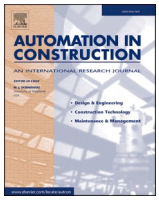
Automation in Construction 140 (2022) 104389

Contents lists available at ScienceDirect

Automation in Construction

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



Automatic image-based brick segmentation and crack detection of masonry walls using machine learning

Dimitrios Loverdos, Vasilis Sarhosis *

School of Civil Engineering, University of Leeds, LS2 9JT, Leeds, UK

ARTICLE INFO

Keywords:
Masonry
Image processing
Documentation
Watershed
Segmentation
Deep learning
CNN

ABSTRACT

This paper aims to improve automation in brick segmentation and crack detection of masonry walls through image-based techniques and machine learning. Initially, a large dataset of hand-labelled images of different in colour, texture, and size of brickwork masonry walls has been developed. Then, different deep learning networks (U-Net, DeepLabV3+, U-Net (SM), LinkNet (SM), and FPN (SM)) were utilised and their quality was assessed. Furthermore, the ability to generate geometric models of masonry structures and the evaluation of the geometric properties of detected cracks was also investigated. Additional metrics were also developed to compare the CNN output with other image-processing algorithms. From the analysis of results it was shown that the use of machine learning, for brick segmentation, provides better outcome than typical image-processing applications. This implementation of deep-learning for crack detection and localisation of bricks in masonry walls highlights the great potential of new technologies for documentation of masonry fabric.

1. Introduction

Masonry is one of the oldest building materials. It is composed of individual masonry units (bricks, blocks, ashlars, irregular stones, etc.) jointed together with or without mortar. Masonry structures represent the highest building stock worldwide. A large portion of masonry structures are “Listed Buildings” and form part of our “Cultural Heritage” [34]. According to UN Sustainable Development Goal 11, Target 11.4, there is a need to “repair and maintenance” rather than “demolish and rebuild” our structures. Currently, for the inspection of existing masonry structures, damage pathologies (i.e., cracking, spalling etc) are captured with traditional techniques (e.g., visual inspection and manual surveying methods [18]), which are labour intensive, subjective, and error prone [37,40]. Additionally, for the documentation of masonry structures, the size and location of masonry units and mortar is of particular interest for the engineers and architects, since such information can be used for the development of high-fidelity computational models for their structural analysis and assessment [17,26,30,31].

In the last ten years, advances in laser scanning and photogrammetry have started to drastically change the building industry since such techniques are able to capture rapidly and remotely digital objects and features in images and points' cloud format. Past research demonstrated that computer-vision and image-processing can be used to create detailed digital records of masonry structures [25] using feature detection [4,7,10,33] and segmentation algorithms [4,6,27]. Applications of Image-processing can be used to quantify deformations on masonry (i.e., when coupled with the installation of simple markers on the structure on key-locations to identify their position [5,42]). Furthermore, feature-extraction has the potential to generate the “as is” numerical model of masonry for detailed analysis [26,32,43,44]. Those applications of image-processing offer a low-cost and reliable alternative to more traditional methods. Although, feature detection and segmentation has already been applied to identify the shape and location of masonry units from images [8,15,36,39,46], their application proves challenging due to digital noise produced by the change in brightness, colour, and texture presented within the digital images.

An alternative approach for image segmentation of masonry units and mortar is with the use of ML (Machine-Learning), such as Deep-Learning (DL; subset of M.L.) [20,41]. Some typical examples of DL include:

- *Classic-Networks:* Multilayer architecture of fully-connected-layers of neurons, which are typically used in data classification and predictions.
- *Convolutional-Neural-Networks (CNN):* Typically used for image classification and segmentation.

* Corresponding author at: School of Civil Engineering, University of Leeds, LS2 9JT Leeds, UK.
E-mail address: v.sarhosis@leeds.ac.uk (V. Sarhosis).

https://doi.org/10.1016/j.autcon.2022.104389
Received 8 March 2022; Received in revised form 3 May 2022; Accepted 26 May 2022
Available online 2 June 2022

- *Fully-Connected-Networks (FCN):* Typically used for image-segmentation, often combined with a CNN backbone.

The most efficient image-segmentation architectures consider the use of FCN [12–14,28,29,38], since they allow any image resolution as input by replacing the final fully-connected-layers of a CNN with convolution layers [29]. CNN and FCN architectures require a large labelled or annotated dataset, trained in representative sample to provide adequate results. However, they have the potential to detect complex features by training the model to a large variety of different cases. Furthermore, for smaller datasets, DL approaches can use a technique called Transfer-Learning which involves pre-training a CNN model on a different and larger dataset with the purpose to learn to detect complex features. This has been shown to provide a reduction to the required computational effort and a boost to overall performance of the model for smaller datasets [21]. Transfer learning can be applied to any CNN and FCN (Fully Convolutional Networks) architecture. Multiple architectures have been used to demonstrate the ability of DL in the semantic segmentation of images.

There are different architectures that could be used for the segmentation of masonry units. U-Net is a FCN architecture that was developed initially for biomedical image-segmentation [38]. It is based on a contracting followed by an expansive path, which initially decreases and then increases the input-size. Due to its performance, U-Net is considered the benchmark in image-segmentation and has been used extensively especially in relatively small datasets. DeepLabV3+ is another state-of-the-art FCN architecture for general use semantic segmentation [13,14]. Additional features on DeepLabV3+ compared to previous iterations and simpler architectures aim to deliver a faster and more accurate network. FPN (Feature-Pyramid-Network) is an FCN network for object detection [28]. It is a feature extractor that follows a bottom-up followed by a top-down path with the addition of lateral connections between the two to merge feature maps of equal spatial size. LinkNet is a light FCN network developed for pixel-wise segmentation optimized for efficiency [12]. LinkNet is a lightweight and fast FCN architecture able to be used for real time applications (i.e., video streaming).

ML applications have already seen use in structural engineering due to their immerse potential to assist with visual inspection and monitoring applications [41]. In cases of damage detection, ML provides the means to identify, locate, and asses detected deterioration on structural elements. Valero et al. [47] extracted statistical data from a 3D point-cloud and used them to train a ML algorithm for the detection and classification of chromatic (i.e., discoloration) and geometric defects on ashlar masonry (using logistic regression with multi-class classification). However, most typical applications of defect detection include the use of DL due to its architecture, which allows the detection of complex features on unstructured data. [11], investigated the use of patch classification using CNN algorithm coupled with a classifier to identify small patches that contained damaged location of historical structures. Their work was continued by Ali [2], where Faster-R-CNN was used for the detection of bounding-boxes that contain locations of damaged bricks on masonry structures. The same year, Wang et al. [48] used a Faster-R-CNN model based on ResNet101 for the real time detection and classification of bounding-boxes that include defected areas on historic masonry buildings. A workflow that utilises mobile-phones for the direct capture and processing of image-data was also proposed by them. Brackenbury et al. [9] discussed the use of GoogleNet-Inception-V3 algorithm and the use of transfer learning for the classification and segmentation of mortar and defects in masonry components. Each defect (i.e., cracking, spalling, or vegetation) was classified separately. Kalfarisi et al. [24] used Mask-RCNN and FRCNN-FED to detect bounding boxes that contain cracks on structures and performed pixel-wise segmentation within the detected areas. Furthermore, they transferred the segmented locations to a 3D reality-mesh object, generated using photogrammetry. Recently, Dais et al. [16] tested different CNN algorithms for the detection of cracks on masonry images. Both patch-classification (with 95.3% accuracy) and semantic-segmentation (with 79.6% F1-score) on pre-trained networks were investigated.

Past research demonstrated that the use of CNN algorithms for the detection of masonry units and mortar is also necessary to provide a complete visualization of the detailed geometry of masonry structures. Ibrahim et al. [23] proposed the use of U-Net for the segmentation of mortar in masonry structures with different bonding pattern (i.e., including rubble). Additionally, they used watershed-transform for the segmentation of each brick unit. Ergün Hatir and İnce [19] proposed the use of Mask-R-CNN for the classification and segmentation of masonry units in historic stone masonry buildings. Each stone detected was classified to a different lithology based on their detected features (i.e., colour, texture, etc).

From the above, although some work has been done in the segmentation of cracks and mortar, there is still no research on coupling brick segmentation with crack detection in masonry structures. The aim of this research is to couple brick segmentation and defect detection techniques and automatically provide holistic and real time information for the documentation, visual inspection, and evaluation of existing masonry structures. In this research, brick segmentation and defect detection (i.e. cracks) were acquired using different state-of-the-art FCN architectures including U-Net, DeepLabV3+, U-Net (SM), LinkNet (SM), and FPN (SM). The work presented here provides algorithms necessary for the automatic documentation and structural inspection of masonry structures from digital images.

2. Development of the database for training and evaluation

Initially, a database was created that includes various images of brick masonry walls of regular pattern (ignoring rubble masonry). Some images were obtained from the internet while others were captured using different sources (i.e., DSLR camera, smartphones) of varied resolution. To improve generalisation, the dataset included masonry walls with cracks, with windows and doors, with varied in colour masonry units as well as with varied illumination and capture-angle. A sample of the raw database used for training and evaluation is shown in Fig. 1.

In total 107 images of masonry structures were fully annotated, including multi-class annotations of masonry blocks, openings, lintels, other/random objects, and background (Fig. 2). Each class was annotated to a different binarized image where black was the background and white was the annotated element. The software used for annotation was the "*SuperAnnotate V.1.1.0*". This specific software was selected since it allowed vector annotations, which permits a simplified annotation of each block, ignoring unnecessary details. It was found that simpler shapes could allow easier transferability to CAD environment, and this will also be represented to the output from the CNN model.

Each image resolution was normalised based on the resolution of the image-slice passed through the network. This was to allow each slice to contain several blocks, but not allow the average block-size to be larger than the image-slice. Doing so, the accuracy of the model has increased, since each image-slice had similar-sized blocks that improved the detection rate. The normalised resolution of each image was evaluated and compared with the normalised resolution of the image-slice (image-part passed through the network). Thus, the image was allowed a specific range of resolution. Although, using a specific resolution of block elements (i.e., pixels contained within a block; by capturing pictures with specific resolution, angle, and distance), would potentially increase the accuracy. However, that would reduce generalisation of the final model and complicate its use (i.e., would require the user to capture images from specific distance/angle). Eqs. (1) to (5) were used to adjust each image. If the image was within the limits of Eq. (3), it retained its resolution. However, if the image was outside the limits of Eq. (3), it was adjusted based on Eq. (5).

$$ImDim = \sqrt{x_{image} * y_{image}} \quad (1)$$

Fig. 1. Sample of the raw database used for training-evaluation.

