios, significantly reducing manual labor; as well as data cleaning and annotation. In Section 4 we describe the statistics analysis of our CUBIT-Det dataset, and compare with other existing datasets of the same task. In Section 5, we adopt nearly 30 state-of-the-art deep learning-based object detection models to evaluate the feasibility of our dataset. Additionally, we conduct an in-depth analysis of the models' performance in identifying diverse defect categories, evaluating its inference speed, and assessing the effects of different annotation methods on the models' performance. We also explore the zero-shot transfer potential of models that have been trained on your CUBIT-Det dataset. Furthermore, in Section 6, we demonstrate the feasibility of the proposed dataset by presenting a real-world infrastructure defect detection experiment. Section 7 concludes this paper and discusses potential research directions for improving infrastructure defect detection.

2. Literature review

2.1. Infrastructure defect detection dataset

Based on our previous work [18], a limited number of publicly available defect detection datasets have been developed for infrastructure defect detection in comparison to defect classification and segmentation datasets. Maeda et al. [27] propose the Road Damage Detection dataset (RDD-2018) for large-scale road damage detection. All the images are collected by a smartphone installed on the dashboard of a vehicle in Japan. RDD-2018 [27] has 9,053 images with a uniform resolution of 600 × 600 and contains 15,435 defect instances. As an extension of RDD-2018, the RDD-2019 [28] dataset expands the data volume to 13,135 images with a uniform resolution of 600 × 600, comprising 30,989 road damage instances. Based on RDD-2019 [28], RDD-2020 [29] incorporates road defect data from the Czech and India,

[1] CUBIT stands for CUHK Building Information Technology.

[2] Our dataset is available at *https://github.com/ZHAOBenyun/CUBIT*.

with the aim of increasing the data diversity and improving the robustness of the neural network model during training. Images from Czech and Japan have a consistent resolution of 600 × 600 while images from India have a larger resolution of 720 × 720. RDD-2020 [29] contains 26,336 frontal-facing road images with more than 31,000 road damage instances under various light and weather conditions. Subsequently, RDD-2022 [30] expand based on RDD-2020 [29] by incorporating data from Norway, Ameraica, and China. Additionally, in RDD-2022 [30], the maximum resolution of the data has been increased to 3650 × 2044, and UAV is utilized for data collection, thereby enhancing the collection efficiency. However, their primary focus remains on road surface crack data, making the scenarios relatively monotonous.

Majidifard et al. develops Pavement Image Dataset (PID) [31] containing 7,237 images of 22 different pavement sections in the United States. All the images are crawled from Google street view. The images, with a resolution of 600 × 600, consist of two camera views: a wide view and a top-down view, which are used to detect pavement distress and calculate the crack density for automated pavement rating in the future, respectively. Murad Al Qurishee et al. [32] also propose a dataset concentrating on detecting pavement damages, which includes 2,620 images with resolution up to 838 × 809. All the images are captured from hand-held phone and hand-held UAV. Recently, Sabouri et al. [33] introduce a relatively high-resolution (4032 × 3024) pavement dataset, SUT-Crack. This dataset encompasses both bounding-box-level and pixel-level annotations, making it suitable for both detection and segmentation tasks. However, the dataset contains only 130 images, which limits its size. Due to this limitation, SUT-Crack [33] is primarily appropriate for use in the testing phase.

Despite the great contribution of the aforementioned datasets, most of their image resolutions are relatively low (no more than 1000 × 1000) and they all focus on road defects only. However, the functional safety of other critical infrastructures such as buildings and bridges are also of great concern. Concrete Defect Bridge Image (CODEBRIM) dataset [34] focuses on detecting the defects of concrete bridges. To detect minor defects on bridges from different scales, cameras on UAVs with high resolution (up to 6000 × 4000) and large focal lengths are adopted to collect data. Nevertheless, the CODEBRIM only contains 1590 images, inadequate to extract defect feature information.

Synthesizing the shortcomings of the existing bounding-box-level datasets described above, we present a large-scale, high-resolution, and multi-scenario defect dataset for civil infrastructure defect detection.

2.2. Data augmentation of defect dataset

In recent times, an increasing number of researchers begin utilizing generative models, specifically Generative Adversarial Networks (GANs [35]), for data augmentation. As a popular deep learning generative model, GANs have been successfully applied in various practical applications. An original GAN consists of a generator and a discriminator. The generator receives random noise as its input, while the discriminator is fed with a combination of real data and fake data generated by the generator. The discriminator's primary objective is to accurately differentiate between the real and generated data. Conversely, the generator aims to intricately map the distribution of the training data, effectively creating realistic imitations. In order to be effective on defects data, modifications to the GAN network. For example, Tian et al. [36] modify the generator's loss to be higher in regions with cracks and lower in areas without cracks, enabling the generator to reconstruct more accurate images. Except the original GAN, Xu et al. [37] utilize DCGAN [38] for dataset expansion, using real collected data as the training set for DCGAN, and then feeding both real and DCGAN-generated data into VGG16 [8] for defect detection. Maeda et al. [28] apply PGGAN [39] to synthesize uncommon "pothole" road surface defects, but the final dataset only contains real-world data. Mei et al. [40] introduce cWGAN to generate connective masks for crack images, replacing ordinary binary masks, thereby aiding the network in better performing detection tasks. While generative networks can achieve data augmentation, the common feature of these methods is the low resolution of the generated images, with the highest being only 256 × 256 in the aforementioned methods. Low-resolution images limit in semantic information, and models trained on these images have restricted scalability. When faced with high-resolution images as input (e.g., 8000 × 6000), resizing them to 256 × 256 results in the loss of significant information, rendering the missing detection of small defects. Therefore, generative models may not be universal approaches in creating a high-quality dataset.

2.3. Defect detection approaches

Deep learning approaches have made great progress in recent years due to the availability of massive datasets and ultra-powerful hardware. The representative networks mentioned in Section 1 are mainly the upstream image feature extraction networks. As one of the most common downstream tasks in various fields and scenarios, object detection needs have emerged in infrastructure inspections.

Few deep learning approaches have been tested in the infrastructure defect dataset mentioned in Section 2.1. Maeda et al. use SSD [41] networks (recast the original backbone to Inception V2 [42] and MobileNetV1 [43]) to train RDD-2018 from scratch with random images flipping during training as the data augmentation method. RDD-2019 is also fed to the SSD network for training, using ResNet50 [9] as the backbone instead of Inception V2. Maeda et al. utilize PG-GAN [39] to generate synthetic images and add these images to the training process for dataset diversity. For RDD-2020 [29] and PID [31], transfer learning is adopted with the weights trained based on ImageNet [15] and MS COCO [17] as a powerful feature extractor. Transfer learning is to employ this powerful feature extractor and then fine-tune this feature extractor for the application to a specific dataset. Except for SSD [41], YOLOv2 [44] and Fast R-CNN [45] are trained on PID [31] as well.

Recently, several trends that bypass human design intuition have also been identified to treat the neural architecture itself from a meta-learning perspective and perform black-box optimizations on the basis of weight training to find an architecture design suitable for a particular task. When training CODEBRIM [34], two meta-learning [46,47] frameworks based on Reinforcement Learning (RL) are added to the convolutional neural network for better frameworks and hyper-parameters on CODEBRIM [34].

Beyond its application in the selection of hyperparameters, the meta-learning has been employed in road surface defect detection tasks that utilize few-shot learning. This approach enables networks to learn in a manner akin to human learning, applying existing knowledge to novel defect categories. Zhou et al. [48] use Faster R-CNN [49] as the detection model, employing the common object dataset FSOD [50] for base learning and conducting transfer training on an upgraded open-source defect dataset, culminating in tests on a real-world highway crack dataset. Dong et al. [51] propose a new few-shot learning framework, enhancing feature extraction by adding an attention module to ResNet18 [9]. During testing, cosine similarity is applied to compare the unlabeled query set with the labeled support set, thereby facilitating more effective classification of road surface cracks.

The aforementioned deep learning methods, being either limited in quantity or outdated, fall short of fully representing the state of the art. Therefore, we have conducted expanded evaluation experiments to assess the performance of various advanced deep learning methods on our proposed dataset, applying these trained models in real-world scenarios to validate the flexibility of our dataset. However, despite the effectiveness of few-shot learning methods [48,51] in addressing data scarcity, the limitation of insufficient and single-scenario crack data restricts the breadth of their application. Therefore, the creation of a high-resolution, multi-scenario dataset is still a necessity.

Fig. 2. Sample images from the CUBIT-Det dataset. The first row of images are crack defects on building surfaces and the second row includes crack defects on pavements (first and second column) and bridges (third and forth column). The third row is about spalling, and the forth row is about moisture.

3. Methodology of dataset establishment

We introduce a more comprehensive dataset for defect detection, CUBIT-Det, encompassing a wide range of infrastructures across different districts in Hong Kong. Representative sample images from our dataset are showcased in Fig. 2. The bulk of the data in CUBIT-Det is meticulously collected using a self-developed unmanned system platform, equipped with corresponding algorithms, from various locations across Hong Kong Island, Kowloon, and the New Territories in the Hong Kong SAR, China. In addition to this, we employ high-resolution Single-lens reflex (SLR) cameras and smartphones to capture supplementary data from diverse angles, thereby enriching the dataset with a broad spectrum of perspectives.

3.1. Defect category selection

We refer to numerous reports, standards, and infrastructure inspection guidelines when selecting the types of defects. Initially, we consult the BSI (British Standards Institution) standards publication about "building and constructed assets – service life planning[3]". This standard primarily focuses on visible defects on the building surface. It discusses the relationship between the degree of deterioration and the exposure time of the building, among which spalling of the concrete cover is a highly dangerous type of defect. This defect can easily lead to the collapse of the building. The occurrence of spalling is often due to moisture erosion, the initial stage of which is manifested as wall seepage and moss. Therefore, we preliminarily select spalling and moisture as two defect categories.

Since the data we collected is from various regions in Hong Kong, we refer to the inspection reports of the Hong Kong Buildings Department as well. The Hong Kong Buildings Department conducts mandatory regular inspections of buildings over 30 years old and higher than three floors to maintain the safety of the buildings. According to some reports from the Buildings Department, we summary that due to aging and wear, buildings in Hong Kong are prone to defects such as cracks (structural cracks, non-structural cracks) and spalling (spalling of concrete and defective external wall finishes). Except for non-structural cracks, other defects require timely contact with the Buildings Department and professionals for regular maintenance and repair work. At this time, our automated unmanned infrastructure inspection system can greatly improve efficiency and assist professionals in completing regular maintenance work.

In addition to the reports from the Hong Kong Buildings Department, a series of guidelines and reports such as the professional guide of building inspections[4] from the Building Surveying Division of the Hong

[3] BS ISO 15686-7:2017.

[4] Volume 1: Pre-1980 Residential&Composite Buildings in Hong Kong.

Kong Institute of Surveyors are also utilized to guide our selection of defect types. Ultimately, combining our previous work (citation) and the aforementioned series of standards, guidelines, and reports, we select crack, spalling, and moisture in CUBIT-Det dataset.

