Structures 81 (2025) 110371

Contents lists available at ScienceDirect

Structures

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

AI-assisted 3D model generation for discontinuum-based analysis of URM buildings

Peter Griesbach [a], Andrei Farcasiu [a], Rhea Wilson [a], Qipei Mei [b], Bora Pulatsu [a,*]

[a] Department of Civil and Environmental Engineering, Carleton University, Ottawa, Ontario K1S 5B6, Canada
[b] Department of Civil and Environmental Engineering, University of Alberta, Edmonton, Alberta T6G 2W2, Canada

ARTICLE INFO

Keywords:
Vision-based pattern recognition
Masonry buildings
URM
Computational modeling

ABSTRACT

This research presents a novel framework for the discontinuum-based analysis of unreinforced masonry (URM) buildings, integrating artificial intelligence (AI) assisted object detection and segmentation into the structural analysis workflow. Recent advancements in machine learning, particularly Convolutional Neural Networks (CNNs), are utilized to digitize a URM building, and the most relevant construction quality parameters (e.g., block size and staggering ratio) are automatically captured from the vision-based data. The collected information is used to inform the implemented block generation algorithm, which places masonry units into wall sections that are not documented (or poorly detected) due to various on-site obstructions. Then, the digital replica of the building is turned into an evidence-based computational model using the discrete element method (DEM), where detected masonry units are represented as discrete rigid blocks in a fully discontinuous setting. The mechanical interaction between rigid blocks is predicted using a cohesive frictional contact model to capture the unit-mortar interface (bond) behavior. The AI-assisted DEM-based model is later used to perform nonlinear pushover analysis to predict the seismic behavior and collapse mechanism of the analyzed building. Hence, it is demonstrated that the proposed approach offers a great potential for discontinuum-based analysis of URM buildings by eliminating the time-consuming model generation process and providing the most representative construction quality features in the structural analysis.

1. Introduction

Technological advancements have significantly influenced the field of structural engineering, as evidenced by the transition from early manual tools to the more recent emergence of computational modeling, which has shifted engineering practices. Integrating artificial intelligence (AI) into design and analysis has become a viable option in structural assessment and health monitoring. However, adopting new technologies often requires time for validation, acceptance, and integration into established workflows. The successful incorporation of AI depends not only on its technical capabilities but also on a careful evaluation of its practical value and limitations. Ensuring that engineering judgment remains central to this integration is essential for maintaining the rigor and accountability of professional practice. In line with this principle, the current research proposes an engineering-driven framework for integrating AI into the structural analysis of unreinforced masonry (URM) buildings. Specifically, the framework leverages convolutional neural networks (CNNs) to automate the generation of 3D discrete block models from vision-based data. The approach aims to streamline a traditionally manual and time-consuming step in the structural engineering workflow, supporting discontinuum-based computational analysis while keeping engineering judgment at its core.

URM buildings are found globally and span a wide range of historical periods, resulting in substantial variability in their design. Factors such as the availability of local materials, geographic context, and mason skill levels contribute to differences in masonry patterns. The structural behavior of unreinforced masonry is strongly influenced by the arrangement of its units, especially the alignment between courses and the connectivity between structural components. Accurately capturing these patterns is therefore essential when using computational models that explicitly represent individual units, such as discontinuum-based approaches. However, reproducing URM structures at a one-to-one scale in a digital environment is often time-consuming and further complicated by site conditions.

To this end, recent advancements in AI, particularly machine learning and computer vision, have enabled the development of

* Corresponding author.
E-mail address: bora.pulatsu@carleton.ca (B. Pulatsu).

https://doi.org/10.1016/j.istruc.2025.110371
Received 20 June 2025; Received in revised form 28 August 2025; Accepted 2 October 2025
Available online 7 October 2025
2352-0124/© 2025 The Author(s). Published by Elsevier Ltd on behalf of Institution of Structural Engineers. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

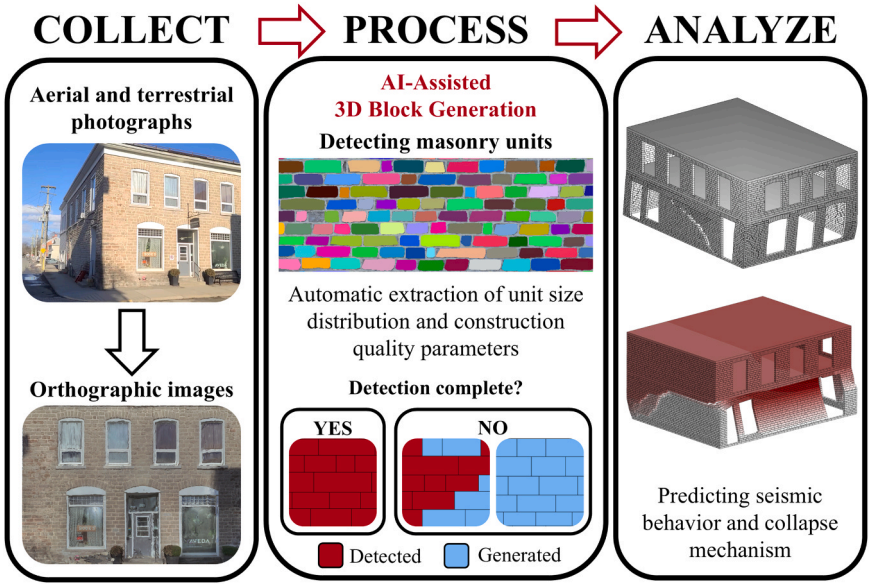

Fig. 1. Proposed AI-assisted three-dimensional discrete block generation framework.

alternative approaches to generate the geometry of URM structures in the digital environment. Researchers have begun exploring CNNs and other advanced techniques to extract masonry patterns from image-based data. These approaches offer a more efficient method of producing digital representations while improving their accuracy and detail. For example, Loverdos and Sarhosis (2023) proposed a workflow that processes orthorectified images to generate geometric digital twins for documentation, inspection, and structural assessment [1]. Their method uses CNNs to identify the locations of masonry units and cracks, converting them into simplified line drawings suitable for subsequent computational analysis. Similarly, Vandenabeele et al. (2024) applied 2D segmentation to a large-scale survey of the Basilica of St. Anthony in Padua, Italy, to identify construction phases and repair interventions across both regular and irregular masonry patterns [2]. However, the authors noted that the quality of the initial survey images significantly affected the segmentation accuracy, with image distortion and foreground obstructions posing challenges. Galanakis et al. (2024) evaluated the Segment Anything Model (SAM) across various masonry types and imaging conditions, highlighting its potential for object detection and damage analysis [3]. Nonetheless, they noted that low color contrast between mortar and masonry units, common in historical structures, remains a significant challenge for reliable segmentation.

AI-based segmentation offers promising solutions, capturing the geometry of URM buildings more efficiently; however, there are still situations where these techniques may fall short. In many older URM structures, accurate one-to-one representations of the masonry pattern are often infeasible due to obstructions such as plaster layers or limited site accessibility. In such cases, alternative block generation strategies must be considered. Rather than replicating the exact position, size, and arrangement of masonry units, several virtual masonry pattern generation tools in the literature adopt probabilistic approaches to create representative masonry patterns based on statistical properties. These tools typically rely on probabilistic approaches, assigning masonry unit dimensions based on statistical distributions to simulate realistic patterns. Zhang et al. (2018) developed a two-dimensional generator in which the height and width of units are determined by random functions [4]. The algorithm initially places rectangular blocks and then applies Voronoi splitting and geometric distortion to selected units to produce more irregular masonry typologies. Later, Pereira et al. (2023) proposed a method that fills a 3D volume using block size and frequency statistics extracted from a representative sample area for three-dimensional, multi-leaf, nonperiodic masonry [5]. The approach avoids vertical alignment of head joints and incorporates features such as through stones. More recently, Szabó et al. (2024) introduced a two-dimensional coursed masonry pattern generator incorporating block arrangement rules based on geometric quality indexes [6]. The generator extracts input parameters, such as block height, aspect ratio (defined by statistical distributions), overlap factor, and block set length, from a survey of a small sample section of the structure. These parameters guide the generation of a masonry pattern that reflects the construction quality of the surveyed area and is naturally related to structural behavior, making it suitable for both detailed and simplified micro-modeling approaches.

Despite recent advancements in AI-based segmentation methods and probabilistic block generation techniques, a gap in the current literature remains between these approaches. AI-based approaches offer efficiency and accuracy in detecting masonry patterns from images, but often fail under real-world conditions where parts of the structure may be obscured by plaster, vegetation, or poor lighting. On the other hand, statistical generation methods are more flexible in such cases but cannot capture the actual geometry of existing structures. This highlights the need for a combined approach that leverages the strengths of both strategies, utilizing AI to detect as much of the masonry pattern as possible and filling in the remaining areas with statistically generated blocks based on engineering logic and construction quality indexes.

To address this gap, the current research proposes an AI-assisted, engineering-driven framework for automatically generating 3D discrete block models of unreinforced masonry structures using image-based data. The workflow is designed specifically for rectangular coursed masonry. An overview of the framework is provided in Fig. 1, which outlines the primary inputs, processing workflow, and outputs of the analyses. It begins with orthographic images, which are processed using a trained YOLO-based convolutional neural network to detect individual masonry units. The YOLO models, trained and evaluated using the Ultralytics Python library, are prepared using datasets organized using the web-based Roboflow toolkit [7]. The detected units are converted into discrete blocks, while undetectable regions are filled using the developed generator. The proposed tool extracts geometric properties from the AI-identified sections and utilizes them to generate a masonry pattern with consistent quality throughout the entire model. By combining AI-assisted segmentation with engineering-informed generation, the proposed framework offers a practical tool for creating reliable, discrete block models, even when parts of the structure are obscured or complete image data is unavailable.

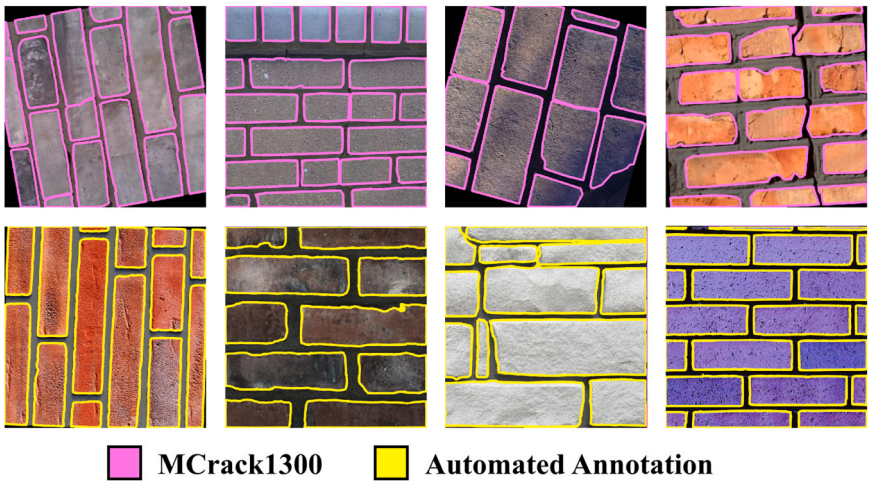

Fig. 2. Annotated images included in the "MCrack1300" dataset and the new images annotated using YOLO and SAM.