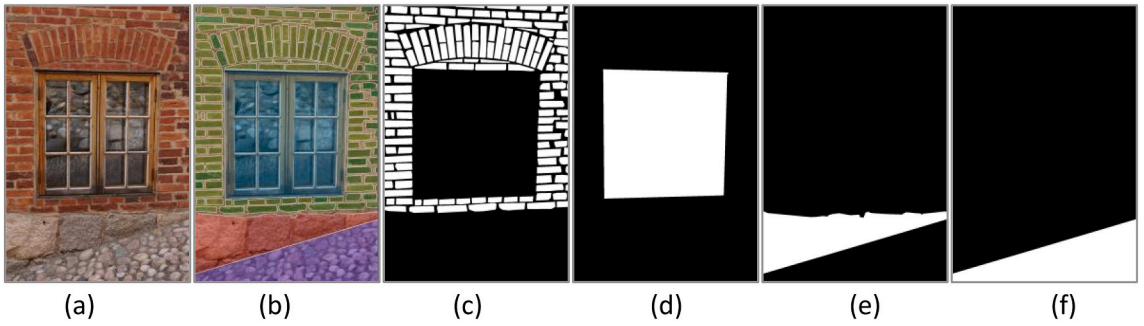


Fig. 2. Annotation of ground-truth data; a) Original image; b) SuperAnnotate vector classification; c) Bricks; d) Openings; e) Structural; f) Background.

$$OutDim = \sqrt{x_{slice} * y_{slice}} \tag{2}$$

$$MaxLimit = OutDim * MaxRatio \& MinLimit = OutDim * MinRatio \tag{3}$$

$$Scale = Limit / ImDim \tag{4}$$

$$\left(x_{final}, y_{final}\right) = \left(x_{image}, y_{image}\right) * Scale \tag{5}$$

where, *ImDim* is the average image-resolution, *OutDim* is the average image-slice resolution, *MaxLimit/MinLimit* is the maximum over the minimum limit that is allowed to disregard further adjustments, and x_{final}/y_{final} is the final resolution per image-axis.

Furthermore, the only variables provided were the *MaxRatio/MinRatio* that are the Maximum over the Minimum preferred ratio to adjust the images based on the size of the image-slices. Those values were adjusted manually until all the image slices contained a satisfactory number of block elements for the specific database. For the current database, the values used were the following:

$$MinRatio = 2, \text{and} \quad MaxRatio = 4 \tag{6}$$

The size of each image slice was equal to 224 × 224 × 3 (i.e., *OutDim* = 224). Each image was padded to adjust its resolution to multiples of the slice-resolution per axis (i.e., 224 pixels) to retain the original aspect-ratio when the image is sliced to smaller parts. The padding value was equal to 255, which created a white border around most of the edges. Then, the white padding was used as filtered locations of post-processed images (i.e., images were background and openings have been replaced with white) and would be instantly disregarded from the CNN output. This provided a total of 2814 image-slices which were used as training and validation (i.e., approx. 25% of them were used for validation), see (Fig. 3). Other slice-resolutions were also tested. The smaller resolutions provided more accurate models. This was due to the increased number of training and validation data.

3. Convolutional neural networks

In this research, the networks evaluated were the U-Net, DeepLabV3+ (Fig. 4), U-Net (SM), LinkNet (SM), and FPN (SM). The latter three (i.e., the *SM*'s) were generated through the python package called "*Segmentation Models*", which includes: ready-to-use semantic-segmentation models, multiple backbones of renown architectures, and pre-trained models for transfer learning. The training procedure involved only the use of the brick class since the dataset of other classes was not considered to be large enough.

Multiple tests were conducted to identify the best combination of backbone, pretrained dataset, loss function, optimiser, and parameters (see Table 1), that would provide the most efficient model. The first test included a combination of different parameters except loss, optimiser, and activation function (Table 1). Every architecture was tested with the "*Adam*" optimiser and "*Weighted-Cross-Entropy*" loss function. The learning-rate of the first test-sequence was equal to 5E-4 with a decay of 5E-6 over 100 epochs.

All architectures provided similar results (between 95.98% to 96.27% validation-accuracy), with DeepLabV3+ (#3) having the highest accuracy (96.27%). For each architecture, the parameters used for the optimal model were:

a) U-Net (#6): Optimized using 64 output filters, 0.0005 L2-Regularization, 0.25 dropout, Batch-Normalization, and "*glorot-uniform*" initializer.

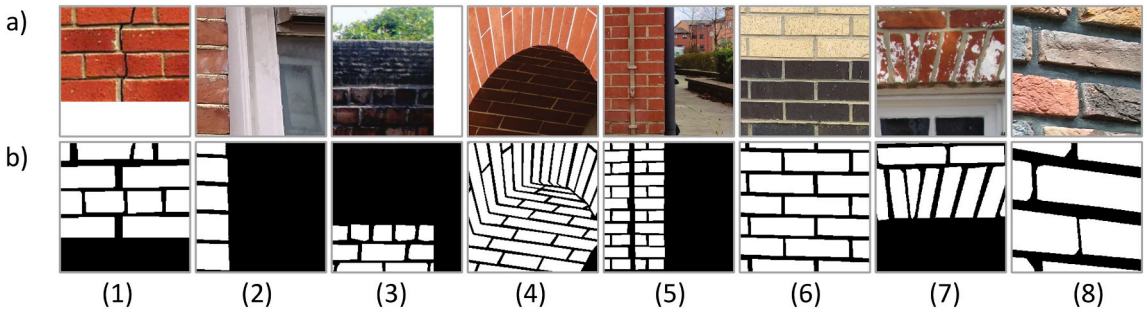


Fig. 3. Sample of the slices used to train and evaluate the model; Top: Original image slice; Bottom: Annotated blocks.

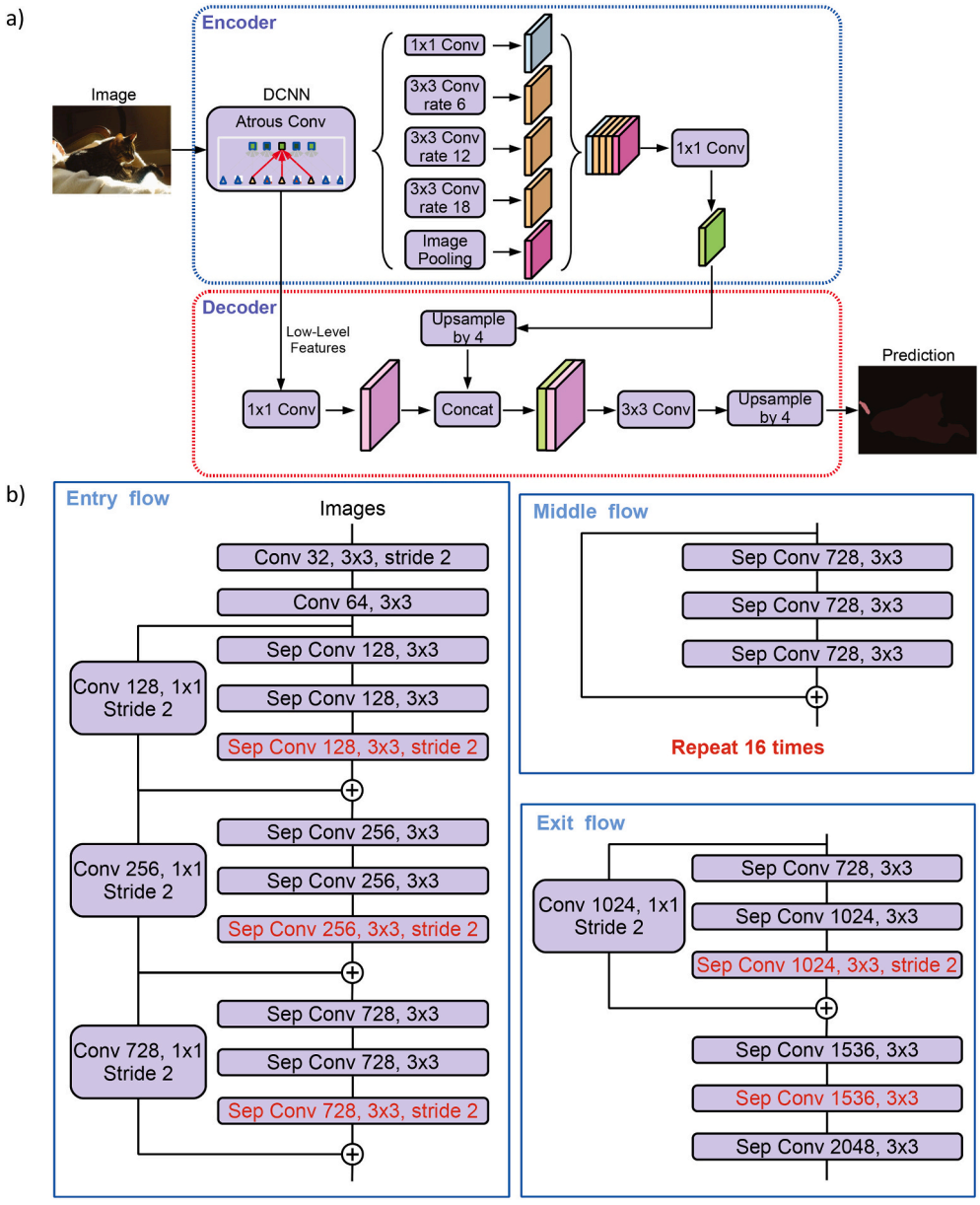


Fig. 4. Architecture of the highest performance model [14]; a) DeepLabV3+ architecture; b) Modified XCEPTION backbone.

b) U-Net-SM (#10): 0.0005 L2-Regularization, "*MobileNet*" backbone, "*ImageNet*" Encoder-Weights, "*Sigmoid*" activation function, and (256, 128, 64, 32, 16) decoder filters.

c) U-Net-SM (#11) is like U-Net-SM (#10) except that (512, 256, 128, 64, 32) decoder filters were used.

Table 1
Testing different parameters for each provided architecture (bold indicates best of each section).