3.2. Data collection

3.2.1. Building

The CUBIT-Det dataset includes abundant images of old buildings in Hong Kong, captured by UAV and DSLR. Building in our dataset comprises stone and concrete walls with three typical types of defects: crack, spalling, and moisture. Data about buildings covers various scenarios, involving different infrastructures, viewing angles, and background scenes. Since the overall population density of Hong Kong is relatively large, especially in Sham Shui Po and Mong Kok in Kowloon Peninsula, there are many pedestrians on the road and narrow spaces between buildings, forming the main difficulty in data collection. We use a relatively small-size UAV for horizontal shooting and DSLR for up-view shooting. According to the path planning method in photogrammetry, we design a energy-saving path algorithm to make the UAV orderly photograph the building to collect images at different locations. A zig-zag pattern, which is designed by ourselves, means flying the UAV in vertical or horizontal strips. All the strips should lie in a plane parallel to the target building façade. Vertical stripes are not recommended because the vertical movement of the lens decreases the clarity and quality of data gathered [52,53].

Fig. 3 illustrates the UAV data collection path in the horizontal zig-zag pattern. At each corner, UAV does not makes right-angle turns as shown in Fig. 3; instead, we opt for a rounded, smooth transition. Such path planning not only reduces the inertial force generated by the UAV from abrupt stops during turns but also shortens the overall distance, thereby achieving energy conservation. Fig. 3(a) is the front view of the zig-zag pattern in horizontal strips, which are proven to be ideal especially when paired with a low flight speed [52,53], Fig. 3(b) and (c) respectively show the UAV flying and shooting following the designed zig-zag route under the top view and right view conditions. There is an overlap between images, similar to a form of image enhancement (i.e., translation). Overlapped parts of adjacent images are labeled consistently. The automated UAV system with our energy-saving path planning algorithm greatly saves labor costs during the collection of data in building scenarios.

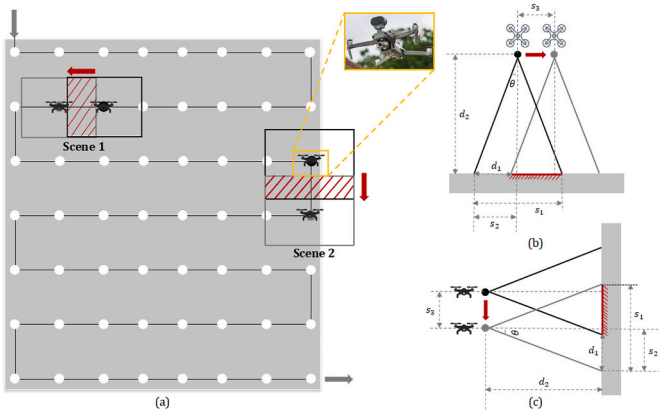


Fig. 3. UAV data collection route with a horizontal zig-zag pattern. (a) Front view. The gray block represents the building façade, and the white dot represents the waypoint, where the UAV takes images in a stop-shoot-go mode. (b) Top view. This view illustrates two horizontally adjacent waypoints with overlapped horizontal fields of view (FoV). (c) Right view. This view illustrates two vertically adjacent waypoints with overlapped vertical FoV. Where $d1$ represents the distance interval between two adjacent waypoints, $d2$ represents the distance from UAV to the building façade, θ represents the angle which is half of the camera FoV and $s1$ represents the maximum distance on the building façade the camera covers.

3.2.2. Pavement

In addition to the defects on the exterior walls of the buildings, we also collect crack data on the pavements. Since the cracks on the pavement surface are relatively long, we use the UGV with external USB cameras to take videos of the pavement surface. After that, the collected videos are sampled at a designed interval. The UGV is named Jackal, a small, fast, customizable robotics research platform from Clearpath Robotics company, shown in Fig. 4(a). We customize the stark Jackal UGV and install some sensors including a mechanical LiDAR Velodyne VLP16, a solid-state LiDAR Livox Mid40, an RGB-D binocular camera Realsense D435i, and a monocular USB camera. When collecting the optical pavement cracks data, we first activate the Velodyne VLP-16 to run SLAM algorithm to realize mapping [54,55]. We only need to manually control the UGV to roughly circle the target area, and the map of this area will be completed. After mapping, we directly mark the target points on the built map, shown in (Fig. 4(b), allowing the UGV to automatically explore the path to the target, which can greatly save labor costs. At the same time, a high-resolution monocular USB camera is always activated to record videos. Finally, we sample the video at intervals of 20 frames per second to obtain images. It should be noted that sampling at a lower FPS (e.g. 5 FPS) could result in overfitting deep neural networks. Therefore, slightly larger sampling intervals are employed.

3.2.3. Bridge

In addition to buildings and pavements, given Hong Kong's hilly terrain, bridges are also a common infrastructure feature, rendering the detection of bridge cracks essential. However, as the collection of bridge data necessitates governmental approval, we confined ourselves to collecting defects on bridges within the CUHK campus to enrich our CUBIT-Det dataset. Unlike cracks data on dark pavement, the bridge has a light colour background, making the cracks more visible while increasing the model's robustness. All bridge images are captured using the backend camera of mobile phones.

3.3. Data cleaning

Upon completing data collection, data cleansing becomes imperative. Given our collection methodology, a portion of the gathered data is unusable. Initially, eliminating such invalid data is necessary to facilitate subsequent annotation. For the building defect data acquired via drones, some images lack the targeted defects, necessitating their removal. Similarly, for road defect data captured using the Jackal UGV, the high-speed movement and sudden stops often result in blurred video segments and unfocused images, which also require exclusion.

Ultimately, after several rigorous rounds of selection and cleansing, we narrow our dataset down from over 8,000 original images to 5,527 high-definition images, each featuring various infrastructure defects. This comprehensive process plays a pivotal role in ensuring the integrity, quality, and applicability of our dataset for thorough analysis and model training purposes.

3.4. Data annotation

Upon successfully completing data filtering and cleaning, we embark on the critical phase of data annotation. The Visual Object Classes (VOC) data format, commonly known as the Pascal VOC format [16], is the most prevalent, standardized, and universally adopted format in both computer vision and industrial applications. Therefore, we choose this format as the designated output for our annotation tools. During the annotation process, we adhere to strict guidelines: (1) Each defect target must be completely enclosed within its respective bounding box, ensuring no overlaps between the box and the defects. (2) Every

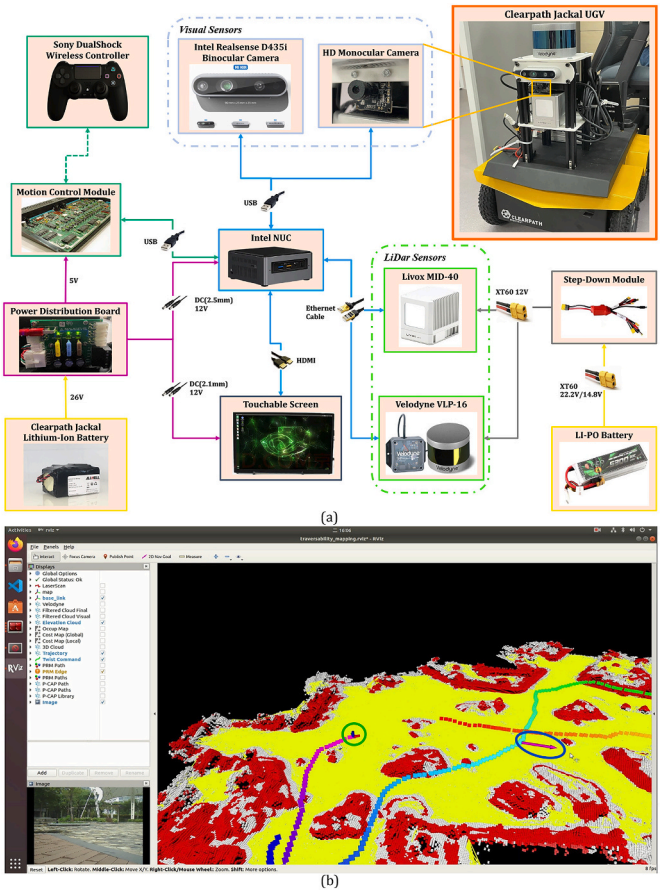


Fig. 4. (a) Self-designed pavement data collection UGV system: Clearpath Jackal. The onboard computer is the Intel NUC. Two LiDARs sensors are Velodyne VLP16 and Livox Mid40, and Visual sensors includes a high-resolution USB monocular camera and a RealSense D435i RGB-D camera. (b) The demonstration of traversibility algorithm based on SLAM algorithm LeGO-LOAM [54,55]. The green circle is the current position of the car, and the arrows in the oval circle are the target points given on the map we have built. The car will explore the path from the initial position to the target point.

discernible defective object in the images must be labeled, with no omissions. Following these stringent rules, our annotation procedure unfolds in three rounds: initially, students annotate the images independently; then, in the second round, they cross-verify each other's work, identifying images with ambiguous labels; finally, in the last round, we consult professors and construction field experts to resolve any remaining ambiguities. This meticulous process ensures the highest level of quality and accuracy in our dataset annotations.

Fig. 5(a) presents the interface of the widely-used annotation tool, LabelImg[5], renowned for its efficacy in bounding box-level object detection tasks. This tool generates labels in the form of Pascal VOC format XML files. The interface's left side features tools essential for image annotation, including options for folder selection, output format choice, and image zooming capabilities. The central section of the interface displays the image being annotated, with various classes distinguished by rectangles in different colors. On the right side, the interface reveals a list of all defects identified in the image and an inventory of all images pending annotation in the selected folder. Fig. 5(b) illustrates the label information encapsulated in the XML file, encompassing details such as the image name, path, dimensions, and the categories and coordinates of all defective objects within the image.

[5] https://github.com/HumanSignal/labelImg.

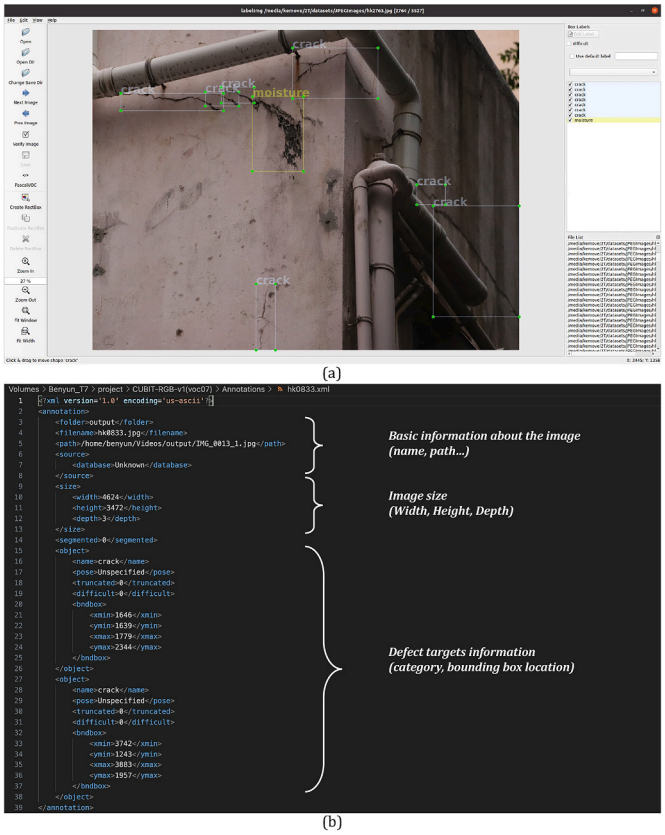


Fig. 5. (a) Interface of LabelImg. The rectangular boxes are used to select defects, with different defects in different colors. (b) XML format file. The annotation information include basic information about the images, image size and defect targets information.

