Periodica Polytechnica Transportation Engineering, 66(2), pp. 516–531, 2022

Lightweight Real-time Detection of Components via a Micro Aerial Vehicle with Domain Randomization Towards Structural Health Monitoring

Isaac Osei Agyemang[1*], Xiaoling Zhang[1], Isaac Adjei-Mensah[1], Joseph Roger Arhin[1], Emmanuel Agyei[1]

[1] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[*] Corresponding author, e-mail: ioagyemang@std.uestc.edu.cn

Received: 01 June 2021, Accepted: 02 February 2022, Published online: 17 February 2022

Abstract

Civil structural component detection plays an integral role in Structural Health Monitoring (SHM) pre and post-construction. Challenges including but not limited to labor-intensiveness, cost, and time constraints associated with traditional methods make it a less optimal approach in SHM. Despite the success of deep convolutional neural networks in diverse detection problems, the required computational resources are a challenge. This has led to rendering a chunk of resource-constrained edge nodes less applicable with deep convolutional neural networks. In this paper, a computational-efficient deep convolutional neural network is presented based on Gabor filters and a color Canny edge detector. Generic Gabor filters are generated and used as initializers in the computational-efficient deep convolutional neural network presented, afterward trained on building components data. Next, extensive offline and online experimentation with a resource-constrained edge node is conducted and evaluated using diverse metrics. The computational-efficient detection model demonstrates to be effective in detection and via NVIDIA GPU profiler, we observe conservation of around 30% of computational resources during training. The computational-efficient detection model adduces almost a 3% mean average precision higher than two state-of-the-art detectors and records a promising frame processing rate during the online experimentation.

Keywords

Gabor filters, color Canny edge detector, micro aerial vehicle, structural health monitoring, deep convolutional neural network

1 Introduction

With the advancement of urban life, the continuous requirement to provide shelter for humanity over natural occurrences (rain, strong winds, intense sunlight, etc.) keeps on rising with diverse construction standards before and post-construction of an infrastructure [1]. Primarily, the construction of buildings that come in variant dimensions, architectural designs, and construction materials addresses the problem of shelter but at a cost (corrosion, deterioration, cracks, etc.). To this end, periodic inspection of the structural health of civil infrastructures such as buildings is knee to aid the prevention of accidental structural collapse. At present, Structural Health Monitoring (SHM) is the branch in civil engineering denoted to inspect the health condition of civil structures [2]. SHM entails an array of diagnostic tools and analyzing techniques that seek to offer timely and accurate diagnoses and analyzes a wide range of civil structures (e.g., buildings, bridges, etc.), and report their health status. Detection of structural parts precedes the diagnostic and analysis aspect of SHM since the different parts of an infrastructure (e.g., building.) require varying maintenance. Besides, detecting the various components of infrastructure is peculiar to the interpretation of damages, health evaluation of the structure, and the optimal maintenance required.

SHM has seen the integration of deep learning which at present is the backbone of many vision-based tasks under categories of classification, detection, and segmentation [3] with contact and non-contact sensors. Contact sensors (fiber optic sensors, strain gauges, etc.) are associated with erroneous readings, costly installation, and maintenance. As a result, non-contacting sensors (high-speed cameras, unmanned aerial vehicles, etc.) are being utilized due to advantages including easy deployment, reliability in data acquisition, and cost-effectiveness [4].

A non-contacting sensor such as an Unmanned Aerial Vehicle (UAV) categorically is in two groups, macro and micro UAVs [5]. The majority of macro UAVs support the integration of extra sensing modalities (flight computers, etc.) to provide extra computing power to meet compute demands of deep learning. Such cannot be said for resource-constrained edge nodes such as Micro Aerial Vehicles (MAVs). This has rendered the majority of MAVs redundant in the civil engineering domain. Tasks such as site monitory, infrastructure inspection (analogous inspections, data collection, etc.) can be accomplished via MAVs together with deep learning before deployment of costly macro UAVs. However, the challenge faced by many civil engineers is the computational requirements of deep learning technologies.

Whence, as a contribution to the SHM domain, we put forward a computational-efficient detector for building components detection task. We present a variant Deeply Supervised Object Detector (DSOD) [6] integrated with a variant Canny edge detector [7] and Gabor filters [8] and introduce the domain randomization technique to mitigate compute requirements without compromising performance. The remaining content of the paper is structured as follows: Sections 2 and 3 present related works and data acquisition and preparation, respectively. The computational-efficient detector is described in Section 4 followed by extensive experiments in Section 5. Section 6 draws the conclusions and future works.

2 Related works

2.1 Deep learning-based SHM

Classification, localization, detection, and segmentation problems in SHM are being mitigated via the use of deep learning technologies [9]. In [10], a pre-trained state-of-the-art classifier VGG-16 is used in the classification of building detrimental (mold, stain, deterioration, etc.) caused by dampness. The authors used VGG-16 to extract features from building data, fed to the network, and with the aid of fully connected layers and a softmax, classification of building damages was attained. An accuracy of 87.50% was adduced in classifying defects of buildings. The authors' approach is promising yet feature specifics related to the training data are not learned since there is no retraining of convolutional blocks. A combination of Convolutional Neural Network (CNN) and conventional machine learning methods (random forest and support vector machine) is presented in [11] for the detection of cracks in concrete building structures. A conclusion drawn by the authors indicates CNNs with conventional machine learning methods such as SVM as a classifier comparative to CNNs with softmax as a classifier, the former outperforms the latter. The experimental results reported support the conclusion of the authors' claims yet generalization of the presented approach is not established.

In another SHM task, corrosion detection is studied in [12] by the proposition of a classifier based on a sliding window and 4 color spaces (RGB, YCbCr, CbCr, and grayscale). Conversion of images to the 4 aforementioned color spaces before input to the Deep Convolutional Neural Network (DCNN) adds up to the computational bottleneck of DCNN. Reference [13] investigates an SHM task based on vibrations to detect defects associated with a civil structure. The authors claim a 100% accurate prediction of defects ranging from minor to extensive damage, this is open to discussion. A one-dimensional CNN is used in the detection of defects associated with civil structures, however, the challenge of requiring large measurements of civil structures to form training data is a limitation to the one-dimensional CNN proposition in [14, 15]. As a result, a non-parametric CNN method is proposed in [16] to address the challenge in [14, 15]. Autoencoders and layer-wise training which constitutes a computer vision merged with deep learning method is used in the classification of a six-class anomaly on a civil structure. An accuracy of 87% was attained including acknowledging the limitation of too many manual representations of anomalies [17]. A fusion CNN for detecting cracks in a steel box girder of bridges is presented in [18] with a classification accuracy of 96.38%. The fusion CNN classifier follows a binary conversion and optimal entropy threshold process in detecting cracks. Crack detection by the fusion CNN classifier has a limitation at certain distances to objects. The introduction of hyperparameters in CNN to alleviate overfitting is presented in [19] in the analysis of a truss bar planar. The conclusions drawn by the authors indicate the success of their proposition, it must be noted the hyperparameters introduced are task-specific and not generic.

2.2 Deep learning-UAV-based SHM

Kang and Cha [20] proposed a DCNN beacon system with geo-tagging for SHM. The authors' proposition constitutes an ultrasonic beacon system for mapping and position localization of the UAV based on a mission planner equipped with navigation maps. A Pixhawk 2.1 flight controller serves as means of controlling the UAV guided by the beacon routers together with a ground station computer for compute processing. The UAV was flown over the region of interest, and a DCNN running on the ground station computer estimated cracked and non-cracked regions. A 97.7% specificity with a sensitivity of 91.9% was attained. A wall-climbing Unmanned Aerial System (UAS) coupled with CNN is proposed in [21] for crack assessment. Through a wireless data transmission, data captured from the UAS is sent to an Android platform for crack detection based on CNN. The authors' UAS CNN-based system for crack detection is made up of the UAS (flight controller, a camera with undistorted lens, soft wheel, data transmitter, battery, and horizontal and vertical motors) and a ground station. The conclusions drawn by the authors indicate an accuracy of 94.48% for crack detection together with less than 5% error in detection of cracks with width measured below 0.1mm. A UAV-based thermographic encoder-to-decoder for segmentation of a bridge deck delamination towards SHM is given in [22]. A DJI Matrix 600 equipped with a thermal camera (FLIR A8300) with an orthogonal field of view medication is used for a deck delamination assessment. An onboard computer together with a customized MATLAB algorithm running off the DJI Matrix 600 was used in the processing of the data before the deep learning-based encoder-to-decoder is applied for segmentation. In [23], a U-Net CNN-based UAV assessment of the serviceability of pavements is presented. Based on UAV photogrammetry data, the authors employed a well-known architecture (U-Net) in the biomedical domain for the segmentation task, 95% accuracy is reported by the authors. DJI Phantom 4 pro, control points, a Topcon ISO1 Robotic Total Station detector, and GPS antennas formed part of the UAV system used. Lee et al. [24] in a field test, applied Recurrent Neural Network (R-CNN) to photographs acquired via a variant EX-drone (Korea) equipped with an HDR-PER 790V and FDR 3000r cameras using a customized MATLAB application.

An identity running through the reiterated works indicates the dominance of macro UAVs in SHM tasks. Based on the reiterated works, by far at the time of writing, this paper would be the first deep learning-based micro UAV SHM attempt with promising signs.