Architecture	Model	Backbone	Val. Acc.	Val. F1	Val. Precision	Val. Recall	Val. Loss
–	–	–	%	%	%	%	–
U-Net	#1	–	95.62	95	0.92	**98.26**	**0.37**
U-Net	#2	–	95.4	94.71	92.35	97.45	0.54
U-Net	#3	–	95.51	94.76	92.18	97.8	0.42
U-Net	#4	–	95.86	95.19	93.01	97.68	0.48
U-Net	#5	–	93.39	92.37	88.03	97.51	0.58
U-Net	**#6**	–	**96.02**	**95.4**	93.05	97.99	0.41
U-Net	#7	–	95.88	95.21	**93.77**	96.86	0.63
U-Net (SM)	#1	VGG16	95.1	94.3	92.1	96.78	0.65
U-Net (SM)	#2	VGG19	95.17	94.32	91.96	97.01	0.59
U-Net (SM)	#3	InceptionV3	95.37	94.6	**94.06**	95.38	0.97
U-Net (SM)	#4	Inception-ResNetV2	89.07	72.46	72.9	75.83	1.23
U-Net (SM)	#5	MobileNet	95.83	95.19	93.79	96.78	0.69
U-Net (SM)	#6	ResNet50	94.65	93.88	91.06	97.09	0.59
U-Net (SM)	#7	SeresNet101	95.13	94.4	92.45	96.61	4.58
U-Net (SM)	#8	SeresNet152	94.77	94.09	90.69	**97.96**	4.21
U-Net (SM)	#9	ResNet152	94.74	93.85	91.2	96.96	0.64
U-Net (SM)	**#10**	MobileNet	**95.98**	95.24	93.64	97.02	0.61
U-Net (SM)	**#11**	MobileNet	95.94	**95.26**	93.41	97.31	**0.56**
U-Net (SM)	#12	MobileNet	95.77	95.09	93.13	97.25	0.59
LinkNet (SM)	**#1**	MobileNet	**96.13**	**95.52**	**94.26**	96.92	0.64
LinkNet (SM)	#2	MobileNet	95.91	95.31	93.4	97.41	0.55
LinkNet (SM)	#3	MobileNet	95.92	95.35	93.3	**97.59**	**0.54**
FPN (SM)	#1	MobileNet	95.99	95.42	93.32	97.71	0.59
FPN (SM)	#2	MobileNet	95.89	95.3	92.9	**97.93**	**0.48**
FPN (SM)	**#3**	MobileNet	**96.07**	**95.53**	93.45	97.77	0.53
FPN (SM)	#4	MobileNet	95.99	95.39	**93.61**	97.35	0.58
DLV3+	#1	Xception	93.12	87.02	91.47	85.33	4.51
DLV3+	#2	Xception	96.26	95.6	**95.12**	96.24	0.81
DLV3+	**#3**	Xception	**96.27**	**95.66**	94.81	96.66	0.75
DLV3+	#4	MobileNetV2	95.85	95.26	93.45	97.22	**0.5**
DLV3+	#5	Xception	95.4	94.71	92.35	**97.45**	0.54

d) LinkNet-SM (#1): 0.0005 L2-Regularization, "*MobileNet*" backbone, "*ImageNet*" Encoder-Weights, "*Sigmoid*" activation function, and (1024, 512, 256, 128, 64) decoder filters.
e) FPN-SM (#3): 0.0005 L2-Regularization, "*MobileNet*" backbone, "*ImageNet*" Encoder-Weights, "*Sigmoid*" activation function, 512 filters and 0.25 dropout.
f) DeepLabV3+ (#3): 16 OS (feature-extractor output ratio), "*Xception*" backbone, "*Pascal-Voc*" pretrained weights, and "*Sigmoid*" activation.

4. Loss function

The loss function was used to minimise the error during training and define the weights to reduce the loss during the next evaluation.

Table 2
Test of loss functions (bold indicates best model of each section).

				Best of three of each model/loss combination				
Architecture	Model	Loss	Epoch	Val Accuracy	Val F1	Val Precision	Val Recall	Val Loss
–	–	–	–	%	%	%	%	–
U-Net	#6	FL	98	96.18	95.3	**97.12**	93.72	3241.99
U-Net	#6	WCE	78	95.44	94.69	92.77	**96.86**	0.66
U-Net	#6	F1L	97	96.09	95.22	95.6	95.09	0.05
U-Net	**#6**	**BCE**	92	**96.27**	**95.57**	96.12	95.14	0.13
U-Net (SM)	#10	FL	89	96.41	95.52	**96.82**	94.5	6022.23
U-Net (SM)	#10	WCE	74	95.99	95.28	94.36	**96.35**	0.97
U-Net (SM)	#10	F1L	74	96.08	95.32	95.98	94.81	0.1
U-Net (SM)	**#10**	**BCE**	99	**96.52**	**95.79**	96.5	95.21	0.18
LinkNet (SM)	#1	FL	99	96.06	95.11	**97.17**	93.29	4998.93
LinkNet (SM)	#1	WCE	80	96.07	95.25	94.3	96.41	0.92
LinkNet (SM)	#1	F1L	87	96.49	95.8	96.15	**95.56**	0.08
LinkNet (SM)	**#1**	**BCE**	95	**96.54**	**95.83**	96.24	95.49	0.17
FPN (SM)	#3	FL	93	96.36	95.55	96.74	94.53	5473.35
FPN (SM)	#3	WCE	87	96.21	95.5	93.95	**97.2**	0.78
FPN (SM)	#3	F1L	100	96.52	95.8	**96.78**	94.92	0.08
FPN (SM)	**#3**	**BCE**	85	**96.58**	**95.88**	96.36	95.49	0.19
DLV3+	#3	FL	59	96.31	95.55	**97.08**	94.17	3632.67
DLV3+	#3	WCE	83	96.09	95.47	93.94	**97.15**	0.57
DLV3+	#3	F1L	69	96.45	95.77	96.53	95.13	0.04
DLV3+	**#3**	**BCE**	97	**96.65**	**96.03**	96.53	95.62	0.15

Multiple loss-functions were tested to identify the most optimal for the current use-case. The loss functions tested were Focal-Loss (FL), Weighted-Cross-Entropy (WCE), F1-Loss (F1L), and Binary-Cross-Entropy (BCE). All cases used the "Adam" optimizer with learning-rate of 1E-4 and decay equal to 1E-6.

Table 2 presents the best of three of each architecture/loss combinations. After evaluating all cases, the most efficient loss-function (highest validation-accuracy) was the BCE. However, it has been noticed that the highest validation-precision was typically acquired when using FL and the highest validation-recall when using WCE. Nonetheless, the target metric was the validation-accuracy. Thus, the optimal loss function was taken equal to the BCE. Furthermore, to exclude any error in the evaluation procedure of the loss-functions, a combination of different optimisers per loss-function was tested. However, the evaluation of the optimiser was only undertaken for the DeepLabV3+ architecture since it had the highest validation-accuracy for both tests. The remaining parameters were equal between the second and third tests. The two optimisers used herein were: a) Adam, Stochastic Gradient Descent (SGD); and b) RMSprop (RMSP).

In Table 3 each optimiser provided the most efficient model with different loss function. So, the use of BCE as the optimal loss function was not universal. Using the SGD optimiser, the most efficient loss function was WCE (86.14% validation accuracy). With RMSP, the optimal loss function was F1L (96.72% validation accuracy). Using Adam, the highest score was obtained through BCE (96.65% validation accuracy). Also, from Table 3, it is concluded that the combinations (Optimiser/Loss) with the highest accuracy were the Adam-BCE and RMSP-F1L and had very similar accuracy. So, both have been considered for the development of the final model. In contrast, the SGD optimiser was disregarded due to the low accuracy score along all loss functions.

5. Final model

The optimal learning-rate used to adjust the final model and decide on the utilisation of F1L and BCE loss functions. RMSP was selected as the target optimiser, since it obtained the highest score, see Table 3. Different learning-rate values were tested over 200 epochs. The decay used was equal to the Learning-Rate over the Max-Epoch. All models used the DLV3+ architecture with the Xception backbone, pretrained to the Pascal-VOC dataset, see Table 4.

The highest score using F1L loss function was obtained with 2E-4 learning rate with validation accuracy equal to 96.86% (Table 4). The highest score using BCE loss function was obtained with 1E-4 learning rate with validation accuracy equal to 96.87% (Table 4). The validation accuracy of both models was very similar. Thus, the selection of the final model considered the progression of the loss on the accuracy/loss graphs (Fig. 5) and the visual representation of the validation set (Fig. 6 and Fig. 7).

The output graph of the BCE model shows moderate overfitting to the dataset provided (Increasing validation-accuracy and validation-loss), which reveals that the BCE model may not be generalising as well as the model with F1L. Moreover, from the samples provided, the model with F1L has reduced noise on complex locations (i.e., images 1–3 in Fig. 6 and Fig. 7). Although using BCE the validation score was slightly higher in the model, the reduction of noise assisted with the detection of individual blocks on more complex images. Also, both models provided very accurate results for images with adequate resolution per block. Moreover, both were able to recognise openings and backgrounds exceptionally well, even for bricks with varied colour (i.e., images 4 and 8 in Fig. 6 and Fig. 7). So, the model with F1L loss function was considered as the best model and adopted here. In more detail, the specified model has a classification error equal to 1.24% for the background and 1.52% for the blocks class (Fig. 8). Additionally, the model is aimed to be used for images that are simpler than the trained dataset. Thus, in practice, the CNN-output is expected to provide improved results.

6. Crack detection algorithm

As mentioned before, the crack detection algorithm adopted in this study used the most efficient model presented in Dais et al. [16]. Both patch classification and pixel wise segmentation was tested. However, for the purposes of this study, only the models for pixel wise segmentation were considered (Table 5). The architectures tested were: a)

Table 3
Test of optimiser/loss combination (Bold indicates best of each section).

				Best of three of each optimiser/loss combination				
Architecture	Optimizer	Loss	Epoch	Val Accuracy	Val F1	Val Precision	Val Recall	Val Loss
–	–	–	–	%	%	%	%	–
DLV3+	SGD**	FL	62	56.72	0	0	0	350,036.3
DLV3+	SGD	FL	62	56.72	0	0	0	350,036.3
DLV3+	**SGD****	**WCE**	72	**86.14**	**84.6**	79.51	**92.45**	1.95
DLV3+	SGD	WCE	96	75.16	73.23	66.15	84.95	4.75
DLV3+	SGD**	F1L	96	76.52	74.38	68.73	84.36	0.26
DLV3+	SGD	F1L	100	72.47	67.84	63.72	74.88	0.33
DLV3+	SGD**	BCE	25	83.59	79.86	**81.19**	81.05	6.08
DLV3+	SGD	BCE	100	73.65	67.11	67.45	70.25	4.58
DLV3+	RMSP*	FL	55	94.93	93.75	96.22	91.67	5921.18
DLV3+	RMSP	FL	44	96.4	95.75	**96.51**	95.1	7827.02
DLV3+	RMSP*	WCE	46	94.09	93.1	92.58	94	1.48
DLV3+	RMSP	WCE	88	96.2	95.5	94.75	**96.39**	1.2
DLV3+	RMSP*	F1L	91	95.87	94.92	95.49	94.72	0.05
DLV3+	**RMSP**	**F1L**	56	**96.72**	**96.09**	96.81	95.47	0.04
DLV3+	RMSP*	BCE	76	95.84	94.9	95.36	94.79	0.26
DLV3+	RMSP	BCE	96	96.57	95.92	96.19	95.75	0.23
DLV3+	Adam	FL	59	96.31	95.55	**97.08**	94.17	3632.67
DLV3+	Adam	WCE	83	96.09	95.47	93.94	**97.15**	0.57
DLV3+	Adam	F1L	69	96.45	95.77	96.53	95.13	0.04
DLV3+	**Adam**	**BCE**	97	**96.65**	**96.03**	96.53	95.62	0.15

Default (No Stars): Momentum = 0, Nesterov = False.
* Momentum = 0.9.
** Momentum = 0.9, Nesterov = True.