4. Statistics of CUBIT-Det

4.1. Overall analysis

Fig. 6(a) illustrates the data collection scenarios. Our dataset primarily consists of building data (**65%**), as buildings are among the most ubiquitous forms of infrastructure in daily life. However, due to the considerable challenge in data collection, no existing datasets include defect data specific for buildings. To tackle this challenge, we concentrate our efforts on building data during dataset establishment. With the help of our unmanned system platform, we greatly reduce the difficulty of collecting building defect data and enhance the efficiency. Pavements are the second-most typical (**29%**) scenario in CUBIT-Det owing to the prevalence of surface cracking. Bridges make up the remaining for **6%**. As the collection of bridge data necessitates governmental approval, we confined ourselves to collecting defects on bridges within the CUHK campus to enrich our dataset.

Fig. 6(b) illustrates the proportion of defect types. Cracks are

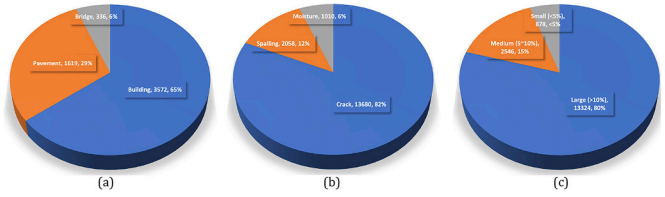


Fig. 6. (a) Defect collection scenarios; (b) Defect categories; (c) Defect target dimensions: Large targets are exceeding 10% of image dimensions, medium-size targets are ranging from 5% to 10% and small targets are less than 5%.

undoubtedly the most significant and common defects, constituting the highest percentage in our dataset (**82%**). Following is spalling, which poses the most significant threat to infrastructure, accounting for **12%**. As mentioned in the Section 3.1, spalling can easily lead to building collapse. The final **6%** pertains to moisture-related issues, which is one of the main causes of spalling.

Regarding the size of the defect objects, the statistics are shown in Fig. 6(c), illustrating that our dataset pays more attention to large objects (**80%**), which are larger than 10% of the entire image size. The medium-size objects are the defects whose width and height are larger than 5% but less than 10% of the image size, accounting for **15%**. And the small objects, whose defect dimension is less than 5% of the whole image, making up **5%** of CUBIT-Det dataset. Large objects are clearer than small objects and contain more defect features and information. For object detection tasks, there is a close relationship between the object size and the receptive field. If the object is small, then the model needs to have a small enough receptive field to detect the object correctly, and a higher resolution to determine the location of the object accurately. In addition, during the training process, if the number of small objects is greater than the number of large objects, the model tends to concentrate more on the small objects, thus affecting its ability to detect large objects. So, we concentrate on high resolution images and large size defect objects.

4.2. Defect targets position distribution

Object distribution is another significant consideration in the object detection dataset. The distribution of objects can affect the model's spatial awareness, which is the model's understanding of the distribution and relative position of objects in space. This distribution can also affect the model's ability to detect offset targets, indicating the model's robustness to target position shifts. When detecting buildings in real scenes, many defects could be sparsely distributed in various locations of the image input to the deep neural network model. Models with strong target offset detection capabilities can easily capture these defects. Furthermore, the generalization ability of the model has a strong relationship with the target position distribution. If the target distribution in the training dataset is not representative, the model will encounter targets with unknown positions in the real scene, and false detections and missed detections will occur. Fig. 7 shows the defect object position distribution of CUBIT-Det, the overall distribution of objects is uniform across the whole image area. The points on the central axis are relatively dense, forming a cross shape, since most objects are located in the middle area of the image. This data-position distribution ensures that the model trained based on our CUBIT-Det dataset has a better spatial perception, offset objects detection ability, and generalization.

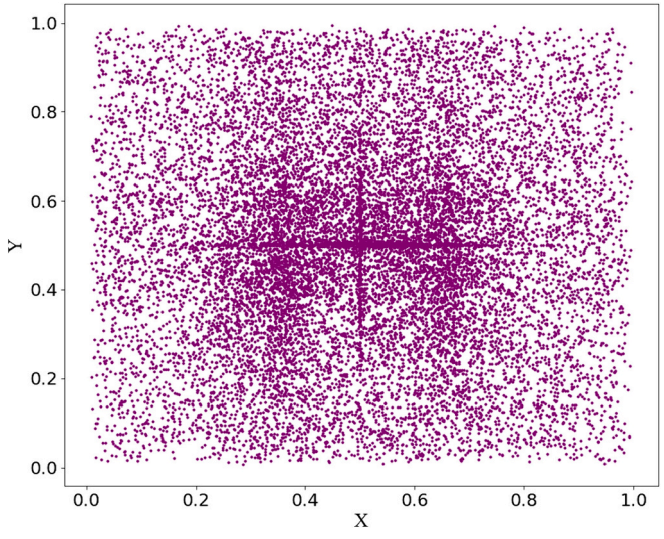


Fig. 7. Object position distribution of CUBIT-Det dataset. The form of scatter plots is used to describe the position distribution of objects' relative center location.

4.3. Comparison with other datasets

The comparative analysis of existing bounding-box-level defect datasets, as discussed in Section 2.1, with our CUBIT-Det dataset is illustrated in Table 1. While our dataset may not rank as the largest in terms of sheer volume, it excels in image resolution and scene diversity. The image resolutions in our dataset range from a minimum of 4624 × 3472 to a maximum of 8000 × 6000. The data encompasses a wide array of infrastructures and scenarios, with a particular emphasis on buildings, which are both challenging to collect and ubiquitous in everyday life. This diversity ensures the effectiveness of models trained on our dataset in detecting defects across various scenarios. The efficacy of CUBIT-Det has been demonstrated through the training of nearly 30 state-of-the-art, deep learning-based, real-time object detection models. Detailed results of this evaluation are provided in Section 5.

5. Evaluation experiments of CUBIT-Det

We train 9 state-of-the-art series (more than 25 models) of real-time object detection algorithms on CUBIT-Det: YOLOv5 [60], YOLOv6 [26], YOLOv7 [61], YOLOX [59], PP-YOLO [56], PP-YOLOv2 [57], PP-YOLOE [58], PP-YOLOE+ [58] and Faster R-CNN [49]. With these algorithms, we adopt the most common two object detection metrics ($AP_{0.5}$ for Pascal VOC [16], and the other is $AP_{0.5:0.95}$ for MS COCO [17]) based on the mean Average Precision (mAP) to evaluate the above networks. The trained models are selected for two reasons. Firstly, all networks are able to achieve real-time while performing the detection task. Secondly, these models are superior to other real-time object detection models with robust capabilities in detecting common objects. Thus, these models are expected to perform better in their infrastructure defect detection task. Additionally, training more network models not only verifies the usability of the dataset but also provides more options in different defect detection scenarios.

Classic object detection models can be broadly divided into two categories: single-stage networks which directly complete classification and regression on the feature map to obtain faster detection results; and two-stage networks which consist of the region proposal network (RPN) generating many candidate boxes and the classification and regression network for recognition and localization of each object. YOLOv5 [60], YOLOv6 [26], YOLOv7 [61], YOLOX [59], PP-YOLO [56], PP-YOLOv2 [57], PP-YOLOE [58] and PP-YOLOE+ [58] are single-stage object detection networks, and Faster R-CNN [49] belongs to two-stage networks.

5.1. Experimental setup

All experiments are conducted on a computer equipped with an Intel i9-10900k CPU and an NVIDIA RTX 3090 GPU. For training and testing various models, our CUBIT-Det dataset is divided into three segments: 3,980 images for training (**72%**), 442 images for validation (**8%**), and 1,105 images (**20%**) for robustness testing. The models train for 400 epochs without using any pre-trained weights from other common object detection datasets. Stochastic gradient descent (SGD) is employed as the optimizer. Additionally, Non-Maximum Suppression (NMS [62]) is utilized to eliminate redundant candidate boxes targeting the same object. Algorithm 1 outlines the detailed procedures for NMS-based object detection. Let B represent the list of initially detected boxes, with S containing the respective detection scores. The threshold of NMS, denoted as N_t, plays a pivotal role. The set D is used to store the final

Table 1
The comparison between existing bounding-box-level defect dataset with CUBIT-Det.

Dataset	Num. of images	Resolution	Data collection platform	Defect type	Structure	Material	Experiments
RDD-2018 [27]	9053	600 × 600	Cameras on ground vehicle	Crack Corrosion	Pavement	Asphalt	- SSD [41] (Inception V2 [42], MobileNet [43])
RDD-2019 [28]	13,135	600 × 600	Cameras on ground vehicle	Crack Corrosion	Pavement	Asphalt	- SSD [41] (ResNet50 [9], (MobileNet [43])
RDD-2020 [29]	26,336	600 × 600 720 × 720	Cameras on ground vehicle	Crack Pothole	Pavement	Asphalt	- SSD [41] (MobileNet [43])
RDD-2022 [30]	47,420	512 × 512 600 × 600 720 × 720 3650 × 2044	Smartphones Hand-held cameras UAV cameras Google street view	Crack Pothole	Pavement	Asphalt	–
PID [31]	7237	640 × 640	Crawled from Internet	Crack	Pavement	Asphalt	a. YOLOv2 [44] b. Fast R-CNN [45]
Murad [32]	2620	up to 838 × 809	Hand-held phones and UAV	Crack	Pavement	Asphalt	- Faster R-CNN [49]
SUT-Crack [33]	130	4032 × 3024	Cameras on ground vehicle	Crack	Pavement	Asphalt	–
CODEBRIM [34]	1590	up to 6000 × 4000	Hand-held cameras Cameras on UAV	Crack Corrosion	Bridge	Concrete	a. MetaQNN [46] b. Efficient Neural Architecture Search [47]
CUBIT-Det	**5527**	4624 × 3472 and 8000 × 6000	**Cameras in Unmanned Systems**	**Crack Spallinig Moisture**	**Building (65%) Pavement (29%) Bridge (6%)**	**Concrete Asphalt Stone**	**a. Faster R-CNN** [49] **(MobileNet** [43]**, ResNet** [9]**)** **b. PP-YOLO** [56] **c. PP-YOLOv2** [57] **b. PP-YOLOE(s,m,l)** [58] **c. PP-YOLOE+(s,m,l)** [58] **d. YOLOX(n,t,s,m,l,x)** [59] **e. YOLOv5(n,s,m,l,x)** [60] **f. YOLOv7(t,normal,x)** [61] **g. YOLOv6(n,s,m,l)** [26] ***i. Real-site experiment***

selection of boxes. The Intersection-over-Union (IoU) threshold N_t is a critical parameter in NMS, assessing the overlap rate between predicted candidate boxes (C-Box) and the ground-truth bounding box (G-Box), with an ideal ratio being 1. In common object datasets, an IoU threshold of 0.5 is typically predefined for classifying whether a predicted candidate box is a true positive or a false positive. During validation and testing phases, the confidence threshold is set at 0.03 and the IoU threshold at 0.6. A visual explanation of IoU is provided in Fig. 8.