2. Digitization of the discontinuous medium

2.1. Convolutional neural networks and CNN-based documentation strategies

The digitization of masonry façades typically involves manually re-creating their geometries using CAD, BIM, or other 3D drawing tools. While these methods are accurate because they rely on the modeler's ability to identify masonry units, they are often inefficient due to the significant time required. Alternative image-processing techniques may be used to detect masonry units and segment the masonry constituents, typically in a semi-automatic fashion. As discussed in a recent study, classic rule-based image processing tools show good performance if the analyzed image is clean and taken in good lighting with high contrast between the units and mortar; otherwise, various factors (e.g., weathering, vegetation growth, etc.) could all interfere and negatively influence the accuracy [8]. However, advancements in machine learning, particularly CNNs, offer a way to automate digitization tasks. CNNs are widely used machine learning implementations for computer vision-based tasks, as they excel in classification, detection, and segmentation tasks with high efficiency. With sufficient training, they can be used to effectively detect and delineate objects of interest within masonry façades [9]. Briefly, CNNs perform object detection tasks by drawing bounding boxes over objects of interest to provide estimates of their locations and spatial boundaries. Subsequently, to provide more granular estimations of spatial boundaries that consider the shape of the objects themselves, instance segmentation is performed to provide vector polygons of the shape of objects [10]. Numerous object detection and instance segmentation CNNs in the literature have proven to be effective in performing these tasks [11], with one such model being YOLOv8, which detects and delineates objects in real-time via a regression-based approach, processing images in a single pass-through. YOLOv8 consists of backbone, head, and neck layers, which extract feature maps, optimize them, and perform detections, respectively [12]. An advantage of using YOLO over other frameworks is its accessibility, due to the availability of intuitive code implementations, such as the Python functions found within the Ultralytics library for training and validation, as well as the wide variety of dataset processing suites, including Roboflow, for efficient dataset organization [13].

Training YOLOv8 models to detect specific objects requires extensive datasets containing annotated images of the target objects. Given the wide variation in masonry's visual characteristics, datasets should be both comprehensive and diverse, containing photographs of masonry façades with bricks of different materials, colors, and sizes. This ensures the models can detect and segment bricks across different façades with high confidence. However, obtaining sufficiently large and diverse photographic datasets can be time-consuming, especially when open-access datasets are not readily available in the literature. Augmentation can be performed to supplement existing photographs in the dataset using various post-processing techniques (*i.e.*, rotating, mirroring, and adjusting the hue, saturation, and brightness of the images) to artificially expand the dataset [14,15]. Post-processing techniques are used to improve a YOLO model's ability to detect target objects in more unconventional configurations, which proves helpful in detecting more irregular masonry assemblies and voussoirs. Hyperparameter tuning is another method for improving the accuracy of models by adjusting high-level training parameters, such as batch size, learning rate, and weight decay, to modify the training process itself to yield improvements in accuracy. However, for this study, the hyperparameters are left as their default values as prescribed by the Ultralytics framework. The focus is on proposing a novel framework that integrates computer vision into structural analysis for improved behavior prediction.

2.2. Masonry unit detection through YOLO and SAM

The implementation of YOLOv8 in this study utilizes "YOLO-Patch-Based-Inference," a Python library that facilitates detection and segmentation tasks on high-resolution images using a patch-based approach, in which the image is split into smaller sections of equal size to ensure all objects are detected and appropriately delineated. A dataset containing 16,275 images (640 $\times$640) is compiled from augmented variations of a pre-existing masonry detection dataset, known as "MCrack1300", and other additional royalty-free images. The original MCrack1300 dataset contains 1300 images of masonry façades with annotations for bricks, broken bricks, and cracks, as shown in Fig. 2 [16]. This dataset comprises a collection of annotated images of bricks and cracks from existing datasets in the literature, and additional photographs taken of masonry buildings located within the vicinity of the University of Birmingham campus.

The annotations for the obtained stock images are prepared using an automated workflow in which bounding boxes of the bricks are detected using a YOLO model trained solely on MCrack1300. Afterward, Ultralytics' automated annotation pipeline, powered by SAM, is employed to automatically generate the segmentation masks for the bricks within the bounding boxes (see Fig. 2). SAM differs from YOLO as it is solely used for instance segmentation, and it relies on user defined prompts to identify the object to be delineated. In this case, the bounding box detections from YOLO are used to direct SAM. The annotated images are organized into a single dataset using Roboflow, a toolset for preparing and disseminating YOLO datasets [7]. This tool assigns images for

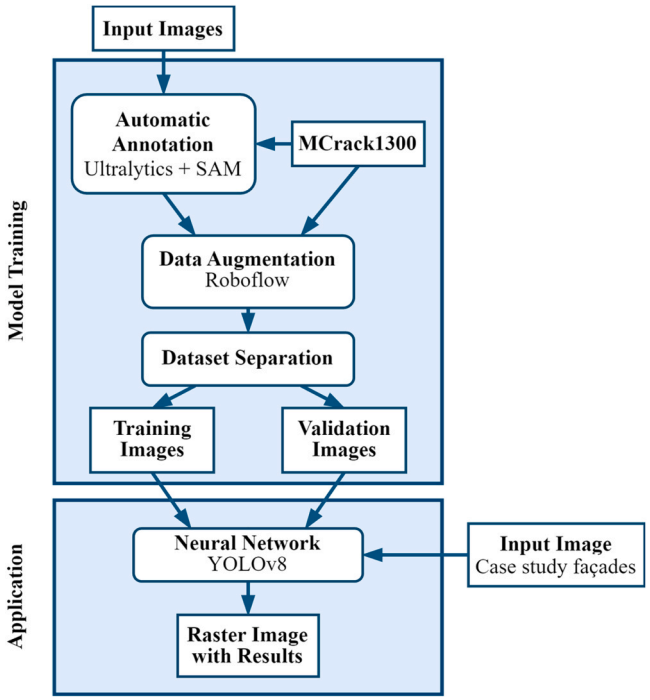

Fig. 3. Overview of the masonry unit detection workflow.

training, testing, or validating YOLO models. For YOLO models, standard practice allocates 80 % for training, 10 % for validation, and 10 % for testing [17]. However, the allocation of images for each task is at the user's discretion. As testing is conducted directly on the orthoimages generated for this study, 80 % of images are allocated for training and the remaining 20 % for validation, as this is also recommended in the literature [18]. Augmentation is also performed using Roboflow, where the images are mirrored and rotated at varying angles to improve the dataset's confidence in detecting bricks and voussoirs.

Once the dataset is finalized, it is exported with Ultralytics, a Python library that facilitates the training and evaluation of YOLO models [19]. YOLO models are trained by processing datasets through "boilerplate" models trained on generic data, which are designed for use as starting points for new models. For this study, the training of the model is completed within an Anaconda environment hosted on an Ubuntu server equipped with a Nvidia A100 graphics processing unit. An overview of the masonry unit detection workflow, including the training and application of the adopted YOLO model, is presented in Fig. 3.

The number of pass-throughs of the dataset performed during training, known as epochs, is a significant factor in determining the accuracy of the YOLO model. For each epoch, all of the annotations contained within the dataset are processed by the model, in which it learns how to detect the target objects in images that are not part of the dataset. The respective allocated images from the dataset are processed through Ultralytics, where, between each epoch, the model learns how to detect the bricks by adjusting its weights accordingly. Upon completion of each epoch, the accuracy of the adjustments is assessed by detecting the bricks in the allocated validation images and comparing the results with the ground truth data, which, in this case, are the hand-drawn annotations set aside for validation. Then, the results of the validation are output to a CSV file. Numerous performance metrics for the YOLO model are provided in this file, with two examples being precision and recall, which measure the quality and quantity of positive detections compared to the hand-drawn validation annotations, respectively [20]. For the case of this study, having higher precision and recall values is ideal, as the objective of the YOLO model is to effectively digitize the masonry units found within a building's façades, without falsely detecting things that are not bricks, such as window panes. Ideally, the model should detect bricks directly, allowing it to replicate the morphology of masonry assemblies, whether through direct replication or evidence-based parametric generation.

As shown in Fig. 4a, both the precision and recall approach 90 %. The displayed precision and recall curves exhibit slightly different convergence behavior, with recall appearing to require more epochs than precision to stabilize. Precision and recall measure different aspects of training results. Both depend on the number of true positives (correct detections), but precision divides this by the total of true and false positives. In contrast, recall divides it by the total of true positives and false negatives. False positives are incorrect detections classified as correct, whereas false negatives are correct detections classified as incorrect. Therefore, the slower convergence of recall may suggest a lower rate of reduction in false negatives per epoch. In other words, precision represents the proportion of correctly identified bricks among all identified bricks, while recall represents the proportion of correctly identified bricks among all actual bricks. The high values of both metrics indicate that the machine learning model is capable of accurately detecting the bricks.

Another important metric is the loss function, which quantifies the difference between the expected and inferred neural network outputs. The training process is guided by the minimization of the loss function, i.e., the loss decreases as the model learns from the data. As shown in Fig. 4b, the training loss decreases sharply within the first few epochs before reaching a plateau, likely due to the YOLO model's activation patterns converging [21]. Around 90 epochs, a sudden drop to 0.60 suggests the model discovered a new activation pattern, improving performance. Validation loss shows a different trend, stabilizing near 0.80 after approximately 60 epochs, without any additional reductions. Despite differences in magnitude, both curves indicate the model effectively minimizes loss until reaching a point of stagnation. Overall, the training and validation losses have changed from 3.9 and 2.7 in epoch 0–0.8 and 0.6 in epoch 100. While there is no baseline for an ideal loss value, as it depends on the model and training data, the final loss

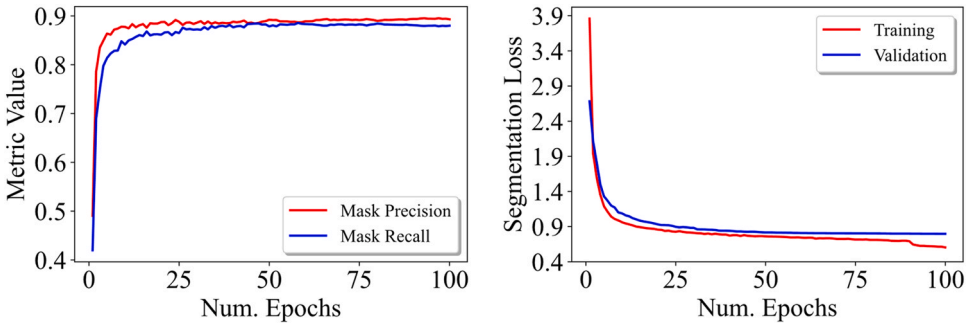

Fig. 4. Validation metrics per epoch: (a) precision and recall, and (c) segmentation loss.