Table 4
Testing different learning-rate values for the selection of the final model (bold indicates best of each section).

				Best of three (DLV3+)				
Optimizer	Loss	Learning Rate	Epoch	Val Accuracy	Val F1	Val Precision	Val Recall	Val Loss
–	–	–	–	%	%	%	%	–
RMSP	F1L	1.00E-04	79	96.73	96.16	96.54	95.82	0.04
RMSP	**F1L**	**2.00E-04**	117	**96.86**	**96.29**	**96.68**	**95.94**	**0.04**
RMSP	F1L	5.00E-05	67	96.57	95.96	96.32	95.65	0.04
RMSP	F1L	0.0005	179	96.62	96.03	96.16	95.93	0.04
RMSP	**BCE**	**1.00E-04**	127	**96.87**	**96.3**	96.46	**96.16**	**0.23**
RMSP	BCE	2.00E-04	131	96.85	96.28	**96.64**	95.94	0.37
RMSP	BCE	5.00E-05	165	96.48	95.85	96.17	95.59	0.26
RMSP	BCE	0.0005	88	96.46	95.84	96.01	95.7	0.24

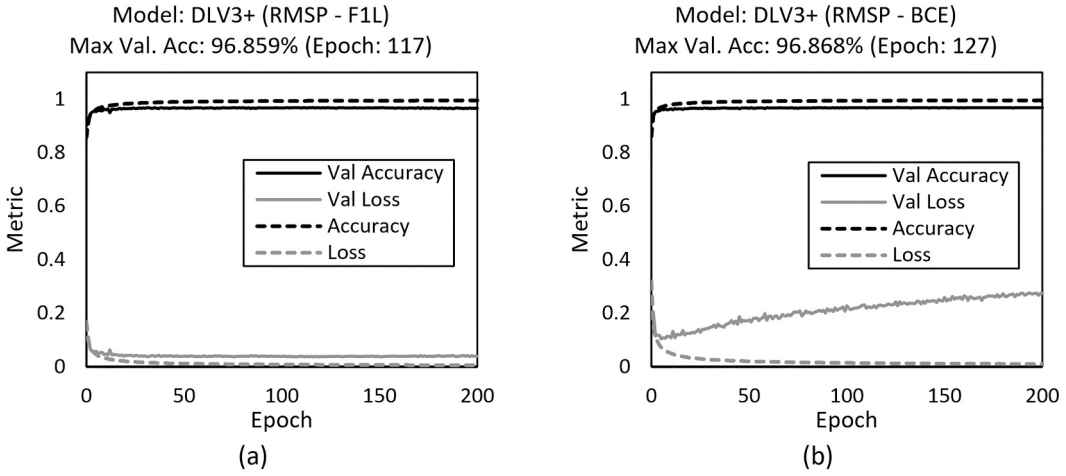


Fig. 5. Graphs of best models; a) DLV3+ model with F1L loss function and learning rate of 2E-4; b) DLV3+ model with BCE loss function and learning rate of 1E-4.

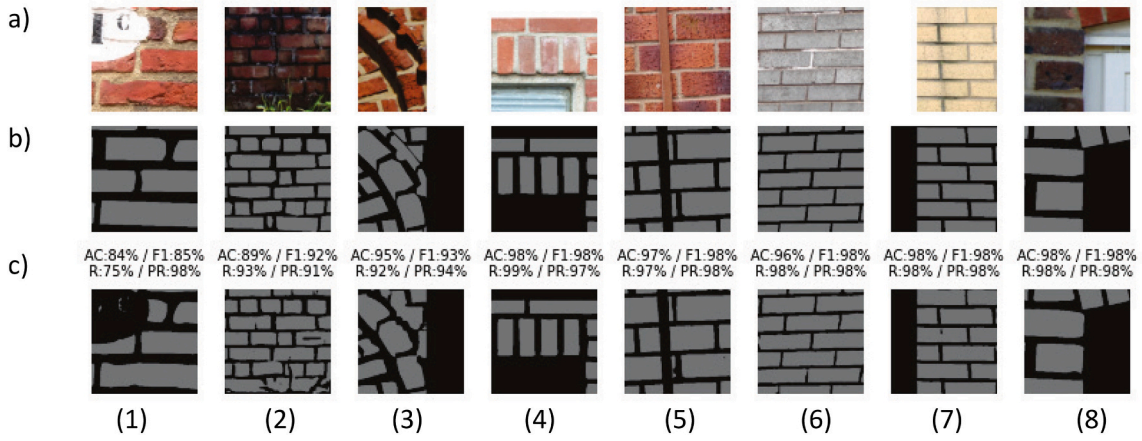


Fig. 6. Evaluation sample from the F1L model; a) Original image-slice; b) Ground truth; c) CNN Output (AC: Accuracy; F1: F1-Score; R: Recall; PR: Precision).

DeepCrack; b) DeepLabV3+; c) FCN based on VGG16; d) U-Net; and e) FPN. The backbones tested were: VGG16, ResNet (multiple), DenseNet (multiple), Inception, MobileNet, and MobileNetV2. Also, multiple loss functions were tested to identify their optimal parameters. The loss functions used were: a) Weighted-Cross-Entropy (WCE), b) Cross-Entropy (CE), c) F1-Score-Loss (F1), and d) Focal-Loss (FL). All models included the use of the Adam optimiser since it provided the highest F1-Score. Transfer learning was also utilised to improve the accuracy of the detection using the ImageNet dataset.

The architecture selected was the U-net-MobileNet with Adam optimiser and WCE loss function. The specified model achieved a validation F1-score equal to 79.6%, validation recall equal to 79.9%, and validation precision equal to 81.4%. The existing CNN-model was used to acquire the damage during the geometrical-model generation. Moreover, the dataset of the damage-detection model is similar to the block-detection model. Thus, it can be used directly to combine the results of both models (blocks and cracks) efficiently. The sample of the validation set used in the evaluation of the model is shown in Fig. 9.

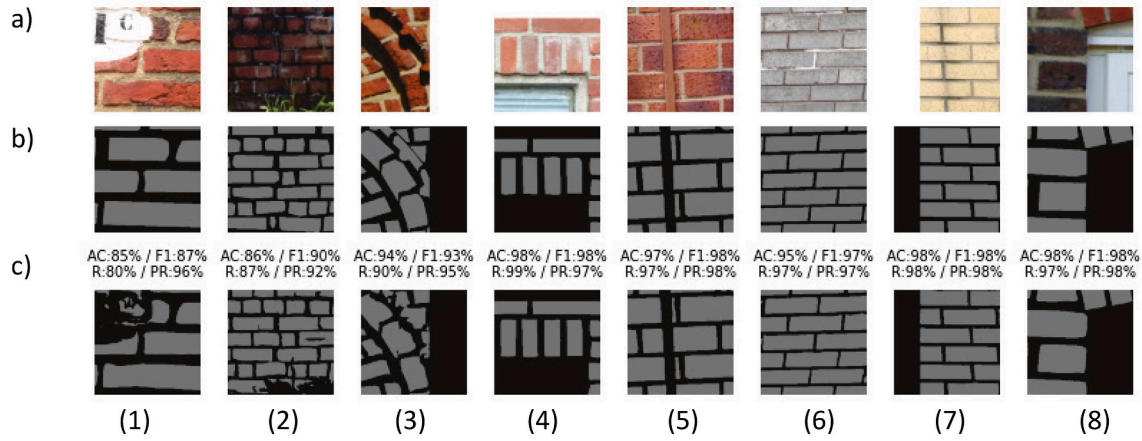


Fig. 7. Evaluation sample from the BCE model; a) Original image-slice; b) Ground truth; c) CNN Output (Metrics: AC: Accuracy, F1: F1-Score, R: Recall, PR: Precision).

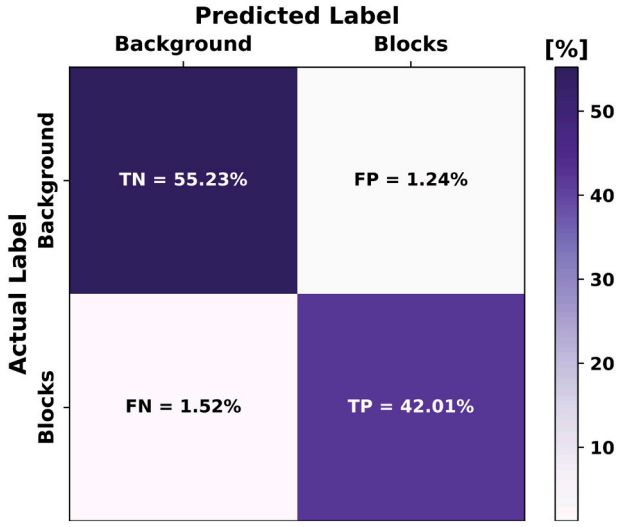


Fig. 8. Confusion matrix; TN: True Negatives; FN: False Negatives; FP: False Positives; TP: True Positives (Model: DLV3+ with RMSP-F1L).

7. Coupling brick segmentation with crack detection

To acquire the final output, image processing of the original image was undertaken. Initially the image was re-sized using the same methodology described in the Development of the database section (Eq. (5)). By combining the image slices directly to the image, distortion effects near the edges of the image-slice (Fig. 10: b) were observed. So, each slice assigned an overlap value. The best results were acquired using an overlap value of 50 pixels for an image slice of 224 × 224. The image was divided into sections of 124 × 124 pixels (224 − 2 × 50) and included a white padding of 100 pixels (2 × 50). This effectively retained only the central section of each slice for use and improved the overall quality of the final output (Fig. 10: c). Furthermore, the use of the models to acquire the location of cracks and masonry elements shown satisfactory results. Fig. 11 presents outputs from both models CNN (blocks and cracks) that were used for verification purposes i.e., not used during the training/evaluation phase.

8. Usage of the developed model

One important use of the damage detection model proposed here is to assist engineers with the inspection and documentation of masonry structures in their care. Using image processing, each individual crack was identified and measured. The isolation of white elements from the CNN output was succeeded by using watershed segmentation to assign a unique label to each crack (Fig. 12: b). Using the individual labels of the segmentation, it was possible to acquire the area of each label by counting the total number of pixels. Each segmentation provided the linearization of its area (Fig. 12: c), which can be used to evaluate the length of the crack. Finally, the results obtained can be scaled to the real dimensions and provide realistic measurements of the crack properties by providing a scale factor (Table 6). The approximate-length (Skeleton (mm)) was acquired under the assumption that the length of each pixel is the average between its horizontal and diagonal distance.