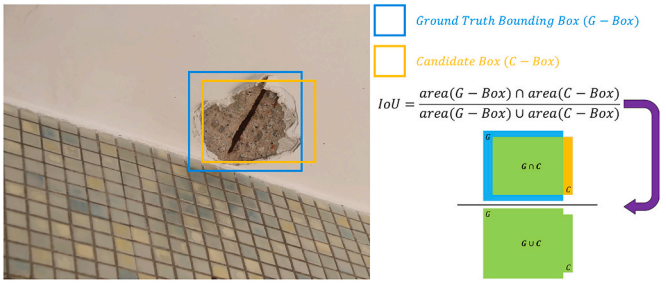


Fig. 8. Visualization of Intersection-over-Union (IoU). Blue rectangle and orange rectangle represent the ground-truth bounding box (G-Box) and candidate box (C-Box) of this spalling sample, respectively. In IoU equation, the denominator symbolizes the union of the G-Box and the C-Box, which is represented by a green rectangle. The overlapping area of the G-Box and the C-Box, which denoted their intersection, is also indicated by the green part. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Input: The **input** initial detection boxes B, the corresponding detection scores S, the related threshold N_t.
Output: The **output** final detection boxes D and the corresponding detection score S.

$D \leftarrow \emptyset$
while $B \neq empty$ **do**
  Select the maximum value in the set of S, and give this value to m. $m \leftarrow argmax(S)$
  $M \leftarrow b_m$
  $D \leftarrow D \cup M$
  $B \leftarrow B - M$
  for b_i *in* B **do**
    if $iou(M, b_i) > N_t$ **then**
      $B \leftarrow B - b_i$; $S \leftarrow S - s_i$;
return D, S

Algorithm 1. The non-maximum suppression-based algorithm for object detection.

5.2. Evaluation metric

Precision (P), Recall (R), and Average Precision (AP) are three most common used metrics in object detection for infrastructure defect detection tasks. Precision (P) is a metric to measure false detection, denoting the ratio of correctly detected target defects among all those predicted by a model. While Recall (R) is a metric to assess miss-detection, denoting the probability that these target defects are correct among all ground truth target defects. Precision and Recall are defined respectively as follows:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

TP, TN, and FN are True Positives, True Negatives, and False Negatives, respectively. If one defect is successfully detected and its predicted box's IoU with the ground truth box is over threshold N_t, the predicted candidate box will be seen as the true positive (TP). Otherwise, it will be regarded as a false positive (FP). In addition, if one target defect fails to be detected, it will be designated as a false negative (FN).

The area under the Precision-Recall curve (AUC-PR) is a metric to judge the performance of an object detection model which considers false detections and miss-detections for varying IoU thresholds. Like the AUC-PR metric, Average Precision is a way to summarize the PR curve into a single value. The Average Precision metric (AP) is the weighted mean of Precision scores achieved at each PR curve threshold, with the increase in Recall from the previous threshold as the weight. Since our detection task is multi-category, after obtaining AP for each category, we also need to average it to obtain a mean value of different AP, for short mAP. The equations of AP and mAP are shown below, where AP_k is the AP of class k and n is the number of classes.

$$AP = \int_0^1 p(r)dr \tag{3}$$

$$mAP = \frac{1}{n}\sum_{k=1}^{n} AP_k \tag{4}$$

5.3. Benchmarking experiment and analysis

5.3.1. Overall analysis

The detailed experimental results of the selected models tested on our CUBIT-Det are shown in Table 2. YOLOv6-l [26] has the best performance among all the models in terms of the accuracy, whose $mAP^{test}_{0.5}$ is 82.9% and $mAP^{test}_{0.5:0.95}$ is 55.9%, while the worst one is PP-YOLOE-s [58], whose results on $mAP^{test}_{0.5}$ and $mAP^{test}_{0.5:0.95}$ are 64.6% and 38.9%, respectively. The two models are single-stage object detection networks. For Faster R-CNN [49] which is the most representative two-stage object detection network, we abandon the original backbone network VGG16 [8] and change it to a more lightweight but stronger backbone, MobileNetV2 [63] and a classic backbone, ResNet50 [9]. However, when the backbone is MobileNetV2 [63], $mAP^{test}_{0.5}$ and $mAP^{test}_{0.5:0.95}$ are 45.6% and 19.4%, respectively, lower compared to the results of the one-stage network in Table 2. When the backbone is ResNet50 [9], compared with MobileNetV2 [63], the detection capability has been greatly improved, $mAP^{test}_{0.5}$ and $mAP^{test}_{0.5:0.95}$ are 71.5% and 43.3% respectively. However, they are still lower than most one-stage detection models.

Among these networks, we visualize the number of parameters of these selected state-of-the-art real-time object detection networks for a complexity comparison. As shown in Fig. 9, YOLO5-n [60] takes the

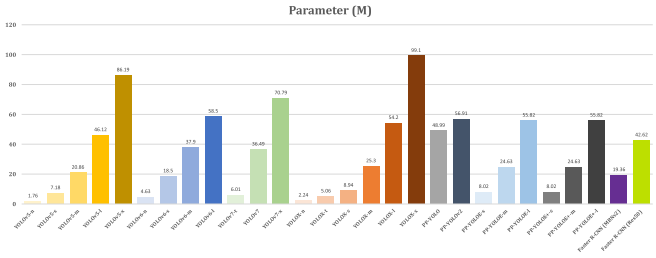


Fig. 9. Comparison of parameters for the state-of-the-art methods. The YOLOv5-n [60] possesses the smallest number of parameters, and the YOLOX-x [59] possesses the largest number of parameters.

Table 2
The test results of selected models based on CUBIT-Det.[1, 2]

Model	Backbone	$mAP^{test}_{0.5}\uparrow$	$mAP^{test}_{0.5:0.95}\uparrow$	Crack		Spalling		Moisture	
				$AP^{test}_{0.5}\uparrow$	$AP^{test}_{0.5:0.95}\uparrow$	$AP^{test}_{0.5}\uparrow$	$AP^{test}_{0.5:0.95}\uparrow$	$AP^{test}_{0.5}\uparrow$	$AP^{test}_{0.5:0.95}\uparrow$
YOLOv5-n [60]	New CSP-Darknet	73.4%	39.9%	68.8%	35.6%	81.9%	49.9%	69.4%	34.4%
YOLOv5-s [60]	New CSP-Darknet	78.5%	47.2%	77.1%	44.1%	85.7%	59.8%	72.6%	37.8%
YOLOv5-m [60]	New CSP-Darknet	80.4%	51.3%	81.3%	50.5%	86.3%	62.3%	**73.8%**	41.2%
YOLOv5-l [60]	New CSP-Darknet	80.6%	52.6%	81.5%	52.1%	87.0%	64.1%	73.2%	41.7%
YOLOv5-x [60]	New CSP-Darknet	81.4%	53.0%	82.0%	52.8%	88.6%	64.1%	73.6%	41.9%
YOLOv6-n [26]	EfficientRep	76.3%	47.9%	80.6%	49.2%	88.8%	60.9%	59.5%	33.6%
YOLOv6-s [26]	EfficientRep	79.0%	48.2%	80.0%	48.4%	86.0%	59.1%	70.9%	37.1%
YOLOv6-m [26]	CSPBep	80.4%	54.1%	83.4%	54.1%	90.1%	64.7%	67.8%	43.6%
YOLOv6-l [26]	CSPBep	**82.9%**	**55.9%**	**85.7%**	**55.8%**	**91.7%**	**67.5%**	71.4%	**44.3%**
YOLOv7-t [61]	*E*-ELAN	71.1%	39.7%	67.4%	36.0%	79.8%	49.8%	66.1%	33.2%
YOLOv7 [61]	E-ELAN	77.5%	47.8%	77.4%	45.9%	82.5%	57.2%	72.7%	40.2%
YOLOv7-x [61]	E-ELAN	79.7%	53%	81.9%	52.8%	85.7%	64.1%	71.5%	41.9%
YOLOX-n [59]	CSP-DarkNet	73.0%	39.5%	71.3%	37.3%	80.7%	48.2%	67.1%	32.9%
YOLOX-t [59]	CSP-DarkNet	77.7%	49.2%	75.9%	48.0%	85.7%	60.2%	71.5%	39.5%
YOLOX-s [59]	CSP-DarkNet	77.8%	50.2%	76.0%	48.5%	85.7%	60.7%	71.6%	41.3%
YOLOX-m [59]	CSP-DarkNet	78.2%	52.2%	76.4%	51.4%	86.3%	62.2%	72.0%	43.0%
YOLOX-l [59]	CSP-DarkNet	78.5%	52.6%	76.8%	52.1%	86.4%	62.6%	72.3%	43.2%
YOLOX-x [59]	CSP-DarkNet	78.8%	53.4%	77.1%	52.2%	86.9%	64.1%	72.5%	43.9%
PP-YOLO [56]	ResNet50-vd-dcn	76.4%	45.1%	75.7%	42.6%	84.3%	56.3%	69.2%	36.5%
PP-YOLOv2 [57]	ResNet50-vd-dcn	77.3%	47.1%	77.8%	44.8%	84.6%	58.1%	69.5%	38.2%
PP-YOLOE-s [58]	CSPRepResNet	64.6%	38.9%	64.6%	36.4%	78.6%	50.1%	50.5%	27.5%
PP-YOLOE-m [58]	CSPRepResNet	74.2%	44.8%	73.9%	43.4%	84.5%	56.7%	64.3%	34.4%
PP-YOLOE-l [58]	CSPRepResNet	75.4%	46.4%	76.7%	46.0%	85.6%	57.9%	63.8%	35.4%
PP-YOLOE+-s [58]	CSPRepResNet	70.6%	44.0%	68.1%	40.4%	82.6%	56.1%	61.0%	35.6%
PP-YOLOE+-m [58]	CSPRepResNet	78.8%	50.9%	79.2%	49.3%	85.2%	60.4%	72.1%	42.9%
PP-YOLOE+-l [58]	CSPRepResNet	78.9%	51.0%	79.1%	49.9%	85.8%	61.7%	71.9%	41.4%
Faster R-CNN [49]	MobileNetV2	45.6%	19.4%	38.5%	15.8%	60.6%	26.3%	30.5%	12.6%
Faster R-CNN [49]	ResNet50	71.5%	43.3%	72.5%	42.3%	83.9%	54.2%	54.2%	29.3%

[1] The best results in each evaluation metric column are in bold.
[2] ↑ (↓) indicates that larger (smaller) values lead to better (worse) performance.

smallest parameters while the YOLOX-x [59] takes the largest.

5.3.2. Categorized detection analysis

Table 2 also presents the test results for different defect categories to further evaluate the detection ability of selected network models trained on our CUBIT-Det dataset.