3 Data acquisition and preparation

At the time of writing this paper, there is no established and open structural image databank such as the ImageNet for scientific and engineering studies within the civil construction set [25]. As such, structural data related to buildings are collected to constitute a mini-structural databank for this study. First, an internet data gathering is conducted using structural digital libraries (National Information Service for Earthquake Engineering (NISEE), Design and Safe) and generic digital libraries (Google Image, Baidu Image, and Yahoo). To add up to the surfed data, a DJI Phantom 3 and a smartphone were used to take pictures within the localities of the authors of this paper. Next, image annotation is key in object detection tasks, as such we manually annotate the sampled images via the use of the VGG Image Annotator tool [26]. Circular region shape, 2D bounding box, Elliptical region shape, Polygon region shape, Polyline region shape, and Point region shape are the annotation shapes available in the VGG Image annotator tool at the time of access. Due to the asymmetrical shapes of the components of buildings, three annotation shapes (2D bounding box, Circular, and Polyline region shapes) were used for the annotation. A synopsis of the databank reposited at [27] is tabulated in Table 1.

4 Lightweight detector

4.1 DSOD overview

The DSOD combines the Single Short Detector (SSD) [28] which is multi-scale proposal-free and DenseNet [29] which introduced deep supervision via dense CNN connections. DSOD network has two main parts, the backbone network for feature extraction and the front-end (detection layers) for object detection. The backbone network comprises a stem block, four dense blocks, two transition layers, and two transition layers without pooling. The front-end contrary to the plain structure of SSD uses an elaborated dense connection to fuse multiscale predicting maps. DSOD is advantageous over other detectors which dwell on pre-trained models. We conjecture that features learned from ImageNet may differ from features of structural imagery (e.g., bridges, etc.). As such training from scratch with few data samples places DSOD ahead of other detectors within this context. However, the network configuration of DSOD overwhelms resource-constrained edge nodes. Reducing the network parameters is

Table 1 Details of building components used in the study

Building components (classes)	Quantity	Source	Annotation type
Door, Window, Pillar, Beam, Shear wall/ Wall, Roof, and Staircase	6,675	Google image, Baidu image, Yahoo, smartphone, NISEE, and DJI Phantom 3	2D bounding box, Circular region shape, and Polyline region shape

a step towards adducing a computational-efficient detector but performance may be compromised. Hence meaningful feature extraction is required to compliment the performance.

4.2 Color Canny edge detector

Canny edge detectors [30] have the advantage of detecting edges with minimum computation yet it is limited to grayscale images. The Canny edge detector operates on a four-stage module; (1) noise reduction, (2) gradient computation, (3) non-maximum suppression, and (4) hysteresis thresholding. An improved color Canny edge detector approach is adopted. In [7] the authors replace the first stage (noise reduction via a Gaussian function) with a quaternion weighted filter which extends a two-dimensional array. As such an RGB image can be expressed in a pure quaternion as:

$$Q_{x,y} = r_{x,y}i + g_{x,y}j + b_{x,y}k \,, \tag{1}$$

where x, y represents coordinates of a color pixel, i, j, k denotes an imaginary part and $r, g, b \in \{0,1,2,\ldots,255\}$ represents the real part. In the second stage, a Sobel kernel operates on an RGB image expressed in a three-dimensional vector $\boldsymbol{I} = r_{x,y}, g_{x,y}, b_{x,y}$ given in a row and column as:

$$\Delta \boldsymbol{H} = \left(H^- \;\; 0 \;\; H^+ \right), \tag{2}$$

$$\Delta V = \begin{bmatrix} V^- \\ 0 \\ V^+ \end{bmatrix}, \tag{3}$$

to compute gradients that detect edge intensity within an RGB color space between 0 to 255. H^+, H^-, V^+ and V^- are given as:

$$H^+\left(x_0, y_0\right) = \frac{1}{4}\begin{bmatrix} I\left(x_0+1, y_0-1\right) + 2I\left(x_0+1, y_0\right) \\ +I\left(x_0+1, y_0+1\right) \end{bmatrix}, \tag{4}$$

$$H^-\left(x_0, y_0\right) = \frac{1}{4}\begin{bmatrix} I\left(x_0-1, y_0-1\right) + 2I\left(x_0-1, y_0\right) \\ +I\left(x_0-1, y_0+1\right) \end{bmatrix}, \tag{5}$$

$$V^+\left(x_0, y_0\right) = \frac{1}{4}\begin{bmatrix} I\left(x_0-1, y_0+1\right) + 2I\left(x_0, y_0+1\right) \\ +I\left(x_0+1, y_0+1\right) \end{bmatrix}, \tag{6}$$

$$V^-\left(x_0, y_0\right) = \frac{1}{4}\begin{bmatrix} I\left(x_0-1, y_0-1\right) + 2I\left(x_0, y_0-1\right) \\ +I\left(x_0+1, y_0-1\right) \end{bmatrix}, \tag{7}$$

where x_0, y_0 denotes coordinates of a color pixel. Then follows the magnitude G and slope θ which are expressed as:

$$G_{x,y} = \sqrt{\left\|\Delta \boldsymbol{H}(x,y)\right\|^2 + \left\|\Delta \boldsymbol{V}(x,y)\right\|^2} \,, \tag{8}$$

$$\theta = arctan\left(\frac{\Delta \boldsymbol{V}(x,y)}{\Delta \boldsymbol{H}(x,y)} \right). \tag{9}$$

Non-maximum suppression is performed to intensify the edges (thin out thick and thin edges, i.e., uniform RGB values) followed by double thresholding which connects the detected edges based on strong and weak pixels. In Fig. 1, we show a non-maximum suppression technique. From Fig. 1, the pixel within the dotted red box is being processed and being compared with pixels within the green dotted box in the direction of an edge, blue dotted arrow. If one of the pixels (a, b + 1) or (a, b – 1) is more intense than the pixel (a, b) being processed, then the more intense pixel (a, b + 1) is kept and the RGB values of the other pixels (a, b – 1) and (a, b) are reduced by the product of the Sobel kernel (i.e., a 3 × 3 = 9). This results in an RGB image where the edges of objects are intensified. To produce a monochrome image, the RGB values of the most intense pixel are replaced with 0 and the less intense pixels with 255.

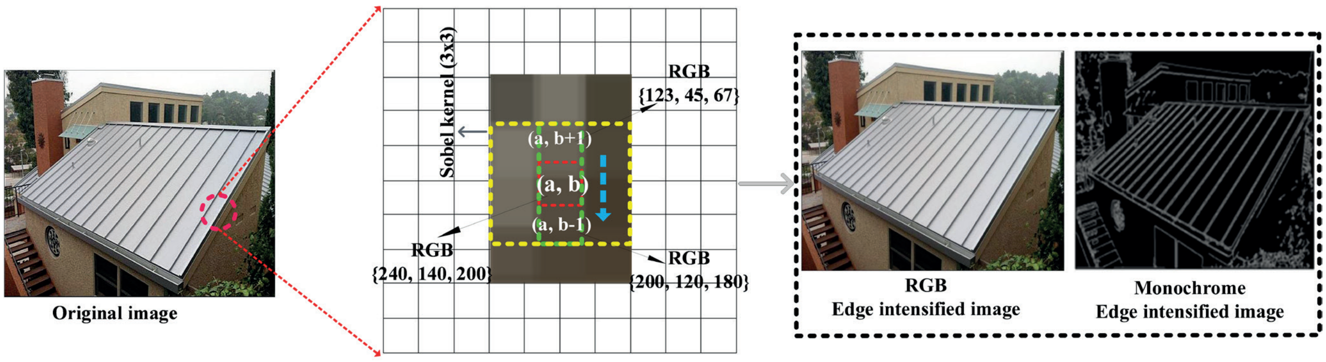


Fig. 1 Illustration of non-maximum suppression within color Canny edge detector

4.3 Gabor filters

Filters are important in image analysis as it is through filters content of an image is understood via transformation of the image into some other domain to detect the presence of a feature. Several image analysis filters such as mean, median, min-max, gaussian, bilateral, Gabor, and convolutional filters do exist with varying pros and cons. Gabor filters, a novel contribution from Denis Gabor are orientation-sensitive filters with capabilities of feature extraction, texture analysis, and disparity estimations of an image. A multiplication of a Gaussian envelope function with a complex oscillation and an extension of the aforementioned functions results in two-dimensional filters. Primarily, Gabor filters are mathematically structured to cope with diverse feature sizes, orientation, shape, and texture within a given image. There have been some studies suggesting the use of Gabor filters in place of convolutional filters [31]. Krizhevsky et al. [32] experimentally show features extracted using convolutional filters have high similarity to that of features extracted using Gabor filters. To some extent, convolved features from convolutional filters share some similarities with color blobs as well. As a result, Gabor filters have been used in some vision-based problems together with CNNs [33, 34]. Gabor filters defined by parameters orientation, wavelength, phase offset, bandwidth, and aspect ratio are some of the key advantages Gabor filters have over convolutional filters. Programmatically using OpenCV-Python, Gabor filters generation follows the structure: *cv.getGaborKernel(ksize, theta, lambda, gamma, psi, ktype),* where *ksize* = Gabor kernel size, *sigma* = bandwidth, *theta* = orientation, *lambda* = wavelength, *gamma* = aspect ratio, *psi* = phase offset, and *ktype* = range of values each Gabor filter can hold. The computational advantage of Gabor filters is attributed to their kernel size, *ksize* = (*a*, *b*) (i.e, *a* × *b* pixels) as opposed to mostly an odd-sized convolutional kernel, *k* (e.g., 3 × 3, 5 × 5, etc.) with the kernel being squared, k^2.