The main use of the feature detection was aimed for the automatic development of geometrical models for documentation and numerical models for analysing the structural capacity of masonry structures. The methodology to convert binary images of masonry blocks and cracks is further explained in Loverdos et al. [32].

The algorithms described in the previous study used binary images acquired using simple photogrammetric applications (i.e., image blurring, image thresholding, edge detection). However, the use of simple image processing applications found to be not reliable and, in some cases, unusable (i.e., for large variations on illumination and/or colour (Fig. 13: c, e, and g)). Furthermore, the process requires to adjust the parameters of each image-processing function manually. The use of CNN, for the feature detection of masonry micro-geometry (i.e., geometry of individual masonry units and mortar), improves the results of the feature extraction by providing a better binarized output and automating the procedure (Fig. 13: d, f, and h).

For both applications (measurement of cracks and generation of geometrical model) the image is required to be either an orthorectified photo or an image captured vertically compared to the masonry element. This will ensure that the detected elements (i.e., blocks, cracks, openings) will have the same scale along the image used.

Moreover, the simplified shapes acquired, using the binarized output from the trained models, demonstrated great improvement when compared to binarized images acquired using image-processing. The blocks were more evenly shaped and better aligned to the actual masonry bricks (Fig. 14). This did not only improve the reliability of the numerical analysis, but also the geometric representation of the structure when used for representation in CAD environments. It should be noted that the original image used in Fig. 13 and Fig. 14 was the same used for the generation of the numerical model in the previous study for comparison purposes. Additionally, the use of simple image-processing applications, for the feature detection, favours the specified image since the contrast between mortar and bricks is highly visible, without large changes to illumination/colour. For general use, the difference between the resultant output is expected to be larger.

Table 5
Architectures tested for defect detection of masonry structures (bold indicates best models).

							Validation Scores		
Network	Pretrained	Loss	Parameters	Model Size	Analysis Time	Best Epoch	F1 Score	Recall	Precision
–	[ImageNet]	–	[Millions]	[MB]	[Hours]	–	%	%	%
DeepCrack	No	WCE	29.5	115.5	5.2	28	74	80.1	71.6
DeepLabv3+	No	WCE	41.3	162.2	5.6	26	74.9	79	73.8
FCN-VGG16	No	WCE	27.8	108.8	2.5	95	75.6	76.6	76.9
U-net	No	WCE	34.5	135.1	5.8	75	75.7	78.9	75.7
U-net-VGG16	Yes	WCE	46.1	180.2	6	37	77.2	81.2	76.2
U-net-ResNet34	Yes	WCE	48	188.1	4.9	61	77.6	78.3	79.5
U-net-ResNet50	Yes	WCE	73.7	288.5	6.8	45	76.3	80.9	74.8
U-net-Densenet121	Yes	WCE	41.6	163.5	6.2	55	78.1	80.7	78.1
U-net-Densenet169	Yes	WCE	54.3	213.4	7.1	63	78.5	83.5	76.2
U-net-InceptionV3	Yes	WCE	68.5	268.1	6.8	31	77.7	79.2	78.9
U-net-MobileNet	**Yes**	**WCE**	37.8	147.9	4.8	45	**79.6**	79.9	81.4
U-net-MobileNet	No	WCE	37.8	147.9	4.8	36	75.4	80.7	73.4
U-net-MobileNet	Yes	CE	37.8	147.9	4.8	36	76.6	73	83
U-net-MobileNet	Yes	F1	37.8	147.9	4.8	29	78.2	77.1	82
U-net-MobileNet	Yes	FL	37.8	147.9	4.8	85	71.2	61.1	**89.4**
U-net-MobileNetV2	Yes	WCE	39.5	154.9	5.3	58	77.7	76.6	81.9
FPN-VGG16	Yes	WCE	32.2	125.8	5.6	79	77.9	82	76.2
FPN-ResNet34	Yes	WCE	38.3	150.2	5.2	36	78	81.5	77.2
FPN-ResNet50	Yes	WCE	42.1	164.8	5.8	27	77.2	81.4	75.8
FPN-Densenet121	Yes	WCE	24.6	97	6.1	31	79	**83.6**	77.2
FPN-Densenet169	Yes	WCE	30.6	120.8	6.6	59	78.6	80	79.5
FPN-InceptionV3	**Yes**	**WCE**	40	157.2	5.7	34	**79.6**	81.3	80.1
FPN-MobileNet	Yes	WCE	20.8	81.4	4.6	40	79.5	79.5	81.7
FPN-MobileNetV2	Yes	WCE	19.9	78.3	4.8	49	78.5	76.7	82.7

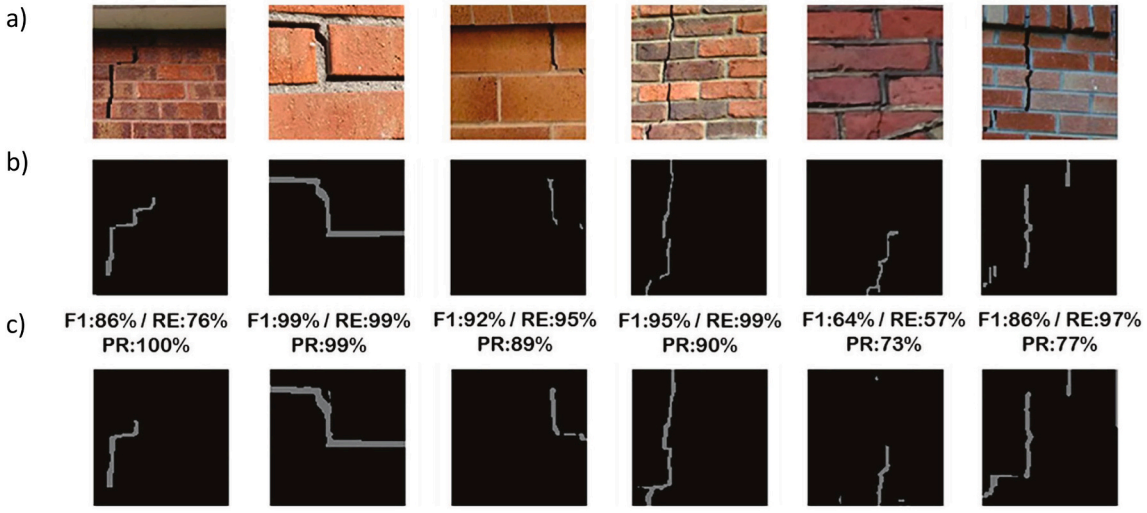


Fig. 9. *Sample images from the crack detection model from [16]; a) Original image-slice; b) Ground truth; c) CNN Output (Metrics: F1: F1-Score, RE: Recall, PR: Precision).*

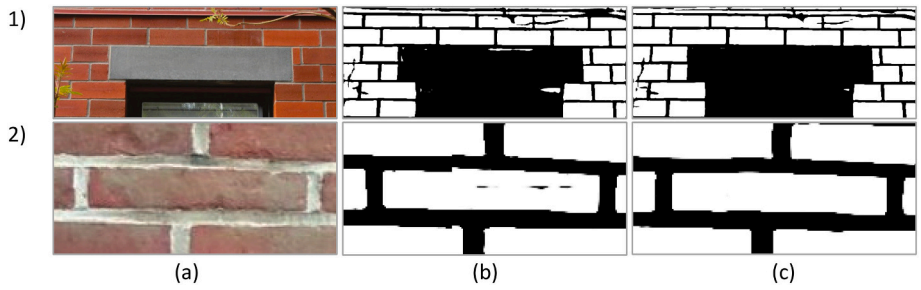


Fig. 10. Effect of using overlap while connecting output-slices; a) Original image; b) Direct connection of 224 × 224 pixel slices; c) Overlap of 50 pixels on 224 × 224 pixel slices.

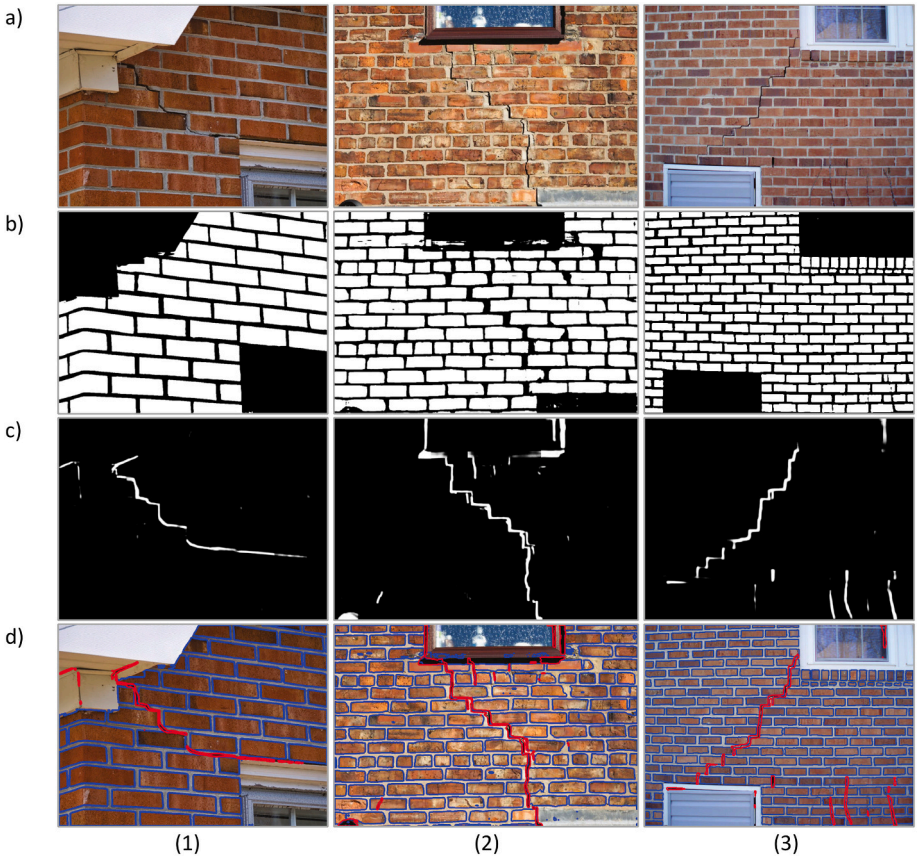


Fig. 11. Combined output of block and crack detection models; a) Original image; b) Block detection; c) Crack detection; d) Marked perimeter of detected elements.