For crack, YOLOv6-l [26] has the strongest detection ability with an $AP^{test}_{0.5}$ and $AP^{test}_{0.5:0.95}$ accuracy of 85.7% and 55.8%, respectively. For the spalling category, YOLOv6-l [26] also has the strongest detection ability, achieving an $AP^{test}_{0.5}$ accuracy of 91.7% (the highest $AP^{test}_{0.5}$ value of all categories) and an $AP^{test}_{0.5:0.95}$ accuracy of 67.5%. But for the moisture category, YOLOv5-m [60] achieves the best performance under the $AP^{test}_{0.5}$ metric, followed by YOLOv5-x [60]. Under the $AP^{test}_{0.5:0.95}$ metric, YOLOv6-l [26] regains the top spot (44.3%) followed by YOLOX-x [59].

The classic two-stage model Faster R-CNN [49] is not well adapted to our dataset. Faster R-CNN with a backbone of MobileNetV2 [63] achieves the lowest scores in each category, especially in the more difficult-to-detect the moisture category, with a $mAP^{test}_{0.5}$ of only 30.5%. After recasting the backbone with ResNet50 [9], the detection ability of Faster R-CNN [49] in the three categories has greatly enhanced, especially for the crack, which is increased from 38.5% to 72.5%.

5.3.3. Network attributes affecting defect detection

The inference time (latency) of the algorithm is another crucial indicator for the practical implementation of real-time infrastructure inspections. Table 3 shows the latency of the selected deep learning algorithm based on our CUBIT-Det dataset. Obviously, the inference time required for single-stage object detection networks is significantly less than that of two-stage object detection networks. The YOLOX-x [59] model, which is the largest model in single-stage networks, takes the longest inference time of 41.2 ms. However, the Faster R-CNN [49] with a lightweight backbone (MobileNetV2 [63]) still needs 55 ms.

In the same algorithm network series, due to the continuous expansion of the backbone network, the parameters and corresponding inference time are constantly increased. However, different networks do not follow this rule that fewer parameters mean shorter inference time. For example, the YOLOv6-n [26] model only needs 2.2 ms to complete the inference of one input image with resolution of 1024 × 1024, but its Params. is 4.63 M. (In deep learning, "Params." refers to the number of trainable parameters in a neural network model. The small "Params" also represents the small storage space occupied by the trained model. "GFLOPs" stands for "giga floating point operations per second". It is a measure of the computational complexity of a model, calculated as the number of floating point operations, for short FLOPs, performed per second, divided by one billion, which is to express the value in billions of FLOPs per second. This metric is often used to estimate the computational cost or efficiency of running a given model on a particular hardware platform) Contrarily, the Params. of YOLOX-n [59] model is only 2.24 M, but it takes 4.2 ms to complete the inference of one image with resolution of 1024 × 1024. Moreover, PP-YOLOv2 [57], which is not an enlarged version of PP-YOLO [56] but an upgraded version with structural improvements to realize the feature extraction ability, takes 11.0 ms to complete the inference for one image, while PP-YOLO [56] needs 11.2 ms to finish inference.

Combined with the mAP results in Table 2, we visualize the latency versus mAP in Fig. 10. For all series of algorithms, as the model size increases, the inference speed will decrease while the detection capability will improve. However, there is a bottleneck in detection capability, which means that simply enlarging the model to realize the enhancement of detection performance cannot always be effective. By magnifying the top-left corner of Fig. 10, it becomes clearer that, on our CUBIT-Det dataset, YOLOv6 [26] series networks demonstrate a fabulous trade-off between accuracy and latency. The YOLOv6-l [26] algorithm with the strongest detection ability can infer an image in 15.9 ms. Compared with the other largest model of each series, YOLOv6-l [26] takes the least time. The reason is that the number of parameters of YOLOv6-l [26] is relatively small, and its backbone is a structurally reparameterized network, which saves a lot of inference time.

Table 3
The inference speed test on CUBIT-Det.

Model	Params. (M)	GFLOPs	Input size	Latency (ms) ↓
YOLOv5-n [60]	1.76	4.10	1024	1.8
YOLOv5-s [60]	7.18	15.80	1024	3.3
YOLOv5-m [60]	20.86	47.90	1024	7.1
YOLOv5-l [60]	46.12	107.70	1024	12.5
YOLOv5-x [60]	86.19	203.80	1024	24.6
YOLOv6-n [26]	4.63	29.03	1024	2.2
YOLOv6-s [26]	18.50	115.64	1024	5.3
YOLOv6-m [26]	37.90	225.55	1024	9.8
YOLOv6-l [26]	65.05	396.57	1024	15.9
YOLOv7-t [61]	6.01	13.00	1024	5.4
YOLOv7 [61]	36.49	61.94	1024	8.4
YOLOv7-x [61]	70.79	188.00	1024	17.9
YOLOX-n [59]	2.24	17.75	1024	4.4
YOLOX-t [59]	5.06	39.00	1024	5.8
YOLOX-s [59]	8.94	68.51	1024	7.6
YOLOX-m [59]	25.30	73.80	1024	13.7
YOLOX-l [59]	54.20	155.60	1024	20.3
YOLOX-x [59]	99.10	281.90	1024	41.2
PP-YOLO [56]	48.99	136.43	1024	11.2
PP-YOLOv2 [57]	56.91	146.50	1024	11.0
PP-YOLOE-s [57]	8.02	20.73	1024	9.4
PP-YOLOE-m [57]	24.63	62.93	1024	11.2
PP-YOLOE-l [57]	55.82	142.13	1024	11.3
PP-YOLOE+-s [57]	8.02	20.73	1024	8.1
PP-YOLOE+-m [57]	24.63	62.93	1024	8.9
PP-YOLOE+-l [57]	55.82	142.13	1024	10.4
Faster R-CNN (MBNv2) [49]	19.36	44.93	1024	55.0
Faster R-CNN (Res50) [49]	42.62	477.24	1024	76.9

5.3.4. Label attributes affecting defect detection

In addition to the influence of different network models on the detection of infrastructure defects, we also experiment with the effect of data labels on the detection. Taking the crack category which is the most defect type in our CUBIT-Det dataset as an example, we classify cracks and labeled different cracks into three categories: "Linear", "Branch", and "Web" given their varied shape, style, and thickness. "Linear" usually refers to a single crack with no branch. "Branch" refers to a crack with a bifurcation, like a tree branch. "Web" refers that the cracks can be

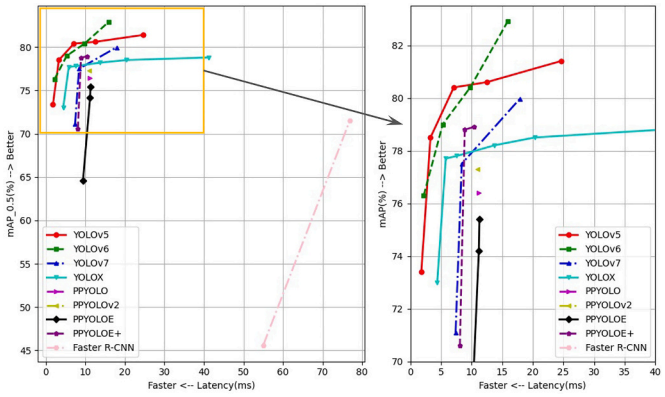


Fig. 10. Trade-off performance of different models about latency versus mAP trained on CUBIT-Det. The further the point is toward the top-left corner, the stronger the detection capability and the shorter the inference time. YOLOv6 [26] series (green dash line) are able to complete inference with relatively little time, but them maintain the highest accuracy.

Fig. 11. Samples of three different crack labels. "Linear" refers to a single crack; "Branch" is like a branch on a tree but forms a circle; "Web" refers to forming multiple rings and gathering together.

formed into rings, and the rings are close together to form a spiderweb-like structure. The visualization samples of 3-classes cracks are shown in Fig. 11.

However, we later hypothesize that such annotations might impact the model's inference speed, leading to detection errors. Furthermore, certain types of cracks, particularly "Branch" and "Web", can be challenging to identify. This complexity could potentially confuse annotators during the labeling process, resulting in inaccurate annotations. Such inaccuracies might also confuse the model, thereby reducing its detection accuracy. Therefore, we select 1000 images containing only cracks from the CUBIT-Det dataset for as precise annotation as possible. We test these images using the lightest models among the three powerful series algorithms - YOLOv5 [60], YOLOv6 [26], and YOLOv7 [61]. The hardware environment for training remains the same as mentioned in Section 5.1, and we still experiment without any pre-trained weights, training from scratch for 200 epochs. After training, we deploy these three models on an onboard computer (NVIDIA Xavier NX) of our UAV.

In the quantitative analysis presented in Table 4, the YOLOv5-n [60], YOLOv6-n [26], and YOLOv7-t [61] models demonstrate that the 1-class labeling approach significantly boosts their detection capabilities. This method not only simplifies the identification of cracks but also minimizes the potential for model confusion. Notably, in the YOLOv6-n [26] model, the detection accuracy, as measured by the $(\mathrm{m})\mathrm{AP}^{test}_{0.5}$ metric, shows an almost 20-percentage-point disparity favoring the 1-class labeling over the 3-classes labeling approach, underscoring the profound impact of annotation methods on model accuracy. Another critical metric, inference time (measured by the latency), also shows differences based on labeling approaches. Typically, models encounter faster decision-making processes when dealing with cracks labeled under a single category compared to those with three categories. However, the use of smaller models in these experiments means that the variance in inference time is relatively minimal. 3 classes labeling takes, on average, 2 ms longer in inference time compared to 1 class labeling, but this is not a greatly small gap in the inference time according to Table 3.

The qualitative comparisons of 1-class labeling versus 3-classes labeling for crack detection, as depicted in Fig. 12, further elucidate these findings. Models trained with a single-category crack annotation approach exhibit a lower propensity for missing crack detections, more precise and confident bounding box delineations, and a reduced tendency for redundant detection of individual cracks, compared to their counterparts trained with a multi-category annotation approach.

Based on the numerical metrics and actual detection results, it can be concluded that labeling cracks into one category has more advantages than subdividing cracks into three categories. It may be attributed to the fact that more categories can lead to more neuron nodes in the last classification layer of deep neural network model, which requires more time to infer. However, the subdivision of cracks into more categories may be helpful for the subsequent crack grade assessment and other related work.