To this end, we employ Gabor filters in the lower layers of a DCNN to mitigate compute demands during the training of the DCNN model. Reference to [33, 34], through a Gabor filter bank, we generate generic Gabor filters as initializers for a DCNN model. Gabor function has parameters as follows:

$$g = \left(x, y, \lambda, \theta, \varphi, \sigma, \gamma\right), \tag{10}$$

where *x*, *y* denotes Gabor kernel size, λ is the wavelength, θ is the orientation, φ represents phase offset, σ denotes bandwidth, and γ is the aspect ratio. A Gabor function is given based on the aforementioned parameters as:

$$g = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)\exp\left(i\left(2\pi\frac{x'}{\lambda} + \varphi\right)\right), \tag{11}$$

where $x' = x\ cos\theta + y\ sin\theta$, $y' = -x\ sin\theta + y\ cos\theta$, and i denotes an imaginary parameter. However, since we use real values for Gabor filters, the expression unfolds in two phases: (1) real, given as:

$$g = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)cos\left(2\pi\frac{x'}{\lambda} + \varphi\right), \tag{12}$$

and (2) the imaginary, given as:

$$g = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)sin\left(2\pi\frac{x'}{\lambda} + \varphi\right). \tag{13}$$

A Gabor filter has a valid orientation θ between 0 and 2π which is given as:

$$\theta_m = \frac{\pi}{2}\left(m-1\right);\ m = 1, 2, \ldots, 8, \tag{14}$$

and an associated wavelength $\boldsymbol{\lambda}$ being either **2** or >**2**, a phase offset $\boldsymbol{\varphi}$ with valid values in the range of $-\boldsymbol{\pi}$ to $\boldsymbol{\pi}$, an aspect ratio $\boldsymbol{\gamma}$ between 0 and 1, and a bandwidth $\boldsymbol{\sigma} > \mathbf{0}$. In Fig. 2 are randomly generated generic Gabor filters via a Gabor filter bank, random data samples processed through the generic Gabor filters, and a visualization of an image processed through an intermittent generic Gabor filter kernel sizes.

4.4 Domain randomization

CNN/DCNN models generalize well as a result of training on diverse data characterized by varying environmental parameters, but in the real world, such data is hard to come by and expensive to produce. For some time now, augmenting training data has been one of the approaches in aiding CNN/DCNN models to attain good generalization. Domain Randomization herein DR can be said to offer an enhanced version of data augmentation. Data augmentation primarily introduces noise such as hue, saturation, rotation, flips, and other image renderings which in a hierarchical level falls within a moderate measure with regards to image rendering techniques. On the other hand, DR offers advanced image renderings including, positioning and orientation of objects of interest, environment control, texture control, lighting control, camera field of view control, and random noise addition, which data augmentation provides. To this end, we adopt DR in the study by utilizing Blender [35], a computer graphics application suitable for rendering. Using diverse 3D models of buildings with varying architectural plans, lighting, camera

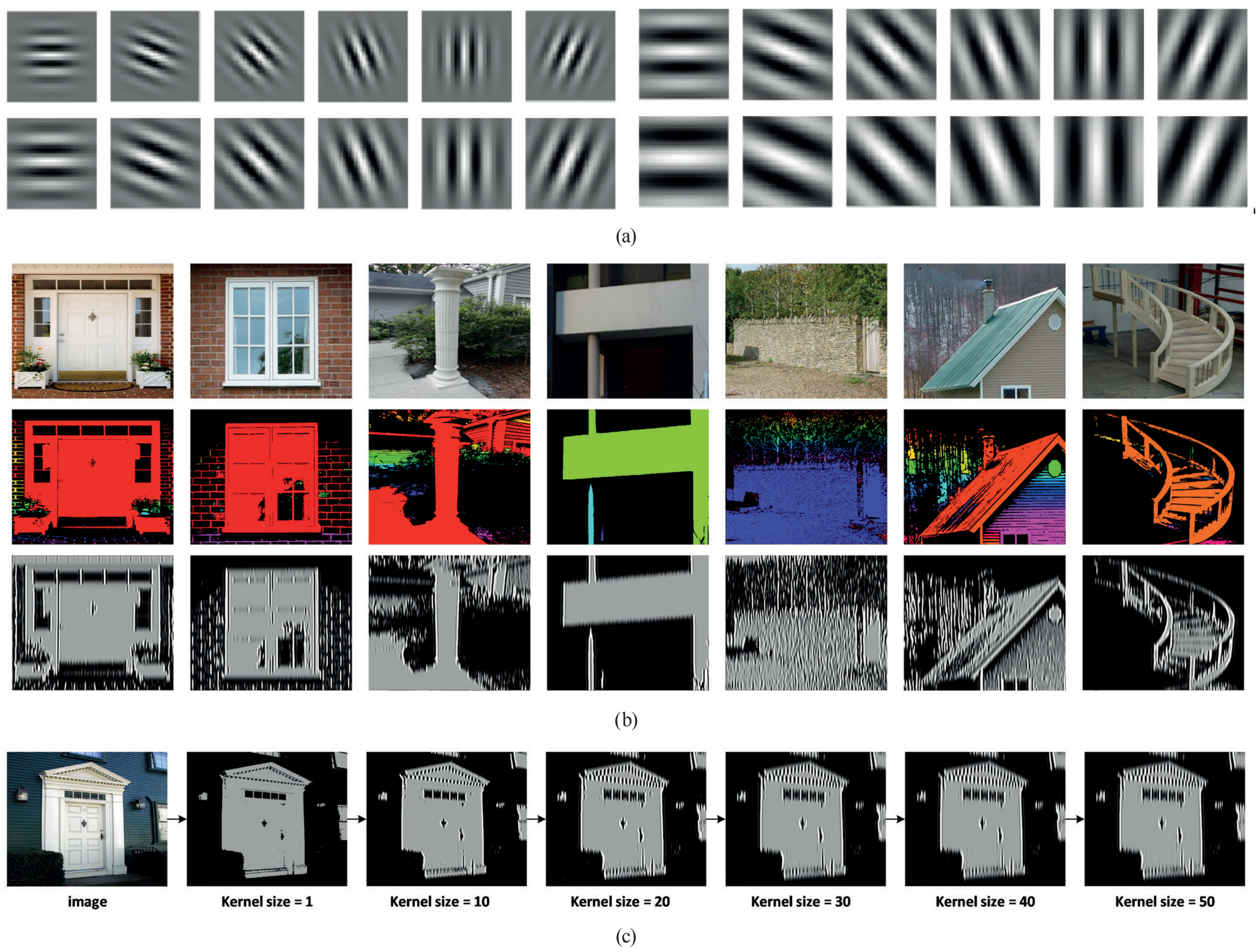


Fig. 2 (a) Generic Gabor filters with varying parameters (orientation, wavelength, phase offset, aspect ratio, and bandwidth), (b) Random selected building components filtered through generated generic Gabor filters, and (c) sample image processed through generic Gabor filters at incremental kernel sizes

field of view, positioning, and orientation of the object of interest, intensity, texture, and random noise parameters, respectively are randomized to create the most varied data in addition to the real data gathered. Dwelling on the automated annotation feature Blender provides, key building components are annotated and cropped to generate synthetic data. By randomizing the parameters less important for detecting and localizing a building component, the DCNN model is intended to learn to generalize faster. Using DR, 1000 varied synthetic data is generated in addition to the collected data. For brevity, randomly selected 3D buildings and cropped annotated building components are visualized in Fig. 3.

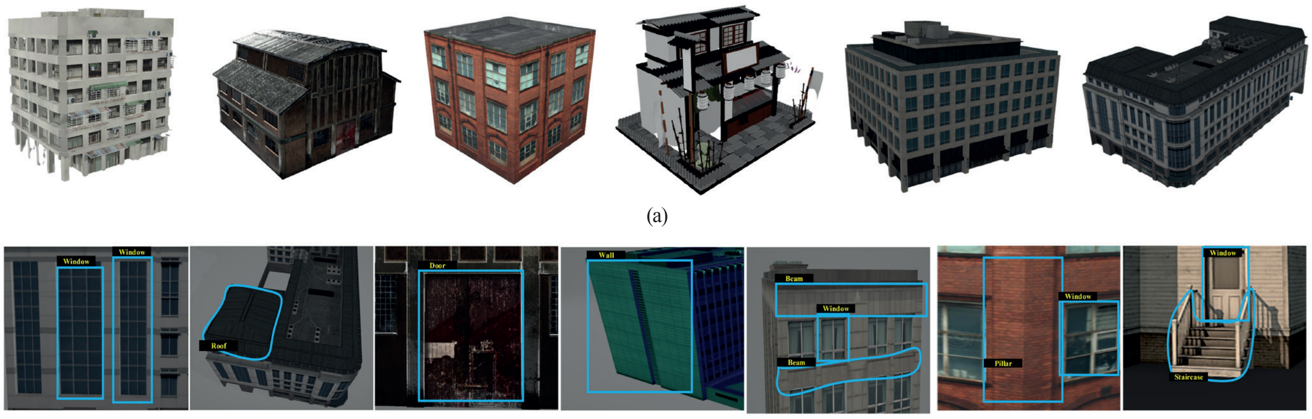
4.5 Network configuration

Inspired by the variant tiny DCNN models, we construct a lightweight DSOD herein CG-DSOD. Our approach in adducing a computational-efficient detector is to reduce the parameters of DSOD yet have an efficient backbone for feature extraction. We conjecture that detecting the edges of objects is paramount to the overall detection of the object. As such we fuse the color Canny edge detector given in Section 4.2 as a convolutional layer in the stem block of CG-DSOD. To further reduce compute resources, we introduce Gabor filters generated in Section 4.3 as a replacement and for feature extraction in the stem block. In principle, the computational complexity of a convolutional filter kernel is $(N \times N)^2$ and that of the Gabor filter kernel is $N \times N$, which gives it a computational urge over the convolutional filter. With these design principles in place, we reduce the parameters of the original DSOD detector and describe the variant CG-DSOD architecture. Similar to DSOD, CG-DSOD is composed of a backbone and a front-end sub-network, respectively. The stem block in CG-DSOD starts with a 3×3 color Canny edge convolution with a stride of 1 which is meant to enhance the edges of objects. This is followed by two Gabor convolutional layers for feature extraction at a low computational rate. Afterward, a convolutional layer follows, then a pooling layer before dense blocks and transition layers with reduced parameters.