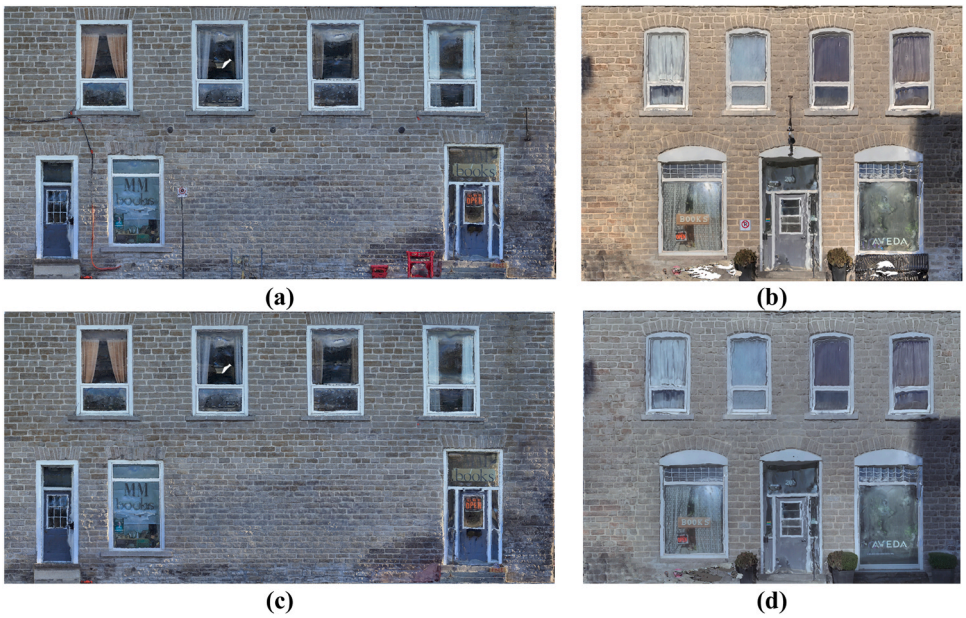

Fig. 5. Orthoimages of the Buchanan Block: (a) Unaltered northwestern façade, (b) Unaltered southwestern façade, (c) Altered northwestern façade, and (d) Altered southwestern façade.

Fig. 6. Southwestern façade masks: (a) no color grading - raw output and (b) color grading - raw output.

values are in line with similar masonry segmentation models in the literature [22], confirming the effectiveness of the trained model.

2.3. Documentation and segmentation of the case study: Buchanan Block

In this section, the trained model is used to develop a digital representation of the Buchanan Block, a historic unreinforced masonry building in Kemptville, Ontario, Canada. Over its lifetime, the Buchanan Block has housed numerous businesses. Built in the 1860s, this stone masonry building is a definitive example of Georgian-style architecture, characterized by its distinct stone masonry and a combination of flat and arched lintels.

To digitally represent the structure, orthographically corrected images of the façade are required to capture the dimensions of the masonry units. These orthoimages are generated using Agisoft Metashape, a photogrammetry software that processes aerial imagery captured by a DJI Mini 3 Pro drone and supplementary terrestrial photographs. The resulting textured photogrammetric mesh provides orthographic views of the southwestern and northwestern façades, shown in Figs. 5a and 5b. Each façade yields two orthoimages, which serve as input for the masonry segmentation workflow. The images capture detailed and orthogonally aligned views of the façades; however, they also contain numerous obstructions such as signs, cables, and outdoor furniture, which limit the visibility of masonry units. Additionally, the lighting in the photographs results in color variation between the two façades. These two issues may hinder detection and segmentation significantly, as obstructions prevent the proper delineation of masonry units, and color variations may lead to uneven detections across both orthoimages.

It is worth noting that the performance of the adopted object detection and segmentation framework is primarily related to the input image quality and resolution. As a result, simple image processing tasks are performed to improve the input vision-based data (orthoimage). The actions performed are not vital for the proposed framework, but are preferable to get better output data. While manual post-processing techniques can be used to remove obstructions and adjust colors, open-source tools with advanced algorithms and machine learning offer more efficient solutions by automating these tasks. In this study, obstructing objects in orthoimages are removed using an AI-based generative fill extension in Krita, an open-source image editing tool. Fig. 5c shows the modified northwestern orthoimage with obstructions removed.

Brick detection and segmentation are performed with an in-house application that provides a streamlined workflow to perform inference tasks on orthoimages of masonry façades using trained YOLO models. The application generates raster images with segmentation masks of masonry units, each displayed in distinct colors for visual clarity. To

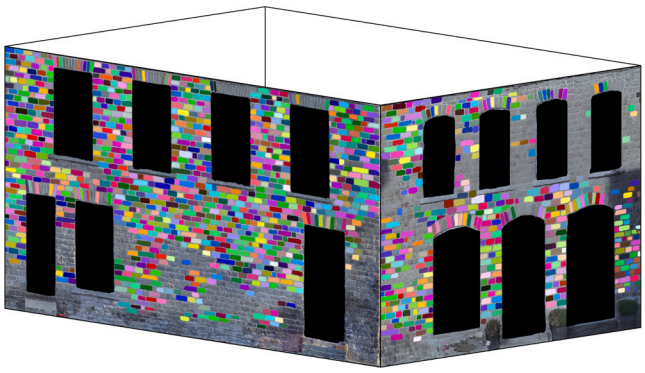

Fig. 7. Northwestern and southwestern façade masks.

reduce false detections of non-brick objects, masks are superimposed over façade openings. The accuracy of the generated masks is assessed through confidence ratios, which quantify the YOLO model's certainty that the segmented object matches with an object class it is trained to detect and delineate [24]. The application allows for the adjustment of a threshold which filters out or includes masks that fall below or above the set value, respectively. As such, the threshold is set as 80 % for this study, which yields a significant amount of brick masks that match the source orthoimage, signifying that the YOLO model is capable of performing its function to a considerable degree of accuracy.

Fig. 6 shows the initial processing of the Buchanan Block orthoimages carried out with the application. The masonry segmentation model successfully detects the majority of bricks; however, it struggles to detect the voussoirs above the openings and with bricks in areas of low contrast between units and mortar joints. To improve detection accuracy, an automated color grading workflow is applied to the southwest façade, which exhibited lower contrast between bricks and mortar due to increased sun exposure. The workflow, facilitated by the "color-matcher" Python library, matches the colors of the southwestern façade with the adjacent façade to ensure consistency in the detection of bricks in both façades [23]. The modified version of the southwestern façade is given in Fig. 5d. The effectiveness of color grading is illustrated in Fig. 6b. After adjustment, the quality of the generated masks is improved significantly, providing a more solid sample set for the geometry generation. Fig. 7 shows the final generated masks for both the northwestern and southwestern façades.

Following visual inspection, additional intervention is employed to delete masks that do not accurately capture the building to ensure no inaccurate bricks are created during this process. To improve both the quality and quantity of the masonry unit masks, improvements can be made in the training workflow and dataset. For example, the current dataset may be expanded with training data either by adding royalty-free images or a new combination of augmentation techniques. Hyperparameter tuning may be also introduced to determine the optimal values that yield increases in accuracy. Although the AI-assisted segmentation does not detect every block in the façades, the results provide a reliable foundation for further reconstruction. To this end, important inferences are made to supplement the AI-generated output. The detected bricks serve as a reference from which undetected units can be inferred probabilistically based on consistent patterns in block size and alignment. The conversion of the AI-generated masks to discrete blocks, along with the subsequent probabilistic filling of undetected areas, is described in the following section.

3. Discrete block generation and construction quality parameters of URM walls

3.1. Automatic block generation for discontinuum-based analysis

The segmented brick and window masks are converted into discrete blocks to perform discrete block simulation. Each block is expanded to include half the mortar joint thickness, which represents brick-mortar texture via a system of discrete blocks, referred to as simplified micro-modeling [25]. This conversion is carried out through image processing in Python, using the Skimage and Numpy libraries to automatically generate discrete blocks from the segmentation masks. The block generation process begins with identifying horizontal bed joints for each wall. Typically, the mortar joints are inferred from blank regions (or white spaces) existing in the segmentation masks. The vertical positions at which blank regions occur are identified and used to define representative bed joints as horizontal lines. A skeletonization procedure is applied at the center of mortar joints while excluding irrelevant blank spaces from undetected blocks. The described procedure is illustrated in Fig. 8. Following bed joint detection, the vertical spacing between

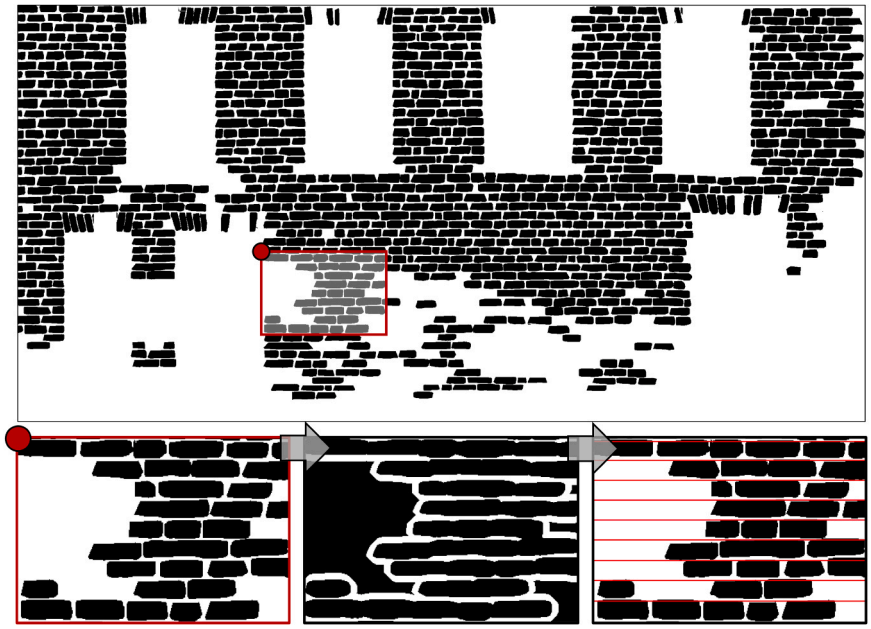

Fig. 8. Detection of bed joints from the brick mask to bed joint placing.