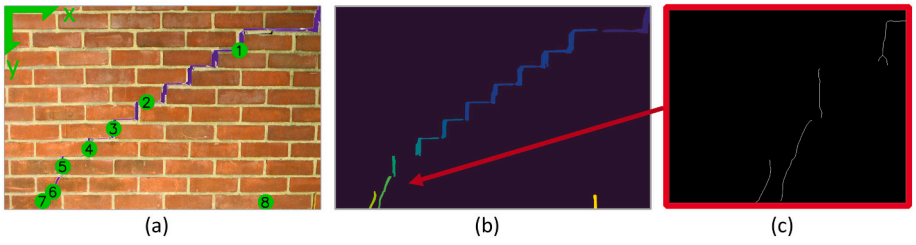


Fig. 12. Evaluation of damage; a) Marked cracks; b) Watershed segmentation; c) Linearization (skeleton).

Table 6
Crack properties acquired using image-processing.

Label	Loc Min	Loc Max	Loc Mid	Area	Skeleton	Area	Skeleton
	(xmin,ymin)	(xmax,ymax)	(x,y)	(pixels)	(pixels)	(mm^2)	(mm)
1	[1908, 0]	[2249, 189]	[2079, 170]	13,474	472	5072	350
2	[923, 155]	[1872, 817]	[1397, 427]	37,425	1482	14,088	1065
3	[753, 804]	[913, 945]	[775, 874]	5537	247	2084	178
4	[568, 936]	[739, 1072]	[595, 993]	5675	272	2136	196
5	[402, 1069]	[428, 1220]	[414, 1144]	3076	137	1158	98
6	[302, 1215]	[397, 1448]	[343, 1325]	3971	229	1495	165
7	[237, 1323]	[289, 1448]	[270, 1390]	2217	121	835	87
8	[1843, 1348]	[1869, 1448]	[1856, 1398]	2190	79	824	57

9. Shape quality

The quality of the segmentation was evaluated to quantify the change between the ground truth and the output from either the CNN model or simple image-processing applications. The simple metrics included the accuracy, recall, precision, and F1-Score, as seen in the use of the CNN model.

However, those metrics tend to check the overall quality of the output. Since the model was aimed to be used for the generation of geometrical models and documentation, additional metrics were included to quantify the quality of the block-shapes. The shape quality was estimated by calculating the coverage, error of area, and quantity of

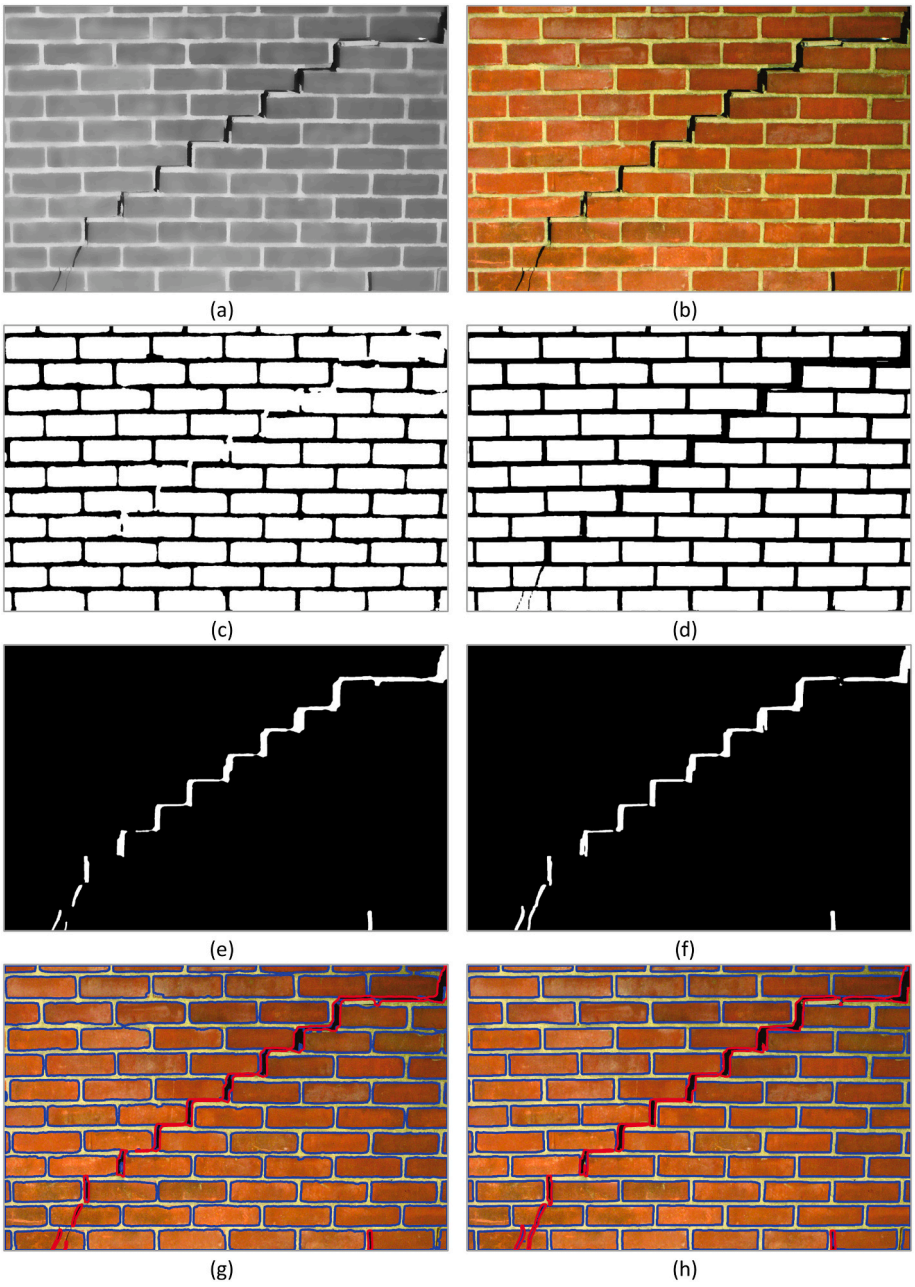


Fig. 13. Comparison of Thresholding and CNN output; a) Blurred/grey image for thresholding; b) Original image for CNN; c) Detected blocks using thresholding; d) Detected blocks using CNN; e) Detected cracks using thresholding; f) Detected cracks using CNN; g) Perimeter of detected elements using thresholding; h) Perimeter of detected elements using CNN.

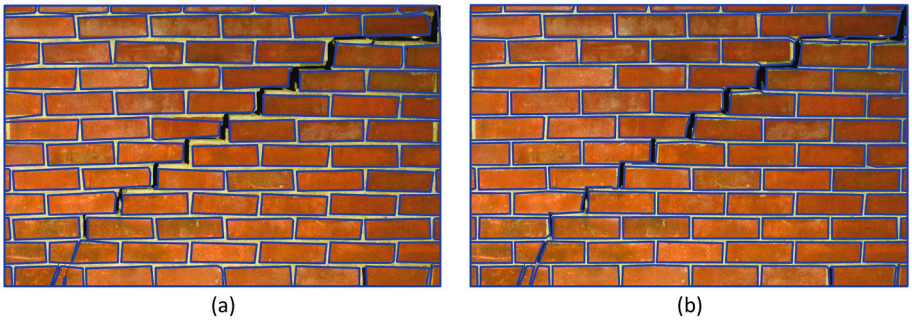


Fig. 14. *Extracted blocks using the methodology described in* Loverdos et al. [32]*; a) Block-detection using image-thresholding; b) Block-detection using CNN.*

undefined blocks (Fig. 15). Each segmentation of the ground-truth was compared with the output of either CNN or image-processing to identify the same object in both images. The first step was to calculate the common area between the two objects (Eq. (7)). The coverage (Eq. (8)) was calculated by comparing the common-area between both objects (ground-truth and output) while the Area-Error was calculated by comparing the area between both objects (Eq. (9)). The quantity of undefined objects was the number of objects that didn't match with a segmentation from the ground truth following certain conditions (i.e., the coverage must be similar to the area of the segmentation). The equations of the additional metrics are provided below:

$$Common = Object1_i \cap Object2_i \tag{7}$$

$$Coverage1 = Common/Area1$$

$$Coverage2 = Common/Area2 \tag{8}$$

$$Area\ Error = Area2/Area1 - 1 \tag{9}$$

$$Missing\ Error = Missing/AllObjects \tag{10}$$

where $Object1_i$ denotes any segmentation on ground truth, $Object2_i$ any segmentation on the output (CNN or Image-processing), $Area1$ the total area of the $Object1$ in pixels, and $Area2$ the total area of $Object2$ in pixels. Individual segmentations were obtained using watershed-segmentation with the binary image as mask. The conditions for segmentation, i.e., the same object as in the ground-truth, were:

$$\text{Condition1(Required)} : Common > 0 \tag{11}$$

$$\text{Condition2(Optional)} : 1 - Coverage1 \leq Threshold1 \tag{12}$$

$$\text{Condition3(Optional)} : 1 - Coverage2 \leq Threshold2 \tag{13}$$

where the *Threshold* is any value between 0 and 1 and denotes the difference between the common area and the total area of the segmentation. The shape-analysis in this case used a Threshold value of 0.2 (i.e., 80% of object area must be common). The first condition was used to verify that two objects have a common area (Eq. (11)). By using the second condition only (Eq. (12)), the evaluation considered objects that were erroneously merged (Fig. 15c), which increased the area-error significantly. By using the third condition only (Eq. (13)), the evaluation considered segmentations that were erroneously broken into smaller elements. If multiple objects satisfy the conditions, then they were all included in the final common area. For this study, both optional conditions were used. Thus, for an object to be considered, must have a common area of at least 80% of both segmentations (ground-truth and output). Fig. 15 shows the undefined objects that did not match both images (Fig. 15: a–b & Fig. 15: a–c). Furthermore, it demonstrates that the CNN output has marginally fewer undefined objects, since image-processing is prone to noise caused by the change in illumination and colour within the same image. Although watershed-segmentation can close open-segmentations (Fig. 13: g), it is not always feasible. In Fig. 16, images were utilised during the validation phase of the model and are shown to compare the output acquired using image-processing. The method applied was the same as the one used to acquire the image in Fig. 13:c. The metrics for the complete evaluation of all images (Fig. 15 & Fig. 16), are shown in Table 7.

Most metrics provide similar values for both test cases of the damaged wall (Fig. 15). More specifically the accuracy on the CNN output was slightly higher, which explains the better representation of the segmentation (95.7% vs 94.8%). Although, the coverage in the CNN image was slightly lower for the objects that were detected correctly (94.4% vs 96.7%). Nonetheless, the missing error of the CNN image was much lower than the thresholding case (5.5% vs 27.5%), which was caused by the presence of multiple open shapes in the thresholding image (Fig. 15: c). On the CNN case, the bottom-left block was undefined due to the damage not separating the blocks completely (Fig. 15: b), as it is on the ground truth image (Fig. 15: a) and detecting the three broken elements as a single object. Also, it should be noted that the large value of the undefined blocks, in the thresholding case, would decrease the coverage and increase the area error, if they were allowed in the evaluation (Fig. 15: c). Thus, a higher coverage does not correspond to an overall better quality of segmentation, since it relates to fewer elements. Furthermore, the image was favourable for block detection using thresholding, due to minimal noise, which explains the better fit of the validated objects. The results will vary depending on alterations to illumination and colour (Fig. 16).