Lastly comes the front-end which comprises 6 scales as in the original DSOD but with reduced parameter configuration. CG-DSOD configuration parameters are tabulated in Table 2 followed by a graphical illustration of the architecture CG-DSOD in Fig. 4.

5 Experiments

The presented detector CG-DSOD is extensively evaluated in two categories, offline and online mode under diverse evaluation metrics. The offline mode entails the use of allocated testing data and the online mode uses a low-cost MAV, a DJI Tello drone in the real world for a detection task. Accuracy and loss, mean Average Precision (mAP), and computational consumption are used as evaluation metrics in the offline mode. A confidence score and frame processing rate is used as a metric in the online mode under two



(b)

Fig. 3 Randomly selected (a) 3D building models and (b) cropped annotated building components

Table 2 CG-DSOD configuration, "cceconv" denotes color canny edge convolution, and "gconv" denotes Gabor convolution

Block	Layer	Output Size	CG-DSOD
-	Input	3 × 300 × 300	-
Stem Block	Color Canny Edge Conv.	64 × 300 × 300	3 × 3 cceconv, stride 1
	Gabor Conv.	64 × 150 × 150	3 × 3 gconv, stride 2
	Gabor Conv.	64 × 150 × 150	3 × 3 gconv, stride 1
	Convolution	128 × 150 × 150	1 × 1 conv, stride 1
	Pooling	128 × 75 × 75	2 × 2 max pool, stride 2
Dense Blocks with transition layers	Dense 0	256 × 75 × 75	$\begin{bmatrix}1\times1\ \ conv\\ 3\times3\ \ conv\end{bmatrix}\times4$
	Transition 0	128 × 38 × 38	1 × 1 conv 2 × 2 max pool, stride 2
	Dense 1	416 × 38 × 38	$\begin{bmatrix}1\times1\ \ conv\\ 3\times3\ \ conv\end{bmatrix}\times6$
	Transition 1	128 × 19 × 19	1 × 1 conv 2 × 2 max pool, stride 2
	Dense 2	512 × 19 × 19	$\begin{bmatrix}1\times1\ \ conv\\ 3\times3\ \ conv\end{bmatrix}\times6$
	Transition 2	256 × 19 × 19	1 × 1 conv
	Dense 3	736 × 19 × 19	$\begin{bmatrix}1\times1\ \ conv\\ 3\times3\ \ conv\end{bmatrix}\times6$
	Transition 3	64 × 19 × 19	1 × 1 conv
Prediction block	CG-DSOD Prediction	-	Plain/Dense

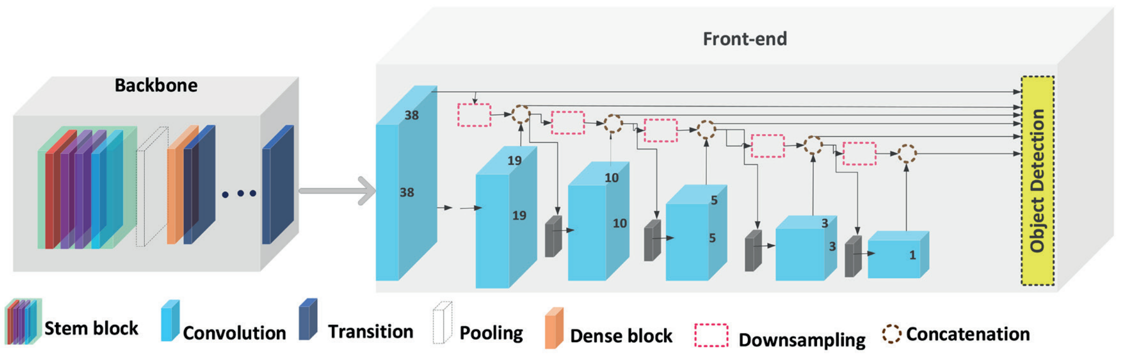


Fig. 4 The architecture of CG-DSOD detector

environmental conditions. We also demonstrate the computational efficiency of the presented approach in an ablation study with a focus on computational complexity and early convergence. Faster R-CNN [36] and YOLOv3 [37] are selected as comparators to CG-DSOD.

5.1 Training of CG-DSOD

CG-DSOD is trained from scratch using a higher ratio of the building component data which is augmented to attain a total of 6,675 images and tested with 20% of the unseen data. Caffe framework is used as the machine learning platform and an NVIDIA Geforce RTX 2070 with Max-Q Design serves as the computing resource. We follow the training technique used to train DSOD, such as the use of SGD Solver, smooth Loss L1 to penalize deviation of estimated bounding box by CG-DSOD from the ground truth, and cross-entropy loss for classification. We set the learning rate at 10^{-3}, using a batch size of 32 and train CG-DSOD for 100 epochs.

5.2 Offline evaluation

5.2.1 Detection accuracy and loss

Both comparators, Faster R-CNN and YOLOv3 go through a similar fine-tuning configuration and both are initialized with ImageNet weights before retraining for feature specifics. Faster R-CNN is trained using ResNet-50 as the backbone and YOLOv3 is trained using DarkNet 19 as the backbone network.

CG-DSOD records a training accuracy of 98.44% accompanied by a loss of 0.0506 and a testing accuracy of 98.01% with a loss of 0.0540 after the 100th epoch. From Fig. 5(a), CG-DSOD shows a steady training and testing pattern on the building component dataset. There are nearly no sudden drifts of both training and loss curves which connotes CG-DSOD attains strong robustness. An observation from Fig. 5(a) shows incorporating Gabor filters and color Canny edge convolution aids the variant model, CG-DSOD to learn faster hence early convergence is adduced. A progressive accuracy of 90% is recorded for both training and testing somewhere into the 10th epoch during training for CG-DSOD (refer to Fig. 5(a)) as opposed to YOLOv3 which records 90% plus of accuracy somewhere beyond the 40th epoch (refer to Fig. 5(b)) and Faster R-CNN which is after the 50th epoch (refer to Fig. 5(c)). It must be noted that despite CG-DSOD progressive accuracy rise at an earlier epoch, it experiences downward accuracy spikes at some epochs yet it rises and is more stable compared to both comparators. Clearly from Fig. 5(a), CG-DSOD has not fitted to the building component data. There is almost nearly no overfitting which indicates good generalization of CG-DSOD. Besides, the rapid depreciation in the loss curves as seen in the subplot of Fig. 5(a) around the 10th

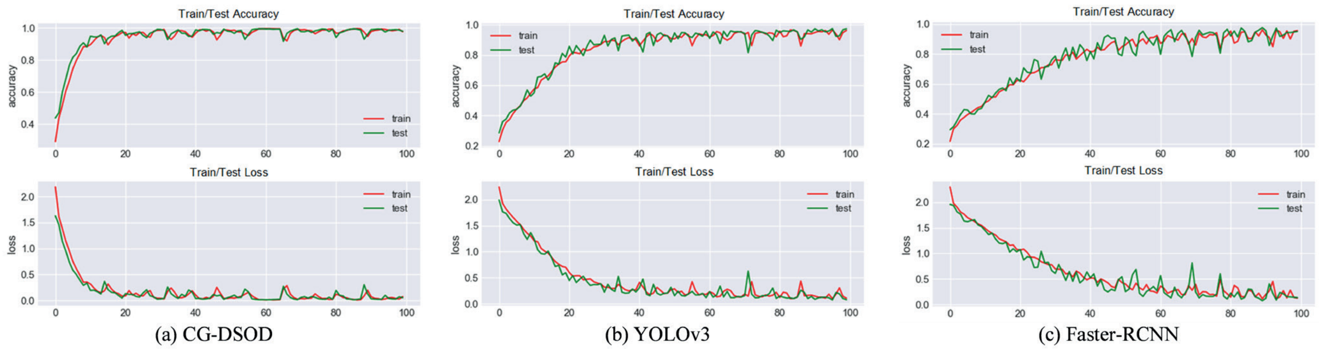


Fig. 5 Graphical comparison of training and testing accuracies and losses for (a) CG-DSOD, (b)YOLOv3, and (c) Faster R-CNN

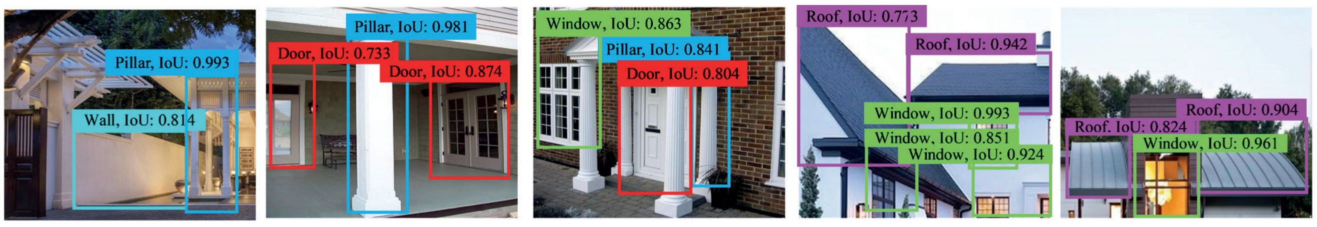
epoch reflects possibly the absence of further relevant features for CG-DSOD to utilize to improve the accuracies. As such, the configuration of CG-DSOD is efficient and training for further epochs may lead to overfitting.