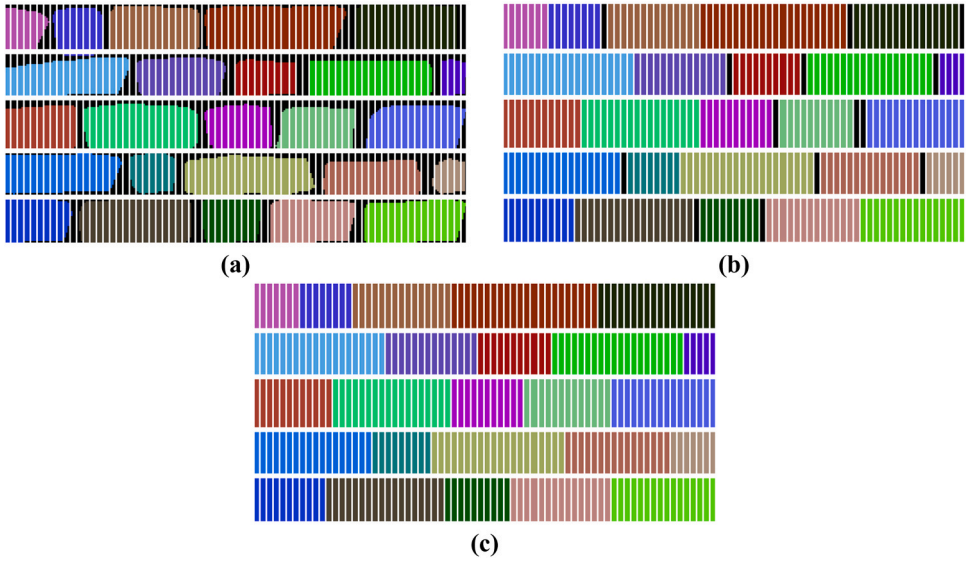

Fig. 9. Conversion of contours to discrete blocks: (a) contours with sliced overlayed, (b) contours expanded into blocks based on slices, and (c) blocks expanded into head joints.

adjacent joints is analyzed to identify outliers. If two joints are separated by a distance that exceeds twice the average row height, additional joints are inserted between them at regular intervals, approximating the average spacing. Since bed joint continuity is assumed in the current geometry preparation workflow, variations in row height across façades present a challenge. To address this, the row heights from the façade with the highest quality brick detection are used as the standard for all other walls.

With all bed joints in the wall defined as horizontal lines, individual rows of brick contours can be isolated from the segmentation mask for further analysis. Each row is partitioned into vertical slices, which are incrementally scanned. Fig. 9a shows these partitions superimposed over the detected block masks in white, where each grid cell represents a slice. These slices span the full height of a masonry course (or row) and can be adjusted by the user. The slice width controls the resolution of the resulting geometry, with smaller widths yielding more precise block length estimations at the expense of increased computational cost. The dominant feature is identified as the one occupying the largest area within this region. The slice is assigned to this dominant feature, representing a brick, a window, or a blank region (corresponding to mortar joints or undetected blocks). Once all slices in a row are evaluated, the entire façade is represented as a series of classified slices, as shown in Fig. 9b. The white outlines indicate individual slices, different colors denote different blocks (i.e., masonry units), and black regions indicate unassigned or blank spaces (Fig. 9b). For illustrative purposes, the slice width in the figure has been increased. Note that to generate a discrete block medium suitable for the selected discontinuum-based analysis, small gaps between adjacent bricks caused by the mortar head joints must be removed, which is accomplished by expanding adjacent blocks. Slices forming a continuous horizontal group less than 5 cm wide are classified as mortar head joints. These slices are reassigned to the nearest brick within the same row, as illustrated in Fig. 9c.

If windows are present in a façade, they are processed using algorithms tailored to their specific geometry. Regarding the reference building, two typologies are encountered: windows with flat and curved arch lintels. For each type, the algorithm begins with an idealized representation of the window and associated lintel, which is then scaled, transformed, and adjusted to best fit the segmented masks. For the flat arches, a bounding rectangle is first fit to the window mask. Then, a trapezoid envelope is positioned with its shorter base aligned to the window's top edge. This trapezoid is vertically subdivided to define the lintel blocks. Parameters such as lintel height, shape, and the number of voussoirs can be modified to reflect actual (*as-is*) geometrical properties. Windows with arch lintels are treated similarly, using a different geometric template adapted through the same transformation and fitting process. Finally, all façades are extruded to a user-defined thickness to generate the three-dimensional geometry required for discontinuum-based modeling.

3.2. *Extracting construction quality parameters based on the vision-based data*

The conversion of segmentation masks into discrete blocks follows the procedure outlined in the previous section. However, in cases where unit detection is incomplete or where AI-assisted detection is not possible, the missing geometry can be generated probabilistically based on the statistical properties observed in the actual wall morphology. It is important to note that the block-based digital replication of the actual URM wall can be obtained following evidence-backed or hypothetical block size and shape distributions. In addition to generating URM wall patterns, the provided statistical properties can also be used to estimate key mechanical parameters of the masonry, including shear strength, Young's modulus, shear modulus, and compressive strength [26].

In this research, two statistical parameters associated with the URM wall morphology are obtained from a portion of the wall shown previously in Section 3.1. The documented region excludes windows, lintels, and significant blank spaces. The first parameter is the horizontal offset of head joints, defined by the minimum length parameter (M_l), also referred to as the staggering ratio in this paper - originally introduced as part of the Masonry Quality Index (MQI) proposed by Borri et al. [26]. The MQI is a qualitative assessment method used to evaluate the construction quality of masonry walls. It is based on the presence or absence of critical construction features that contribute to structural integrity and behavior, such as regular unit dimensions, proper joint alignment, and effective interlocking between wall leaves, among other factors. The second parameter is the block size, which is denoted as block length (l_b) given the approximately equal height of masonry courses.

The staggering ratio (M_l) is a non-dimensional quantity defined as the ratio between the shortest path that connects two points on the wall surface by following only mortar joints and the straight-line distance between those two points. Within the MQI framework, if M_l exceeds 1.6, the courses are considered to be well-staggered. Conversely, if it falls below 1.4, the joints are aligned, which may indicate poor construction quality and a greater risk of premature failure. In the current research,

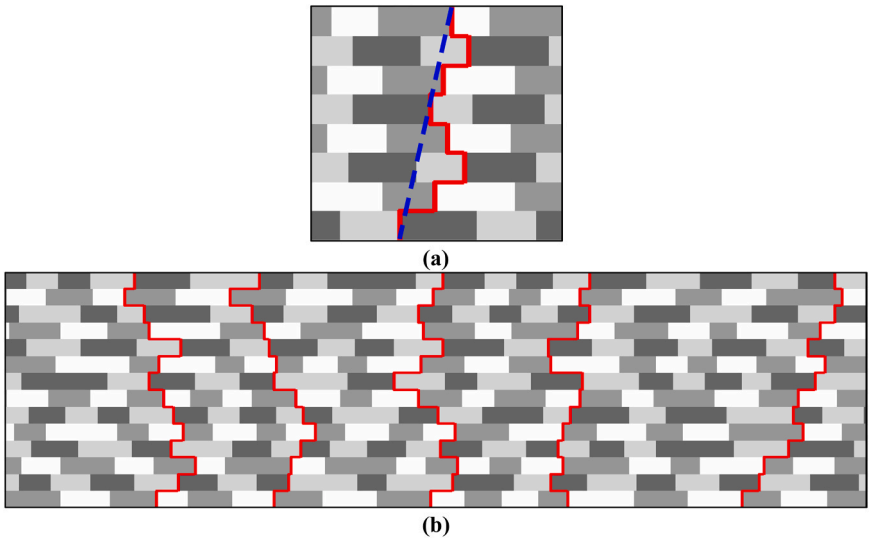

Fig. 10. (a) Illustration of staggering ratio (M_l) defined as the ratio between the shortest path that connects two points on the wall surface by following only mortar joints (continuous red line) and the straight-line distance between those two points (blue dashed line); (b) Measurements of a wall's average M_l value using multiple starting locations.

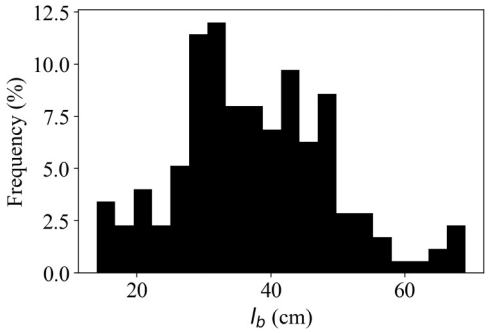

Fig. 11. Block length distribution regarding the actual wall morphology.

the staggering ratio is used to ensure that the generated wall geometry exhibits similar in-plane performance to the analyzed reference URM building. To compute the staggering ratio from the discrete block geometry, a grid-based representation is used. Head joints are identified by locating horizontally adjacent cells that belong to different blocks. A head joint at the top of the selected region is chosen as the starting point. The horizontal distance to the head joints in the next course is measured, and the head joint with the smaller of the two distances is selected as the next starting point. This process continues through each course until bottom of the wall is reached. The final M_l is calculated as the ratio of the total length of the joint-following path to the direct vertical distance between the start and end points (Fig. 10a). This process is repeated for multiple starting location across the top course to account for variability and an average M_l value is calculated for the wall as shown in Fig. 10b.

The block size distribution is a relevant parameter related to the structural performance of a URM wall, and it is also a key parameter in the MQI analysis. According to the structural assessment framework of MQI, the unit size criterion is fulfilled if more than 50 % of elements have a large dimension greater than 40 cm. The criterion is not fulfilled if more than 50 % are smaller than 20 cm. Throughout this research, the same wall region used to compute M_l is considered to document the block length distribution. Not that the analysis does not include block height, as it remains constant within each course. The length of each unit is calculated by summing the number of grid cells associated with each unique block identifier, discussed earlier. Units intersecting the wall boundary are excluded to avoid skewing the dataset with partial blocks. The resulting block length distribution is shown in Fig. 11.