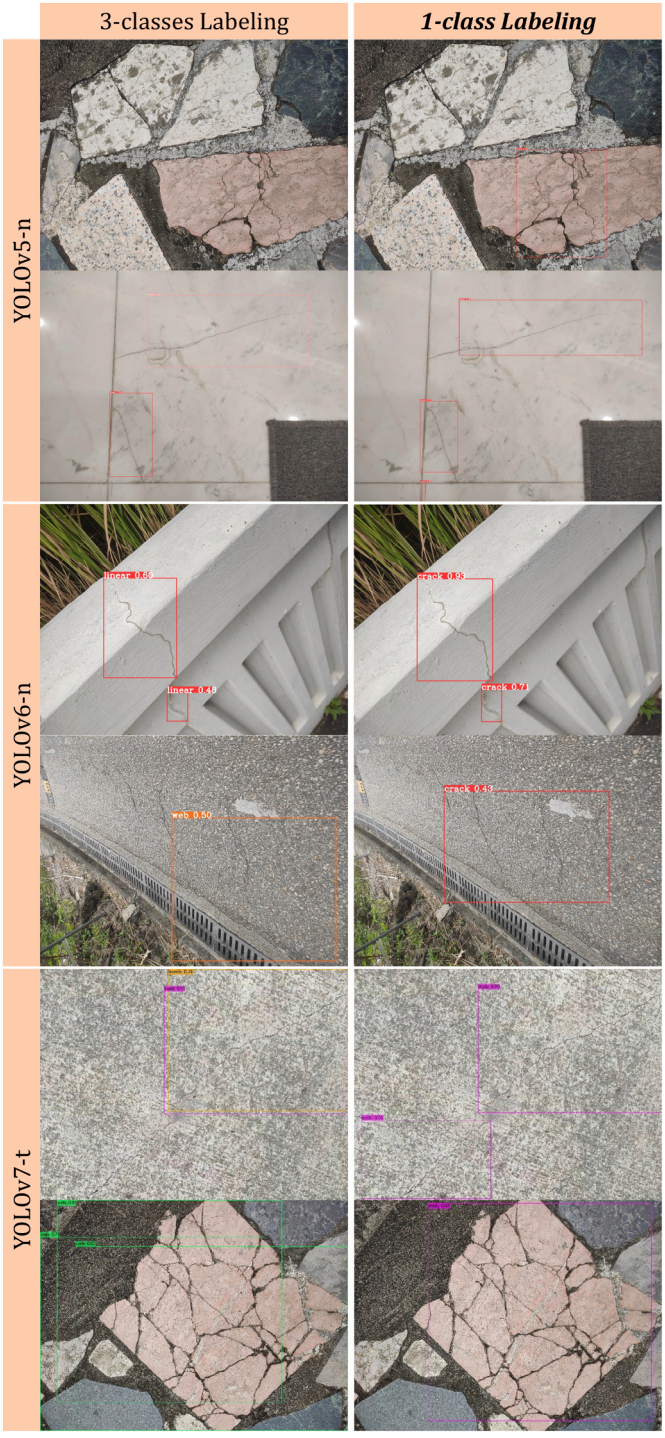


Fig. 12. Comparison of crack prediction results under 3-classes labeling and 1-class labeling. The left column shows the crack detection results under 3-classes labeling and the right column shows the results under 1-class labeling. The better detection results are demonstrated by the 1-class crack labeling method.

Table 4
The crack detection results about different annotation methods.

Label type	Model	Input size	$(\mathrm{m})\mathrm{AP}^{test}_{0.5}$	Latency (ms)
1-class Labeling	YOLOv5-n [60]	1024	44.6%	19.1
3-classes Labeling	YOLOv5-n [60]	1024	33.2%	22.0
1-class Labeling	YOLOv6-n [26]	1024	53.6%	21.8
3-classes Labeling	YOLOv6-n [26]	1024	34.3%	24.5
1-class Labeling	YOLOv7-t [61]	1024	43.8%	24.7
3-classes Labeling	YOLOv7-t [61]	1024	30.9%	26.8

5.3.5. Transferability of the CUBIT-Det dataset

In addition to the analyses previously discussed about CUBIT-Det, an effective approach to validate the newly proposed dataset involves directly assessing models trained on it using similar open-source datasets. Following an extensive review and analysis of datasets akin to our CUBIT-Det dataset, as detailed in Table 1, we select the SUT-Crack [33] dataset as the testing ground. Then, We conduct the zero-shot tests by using YOLOv5 [60], YOLOv6 [26], and YOLOv7 [61] models that have consistently shown superior detection performance on our dataset.

The decision to select the SUT-Crack dataset [33] for evaluation is based on its high-resolution data, which aligns with the quality of our CUBIT-Det dataset. And its labeling methodology, with a significant focus on "crack", closely resembles ours. Moreover, a more challenging aspect is that the SUT-Crack [33] contains numerous images with shadows caused by varying angles of light exposure. While other datasets referenced in our study, such as RDD [27–30] and CODEBRIM [46], differ significantly in terms of image scenarios, resolution, and the granularity of defect labels (ranging from overly detailed to excessively broad), they lack dedicated test sets. These datasets are primarily designed for training purposes and are not optimized for comprehensive testing. Given that there are no open-source datasets specifically tailored to building exterior surface defects, and considering the trailblazing role of our CUBIT-Det in providing an extensive range of building defect data, finding a perfectly compatible open-source test dataset is challenging. Therefore, taking all these factors into account, the SUT-Crack dataset [33] stands out as the most suitable option for our transferability testing, effectively demonstrating the adaptability and robustness of our dataset.

We conduct the zero-shot experiments under the same hardware environment as mentioned in Section 5.1, and the quantitative test results are demonstrated in Table 5. It is evident that models trained on our CUBIT-det dataset demonstrate considerable scalability, which also attests to the notable transferability of our dataset. Although our models are not trained on this SUT-Crack [33] dataset and are directly subjected to zero-shot testing, the performance on larger models of YOLOv5 [60], YOLOv6 [26] and YOLOv7 [61] are still quite favorable. Particularly noteworthy is the YOLOv6-l [26] model, which not only performed exceptionally well on our dataset but also continued to excel in this transferability test. It achieved an $AP_{0.5}$ of 89.6%, nearing 90%, and an $AP_{0.5:0.95}$ of 58.2%. This performance surpasses that on the test set of CUBIT-Det, validating both the transferability of our dataset and the scalability of models trained on it.

In Fig. 13, we demonstrate the qualitative detection results of SUT-Crack [33] from YOLOv6-l [26] trained on our CUBIT-Det. Despite the challenges posed by varied shadow and lighting conditions in the SUT-Crack [33] dataset and no more any other training, the YOLOv6-l [26] model, which is trained on our CUBIT-Det dataset, consistently exhibits exceptional performance in this zero-shot test. It accurately identifies and delineates almost every crack in its entirety, showcasing its robust detection capabilities.

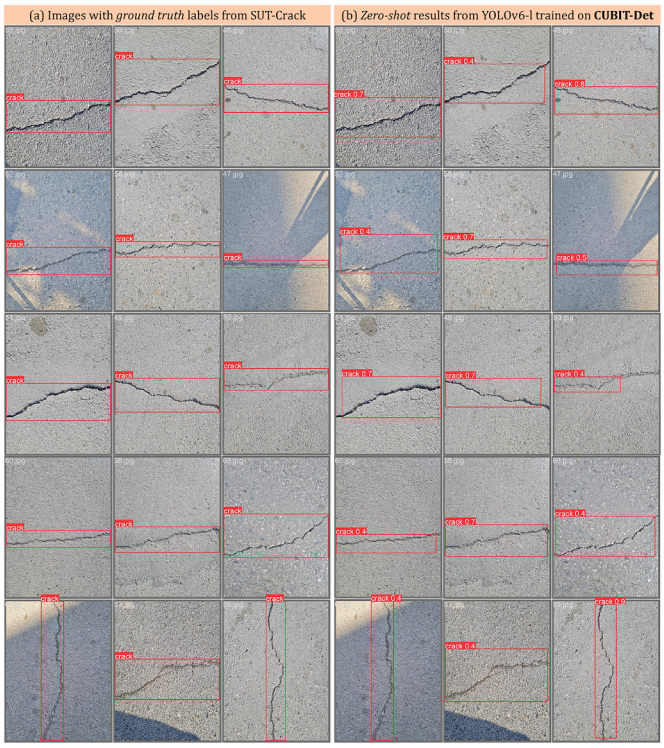


Fig. 13. The validation about the transferability of CUBIT-Det dataset. (a) Original images with groud-truth labels from SUT-Crack [33]; (b) Detection results from YOLOv6-l trained based on CUBIT-Det. Despite the interference caused by shadow and light exposure in the data, YOLOv6-l [26] trained on our CUBIT-Det dataset manage to figure out the cracks in zero-shot task.

6. Real-world experiment

To further verify the feasibility of our CUBIT-Det dataset, a real-world infrastructure inspection is conducted on an industrial building in Fo Tan, Sha Tin District, New Territories in Hong Kong. YOLOv6-l [26] is chosen because of its particularly good performance for crack detection, and the fact that the target building is relatively new, nearly without apparent spalling and moisture defects.

Three drones are used to ensure a high-efficiency and complete and coverage of the building façades and an exhaustive detection of cracks. The left part of Fig. 14 illustrates that the automated multi-UAVs system is utilized to capture the images of this industrial building equipped with our energy-saving path planning algorithm. At the same time, the detection model, YOLOv6-l [26], pre-trained based on our CUBIT-Det dataset, has a great detection performance, with almost no false detection and a low miss detection rate, accurately locating and detecting small defects.

In total, 1016 images are detected according to different wall surfaces, and some detected results have been shown in the right part of Fig. 14, including 129 for northeast façade, 200 for southeast façade, 393 for northwest façade, and 294 for southwest façade. Of the 1016 exterior wall images, our model detects 1095 cracks. Most of these defects belong to light, the detailed information is presented in Table 6. The Precision and Recall of this inspection task is 97.8% and 85.7%, respectively. The results are endorsed by the company of this industrial building. In addition to the visual inspection results in the images, the defect information (Table 7) also includes the defect class (1, 2, 3 represent crack, spalling, moisture), relative position (coordinates of the center point of the prediction box, with the upper left corner of the detected image as coordinate origin), dimension (width and height of the prediction box), and the confidence score of the detected defects. All these defect data information will be used in the subsequent workflow, for instance, within a GIS (Geographic information system) platform, the detected defects are registered onto a point cloud entity model reconstructed by our developed learning-based algorithm [64], utilizing both

Table 5
The test results on SUT-Crack [33] dataset.