YOLOv3 which is the first runner-up in detecting building components in this study records training and testing accuracies of 97.56% and 97.94% with losses of 0.1072 and 0.0792, respectively. Faster R-CNN records an accuracy of 96.29% and a loss of 0.1097 during training and at testing records 95.02% as accuracy and 0.1371 as a loss. Both comparators attain good generalization as it can be seen from Fig. 5(b) and 6(c), little overfitting is attained by both comparators as compared to CG-DSOD. Although all three detection models (CG-DSOD, YOLOv3, and Faster R-CNN) records training and testing accuracies above 95%, YOLOv3 and Faster R-CNN lag behind CG-DSOD. CG-DSOD surpasses YOLOv3 and Faster R-CNN concerning accuracy, loss, and early convergence. Table 3 is a tabulation of training and testing accuracies and losses of the detectors followed by visualization of the performance of each model in Fig. 5.

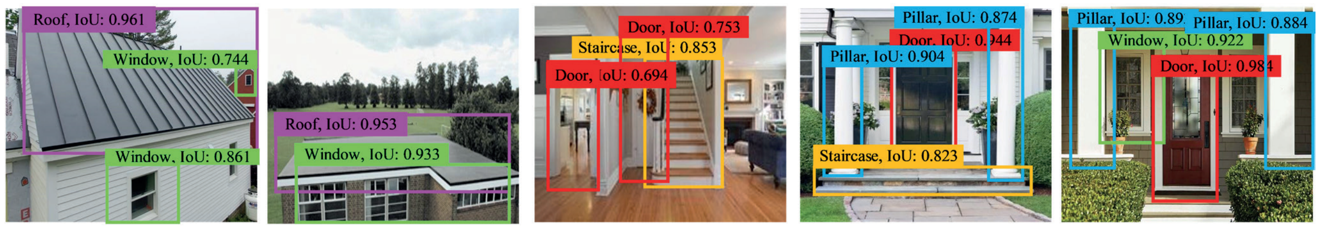
Using Intersection over Union (IoU) at a threshold of 0.6 and Per-Class Regression (PCR) which designates separate predicted bounding boxes for the classes, randomly selected correct and incorrect detection of building components by CG-DSOD is presented in Fig. 6 excluding that of the comparators for reasons of brevity. Besides, it must be noted the accuracies adduced by each detection model do reflect on the detection of the object of interest in the test data. IoU here is the intersection of bounding box estimations by the detectors (CG-DSOD, YOLOv3, and Faster R-CNN) against the ground truth, more details are given in Section 5.2.2. Interpretation of detection results in Fig. 6 is read from left to right in corresponding rows a, b, and c. Correct detected results are given in rows a and b with the incorrect detections in row c. Incorrect results in Fig. 6 fall within the False Positive (FP) and False Negative (FN) categories. The former, FP denotes correct classification and localization but IoU is less than the given threshold 0.6, and the latter, FN denotes an instance such as images 1, 3, and 4 in row c where wrong classifications are assigned by CG-DSOD.

Table 3 Training and testing accuracies and losses for the three detection models used in this study

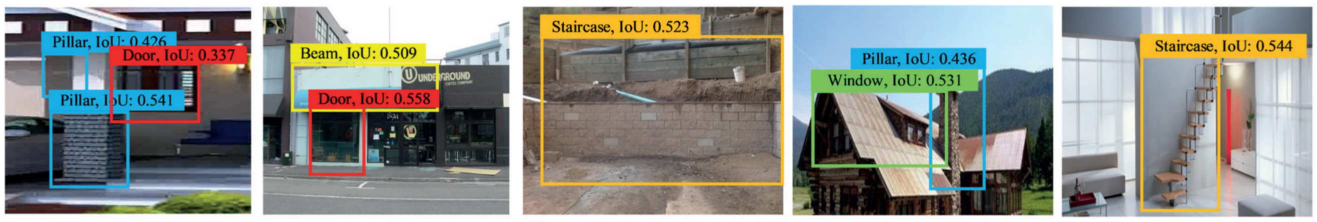
Detection model	Training accuracy	Training loss	Testing accuracy	Testing loss
CG-DSOD	**98.44%**	**0.0506**	**98.01%**	**0.05040**
YOLOv3	97.56%	0.1072	97.94%	0.0792
Faster R-CNN	96.29%	0.1097	95.02%	0.1371



(a)



(b)



(c)

Fig. 6 Random selection of correct detections by CG-DSOD in rows (a) and (b) and incorrect detections in row (c) based on IoU threshold at 0.6

5.2.2 Mean average precision

We employ the evaluation approach used in the MS COCO 2014 challenge to measure the performance of each detection model (CG-DSOD, YOLOv3, and Faster R-CNN). As such, we compute the Average Precision (AP) of the 7 classes (refer to Table 1) over specified IoU thresholds and average the adduced APs. Afterward, the mean of the averaged APs at specified thresholds are taken, hence the mean Average Precision (mAP) is attained. The threshold used in the benchmark is AP @ IoU = {0.5,0.55,…,0.95}, i.e., an AP at specified IoU. Here we interpret the threshold in two folds, thus, (1) as confidence threshold for classifying the object, and (2) as IoU to measure the predicted location of the object from the ground-truth bounding box (polyline region, 2D box, or circular region). Briefly, in the context of AP, the precision-recall curve (precision measures false positive rate and recall measures false-negative rate) is used to compute the AP of each of the object instances (classes) at specified thresholds. Afterward, averaged to adduce the AP for a detection model (CG-DSOD, YOLOv3, and Faster R-CNN). Using an 11 point interpolation, AP is expressed as:

$$AP = \frac{1}{11} \sum_{Recall_i} Precision\left(Recall_i\right) \tag{15}$$

where $Recall_i = [0, 0.1, 0.2, \ldots, 1.0]$.

In measuring the predicted location of an object based on the specified thresholds, the IoU is given as:

$$IoU = \frac{area\ of\ overlap}{area\ of\ union}. \tag{16}$$

Based on the IoU and precision-recall curve which leads to the computation of APs, we report the performance of the detection models used in this study.

Table 4 is a statistical representation of APs adduced for each detection model used in the study under the variant thresholds. Vividly from Table 4, it can be seen as the threshold increases there is a drop in the AP connoting the detection models are more stringent about the true positives. The steepness of the precision-recall curve as seen in Fig. 7 which computes the APs under variant thresholds varies across the detection models (CG-DSOD, YOLOv3, and Faster R-CNN). Fig. 7 gives a graphical insight into the performance of each detection model under increasing thresholds. Using both Table 4 a numerical view, and Fig. 7 a graphical view, comparatively CG-DSOD shows a steadier drop in the precision-recall curve as opposed to the comparators. As the threshold increases from 0.8 to 0.95, the volatility of CG-DSOD increases yet is advantageous over the two comparators which attains much higher volatilities at higher thresholds. To this end, we are of the view a reflection of the training and testing accuracies and losses can be seen in the precision-recall curves which leads to APs and finally a mAP for each detection model. Percentage-wise, there is nearly a 2% apart difference adduced among the three detection models under a mean training and testing accuracies. A similar inference is deduced in the mAP which is represented in Fig. 8 graphically, and in the last column of Table 4. As such, this analytical insight buttresses our initial view of the linkage of a reflection of training and testing accuracies in the mAP for the detection models (CG-DSOD, YOLOv3,

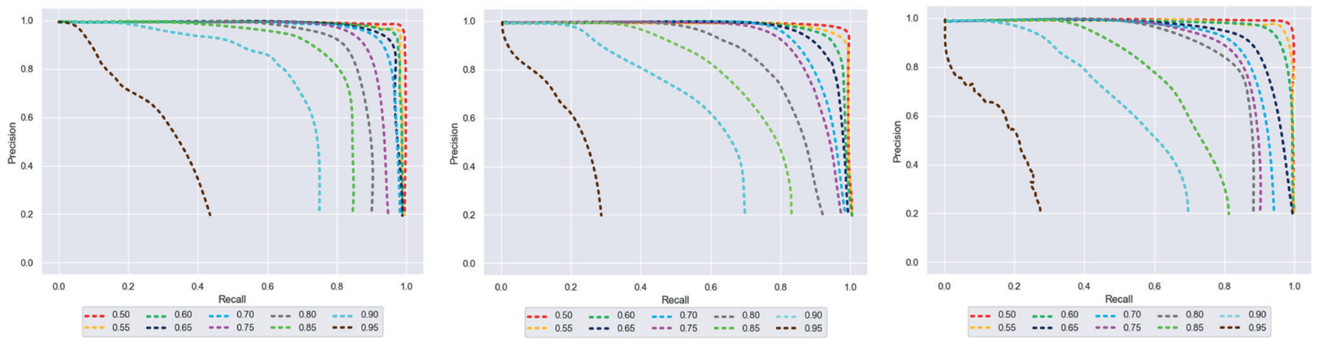


Fig. 7 The precision-recall curve adduced for the detection models under specified thresholds (a) CG-DSOD, (b) YOLOv3, (c) Faster R-CNN