3.3. Generating URM walls matching statistically the morphological features of the reference building

As discussed in previous sections, the proposed AI-assisted framework offers automatic documentation of masonry units, extracting

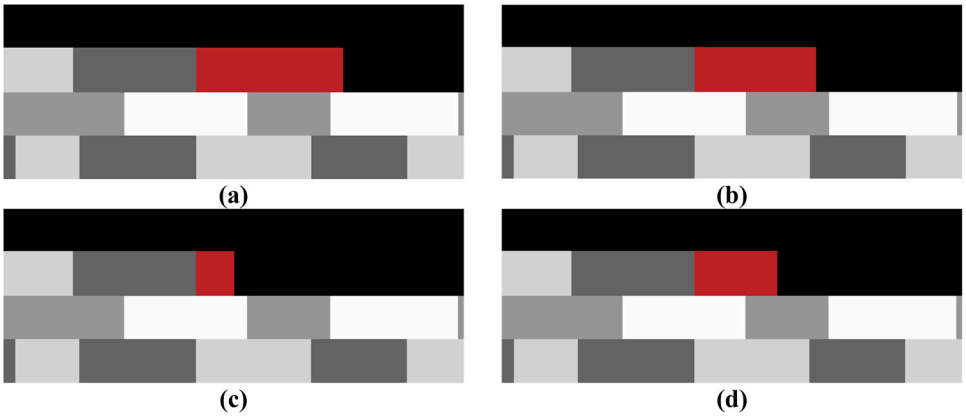

Fig. 12. Candidate blocks (denoted via red color) placed into the partially empty wall space under consideration.

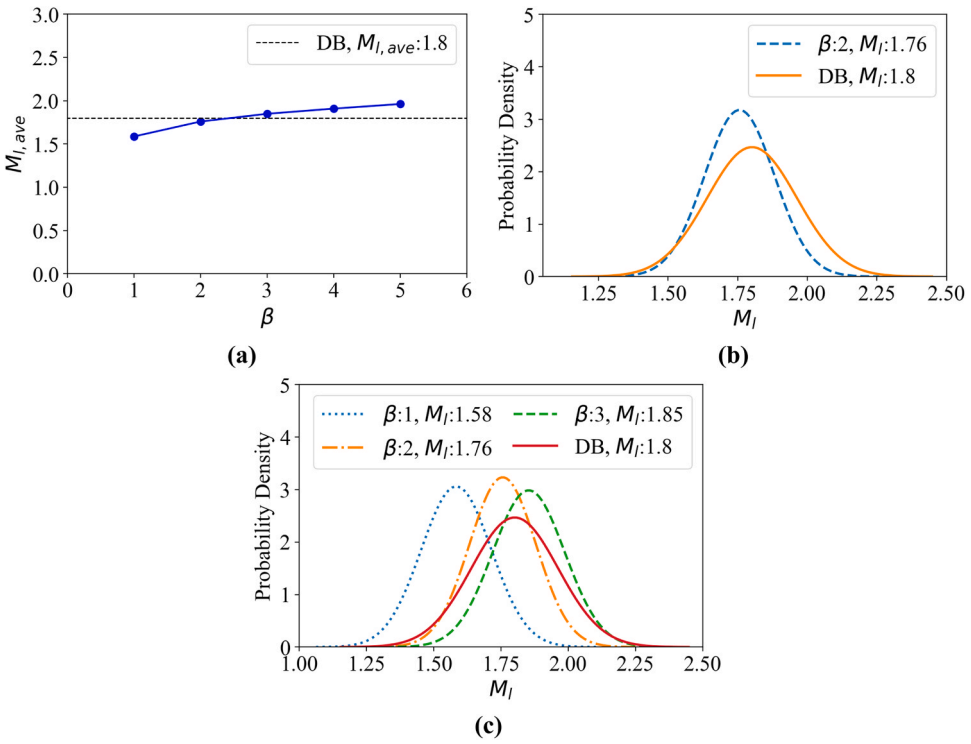

Fig. 13. (a) Relationship between the predefined number of candidate blocks (β) and the average value of the staggering ratio ($M_{l,ave}$); (b) Distribution of M_l ratios measured from a wall section using detected blocks (DB) and a statistically generated wall where β= 2; and (c) representation of M_l distributions corresponding to different the number of candidate blocks (β).

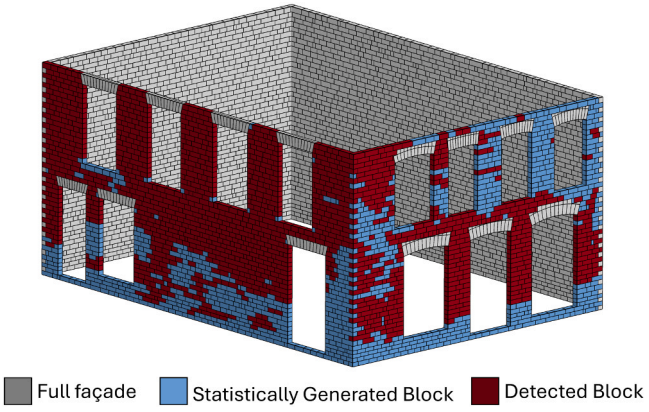

Fig. 14. AI-assisted discrete block geometry of Buchanan Block.

several construction features and their statistical variations. In cases where no vision-based data is available, the documented statistical parameters may be beneficial in generating representative wall cross-sections. In other words, if a detected block mask is unavailable, data-driven representative URM walls can be created and utilized in structural analysis. In this section, the proposed workflow is presented to generate façades of the reference building where no visual information is available due to physical obstructions.

The implemented algorithm starts with placing blocks sequentially row by row, from left to right. In a wall where some portion of the masonry units have already been detected, each block's height is set to the height of the row determined earlier through the proposed framework. The length of the block being placed is determined based on the location of head joints in the adjacent row below to the current one and relies on the provided block length distribution. Overall, a potential block being placed (referred to as a *candidate block*) is sampled from the pre-determined block length distribution. Fig. 12a-d shows a different selection of candidate blocks. From the given candidate block set, the one that creates the greatest horizontal offset between the newly generated head joint on its right and the nearest head joint in the vertically adjacent layer is selected. Given four candidate blocks in Fig. 12, the selected block is depicted in Fig. 12d following the implemented strategy.

The number of candidate blocks considered for each block placement directly influences the staggering ratio of the statistically generated wall sections. Therefore, the proposed framework adjusts the number of candidate blocks selected (denoted as β) to match the desired M_l value. The ideal candidate block size is selected from a range of candidates by choosing the one that best matches the M_l of the detected wall. Fig. 13a shows the relationship between the number of candidate blocks and the corresponding average staggering ratio ($M_{l,ave}$). The results show a converging trend of $M_{l,ave}$ when the group of candidate blocks gets larger. Users can pick the candidate block set which provides a staggering ratio that aligns with the required construction morphology (denoted via a dashed line and titled as $M_{l,ave}$ detected blocks (DB) – see Fig. 13a). Here, a group of two candidate blocks provides a satisfactorily close M_l distribution obtained from the reference building, shown in Fig. 13b. The outcomes of various β on the M_l distribution is provided in the Fig. 13c.

The procedure for creating a wall with no detected blocks (or masonry units) by using only statistical wall data is similar to the methodology when detected blocks are present. A key difference is that there are no predefined rows within the wall. If the façade is isolated from the load-bearing wall system, the average block height can be provided as an input, acting as a target row height when the wall is divided into equally sized rows. In the reference building, the row heights in the obscured walls can be determined using information from the neighboring façades. Once the masonry course positions are determined, the block placement algorithm is executed and URM walls are created using the statistical data describing the wall (l_b and M_l measured from adjacent

Table 1
Proposed procedure to generate discrete block medium, provided as pseudo-code.

Input: *SM* (Segmentation mask), *WM* (Window mask), *D* (Wall dimensions), *SG* (Grid-cell length), β (Candidate block set size), HJ_{Max} (Maximum headjoint thickness)

1: $BJH \leftarrow$ Bed joint heights determined based on SM – (Figure 8)

2: $B_{Grid} \leftarrow$ 2D array of zeros [size(BJH) – 1 × round(D_{Length}/SG)] // *used to represent masonry segmentation as discrete geometry*

3: **For** each *grid-cell* in B_{Grid} **do**:

  $AoI \leftarrow$ Area of interest in *SM* corresponding to *grid-cell* location in B_{Grid}

  Modify value of the *grid-cell* to Block ID most represented in *AoI* – (Figure 9b)

4: // *For each blank space in* B_{Grid} *a submatrix* B_{Blank} *is defined*

For each B_{Blank} in B_{Grid} **do**:

  If size(B_{Blank}) < (HJ_{Max} / *SG*) **then**:

    Fill B_{Blank} region in B_{Grid} by expanding adjacent B_{Grid} entries – (Figure 9c)

5: // $grid\text{-}cell_{Blank}$ *refers to entries with no assigned block ID*

If B_{Grid} contains $grid\text{-}cell_{Blank}$ **then**:

  $BL \leftarrow$ The detected block lengths as a function of B_{Grid} – (Figure 11)

  For each $grid\text{-}cell_{Blank}$ in B_{Grid} **do**:

    $CB_{List} \leftarrow$ Candidate blocks - randomly selected from *BL* according to β – (Figure 12)

    CB ← Block from CB_{List}, maximizing the distance between its rightward head joint and the nearest head joint in the adjacent row // *random block selected in first row* – (Figure 12d)

    CB inserted into B_{Grid} by assigning a new Block ID to $grid\text{-}cell_{Blank}$ and neighbouring blank *grid-cells* // *quantity of grid-cells modified based on size of CB*

6: Generate *DB* via B_{Grid} where *grid-cells* with the same block ID represent one block (associated with a masonry unit) // *Block height scaled based on BJH*
// *Block length scaled based on the number of assigned grid-cells and SG*

7: Lintel and opening template scaled based on *WM* and then imposed onto *DB*

Output: *DB* (Discrete block medium) – (Figure 14)

façades). The resulting geometry can be seen in Fig. 14, where red blocks are "detected", blue blocks are "statistically generated" regions, and gray façades are entirely statistically generated (denoted as "Full façade"). The full discrete block generation procedure is outlined in Table 1, summarizing the key steps of the algorithm.

4. Evidence-based discontinuum-based analysis

This section outlines the state-of-the-art application of the proposed AI-assisted computational modeling framework using the discrete element method (DEM). First, the mathematical background of the DEM is discussed briefly, and then its application to the generated discrete block medium is demonstrated.

4.1. An overview of DEM

The computational modeling of unreinforced stone and brickwork masonry structures can be performed following continuum and discontinuum-based analysis. Traditionally, the former approach represents masonry composite as a homogenous medium without directly addressing the morphological features of masonry construction. Conversely, discontinuum-based analysis explicitly captures cracking, which develops and localizes primarily along weak mortar joints, by adopting a discrete-block medium that closely follows the actual wall morphology. The present research utilizes the discrete element method, falling into the category of discontinuum-based analysis, to simulate the reference building. As presented earlier, the block-based representation of the building is generated using the proposed AI-assisted novel framework.