Model	Input size	Precision	Recall	$AP^{test}_{0.5}$	$AP^{test}_{0.5:0.95}$
YOLOv5-m [60]	1024	57.9%	68.7%	59.9%	25.4%
YOLOv5-l [60]	1024	63.7%	73.9%	68.4%	33.8%
YOLOv5-x [60]	1024	72.3%	74.4%	71.6%	37.9%
YOLOv6-m [26]	1024	76.0%	82.8%	80.7%	48.4%
YOLOv6-l [26]	1024	82.7%	85.8%	89.6%	58.2%
YOLOv7 [61]	1024	70.8%	72.4%	66.2%	31.5%
YOLOv7-x [61]	1024	76.8%	69.4%	74.0%	41.8%

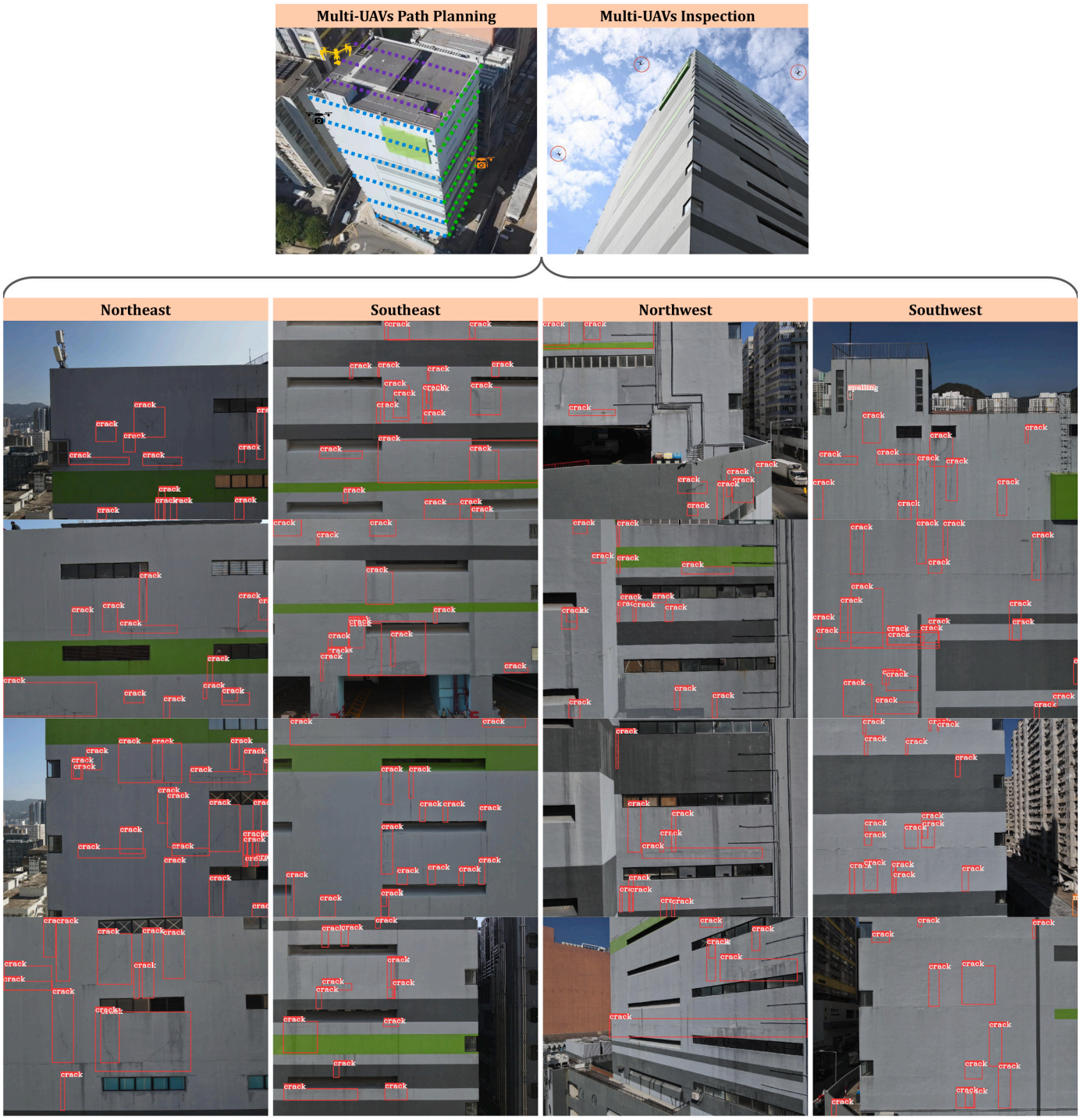


Fig. 14. Visualization of this real-world building inspection task. On the left, three of our UAVs (framed by the red ellipse) with path planning algorithm have cooperated to inspect the building. On the right, each of the four columns shows the results of the façade in one direction of this building.

Table 6
The distribution of cracks per direction in the target building.

Direction	Light	Mild	Moderate	Critical	Sum
Southeast	315	0	1	5	321
Southwest	291	9	9	26	335
Northeast	211	0	5	13	229
Northwest	189	0	3	18	210

global and relative positional information. This real-site detection results of this industrial building inspection task demonstrate that the deep learning models trained on our CUBIT-Det can be easily transferred to real-world applications.

7. Conclusions and discussion

This paper presents a multi-scenario, high-resolution, and sufficient dataset named CUBIT-Det, which is collected by our self-developed

Table 7
The example of output information from one detected image.

Defect class	Horizontal position	Vertical position	Box width	Box height	Confidence coefficient
1	0.938375	0.731833	0.010250	0.012000	0.217123
1	0.754062	0.992583	0.116625	0.013833	0.237203
1	0.679397	0.774417	0.033875	0.025833	0.533062
1	0.793875	0.618500	0.028500	0.016333	0.610724
1	0.790625	0.819833	0.030750	0.018333	0.676300
1	0.936188	0.171500	0.127125	0.343000	0.809036

automated unmanned system, for infrastructure inspections, especially in building applications. And we have evaluated nearly 30 state-of-the-art real-time deep-learning object detection algorithms based on our dataset to further verify its reliability. The detection performance of each model is comprehensively compared regarding the accuracy and the inference speed metrics, while the effect of different labeling approaches on the detection performance is also analyzed. In the selected models, YOLOv6 series demonstrate the best trade-off in latency versus mAP. Among them, YOLOv6-l performs the best, which can reach 82.9% and 55.9% under the two indicators $mAP^{test}_{0.5}$ and $mAP^{test}_{0.5:0.95}$, respectively. And YOLOv6-l only takes 15.9 ms to complete the inference of one 1024×1024 input image. More importantly, to validate the feasibility and expansibility of our CUBIT-Det, we conduct a real-world experiment on an industrial building in Hong Kong by using the model trained on our dataset, and excellent inspection results have been achieved.

Firstly, while we introduce a dataset centered on diverse infrastructure defects, especially for building façade, marking it as a pioneering collection, its volume of samples still remains suboptimal. The richness and diversity of a dataset, particularly in quantity, play a crucial role in ensuring the robustness and generalizability of a trained model. And, an insufficiently large dataset may lead to an overfitted model and reduced performance on unseen or real-world data. Therefore, in our future work, we aim to further optimize our unmanned system platforms to fully leverage its capabilities for collecting more data, thus expanding our dataset. We specifically intend to utilize drone platforms to gather data on exterior building surface defects, a challenging and underrepresented scenario in existing infrastructure datasets. Then, in the process of expanding the dataset, it is imperative to include images captured under various lighting conditions (such as daytime and nighttime) and different weather scenarios (overcast, rainy, and snowy conditions) to enhance the diversity of the dataset. Additionally, there should be an effort to incorporate a broader range of defect types, with particular attention paid to less common defect categories. These diverse data inputs will enable the trained models to become more robust and versatile, allowing for their application in a wider array of detection scenarios.

Furthermore, even though we rigorously test across nearly 30 object detection networks, emphasizing the extensive nature of our experiments, relying solely on existing algorithmic frameworks might not unveil the full potential of our dataset. The absence of a network tailored to infrastructure defect detection might inadvertently lead to the overlook of certain domain-specific defect types.

Lastly, but of utmost importance, in order to enhance the scalability and application scope of our dataset, it is crucial to extend beyond merely bounding box level annotations and incorporate pixel level segmentation masks as labels. Segmentation labels, being more granular, focus exclusively on the defect itself and better mitigate the influence of background noise. The addition of segmentation labels will render our dataset more comprehensive. Models trained on our dataset, augmented with these detailed annotations, will be equipped to handle a variety of task requirements, thus offering a more robust foundation for community.

In summary, this research provides a foundational work and indicates future directions for more efficiency infrastructure defect detection, despite space for further refinement and exploration. Future endeavors should focus on not only augmenting the volume, diversity and richness of dataset but also specializing a more lightweight but accurate algorithm tailored for the real-world infrastructure inspection tasks.

CRediT authorship contribution statement

Benyun Zhao: Conceptualization, Investigation, Formal analysis, Writing – original draft. **Xunkuai Zhou:** Conceptualization, Investigation, Methodology, Formal analysis, Writing – original draft. **Guidong Yang:** Investigation, Formal analysis, Writing – original draft. **Junjie Wen:** Investigation, Formal analysis, Writing – original draft. **Jihan Zhang:** Investigation, Formal analysis, Writing – original draft. **Jia Dou:** Investigation, Formal analysis, Writing – original draft. **Guang Li:** Investigation, Formal analysis, Writing – original draft. **Xi Chen:** Conceptualization, Resources, Supervision, Writing – review & editing, Project administration. **Ben M. Chen:** Conceptualization, Funding acquisition, Resources, Supervision, Writing – review & editing, Project administration.

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 the paper.

Data availability

Data will be made available on request.

Acknowledgements

This work is supported by the InnoHK of the Government of the Hong Kong Special Administrative Region via the Hong Kong Centre for Logistics Robotics (HKCLR).