Table 4 Numerical representation of detection models under specified IoUs

Model	AP .50	AP .55	AP .60	AP .65	AP .70	AP .75	AP .80	AP .85	AP .90	AP .95	mAP
CG-DSOD	**0.9909**	**0.9805**	**0.9793**	**0.9462**	**0.9379**	**0.9016**	**0.8663**	**0.7934**	**0.6840**	**0.3367**	**0.8417**
YOLOv3	0.9899	0.9771	0.9757	0.9397	0.9233	0.8912	0.8232	0.7325	0.6001	0.2672	0.8120
Faster R-CNN	0.9903	0.9871	0.9768	0.9039	0.8841	0.8593	0.7984	0.7031	0.5701	0.2395	0.7913

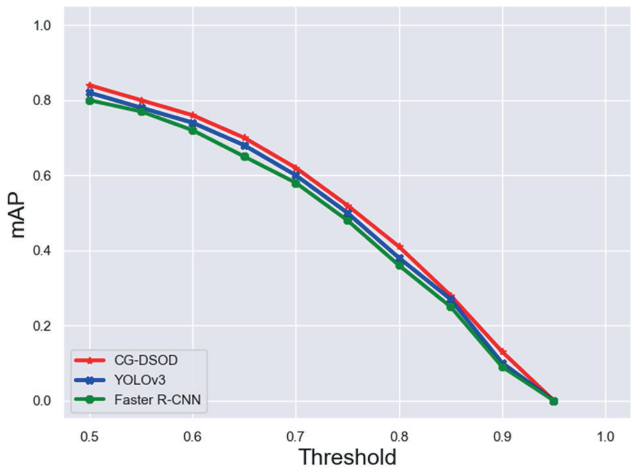


Fig. 8 mAP results for CG-DOSD, YOLOv3, and Faster R-CNN detectors under diverse thresholds, i.e., $0.5 \leq \text{IoU} \leq 0.95$

and Faster R-CNN). All in all, CG-DSOD records a mAP of 84.17%, YOLOv3 records 81.20%, and Faster R-CNN records 79.13% in the building component detection task.

5.2.3 Computational cost

In this section, we measure the computational resources utilized by each detection model during training to infer the computational consumption. We employ a modified version of the computational cost procedure in [38] by introducing a parameter *windows_bias* to account for resources being utilized by the background process and windows processes during training. The computational parameters entail:

- Runtime: time used in training of each model separately.
- GPU Load: the current supplied to the GPU cores during training given in the form of voltage.
- GPU Mem.: the total amount of GPU memory consumed.
- DataTrans: this is a measure of the amount of data transferred between the cores of the GPU during training.
- Windows_bias: the new parameter introduced to account for GPU and memory usage for windows processes (Client Server Runtime Process, Desktop Window Manager, Registry, etc.) and background processes (Networkcap, IntelAudioService, etc.) during training.

Numerical measures for the aforementioned computational parameters are via the NVIDIA profiling tools (NSIGHT tools, NVVP, and NVML). The computational cost during training can then be obtained as:

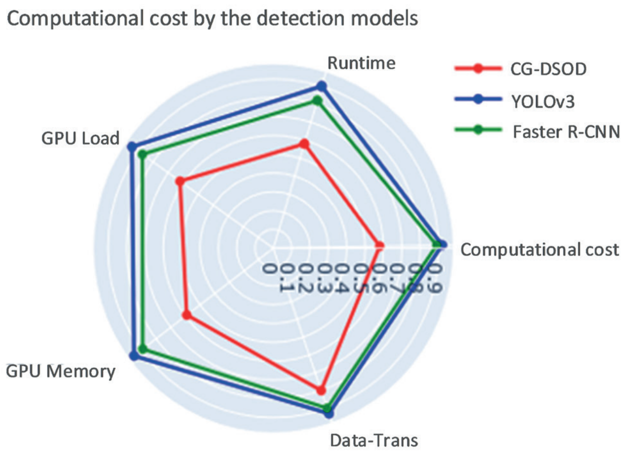


Fig. 9 Computational cost by the CG-DSOD, YOLOv3, and Faster R-CNN

$$modelE = runtime \times gpuload \times gpumem \times datatrans \, , \quad (17)$$

$$Computational\ cost = \frac{modelE - windows\_bias}{number\ of\ epoches} \, . \quad (18)$$

Fig. 9 shows the normalized plotting of the computational parameters which also serve as standalone evaluation metrics and the computational cost by each detection model (CG-DSOD, YOLOv3, and Faster R-CNN). From Fig. 9, it can be seen that CG-DSOD can conserve variation of a little over 20% on the normalized values for Runtime, GPU load, GPU mem., and DataTrans as opposed to both YOLOv3 and Faster R-CNN. As such in the overall computational cost derived using Eq. (10), CG-DSOD conserves nearly 30% of the compute resources required during training as opposed to the two comparators. The computational reduction is much attributed to the use of Gabor filters in the lower blocks of CG-DSOD architecture. It must be included that, color Canny edge convolutions introduced also contributes to the reduction of computation required in the training of CG-DSOD. Besides in [39], the authors designed circuits in Verilog mapped onto a 45 nm field-programmable gate array board to measure the computational consumption of a DCNN model. It occurred the first two convolutional blocks consume nearly 47% of the computational resource required to train a DCNN. The architectures of YOLOv3 and Faster R-CNN reflect on the computational cost by the two detection models, respectively. With YOLOv3 having 106 convolutional layers and Faster R-CNN having 50 convolutional layers as their base architectures, the high computational requirements adduced during training indicate the depth of a DCNN architecture contributes to the high computations.

5.3 Online evaluation

In this section, we give an account of an online evaluation of CG-DSOD together with the comparators. A low-cost MAV, a DJI Tello drone together with a ground station control equipped with an NVIDIA Geforce RTX 2070 with Max-Q Design and 16GB of memory is used for the online experiment. Due to range limitations, a Xiaomi wifi 2 repeater is used to extend the range. An insight into the real-world experimentation setup is visualized in Fig. 10(a). The DJI Tello drone operates on an Intel 14 core processor with a default frame rate of 30 frames per second (fps) and has an extremely low compute power which prevents direct DCNN model implementation on the DJI Tello drone. To control the DJI Tello drone, the *djitellopy* module is used to implement a navigation controller feasible with a computer keyboard. As such, keys *w*, *s*, *a*, *d* denotes changes in altitude and orientation and keys *up*, *down*, *left*, *right* denotes changes in the aspect of pitch and roll, respectively. Readers are to refer to Fig. 10(b) for graphical understanding. Through the setup described, a human pilot flies the DJI Tello within the hallways and outskirts of a dormitory building within the localities of the authors of this study whiles the detection models process the video feed from the DJI Tello in real-time. Since there are no ground truth labels in the real-world environment, we evaluate the detectors with the confidence scores and the frame processing rate. The report for the two environmental scenarios is based on similar detections within similar frames by the three detectors for comparative reasons. First, an account is given for environment 1 which has no objects obstructing the percept of the DJI Tello drone, Fig. 11(a). Here, random sample frames are selected, then we compute average confidence *avgC* for each frame $f_1,\ldots,f_n$ afterward, we take the mean confidence *mC* of all the average confidence *avgC*. CG-DSOD records a higher mean confidence score using a threshold of 0.6 compared to YOLOv3 and Faster R-CNN. Moreover, CG-DSOD has higher accuracy and mAP as opposed to both comparators which reflect in the estimated confidence scores.

In the second scenario, environment 2, a human poses as an obstruction to infer the performance of the detectors. The human pilot flies the DJI Tello through the occluded environment for the detection of objects of interest. We observed that in the occluded environment, the confidence scores estimated by all three detectors decrease for similar objects detected in environment 1 which has no obstacles. Regardless of the decrease in confidence scores, CG-DSOD has much better confidence than both comparators as seen in Fig. 11(b). We are of the view, the presence of occlusion contributes to the decrease in the confidence scores. For brevity, we report a handful of correct detection frames for both environments in Fig. 11 excluding the acknowledged false detections. For detection speed, YOLOv3 achieves the best result at a processing rate of 65.6fps while our detector CG-DSOD attains a promising frame processing rate of 42.7fps and Faster R-CNN records 9.4fps.