The metrics acquired for Fig. 16 demonstrate that the quality of shapes when using simple thresholding was detrimental for the accuracy of the model. This can be observed initially from the median accuracy, which was equal to 94.5% vs 79.3%, for the CNN and thresholding methods respectively. In general, thresholding provided marginally fewer validated blocks, as it was observed by the median missing error, which was equal to 22.9% vs 69.1%, for the CNN and thresholding methods respectively (Fig. 16: c, d). Furthermore, the overall fitting of the shapes was higher in the CNN case since the coverage calculated was 96.7% vs 94.6% for the CNN and thresholding methods respectively.

Moreover, simple image-processing methods can not recognise openings, damage, or background (Fig. 16: d4 & Fig. 16: d8). They are only able to identify either edges (edge detection) or pixel intensity (thresholding). Every image that contained locations of background, required modification before it was used. Thus, the use of CNN for the object detection was preferred for the current test-case, due to its higher accuracy and reliability to identify correctly almost every object (Fig. 15: b), except for highly complex images (Fig. 16: c1, c2, c3).

10. Conclusions

This research is contributing towards automating procedures that engineers would require considerable amounts of effort, expertise, and time to perform documentation and structural inspection of masonry fabric, while at the same time minimises the human error. Also, the

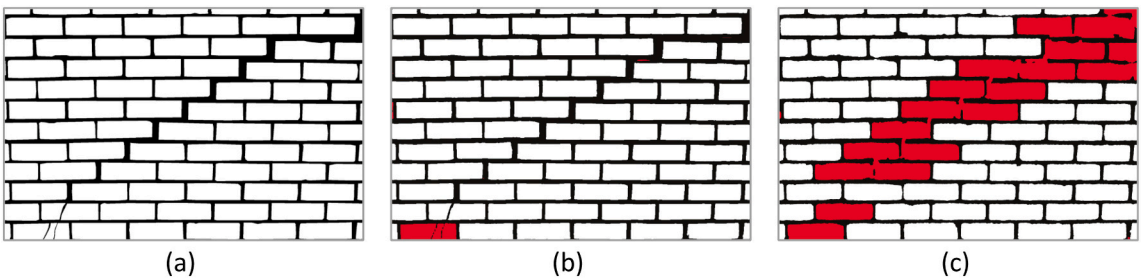


Fig. 15. Comparing segmentation quality of CNN and Thresholding (Red: Unidentified blocks with large change to location/area): a) Ground-truth image (Annotated); b) CNN-Output compared with annotated; c: Thresholding-output compared with annotated. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

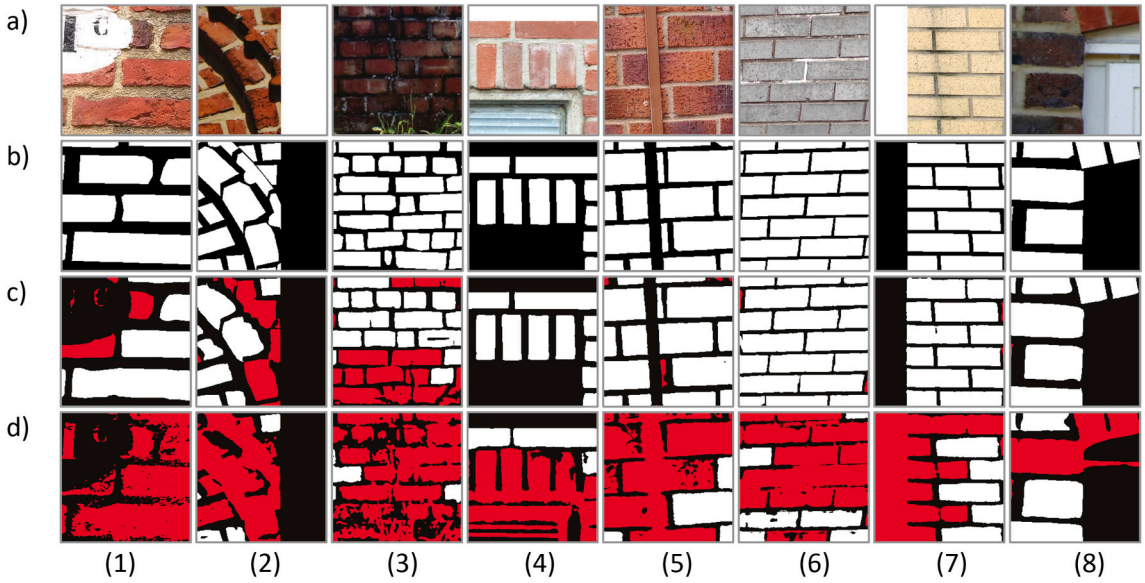


Fig. 16. Comparing segmentation quality of the train/validation set (Red: Unidentified blocks with large change to location/area): a) Original image; b) Ground-truth image (Annotated); c) CNN-Output compared with annotated; d) Thresholding-output compared with annotated. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 7
Metrics to quantify the segmentation quality of the output.

Name	Image1 (Gr. Truth)	Image2 (Output)	Acc.	Recall	Precision	F1 Score	Coverage (1)	Area Error	Missing Error (1)
–	–	–	%	%	%	%	%	%	%
Fig. 15: b	Annotated	CNN	95.68	94.63	99.78	97.13	94.4	−5.39	5.5
Fig. 15: c	Annotated	Thresh.	94.76	96.68	96.55	96.62	96.7	−1.64	27.47
Fig. 16: 1c	Annotated	CNN	83.63	75.24	97.65	84.99	97.31	0.76	44.44
Fig. 16: 1d	Annotated	Thresh.	66.67	68.08	75.44	71.57	0	n.a.	100
Fig. 16: 2c	Annotated	CNN	94.65	92.43	94.48	93.44	93.95	−2.11	50
Fig. 16: 2d	Annotated	Thresh.	83.24	87.17	75.82	81.1	92.03	−4.8	75
Fig. 16:3c	Annotated	CNN	89.35	93.5	91.47	92.47	96.2	3.63	50
Fig. 16: 3d	Annotated	Thresh.	70.97	76.69	80.83	78.71	94.27	3.58	87.5
Fig. 16: 4c	Annotated	CNN	98.25	98.56	96.93	97.74	98.42	1.76	9.09
Fig. 16: 4d	Annotated	Thresh.	70.7	95.07	57.08	71.33	95.46	12.58	72.73
Fig. 16:5c	Annotated	CNN	96.69	97.12	98.13	97.62	96.24	−1.66	16.67
Fig. 16: 5d	Annotated	Thresh.	78.28	91.85	80.03	85.53	96.99	0.81	77.78
Fig. 16: 6c	Annotated	CNN	96.29	97.63	97.77	97.7	97.91	0.28	10
Fig. 16: 6d	Annotated	Thresh.	87.8	92.68	92.21	92.45	90.35	−7.01	70
Fig. 16: 7c	Annotated	CNN	97.59	98.35	97.63	97.99	98.53	0.92	7.69
Fig. 16: 7d	Annotated	Thresh.	69.58	92.77	67.98	78.47	95.14	−2.07	61.54
Fig. 16: 8c	Annotated	CNN	98.22	98.17	97.99	98.08	97.21	−0.97	12.5
Fig. 16: 8d	Annotated	Thresh.	92.14	98.58	86.42	92.1	95.68	8.01	50
		Median CNN:	94.48	93.96	96.87	95.24	96.69	−0.31	22.88
		Median Thresh:	79.35	88.84	79.15	83.1	94.58	0.29	69.11

proposed approach is suitable for cases in which visual inspection is in locations difficult to reach by humans. The study demonstrates that both crack and block (i.e. masonry unit) detection can be achieved with adequately high accuracy by utilising deep learning approaches (i.e. block-detection model achieved a validation-accuracy of 96.86% and the crack detection model an F1-Score of 79.6%). The quality of the binarized output has also been assessed showing that the CNN output outperforms simple image-processing functions even for clean images. Especially considering that simple image-processing applications do not differentiate between detected elements and background/openings. Additionally, deep learning methods allow for the improvement of the model by increasing the dataset used for training and validation. Consequently, the performance of the model can always be enhanced by acquiring additional samples of the classified elements.

The main limitation of the demonstrated application of deep learning, for the detection of features in masonry structures, is that a similar sample should be provided during training of the model to detect specific features. i.e., to be able to reliably identify irregular masonry units, images with irregular masonry should be included in the dataset. Furthermore, features not shown in the image will not be identified by the model (i.e., cracks of extremely small size). Thus, the engineer must ensure that the desired features should be visible on the image-slice passed through the network. Also, the use orthorectified images is important for the accurate evaluation of detected features (i.e., if used for numerical modelling, or crack measurements).

Future work includes the implementation of the developed models to a framework that will automatically generate numerical models and analyse changes on the structure in real time. This includes a detailed report of the measurement of detected defects on the structure in a form of a geometrical digital twin. Moreover, currently only cracks are

detected but additional classifications of defects are considered, such as spalling, vegetation, and discolouration. Additionally, the results obtained from the block/crack-detection can be coupled directly with algorithms for numerical modelling to automatically evaluate the crack patterns on the structure by performing numerical analysis or compare the results from inverse analysis by matching the outputs (from the block/crack detection models and evaluation method used, i.e., [1,3,22,35,45]). In all cases, the capture procedure can be replaced by remote sensing applications such as drones to remotely capture image/video data paired with semantic segmentation for the identification of structural elements; hence, providing a digital twin of the structure considered for real time monitoring.

Declaration of Competing Interest

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

Acknowledgements

Several photos obtained by engineers from Network Rail and Helifix, UK, and were kindly offered to expand our masonry dataset. Therefore, their support in this study is highly recognised and appreciated. Ihsan Bal, Eleni Smyrou and Dimitris Dais are acknowledged for their insightful comments on the implementation of the deep learning network for the crack detection algorithm (github.com/dimitrisdais/crack_detection_CNN_masonry). This work was funded by the EPSRC project "Exploiting the resilience of masonry arch bridge infrastructure: a 3D multi-level modelling framework" (ref. EP/T001348/1). The financial contribution is very much appreciated.

References

[1] C. Alessandri, M. Garutti, V. Mallardo, G. Milani, Crack patterns induced by foundation settlements: integrated analysis on a renaissance masonry palace in Italy, Int. J. Architectural Heritage 9 (2015) 111–129, https://doi.org/10.1080/15583058.2014.951795.