References

[1] Ali Akbar Shirzadi Javid, Parviz Ghoddousi, Gholamreza Ghodrati Amiri, Khalil Donyadideh, A new 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, https://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, https://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, https://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, https://doi.org/10.1109/LRA.2021.3062599.
[5] Khashayar Asadi, Akshay Kalkunte Suresh, Alper Ender, Siddhesh Gotad, Suraj Maniyar, 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, https://doi.org/10.1016/j.autcon.2019.103068.
[6] Qingxiang Li, Guidong Yang, Chuanxiang Gao, Yijun Huang, Jihan Zhang, Dongyue Huang, Benyun Zhao, Xi Chen, Ben M. Chen, Single drone-based 3d reconstruction approach to improve public engagement in conservation of heritage buildings: a case of hakka tulou, J. Build. Eng. (2024) 108954, https://doi.org/10.1016/j.jobe.2024.108954.
[7] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90, https://doi.org/10.1145/3065386.
[8] Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv, 2014, https://doi.org/10.48550/arXiv.1409.1556.
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (June 2016), https://doi.org/10.1109/CVPR.2016.90.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, arXiv, 2020, https://doi.org/10.48550/arXiv.2010.11929.
[11] Navneet Dalal, Bill Triggs, Histograms of oriented gradients for human detection, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (July 2005), https://doi.org/10.1109/CVPR.2005.177.
[12] Pedro Felzenszwalb, David McAllester, Deva Ramanan, A discriminatively trained, multiscale, deformable part model, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (June 2008), https://doi.org/10.1109/CVPR.2008.4587597.
[13] 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, https://doi.org/10.1016/j.autcon.2021.103959.
[14] Qiuchen Zhu, Quang Phuc Ha, A bidirectional self-rectifying network with bayesian modeling for vision-based crack detection, IEEE Trans. Industr. Inform. 19 (3) (2022) 3017–3028, https://doi.org/10.1109/TII.2022.3172995.
[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, Li Fei-Fei, Imagenet: a large-scale hierarchical image database, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (June 2009), https://doi.org/10.1109/CVPR.2009.5206848.
[16] S.M. Mark Everingham, Ali Eslami, Luc Van Gool, Christopher K.I. Williams, John Winn, Andrew Zisserman, The pascal visual object classes challenge: a retrospective, Int. J. Comput. Vis. 111 (2015) 98–136, https://doi.org/10.1007/s11263-014-0733-5.
[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C. Lawrence Zitnick, Microsoft coco: common objects in context, Proc. Eur. Conf. Comput. Vis. (September 2014), https://doi.org/10.1007/978-3-319-10602-1_48.
[18] 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, https://doi.org/10.1016/j.conbuildmat.2022.129226.
[19] 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, Proc. Int. Joint Conf. Neural Netw. (May 2017), https://doi.org/10.1109/IJCNN.2017.7966101.
[20] Ronny Stricker, Markus Eisenbach, Maximilian Sesselmann, Klaus Debes, Horst-Michael Gross, Improving visual road condition assessment by extensive experiments on the extended gaps dataset, Proc. Int. Joint Conf. Neural Netw. (July 2019), https://doi.org/10.1109/IJCNN.2019.8852257.
[21] Mateusz Żarski, Bartosz Wójcik, Jarosław Adam Miszczak, Krakn: transfer learning framework and dataset for infrastructure thin crack detection, SoftwareX 16 (2021) 100893, https://doi.org/10.1016/j.softx.2021.100893.
[22] 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) (2020) 1525–1535, https://doi.org/10.1109/TITS.2019.2910595.
[23] Qipei Mei, Mustafa Gül, Md Riasat Azim, Densely connected deep neural network considering connectivity of pixels for automatic crack detection, Autom. Constr. 110 (2020) 103018, https://doi.org/10.1016/j.autcon.2019.103018.
[24] Ç.F. Özgenel, A. Gönenç Sorguç, Performance comparison of pretrained convolutional neural networks on crack detection in buildings, in: Proceedings of the International Symposium on Automation and Robotics in Construction, July 2018, https://doi.org/10.22260/ISARC2018/0094.
[25] Yongsheng Bai, Bing Zha, Halil Sezen, Alper Yilmaz, Deep cascaded neural networks for automatic detection of structural damage and cracks from images, in: International Society for Photogrammetry and Remote Sensing Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, V-2-2020, 2020, pp. 411–417, https://doi.org/10.5194/isprs-annals-V-2-2020-411-2020, 2020.
[26] Chuyi Li, Lulu Li, Yifei Geng, Hongliang Jiang, Meng Cheng, Bo Zhang, Zaidan Ke, Xiaoming Xu, Xiangxiang Chu, Yolov6 v3. 0: A Full-Scale Reloading, arXiv, 2023, https://doi.org/10.48550/arXiv.2301.05586.
[27] Hiroya Maeda, Yoshihide Sekimoto, Toshikazu Seto, Takehiro Kashiyama, Hiroshi Omata, Road damage detection using deep neural networks with images captured through a smartphone, Comput. Aided Civ. Inf. Eng. 33 (12) (2018) 1127–1141, https://doi.org/10.1111/mice.12387.
[28] Hiroya Maeda, Takehiro Kashiyama, Yoshihide Sekimoto, Toshikazu Seto, Hiroshi Omata, Generative adversarial network for road damage detection, Comput. Aided Civ. Inf. Eng. 36 (1) (2021) 47–60, https://doi.org/10.1111/mice.12561.
[29] Deeksha Arya, Hiroya Maeda, Sanjay Kumar Ghosh, Durga Toshniwal, Yoshihide Sekimoto, Rdd2020: an annotated image dataset for automatic road damage detection using deep learning, Data Brief 36 (2021) 107133, https://doi.org/10.1016/j.dib.2021.107133.
[30] Deeksha Arya, Hiroya Maeda, Sanjay Kumar Ghosh, Durga Toshniwal, Yoshihide Sekimoto, Rdd2022: A Multi-National Image Dataset for Automatic Road Damage Detection, arXiv, 2022, https://doi.org/10.48550/arXiv.2209.08538.
[31] Hamed Majidifard, Peng Jin, Yaw Adu-Gyamfi, William G. Buttlar, Pavement image datasets: a new benchmark dataset to classify and densify pavement distresses, Transp. Res. Rec. 2674 (2) (2020) 328–339, https://doi.org/10.1177/0361198120907283.
[32] Murad Al Qurishee, Weidong Wu, Babatunde Atolagbe, Joseph Owino, Ignatius Fomunung, Mbakisya Onyango, Creating a dataset to boost civil engineering deep learning research and application, Engineering 12 (3) (2020) 151–165, https://doi.org/10.4236/eng.2020.123013.
[33] Mohammadreza Sabouri, Alireza Sepidbar, Sut-crack: a comprehensive dataset for pavement crack detection across all methods, Data Brief 51 (2023) 109642, https://doi.org/10.1016/j.dib.2023.109642.
[34] Martin Mundt, Sagnik Majumder, Sreenivas Murali, Panagiotis Panetsos, Visvanathan Ramesh, Meta-learning convolutional neural architectures for multi-target concrete defect classification with the concrete defect bridge image dataset, Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (June 2019), https://doi.org/10.1109/CVPR.2019.01145.
[35] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative adversarial nets, Proc. Adv. Neural Inf. Process. Syst. (December 2014), https://doi.org/10.1145/3422622.
[36] Lulu Tian, Zidong Wang, Weibo Liu, Yuhua Cheng, Fuad E. Alsaadi, Xiaohui Liu, A new Gan-based approach to data augmentation and image segmentation for crack detection in thermal imaging tests, Cogn. Comput. (2021) 1263–1273, https://doi.org/10.1007/s12559-021-09922-w.
[37] Boqiang Xu, Chao Liu, Pavement crack detection algorithm based on generative adversarial network and convolutional neural network under small samples, Measurement (2022) 111219, https://doi.org/10.1016/j.measurement.2022.111219.
[38] Alec Radford, Luke Metz, Soumith Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, arXiv, 2015, https://doi.org/10.48550/arXiv.1511.06434.
[39] Tero Karras, Timo Aila, Samuli Laine, Jaakko Lehtinen, Progressive Growing of Gans for Improved Quality, Stability, and Variation, arXiv, 2017, https://doi.org/10.48550/arXiv.1710.10196.
[40] Qipei Mei, Mustafa Gül, A cost effective solution for pavement crack inspection using cameras and deep neural networks, Constr. Build. Mater. 256 (2020) 119397, https://doi.org/10.1016/j.conbuildmat.2020.119397.
[41] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, Ssd: single shot multibox detector, Proc. Eur. Conf. Comput. Vis. (October 2016), https://doi.org/10.1007/978-3-319-46448-0_2.
[42] Hadi Daneshmand, Jonas Kohler, Francis Bach, Thomas Hofmann, Aurelien Lucchi, Batch Normalization Provably Avoids Ranks Collapse for Randomly Initialised Deep Networks, arXiv, 2020, https://doi.org/10.48550/arXiv.2003.01652.
[43] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv, 2017, https://doi.org/10.48550/arXiv.1704.04861.
[44] Joseph Redmon, Ali Farhadi, Yolo9000: better, faster, stronger, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (July 2017), https://doi.org/10.1109/CVPR.2017.690.
[45] Ross Girshick, Fast r-cnn, Proc. IEEE Int. Conf. Comput. Vis. (December 2015), https://doi.org/10.1109/ICCV.2015.169.
[46] Bowen Baker, Otkrist Gupta, Nikhil Naik, Ramesh Raskar, Designing Neural Network Architectures using Reinforcement Learning, arXiv, 2016, https://doi.org/10.48550/arXiv.1611.02167.
[47] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, Jeff Dean, Efficient Neural Architecture Search via Parameters Sharing, arXiv, 2018, https://doi.org/10.48550/arXiv.1802.03268.
[48] Wei Zhou, Yunfei Zhan, Hancheng Zhang, Lei Zhao, Chen Wang, Road defect detection from on-board cameras with scarce and cross-domain data, Autom. Constr. 144 (2022) 104628, https://doi.org/10.1016/j.autcon.2022.104628.
[49] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, Faster r-cnn: towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2017) 1137–1149, https://doi.org/10.1109/TPAMI.2016.2577031.
[50] Qi Fan, Wei Zhuo, Chi-Keung Tang, Yu-Wing Tai, Few-shot object detection with attention-rpn and multi-relation detector, Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (June 2020), https://doi.org/10.1109/CVPR42600.2020.00407.
[51] Hongwen Dong, Kechen Song, Qi Wang, Yunhui Yan, Peng Jiang, Deep metric learning-based for multi-target few-shot pavement distress classification, IEEE Trans. Industr. Inform. 18 (3) (2021) 1801–1810, https://doi.org/10.1109/TII.2021.3090036.
[52] Tarek Rakha, Alice Gorodetsky, Review of unmanned aerial system (uas) applications in the built environment: towards automated building inspection procedures using drones, Autom. Constr. 93 (2018) 252–264, https://doi.org/10.1016/j.autcon.2018.05.002.
[53] Christian Eschmann, Timo Wundsam, Web-based georeferenced 3d inspection and monitoring of bridges with unmanned aircraft systems, J. Surv. Eng. 143 (3) (2017) 04017003, https://doi.org/10.1061/(ASCE)SU.1943-5428.0000221.
[54] Tixiao Shan, Brendan Englot, Lego-loam: lightweight and ground-optimized lidar odometry and mapping on variable terrain, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (October 2018), https://doi.org/10.1109/IROS.2018.8594299.
[55] Tixiao Shan, Jinkun Wang, Brendan Englot, Kevin Doherty, Bayesian generalized kernel inference for terrain traversability mapping, Proc. Conf. Robot Learn. (October 2018). PMLR 87:829-838, https://proceedings.mlr.press/v87/shan18a.html.
[56] Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, Shilei Wen, Pp-yolo: An Effective and Efficient Implementation of Object Detector, arXiv, 2020, https://doi.org/10.48550/arXiv.2007.12099.
[57] Xin Huang, Xinxin Wang, Wenyu Lv, Xiaying Bai, Xiang Long, Kaipeng Deng, Qingqing Dang, Shumin Han, Qiwen Liu, Xiaoguang Hu, Dianhai Yu, Yanjun Ma,

Yoshie Osamu, Pp-yolov2: A Practical Object Detector, arXiv, 2021, https://doi.org/10.48550/arXiv.2104.10419.
[58] Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui, Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, Baohua Lai, Pp-yoloe: An Evolved Version of Yolo, arXiv, 2022, https://doi.org/10.48550/arXiv.2203.16250.
[59] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, Jian Sun, Yolox: Exceeding Yolo Series in 2021, arXiv, 2021, https://doi.org/10.48550/arXiv.2107.08430.
[60] Glenn Jocher, YOLOv5 by Ultralytics, 2020, p. 5.
[61] Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao, Yolov7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (June 2023), https://doi.org/10.1109/CVPR52729.2023.00721.
[62] Jan Hosang, Rodrigo Benenson, Bernt Schiele, Learning non-maximum suppression, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (July 2017), https://doi.org/10.1109/CVPR.2017.685.
[63] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, Mobilenetv2: inverted residuals and linear bottlenecks, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (June 2018), https://doi.org/10.1109/CVPR.2018.00474.
[64] Guidong Yang, Xunkuai Zhou, Chuanxiang Gao, Xi Chen, Ben M. Chen, Learnable cost metric-based multi-view stereo for point cloud reconstruction, IEEE Trans. Ind. Electron. (2023) 1–10, https://doi.org/10.1109/TIE.2023.3337697.