The adopted computational modeling strategy, pioneered by Cundall [27], relies on the mechanical interaction between the adjacent rigid blocks considered at point contacts (denoted as sub-contacts). An explicit solution scheme is utilized to integrate the equations of motion for rigid blocks, where translational and rotational block velocities are predicted at each step. Accordingly, block positions are updated, and relative sub-contact displacements are computed. At each point contact, three orthogonal springs are defined (one in the normal and two in the shear direction), illustrated in Fig. 15a. The cohesive bond behavior is predicted along the contact plane following the Mohr-Coulomb joint model with tension and compression cut-off. Fig. 15a shows a

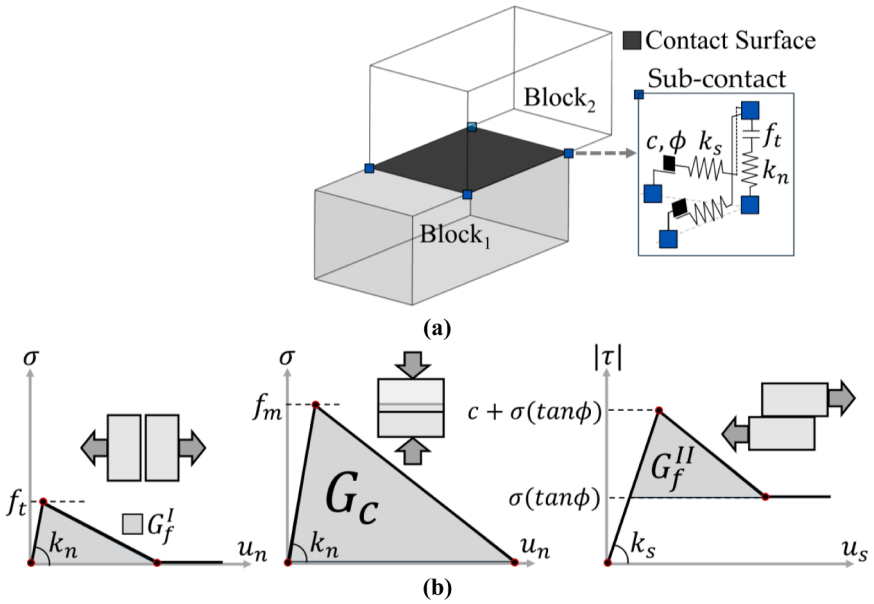

Fig. 15. (a) Illustration of sub-contact and frictional cohesive spring model; (b) Adopted contact constitutive laws in tension, compression, and shear (from left to right).

Table 2
Defined contact properties corresponding to the mechanical characteristics of the bond.

Bond Properties			
k_n, k_s (GPa/m)	f_T, c, f_m (MPa)	ϕ (°)	G_f^I, G_f^{II}, G_c (N/m)
20, 8	0.1, 0.1, 8.0	31	3, 30, 25600

sub-contact where the bond tensile strength (f_t), cohesion (c), and friction angle (θ) are required as input parameters. Note that elastic behavior is governed by the normal (k_n) and shear (k_s) springs and nonlinear cohesive bond behavior is captured using an elasto-softening contact constitutive law. In this study, the mechanical interaction between rigid blocks is computed following the recently implemented coupled fracture-energy-based contact model, discussed comprehensively in [28]. The sub-contact normal (σ) and shear (τ) stress is calculated as a function of relative displacement between adjacent blocks that are in contact. Tension-compression behavior is simulated via bi-linear stress-displacement relationship (σ-u_n) where the ultimate sub-contact normal displacement is computed based on the defined fracture energy, denoted by G_f^I and G_c, respectively (Fig. 15b). Similarly, the bond shear behavior is replicated considering a bilinear stress-displacement relationship (τ-u_s) with a residual plateau reflecting pure frictional resistance of the contact plane (*i.e.*, $\tau = \sigma\tan\phi$) controlled by the mode-II fracture energy G_f^{II}, shown in Fig. 15b. It is worth noting that the compression failure is captured considering macro-compression strength of masonry composite, denoted as f_m (see Fig. 15b). The contact stiffness degradation is considered in unloading-reloading regimes in the normal direction as a function of post-peak strength reduction. In contrast, no stiffness degradation is considered in the shear direction. Further details and applications of the utilized contact model can be found in the reference studies [28,29].

The computed sub-contact stresses are multiplied by the associated sub-contact area to get point contact forces, which are used in the translational and rotational equations of motion together with external loads, body forces, and moments. This study only performs pushover analysis, considering the reference building where the lateral loads are incremented upon reaching equilibrium at each loading level. The quasi-static solutions are obtained by adopting Cundall's Local damping formulation, discussed in [30]. The readers are referred to the reference studies discussing the computational procedure and implementation of DEM [31,32]. Moreover, the validation of the DEM on the structural assessment and analysis of URM buildings can be found in the provided references (e.g., [29,33–41] among others). Throughout this study, DEM-based simulations are prepared using a commercial

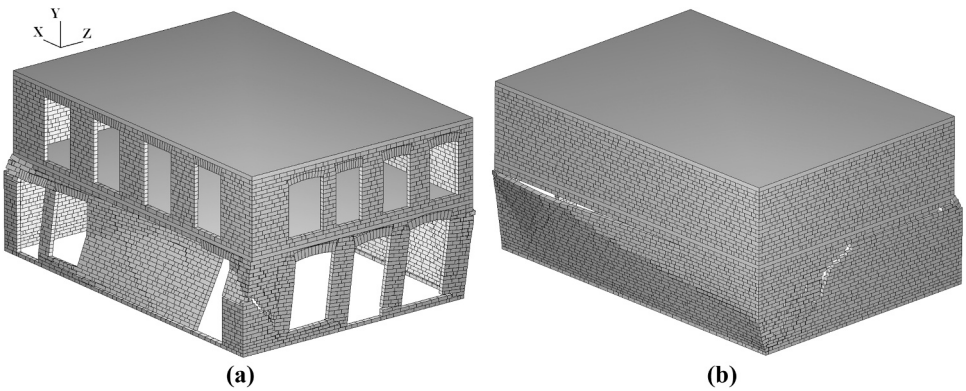

Fig. 16. The predicted collapse mechanism obtained via the proposed AI-assisted discontinuum-based analysis strategy.

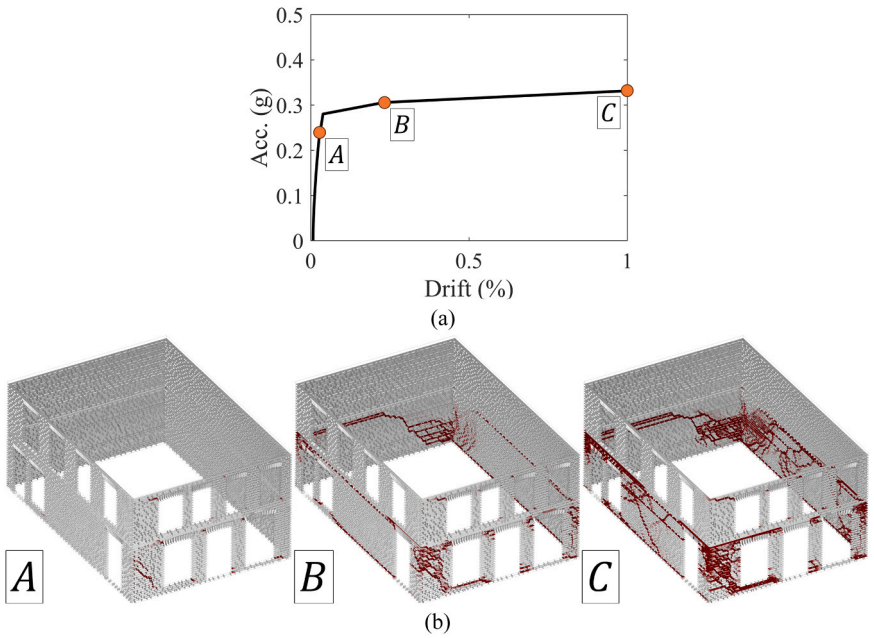

Fig. 17. Structural analysis of the reference URM building (a) Pushover curve (acceleration vs. drift) and (b) predicted crack progression at different drift ratios.

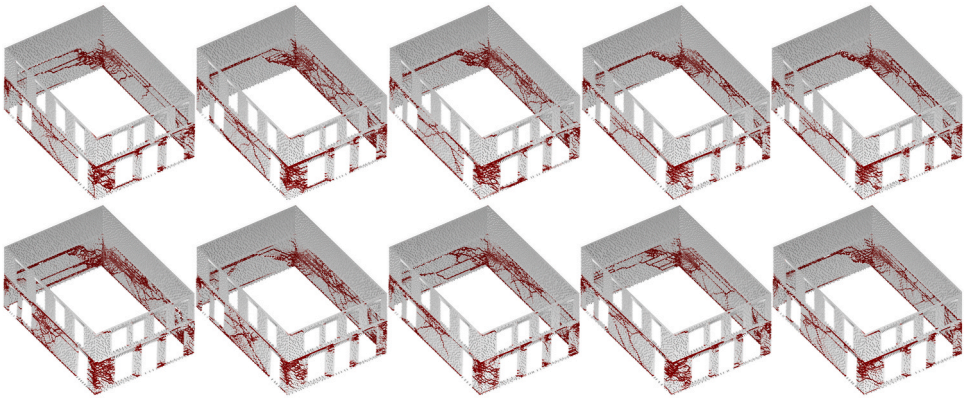

Fig. 18. Predicted crack pattern associated with the collapse mechanisms of the URM buildings generated via the presented statistical wall generation algorithm.

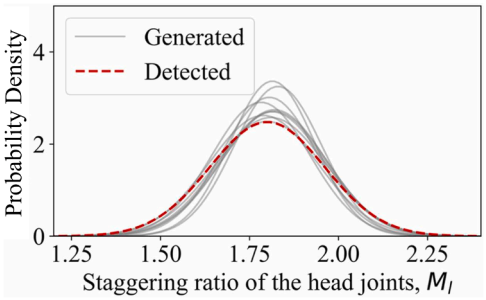

Fig. 19. Distribution of detected and generated staggering ratio of head joints.

three-dimensional discrete element code, 3DEC developed by Itasca [42].

4.2. Application of AI-assisted discontinuum-based analysis

The proposed strategy is applied through a DEM-based simulation, where the defined contact properties are taken from the relevant studies and following suggested values in the literature (e.g., [43,44]). Given the natural consequence of strong-brick weak-mortar joint combination, relatively low strength properties are considered for bond (or unit-mortar interface), presented in Table 2. The elastic modulus of masonry (E_m) is considered as 3 GPa, and the contact stiffness is predicted accordingly (i.e., $k_n = E_m/h$, h: average vertical spacing between bed-joints - $h \approx 0.15$).