[2] L. Ali, Damage detection and localization in masonry structure using faster region convolutional networks, Int. J. GEOMATE 17 (2019) 98–105, https://doi.org/10.21660/2019.59.8272.

[3] M. Angelillo, The model of Heyman and the statical and kinematical problems for masonry structures, Int. J. Masonry Res. Innovation 4 (2019) 14–21, https://doi.org/10.1504/IJMRI.2019.096820.

[4] P. Arbeláez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2011) 898–916, https://doi.org/10.1109/TPAMI.2010.161.

[5] İ.E. Bal, D. Dais, E. Smyrou, V. Sarhosis, Novel invisible markers for monitoring cracks on masonry structures, Constr. Build. Mater. 300 (2021), https://doi.org/10.1016/j.conbuildmat.2021.124013.

[6] S. Beucher, F. Meyer, Advances of mathematical morphology in image processing, in: Mathematical Morphology in Image Processing, Marcel Dekker Inc, New York, 1993, pp. 433–481, https://doi.org/10.1201/9781482277234-12.

[7] D.J. Bora, A novel approach for color image edge detection using multidirectional Sobel international journal of computer sciences and engineering open access a novel approach for color image edge detection using multidirectional Sobel filter on HSV color space, Int. J. Comput. Sci. Eng. 5 (2017) 154–159, https://doi.org/10.6084/m9.figshare.4732951.

[8] D. Brackenbury, M. Dejong, Mapping Mortar Joints in Image Textured 3D Models to Enable Automatic Damage Detection of Masonry Arch Bridges, Tampere, Finland, 2018.

[9] D. Brackenbury, I. Brilakis, M. Dejong, Automated defect detection for masonry arch bridges, in: International Conference on Smart Infrastructure and Construction 2019, ICSIC 2019: Driving Data-Informed Decision-Making 2019, 2019, pp. 3–10, https://doi.org/10.1680/icsic.64669.003.

[10] J. Canny, A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8, 1986, pp. 679–698, https://doi.org/10.1109/TPAMI.1986.4767851.

[11] K. Chaiyasarn, M. Sharma, L. Ali, W. Khan, N. Poovarodom, Crack detection in historical structures based on convolutional neural network, Int. J. GEOMATE 15 (2018) 240–251, https://doi.org/10.21660/2018.51.35376.

[12] A. Chaurasia, E. Culurciello, LinkNet: Exploiting encoder representations for efficient semantic segmentation, in: 2017 IEEE Visual Communications and Image Processing, VCIP 2017 2018-Janua, 2018, pp. 1–4, https://doi.org/10.1109/VCIP.2017.8305148.

[13] L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets and fully connected CRFs, in: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings 40, 2015, pp. 834–848.

[14] L.C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, DeepLabv3+: Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11211 LNCS, 2018, pp. 833–851, https://doi.org/10.1007/978-3-030-01234-2_49.

[15] F. Cluni, D. Costarelli, A.M. Minotti, G. Vinti, Enhancement of thermographic images as tool for structural analysis in earthquake engineering, NDT E Int. 70 (2015) 60–72, https://doi.org/10.1016/j.ndteint.2014.10.001.

[16] D. Dais, İ.E. Bal, E. Smyrou, V. Sarhosis, Automatic crack classification and segmentation on masonry surfaces using convolutional neural networks and transfer learning, Autom. Constr. 125 (2021), https://doi.org/10.1016/j.autcon.2021.103606.

[17] A.M. D'Altri, V. Sarhosis, G. Milani, J. Rots, S. Cattari, S. Lagomarsino, E. Sacco, A. Tralli, G. Castellazzi, S. de Miranda, Modeling Strategies for the Computational Analysis of Unreinforced Masonry Structures: Review and Classification, Archives of Computational Methods in Engineering, Springer, Netherlands, 2020, https://doi.org/10.1007/s11831-019-09351-x.

[18] J. Eaton, M. Edwards, M. Crapper, Heritage Railway Association: The Inspection and Maintenance of Civil Engineering Assets, 2014.

[19] M. Ergün Hatir, İ. İnce, Lithology mapping of stone heritage via state-of-the-art computer vision, J. Build. Eng. 34 (2021), https://doi.org/10.1016/j.jobe.2020.101921.

[20] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, J. Garcia-Rodriguez, A Review on Deep Learning Techniques Applied to Semantic Segmentation, 2017, pp. 1–23.

[21] M. Hussain, J.J. Bird, D.R. Faria, A study on CNN transfer learning for image classification, Advan. Intelligent Sys. Computing 840 (2019) 191–202, https://doi.org/10.1007/978-3-319-97982-3_16.

[22] A. Iannuzzo, F. de Serio, A. Gesualdo, G. Zuccaro, A. Fortunato, M. Angelillo, Crack patterns identification in masonry structures with a C° displacement energy method, Int. J. Masonry Res. Innovation 3 (2018) 295–323, https://doi.org/10.1504/IJMRI.2018.093490.

[23] Y. Ibrahim, B. Nagy, C. Benedek, Cnn-based watershed marker extraction for brick segmentation in masonry walls, in: F. Karray, A. Campilho, A. Yu (Eds.), Image Analysis and Recognition, ICIAR, vol. 2019, Lecture Notes in Computer Science. Springer, Cham, Waterloo, ON, Canada, 2019, pp. 332–344, https://doi.org/10.1007/978-3-030-27202-9_30.

[24] R. Kalfarisi, Z.Y. Wu, K. Soh, Crack detection and segmentation using deep learning with 3D reality mesh model for quantitative assessment and integrated visualization, J. Comput. Civ. Eng. 34 (2020) 04020010, https://doi.org/10.1061/(asce)cp.1943-5487.0000890.

[25] N. Kassotakis, V. Sarhosis, Employing non-contact sensing techniques for improving efficiency and automation in numerical modelling of existing masonry structures: a critical literature review, Structures. (2021), https://doi.org/10.1016/j.istruc.2021.03.111.

[26] N. Kassotakis, V. Sarhosis, M.V. Peppa, J. Mills, Quantifying the effect of geometric uncertainty on the structural behaviour of arches developed from direct measurement and Structure-from-Motion (SfM) photogrammetry, Eng. Struct. 230 (2021), 111710, https://doi.org/10.1016/j.engstruct.2020.111710.

[27] A.S. Kornilov, I.V. Safonov, An overview of watershed algorithm implementations in open source libraries, J. Imaging 4 (2018) 123, https://doi.org/10.3390/jimaging4100123.

[28] T.Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 2017-Janua, 2017, pp. 936–944, https://doi.org/10.1109/CVPR.2017.106.

[29] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 07-12-June, 2015, pp. 431–440, https://doi.org/10.1109/CVPR.2015.7298965.

[30] P.B. Lourenço, Computational Strategies for Masonry Structures, PhD Thesis, Delft University Press, 1996.

[31] P.B. Lourenço, Computational strategies for masonry structures: multi-scale modeling, dynamics, engineering applications and other challenges, in: Congreso de Métodos Numéricos En Ingeniería, 2013, pp. 451–472.

[32] D. Loverdos, V. Sarhosis, E. Adamopoulos, A. Drougkas, An innovative image processing-based framework for the numerical modelling of cracked masonry structures, Autom. Constr. 125 (2021), 103633, https://doi.org/10.1016/j.autcon.2021.103633.

[33] D.R. Martin, C.C. Fowlkes, J. Malik, Learning to detect natural image boundaries using brightness and texture, Adv. Neural Inf. Proces. Syst. 26 (2003) 530–549, https://doi.org/10.1109/TPAMI.2004.1273918.

[34] L. McKibbins, C. Melbourne, C. Gaillard, Masonry Arch Bridges: Condition Appraisal and Remedial Treatment (C656), 2006.

[35] R. Napolitano, B. Glisic, Methodology for diagnosing crack patterns in masonry structures using photogrammetry and distinct element modeling, Eng. Struct. 181 (2019) 519–528, https://doi.org/10.1016/j.engstruct.2018.12.036.

[36] N. Oses, F. Dornaika, A. Moujahid, Image-based delineation and classification of built heritage masonry, Remote Sens. 6 (2014) 1863–1889, https://doi.org/10.3390/rs6031863.

[37] B.M. Phares, G.A. Washer, D.D. Rolander, B.A. Graybeal, M. Moore, Routine highway bridge inspection condition documentation accuracy and reliability,

J. Bridg. Eng. 9 (2004) 403–413, https://doi.org/10.1061/(asce)1084-0702(2004)9:4(403).
[38] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, arXiv (2015) 1–8.
[39] G. Sithole, Detection of Bricks in a Masonry Wall, in: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2008, pp. 567–572.
[40] A.M. Sowden, The Maintenance of Brick and Stone Masonry Structures, 1st ed., CRC Press, London, 1990 https://doi.org/10.1201/9781003062066.
[41] B.F. Spencer, V. Hoskere, Y. Narazaki, Advances in computer vision-based civil infrastructure inspection and monitoring, Engineering 5 (2019) 199–222, https://doi.org/10.1016/j.eng.2018.11.030.
[42] G. Stockdale, Y. Yuan, G. Milani, The behavior mapping of masonry arches subjected to lumped deformations, Constr. Build. Mater. 319 (2022), https://doi.org/10.1016/j.conbuildmat.2021.126069.
[43] S. Tiberti, G. Milani, 2D pixel homogenized limit analysis of non-periodic masonry walls, Comput. Struct. 219 (2019) 16–57, https://doi.org/10.1016/j.compstruc.2019.04.002.
[44] S. Tiberti, G. Milani, 3D homogenized limit analysis of non-periodic multi-leaf masonry walls, Comput. Struct. 234 (2020), 106253, https://doi.org/10.1016/j.compstruc.2020.106253.
[45] S. Tiberti, N. Grillanda, V. Mallardo, G. Milani, A Genetic Algorithm adaptive homogeneous approach for evaluating settlement-induced cracks in masonry walls, Eng. Struct. 221 (2020), 111073, https://doi.org/10.1016/j.engstruct.2020.111073.
[46] E. Valero, F. Bosché, A. Forster, Automatic segmentation of 3D point clouds of rubble masonry walls, and its application to building surveying, repair and maintenance, Autom. Constr. 96 (2018) 29–39, https://doi.org/10.1016/j.autcon.2018.08.018.
[47] E. Valero, A. Forster, F. Bosché, E. Hyslop, L. Wilson, A. Turmel, Automated defect detection and classification in ashlar masonry walls using machine learning, Autom. Constr. 106 (2019), 102846, https://doi.org/10.1016/j.autcon.2019.102846.
[48] N. Wang, X. Zhao, P. Zhao, Y. Zhang, Z. Zou, J. Ou, Automatic damage detection of historic masonry buildings based on mobile deep learning, Autom. Constr. 103 (2019) 53–66, https://doi.org/10.1016/j.autcon.2019.03.003.