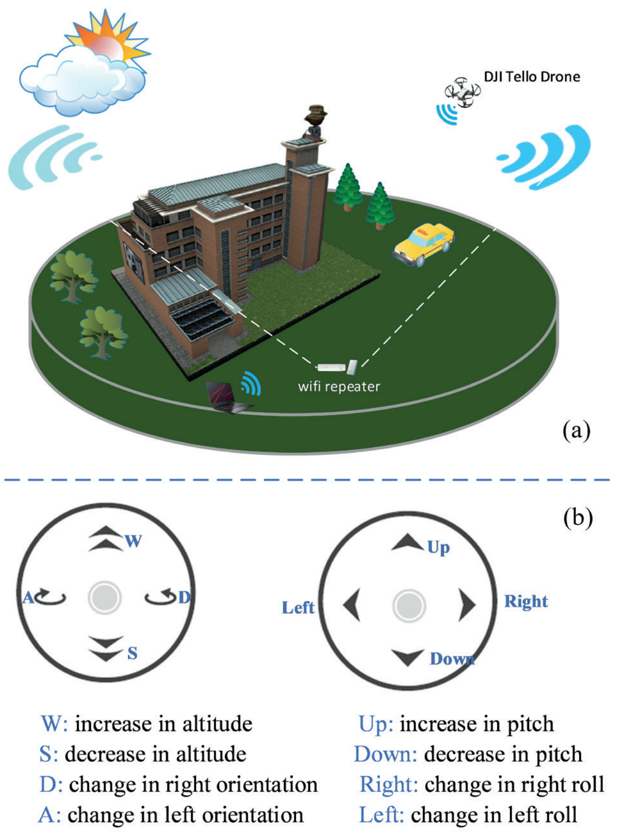


Fig. 10 (a) schematic of real-time experiment setup and (b) illustration of DJI Tello controller

5.4 Ablation study

We infer the contribution of Gabor filters and color Canny edge convolution in CG-DSOD detector by removing the generic Gabor filters and color Canny edge convolution hence leaving us with the DSOD detector with reduced parameters as in Table 2, herein DSOD-small. The focus is on the computational consumption, how fast the detector (DSOD-small) learns as well as the accuracies recorded. CG-DSOD has total training parameters of 0.83M as compared to the DSOD-small which has 0.96M. After training for 100 iterations using the same dataset we observe DSOD-small fails to attain early convergence as seen from Fig. 12(b). DSOD-small attains training and testing accuracies of 97.58% and 97.13 % with losses 0.9641 and

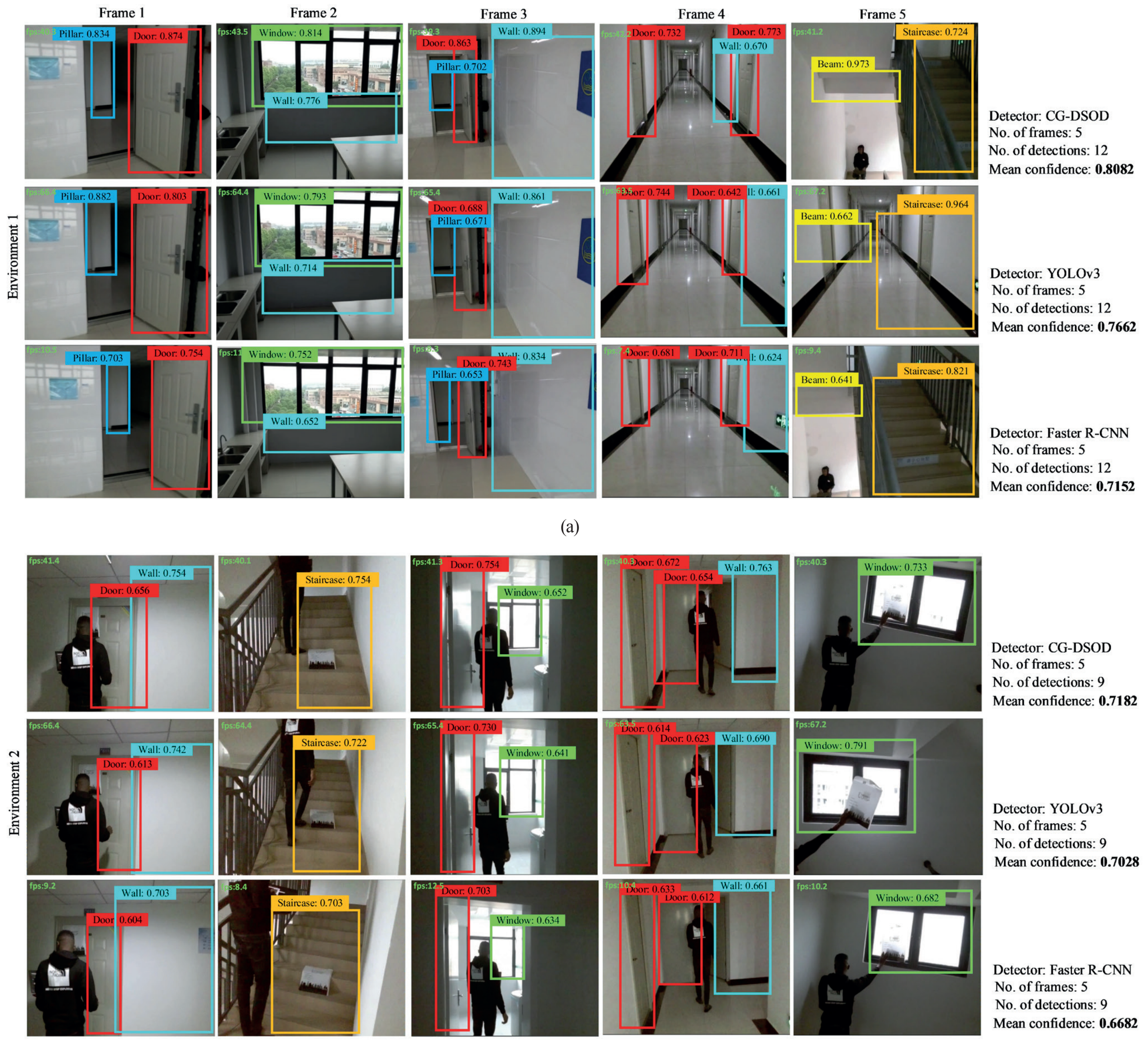
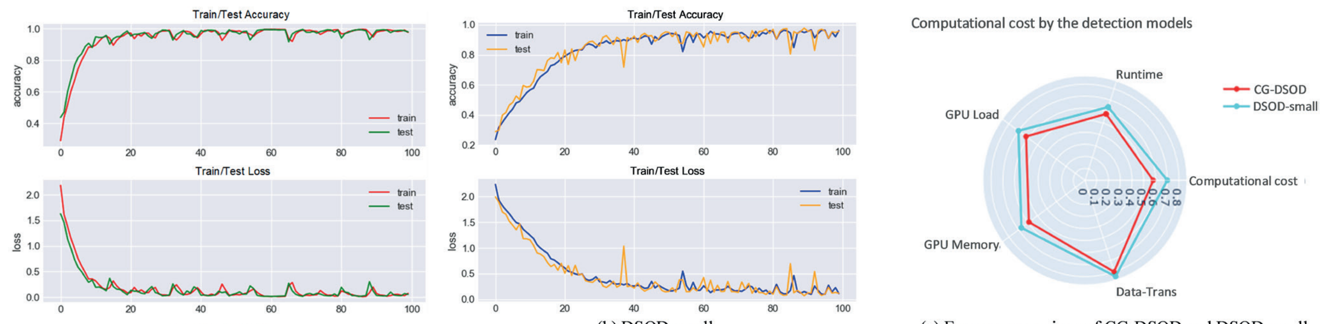


Fig. 11 (a) represents detection results in environment 1 and (b) denotes detection results for environment 2 where a human poses as a weak obstruction



(a) CG-DSOD (b) DSOD-small (c) Energy comparison of CG-DSOD and DSOD-small

Fig. 12 Ablation study aiming at convergence, and accuracies (a and b) and computational cost (c) of CG-DSOD and DSOD-small

0.01482, respectively which trails the accuracies and losses of CG-DSOD, refer to Table 3. From Fig. 12(b), it can be seen that DSOD-small starts to converge somewhere after the 40th epoch compared to CG-DSOD which is somewhere after the 10th epoch, refer to Fig. 12(a). The computational cost computation follows the same approach in Section 5.2.3. After training DSOD-small for 100 epochs, we observed DSOD-small uses about 12% more compute resources as compared to CG-DSOD. The computational cost depicted in Fig. 12(c) correlates CG-DSOD and DSOD-small, respectively.

6 Conclusions

In this study, we have presented a lightweight detection model dubbed CG-DSOD. Our detection model (CG-DSOD) is a variant of the DSOD detector coupled with Gabor filters and color Canny edge detectors in the lower blocks with reduced parameters for computing reduction and faster learning. We train CG-DSOD on building structural components and also introduce domain randomization, a much better data augmentation technique. We investigated the capabilities of CG-DSOD over three main axes: accuracy, mAP, and computational cost. We conduct two extensive experiments, offline and online (real-world) with a resource-constrained edge node for verification of CG-DSOD. Basing our conclusion on the experiments, CG-DSOD outperforms the two comparators in the study. CG-DSOD attains a nearly 2% advantage in accuracy over its comparators, somewhere around 3% mAP higher, and nearly a 30% reduction in computational resources as opposed to YOLOv3 and Faster R-CNN. An online verification using a DJI Tello drone shows the detection speed of CG-DSOD is enough for real-time detection tasks. CG-DSOD shows promising detection results in occluded environments which is a sign of CG-DSOD having the urge to overcome complexity within environments and making use of partial percept of objects of interest. Last, as future work, using Gabor filters as a convolutional modulator and using a DCNN to attain autonomous navigation of a MAV is a step towards autonomous SHM.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grants 61571099, 61501098.