After reaching equilibrium under self-weight, lateral loads are applied parallel to the building's short axis as a uniform acceleration field. The acceleration is increased in increments of 0.02 g following equilibrium, reaching a maximum slightly above 0.3 g. Fig. 16a-b illustrate the predicted collapse mechanism of the URM building from different perspectives. The results indicate the complex failure mode of the reference structure. The long façade fails due to an out-of-plane mechanism, whereas the short façade demonstrates an in-plane mechanism, specifically rocking behavior in slender piers (Fig. 16a) and a diagonal tension crack in the squat URM wall (Fig. 16b). These results demonstrate that the realistic wall morphology captured by the proposed AI-assisted approach enables robust prediction of both global and local failure mechanisms of the building.

Figs. 17a and 17b show the pushover curve (acceleration vs. lateral drift) obtained via the performed nonlinear quasi-static analysis and predicted damage progression, respectively. Note that the drift is

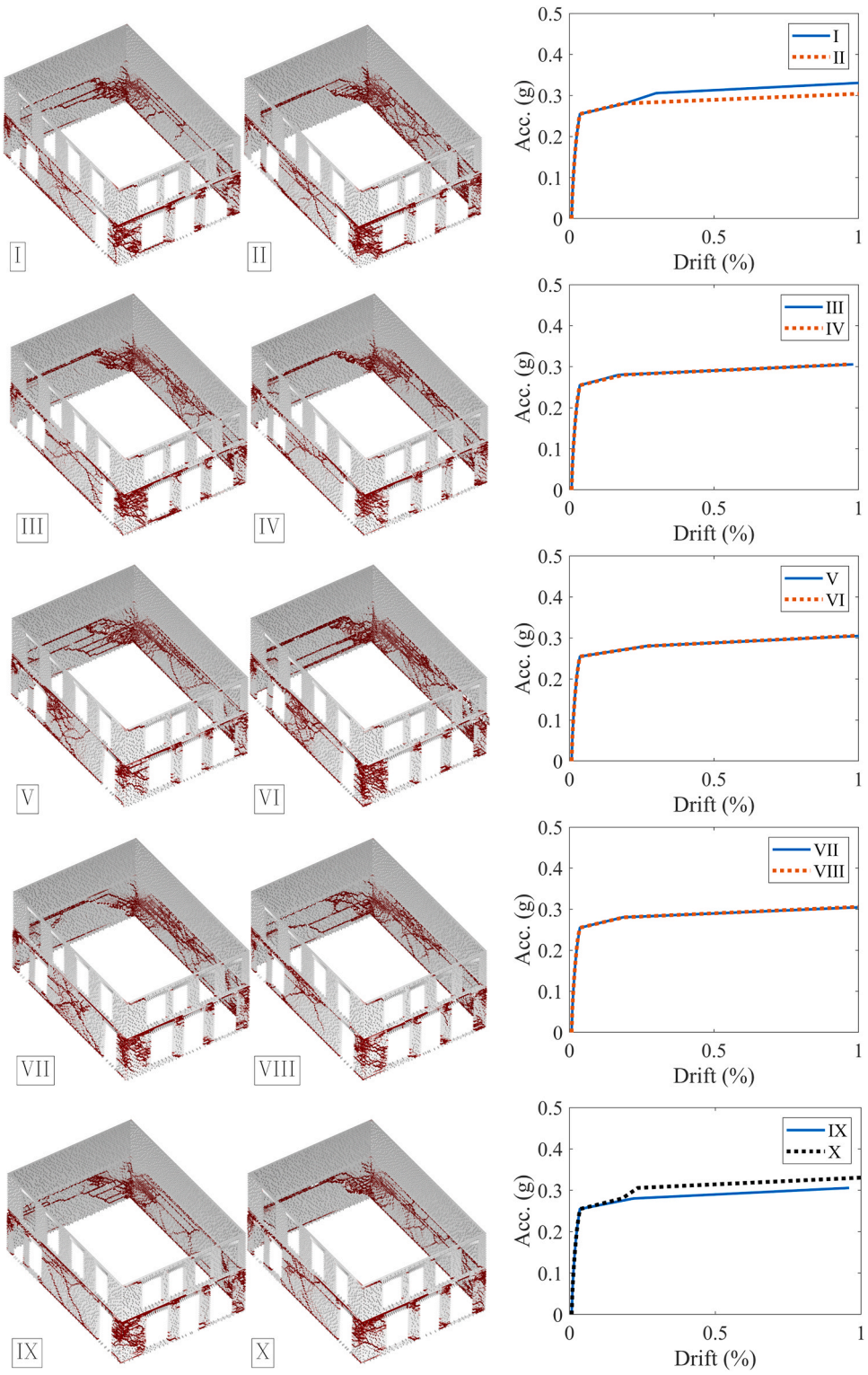

Fig. 20. The results of ten pushover analysis including collapse mechanism and associated acceleration-drift curves.

computed as the ratio of the displacement recorded at the roof to the height of the building. The provided crack patterns (indicated via red contact planes in Fig. 17b) demonstrate the complex damage evolution in the analyzed stone masonry building. The three failure stages corresponding to different lateral drift ratios are demonstrated in Fig. 17. The results highlight that upon reaching the in-plane (IP) capacity of the URM walls along the short axis of the building, excessive damage is captured through the orthogonal walls collapsing in the out-of-plane (OOP) direction. Since the developed strategy offers a three-dimensional setting, the combined IP-OOP responses are simulated.

4.3. Pushover analysis of statically generated discrete block models based on DEM

As discussed earlier, the implemented block generation algorithm aims to capture the block length and head joint staggering ratio of the

masonry buildings as closely as possible by taking advantage of recently developed AI-based object detection techniques. The proposed strategy reconstructs the undetected wall sections by statistically matching the construction morphology of the detected wall cross-sections. To further understand the necessity of including detected blocks in the discontinuum-based analysis, ten discrete-block models of the same reference building are analyzed using only the statistically generated blocks. In line with Section 4.2, pushover analysis is performed on each model considering identical boundary conditions and loading procedures. It is important to note that since the proposed workflow has a probabilistic nature, the obtained M_l distributions may show slight variations from one model to another (see Fig. 19 in the Appendix). Overall, comparable crack patterns and corresponding collapse mechanisms are obtained from the performed pushover analyses, as shown in Fig. 18. Although the damage locations (i.e., joint openings and separations) vary locally, all discrete-block models demonstrate a similar structural behavior under prescribed lateral loading conditions and collapse around the same maximum acceleration (see Fig. 20 in the Appendix). Therefore, the results suggest that the generated discrete block medium, including no detected blocks but adopting similar construction morphologies related to actual wall cross-sections, provides analogous results in terms of failure mode and capacity

This outcome is encouraging in cases where physical obstructions, site conditions, or the lack of photographic data may render the proposed segmentation workflow impractical. In such cases, a fully statistical generation of discrete block media may be considered a viable alternative. However, the advantages of a one-to-one geometric representation should be considered. Capturing the existing morphology does not add any computational burden to the generation of a discrete-block medium in the proposed framework; therefore, there is no extra cost from a computational modeling perspective. Additionally, accurately representing the as-is geometry allows for the identification of outlier conditions that may not be reflected in statistical averages. Common features in historical URM buildings, such as localized regions with small units, poor workmanship, or irregular construction, can significantly influence local failure mechanisms and may be missed when using purely statistical generation. A high-fidelity geometric model also provides a foundation for creating digital twins. Generating detailed block-level geometry using an AI-assisted workflow is an important step toward digitizing the built environment and supports future integration into digital twin initiatives. Finally, representing the current morphology enables the incorporation of accurate pre-existing conditions such as cracking, damaged units, or degraded mortar joints, which may be essential for structural health monitoring or post-earthquake damage assessments. With these considerations in mind, the choice between statistical generation and segmented one-to-one representation should be guided by the specific objectives, data availability, and desired level of fidelity for each case study.

5. Conclusions

The discontinuum-based analysis of unreinforced masonry (URM) buildings requires a block-based representation that reflects the composite nature of masonry constructions. Depending on the expected accuracy and crack resolution, either hypothetical or actual wall morphologies are used to create the digital replicas. While the actual morphology provides the most realistic prediction of collapse mechanisms, it is time-consuming and demands considerable drawing expertise. To this end, the proposed AI-assisted workflow offers an innovative approach using an advanced computer vision technique for object detection and segmentation to generate a discrete block medium. The implemented strategy accelerates the conversion of vision-based data into discrete block models and offers necessary information regarding the construction quality of the load-bearing wall system. According to the presented investigation, the following conclusions are derived.

- The proposed framework demonstrates that recent advances in AI can support DEM-based structural analysis of URM buildings by automating time-intensive steps in the workflow. The results highlight the value of a combined approach, where engineering-informed geometry generation works together with vision-based segmentation to produce high-fidelity discrete block models.
- The segmentation workflow accurately detects the majority of the stone units in the façades of the case study building, producing precise block identifications. However, gaps in the detected masks remain due to external factors such as visual obstructions and variations in lighting. To address this, a probabilistic block generator is used to reconstruct the missing sections while preserving the overall wall morphology. This study offers an overall modelling workflow that is modular in nature. Therefore, the adopted CNN can be easily substituted with alternate CNN frameworks.
- The automatically extracted geometrical and morphological properties (e.g., block size and staggering ratio distributions) support preliminary assessments and provide valuable information regarding construction quality. These parameters are also used to probabilistically fill gaps in the detected masonry wall sections, ensuring consistency with the wall's as-is geometry.
- Moreover, the documented construction quality parameters can also be used to generate complete URM walls automatically. The results of the analyses demonstrate that fully probabilistic representations of the masonry walls reproduce global and local failure mechanisms similar to those of the as-is structure. This highlights the significant potential and practical applications of the developed framework when complete visual data is unavailable. Thus, the implemented evidence-backed block generation algorithm provides an alternative solution when the visibility of the URM walls is obscured by plaster, vegetation, or poor lighting.
- Overall, the study exemplifies the importance of an AI-assisted computational modeling framework that elevates discontinuum-based analysis to the next level. The presented knowledge-guided workflow changes the conventional way of block generation for discontinuum-based analysis; hence, it opens new avenues for the fully automated DEM-based computational modeling of the current conditions of URM buildings.

Future studies will focus on enhancing the workflow to address rubble and highly irregular (non-coursed) masonry constructions. Furthermore, the adopted framework will also be coupled with crack detection and inclusion into the DEM-based computational model.

CRediT authorship contribution statement

Rhea Wilson: Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Andrei Farcasiu:** Writing – review & editing, Validation, Methodology, Investigation. **Bora Pulatsu:** Writing – review & editing, Supervision, Software, Resources, Project administration, Funding acquisition. **Qipei Mei:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Peter Griesbach:** Writing – review & editing, Software, Methodology, Conceptualization.

Declaration of Competing Interest

The authors declare no conflict of interest.

Acknowledgement

The support of the Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN- 2022–04938 is gratefully acknowledged.

Appendix

In Fig. 19, ten generated M_l (using β=2) are compared to the distribution of M_l measured from a wall section using detected blocks (DB). The results of the pushover analyses and associated acceleration-drift curves (corresponding to each generated M_l) are also presented in Fig. 20.

References

[1] Loverdos D, Sarhosis V. Image2DEM: a geometrical digital twin generator for the detailed structural analysis of existing masonry infrastructure stock. SoftwareX 2023;22:101323. https://doi.org/10.1016/j.softx.2023.101323.

[2] Vandenabeele L, Loverdos D, Pfister M, Sarhosis V. Deep learning for the segmentation of Large-Scale surveys of historic masonry: a new tool for building archaeology applied at the basilica of st anthony in Padua. Int J Arch Herit 2023; 18:1749–61. https://doi.org/10.1080/15583058.2023.2260771.

[3] Galanakis D, Lucho S, Maravelakis E, Bolanakis N, Konstantaras A, Vidakis N, et al. Segment anything model for Scan-to-Structural analysis in cultural heritage, Electronic Engineering, Information Technology & Education (EEITE), International Conference. Chania, Greece: IEEE; 2024. p. 1–7. https://doi.org/10.1109/EEITE61750.2024.10654401. . 5th int. Conf. electron2024.

[4] Zhang S, Hofmann M, Beyer K. A 2D typology generator for historical masonry elements. Constr Build Mater 2018;184:440–53. https://doi.org/10.1016/j.conbuildmat.2018.06.085.

[5] Pereira M, D'Altri AM, de Miranda S, Glisic B. Automatic multi-leaf nonperiodic block-by-block pattern generation and computational analysis of historical masonry structures. Eng Struct 2023;283:115945. https://doi.org/10.1016/j.engstruct.2023.115945.

[6] Szabó S, Funari MF, Lourenço PB. A mason-inspired pattern generator for historic masonry structures using quality indexes. Eng Struct 2024;304. https://doi.org/10.1016/j.engstruct.2024.117604.

[7] Dwyer B., Nelson J., Hansen T. Roboflow (Version 1.0) [Software] 2024. ⟨https://roboflow.com⟩.

[8] Griesbach P, Wilson R, Karakus B, Pulatsu B. Transferring vision-based data to discontinuum analysis for the assessment of URM walls. Eur J Environ Civ Eng 2023;0:1–16. https://doi.org/10.1080/19648189.2023.2254376.

[9] Raitoharju J. Convolutional neural networks, Deep Learn Robot Percept Cogn. Elsevier; 2022. p. 35–69. https://doi.org/10.1016/B978-0-32-385787-1.00008-7.

[10] Lakshmanan V., Gorner M., Ryan G. Practical Machine Learning for Computer Vision. First Edit. O'Reilly; 2021.

[11] Karimi N, Mishra M, Lourenço PB. Automated surface crack detection in historical constructions with various materials using deep Learning-Based YOLO network. Int J Arch Herit 2024;00:1–17. https://doi.org/10.1080/15583058.2024.2376177.

[12] Yaseen M. What is YOLOv8: an In-Depth exploration of the internal features of the Next-Generation object. Detector 2024;8:1–10.

[13] Kukartsev VV, Ageev RA, Borodulin AS, Gantimurov AP, Kleshko II. Deep learning for object detection in images development and evaluation of the YOLOv8 model using ultralytics and roboflow libraries. Lect Notes Netw Syst 2024:629–37. https://doi.org/10.1007/978-3-031-70285-3_48.

[14] Gupta HS, Sameer M, Ahmad G. Real-Time vehicle detection using YOLOv8 and data augmentation approach. IEEE Fifth Int. Conf. Adv. Electron. Comput. Commun, 2023. Bengaluru, India: IEEE; 2023. p. 1–6. https://doi.org/10.1109/ICAECC59324.2023.10560330.

[15] Mumuni A, Mumuni F. Data augmentation: a comprehensive survey of modern approaches. Array 2022;16:100258. https://doi.org/10.1016/j.array.2022.100258.

[16] Ye Z, Lovell L, Faramarzi A, Ninić J. Sam-based instance segmentation models for the automation of structural damage detection. Adv Eng Inform 2024;62. https://doi.org/10.1016/j.aei.2024.102826.

[17] Kumar A, Kalia A, Verma K, Sharma A, Kaushal M. Scaling up face masks detection with YOLO on a novel dataset. Optik 2021;239:166744. https://doi.org/10.1016/j.ijleo.2021.166744.

[18] Vabalas A, Gowen E, Poliakoff E, Casson AJ. Machine learning algorithm validation with a limited sample size. PLoS One 2019;14:e0224365. https://doi.org/10.1371/journal.pone.0224365.

[19] Jocher G., Chaurasia A., Qiu J. Ultralytics YOLOv8 2023. ⟨https://github.com/ultralytics/ultralytics⟩.

[20] Goutte C, Gaussier E. A probabilistic interpretation of precision, recall and F-Score, with implication for evaluation. Eur Conf Inf Retr 2005:345–59. https://doi.org/10.1007/978-3-540-31865-1_25.

[21] Ainsworth M, Shin Y. Plateau phenomenon in gradient descent training of RELU networks: explanation, quantification, and avoidance. SIAM J Sci Comput 2021;43: A3438–68. https://doi.org/10.1137/20M1353010.

[22] Loverdos D, Sarhosis V. Automatic image-based brick segmentation and crack detection of masonry walls using machine learning. Autom Constr 2022;140: 104389. https://doi.org/10.1016/j.autcon.2022.104389.

[23] Hahne C, Aggoun A. PlenoptiCam v1.0: a Light-Field imaging framework. IEEE Trans Image Process 2021;30:6757–71. https://doi.org/10.1109/TIP.2021.3095671.

[24] Mandal S, Mones SMB, Das A, Balas VE, Shaw RN, Ghosh A. Single shot detection for detecting real-time flying objects for unmanned aerial vehicle. Artif. Intell. Futur. Gener. Robot. Elsevier; 2021. p. 37–53. https://doi.org/10.1016/B978-0-323-85498-6.00005-8.

[25] Lourenço PB. Computations on historic masonry structures. Prog Struct Eng Mater 2002;4:301–19. https://doi.org/10.1002/pse.120.

[26] Borri A, Corradi M, Castori G. A method for the analysis and classification of historic masonry. Bull Earthq Eng 2015:2647–65. https://doi.org/10.1007/s10518-015-9731-4.

[27] Cundall PA. A computer model for simulating progressive, large-scale movements in blocky rock systems. Int Symp Rock Mech 1971;2:47–65.

[28] Pulatsu B. Coupled elasto-softening contact models in DEM to predict the in-plane response of masonry walls. Comput Part Mech 2023;10:1759–70. https://doi.org/10.1007/s40571-023-00586-x.

[29] Pulatsu B, Wilson R, Lemos JV, Mojsilović N. Exploring the cyclic behaviour of URM walls with and without Damp-Proof course (DPC) membranes through discrete element method. Infrastructures 2024;9:11. https://doi.org/10.3390/infrastructures9010011.

[30] Cundall PA, C D. Dynamic relaxation applied to continuum and discontinuum numerical models in geomechanics. 1st Ed. Boca Raton, FL: CRC Press; 2017.

[31] Cundall PA. Formulation of a Three-dimensional disctinct element model - part I. A scheme to detect and represent contacts in a system composed of many polyhedral blocks. Int J Rock Mech Min Sci Geomech 1988;25:107–16.

[32] Hart R, Cundall PA, Lemos JV. Formulation of a three-dimensional distinct element model - part II. Mechanical calculations for motion and interaction of a system composed of many polyhedral blocks. Int J Rock Mech Min Sci Geomech 1988;25: 117–25.

[33] Pulatsu B, Bretas EM, Lourenço PB. Discrete element modeling of masonry structures: validation and application. Earthq Struct 2016;11:563–82. https://doi.org/10.12989/eas.2016.11.4.563.

[34] Pulatsu B, Erdogmus E, Lourenço PB, Lemos JV, Tuncay K. Simulation of the in-plane structural behavior of unreinforced masonry walls and buildings using DEM. Structures 2020;27:2274–87. https://doi.org/10.1016/j.istruc.2020.08.026.

[35] Pulatsu B, Tuncay K. Computational modeling of damage progression in unreinforced masonry walls via DEM. Turk J Civ Eng 2024;35. https://doi.org/10.18400/tjce.1323977.

[36] Malomo D, DeJong MJ. A Macro-Distinct element model (M-DEM) for out-of-plane analysis of unreinforced masonry structures. Eng Struct 2021;244. https://doi.org/10.1016/j.engstruct.2021.112754.

[37] Malomo D, DeJong MJ, Penna A. Influence of bond pattern on the in-plane behavior of URM piers. Int J Arch Herit 2021;15:1492–511. https://doi.org/10.1080/15583058.2019.1702738.

[38] Pulatsu B, Gonen S, Parisi F, Erdogmus E, Tuncay K, Funari MF, et al. Probabilistic approach to assess URM walls with openings using discrete rigid block analysis (D-RBA). J Build Eng 2022;61:105269. https://doi.org/10.1016/j.jobe.2022.105269.

[39] Mendes N, Zanotti S, Lemos JV. Seismic performance of historical buildings based on discrete element method: an adobe church. J Earthq Eng 2020;24:1270–89. https://doi.org/10.1080/13632469.2018.1463879.

[40] Foti D, Vacca V, Facchini I. DEM modeling and experimental analysis of the static behavior of a dry-joints masonry cross vaults. Constr Build Mater 2018;170: 111–20. https://doi.org/10.1016/j.conbuildmat.2018.02.202.

[41] Alexakis H, Makris N. Hinging mechanisms of masonry Single-Nave barrel vaults subjected to lateral and gravity loads. J Struct Eng 2017;143:4017026. https://doi.org/10.1061/(asce)st.1943-541x.0001762.

[42] Itasca Consulting Group Inc. 3DEC Three Dimensional Distinct Element Code 2013.

[43] Lourenço PB, Gaetani A. Recommended properties for advanced numerical analysis. Finite Elem Anal Build Assess. New York: Routledge; 2022. p. 209–320. https://doi.org/10.1201/9780429341564-4.

[44] Gonen S, Pulatsu B, Erduran E, Pelà L, Soyoz S. Dynamic characteristics of stone masonry walls before and after damage: experimental and numerical investigations. Eng Struct 2024;306:117808. https://doi.org/10.1016/j.engstruct.2024.117808.