References

[1] Kong, Q., Allen, R. M., Kohler, M. D., Heaton, T. H., Bunn, J. "Structural health monitoring of buildings using smartphone sensors", Seismological Research Letters, 89(2A), pp. 594–602, 2018. https://doi.org/10.1785/0220170111

[2] Vardanega, P. J., Webb, G. T., Fidler, P. R. A., Middleton, C. R. "Assessing the potential value of bridge monitoring systems", Proceedings of the Institution of Civil Engineers: Bridge Engineering, 169(2), pp. 126–138, 2016. https://doi.org/10.1680/jbren.15.00016

[3] Djenouri, D., Laidi, R., Djenouri, Y., Balasingham, I. "Machine learning for smart building applications: Review and taxonomy", ACM Computing Surveys, 52(2), Article number: 24, 2019. https://doi.org/10.1145/3311950

[4] Sony, S., Dunphy, K., Sadhu, A., Capretz, M. "A systematic review of convolutional neural network-based structural condition assessment techniques", Engineering Structures, 226, Article number: 111347, 2021. https://doi.org/10.1016/j.engstruct.2020.111347

[5] Ramezani Dooraki, A., Lee, D.-J. "An innovative bio-inspired flight controller for quad-rotor drones: Quad-rotor drone learning to fly using reinforcement learning", Robotics and Autonomous Systems, 135, Article number: 103671, 2021. https://doi.org/10.1016/j.robot.2020.103671

[6] Shen, Z., Liu, Z., Li, J., Jiang, Y.-G., Chen, Y., Xue, X. "DSOD: Learning Deeply Supervised Object Detectors from Scratch", In: 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 1937–1945. https://doi.org/10.1109/ICCV.2017.212

[7] Xin, G., Ke, C, Xiaoguang, H. "An improved Canny edge detection algorithm for color image", In: IEEE 10th International Conference on Industrial Informatics, Beijing, China, 2012, pp. 113–117. https://doi.org/10.1109/INDIN.2012.6301061

[8] Gabor, D. "Theory of communication. Part 1: The analysis of information", Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering, 93(26), pp. 429–441, 1946. https://doi.org/10.1049/ji-3-2.1946.0074

[9] Cha, Y.-J., Choi, W., Suh, G., Mahmoudkhani, S., Büyüköztürk, O. "Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types", Computer-Aided Civil and Infrastructure Engineering, 33(9), pp. 731–747, 2018. https://doi.org/10.1111/mice.12334

[10] Perez, H., Tah, J. H. M., Mosavi, A. "Deep Learning for Detecting Building Defects Using Convolutional Neural Networks", Sensors, 19(16), Article number: 3556, 2019. https://doi.org/10.20944/preprints201908.0068.v1

[11] Cha, Y.-J., Choi, W., Büyüköztürk, O. "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks", Computer-Aided Civil and Infrastructure Engineering, 32(5), pp. 361–378, 2017. https://doi.org/10.1111/mice.12263

[12] Atha, D. J., Jahanshahi, M. R. "Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection", Structural Health Monitoring, 17(5), pp. 1110–1128, 2018. https://doi.org/10.1177/1475921717737051

[13] Khodabandehlou, H., Pekcan, G., Fadali, M. S. "Vibration-based structural condition assessment using convolution neural networks", Structural Control and Health Monitoring, 26(2), Article ID e2308, 2019.
https://doi.org/10.1002/stc.2308

[14] Abdeljaber, O., Avci, O., Kiranyaz, S., Gabbouj, M., Inman, D. J. "Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks", Journal of Sound and Vibration, 388, pp. 154–170, 2017.
https://doi.org/10.1016/j.jsv.2016.10.043

[15] Avci, O., Abdeljaber, O., Kiranyaz, S., Inman, D. "Structural damage detection in real time: Implementation of 1D convolutional neural networks for SHM applications", Structural Health Monitoring & Damage Detection, 7, pp. 49–54, 2017.
https://doi.org/10.1007/978-3-319-54109-9_6

[16] Abdeljaber, O., Avci, O., Kiranyaz, M. S., Boashash, B., Sodano, H., Inman, D. J. "1-D CNNs for structural damage detection: Verification on a structural health monitoring benchmark data", Neurocomputing, 275, pp. 1308–1317, 2018.
https://doi.org/10.1016/j.neucom.2017.09.069

[17] Bao, Y., Tang, Z., Li, H., Zhang, Y. "Computer vision and deep learning–based data anomaly detection method for structural health monitoring", Structural Health Monitoring, 18(2), pp. 401–421, 2019.
https://doi.org/10.1177/1475921718757405

[18] Xu, Y., Bao, Y., Chen, J., Zuo, W., Li, H. "Surface fatigue crack identification in steel box girder of bridges by a deep fusion convolutional neural network based on consumer-grade camera images", Structural Health Monitoring, 18(3), pp. 653–674, 2019.
https://doi.org/10.1177/1475921718764873

[19] Lee, S., Ha, J., Zokhirova, M., Moon, H., Lee, J. "Background Information of Deep Learning for Structural Engineering", Archives of Computational Methods in Engineering, 25(1), pp. 121–129, 2018.
https://doi.org/10.1007/s11831-017-9237-0

[20] Kang, D., Cha, Y.-J. "Autonomous UAVs for Structural Health Monitoring Using Deep Learning and an Ultrasonic Beacon System with Geo-Tagging", Computer-Aided Civil and Infrastructure Engineering, 33(10), pp. 885–902, 2018.
https://doi.org/10.1111/mice.12375

[21] Jiang, S., Zhang, J. "Real-time crack assessment using deep neural networks with wall-climbing unmanned aerial system", Computer-Aided Civil and Infrastructure Engineering, 35(6), pp. 549–564, 2020.
https://doi.org/10.1111/mice.12519

[22] Cheng, C., Shang, Z., Shen, Z. "Automatic delamination segmentation for bridge deck based on encoder-decoder deep learning through UAV-based thermography", NDT and E International, 116, Article number: 102341, 2020.
https://doi.org/10.1016/j.ndteint.2020.102341

[23] Garilli, E., Bruno, N., Autelitano, F., Roncella, R., Giuliani, F. "Automatic detection of stone pavement's pattern based on UAV photogrammetry", Automation in Construction, 122, Article number: 103477, 2021.
https://doi.org/10.1016/j.autcon.2020.103477

[24] Lee, J.-H., Yoon, S.-S., Jung, H.-J., Kim, I.-H. "Diagnosis of crack damage on structures based on image processing techniques and R-CNN using unmanned aerial vehicle (UAV)", Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems, Article number: 1059811, 2018.
https://doi.org/10.1117/12.2296691

[25] Kim, H., Kim, K., Kim, H. "Data-driven scene parsing method for recognizing construction site objects in the whole image", Automation in Construction, 71(Part 2), pp. 271–282, 2016.
https://doi.org/10.1016/j.autcon.2016.08.018

[26] Visual Geometry Group "VGG Image Annotator (2.0.11)", [computer program] Available at: https://www.robots.ox.ac.uk/~vgg/software/via/via.html [Accessed: 5 December 2020]

[27] Agyemang, I. O. "Civil Structures/Infrastructures construction components data", [Online] Available at: https://www.scidb.cn/en/s/quEJFr [Accessed: 25 December 2020]

[28] Konishi, Y., Hanzawa, Y., Kawade, M., Hashimoto, M. "Fast 6D Pose Estimation Using Hierarchical Pose Trees", In: Computer Vision – ECCV 2016, 14th European Conference, Amsterdam, The Netherlands, 2016, pp. 398–413.
https://doi.org/10.1007/978-3-319-46448-0_24

[29] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q. "Densely connected convolutional networks", In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 2261–2269.
https://doi.org/10.1109/CVPR.2017.243

[30] Canny, J. "A computational approach to edge detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6), pp. 679–698, 1986.
https://doi.org/10.1109/TPAMI.1986.4767851

[31] Meshgini, S., Aghagolzadeh, A., Seyedarabi, H. "Face Recognition Using Gabor Filter Bank, Kernel Principle Component Analysis and Support Vector Machine", International Journal of Computer Theory and Engineering, 4(5), pp. 767–771, 2012.
https://doi.org/10.7763/ijcte.2012.v4.574

[32] Krizhevsky, A., Sutskever, I., Hinton, G. E. "ImageNet Classification with Deep Convolutional Neural Networks", Communications of the ACM, 60(6), pp. 84–90, 2012.
https://doi.org/10.1145/3065386

[33] Yuan, Y., Zhang, J., Wang, Q. "Deep Gabor convolution network for person re-identification", Neurocomputing, 378, pp. 387–398, 2020.
https://doi.org/10.1016/j.neucom.2019.10.083

[34] Chang, S.-I., Morgan, N. "Robust CNN - based Speech Recognition With Gabor Filter Kernels", In: Proceedings of Interspeech 2014, Singapore, 2014, pp. 905–909.
https://doi.org/10.21437/Interspeech.2014-226

[35] Blender Foundation "Blender 3D creation suite (3.0.1)", [computer program] Available at: https://www.blender.org/download/ [Accessed: 15 March 2021]

[36] Ren, S., He, K., Girshick, R., Sun, J. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), pp. 1137–1149, 2017.
https://doi.org/10.1109/TPAMI.2016.2577031

[37] Redmon, J., Farhadi, A. "YOLOv3: An Incremental Improvement", University of Washington, Washington, DC, USA, Tech report, 2018. [online] Available at: http://arxiv.org/abs/1804.02767

[38] Anwar, A., Raychowdhury, A. "Autonomous Navigation via Deep Reinforcement Learning for Resource Constraint Edge Nodes Using Transfer Learning", IEEE Access, 8, pp. 26549–26560, 2020. https://doi.org/10.1109/ACCESS.2020.2971172

[39] Sarwar, S. S., Panda, P., Roy, K. "Gabor filter assisted energy efficient fast learning Convolutional Neural Networks", presented at Proceedings of the International Symposium on Low Power Electronics and Design, Taipei, Taiwan, July, 24–26, 2017. https://doi.org/10.1109/ISLPED.2017.8009202