Automation in Construction 125 (2021) 103633

Contents lists available at ScienceDirect

Automation in Construction

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

An innovative image processing-based framework for the numerical modelling of cracked masonry structures

Dimitrios Loverdos [a], Vasilis Sarhosis [a,*], Efstathios Adamopoulos [b], Anastasios Drougkas [a]

[a] University of Leeds, School of Civil Engineering, Woodhouse Ln, Leeds LS2 9DY, United Kingdom
[b] University of Turin, Department of Computer Sciences, Corso Svizzera 185, Turin 10149, Italy

ARTICLE INFO

Keywords:
Masonry
Image processing
Watershed transform segmentation
Feature extraction
Numerical modelling
DEM

ABSTRACT

A vital aspect when modelling the mechanical behaviour of existing masonry structures is the accuracy in which the geometry of the real structure is transferred in the numerical model. Commonly, the geometry of masonry is captured with traditional techniques (e.g. visual inspection and manual surveying methods), which are labour intensive and error-prone. Over the last ten years, advances in photogrammetry and image processing have started to change the building industry since it is possible to capture rapidly and remotely digital records of objects and features. Although limited work exists in detecting distinct features from masonry structures, up to now there is no automated procedure leading from image-based recording to their numerical modelling. To address this, an innovative framework, based on image-processing, has been developed that automatically extracts geometrical features from masonry structures (i.e. masonry units, mortar, existing cracks and pathologies, etc.) and generate the geometry for their advanced numerical modelling. The proposed watershed-based algorithm initially deconstructs the features of the segmentation, then reconstructs them in the form of shared vertices and edges, and finally converts them to scalable polylines. The polylines extracted are simplified using a contour generalisation procedure. The geometry of the masonry elements is further modified to facilitate the transition to a numerical modelling environment. The proposed framework is tested by comparing the numerical analysis results of an undamaged and a damaged masonry structures, using models generated through manual and the proposed algorithmic approaches. Although the methodology is demonstrated here for use in discrete element modelling, it can be applied to other computational approaches based on the simplified and detailed micro-modelling approach for evaluating the structural behaviour of masonry structures.

1. Introduction

Assessing the structural performance of ageing masonry structures is a difficult task. Over the last three decades, significant efforts have been devoted to developing numerical models to represent the complex and non-linear behaviour of existing unreinforced masonry structures subjected to external loads. Such models range from considering masonry as a continuum (macro-models) to the more detailed ones that consider masonry as an assemblage of units and mortar joints (micro-models or meso-scale models), see [22]. Since old and deteriorated masonry is typically characterised by low bond strength [29], cracking is often a result of the masonry units' de-bonding from the mortar joints. Given the importance of the masonry unit-to-mortar interface on the structural behaviour of aged masonry structures, micro-modelling approaches (i.e. those based on Discrete Element Method; in which the mortar is described as zero-thickness interfaces between the masonry units) are better suited for simulating their serviceability and load carrying capacity [27,29]. A vital aspect when modelling masonry structures based on the micro-modelling approach is the accuracy in which the geometry and material performance characteristics are transferred in the numerical model [14,15]. Even though current numerical modelling strategies for masonry are focusing primarily on idealised geometry [3], examples in the literature (e.g. [13]) demonstrate that a more representative visualisation of the masonry leads to more accurate results.

Some efforts are being made by the scientific community to accurately capture the geometrical characteristics of masonry structures using traditional techniques (e.g. on-site inspection and manual surveying methods). However, such methods have been found to be labour intensive and error-prone [38]. Over the last ten years, advances in laser scanning and photogrammetry have started to drastically change

* Corresponding author.
E-mail address: v.sarhosis@leeds.ac.uk (V. Sarhosis).

https://doi.org/10.1016/j.autcon.2021.103633
Received 2 December 2020; Received in revised form 4 February 2021; Accepted 14 February 2021
Available online 21 February 2021

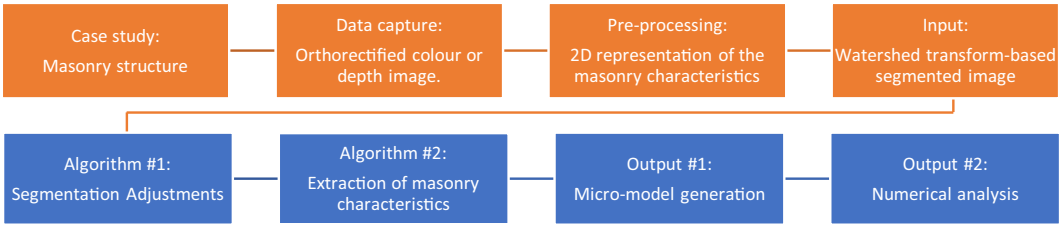

Fig. 1. Suggested workflow of the overall framework. The work presented in this document is shown in blue (Second row).

the building industry since similar techniques are able to capture rapidly and remotely digital records of building elements and features in three-dimensional (3D) point-cloud and ortho-image format [1,24,25]. However, current approaches for extracting geometrical features (i.e. size and positioning of masonry units, location and size of cracks, etc.) from imagery, lack automation, while protocols for the systematic generation of meso-scale models for assessing the structural behaviour of masonry structures are absent. Thus, even today, the feature extraction of masonry units' geometry is done manually using either computer-aided design (CAD), image, or point-cloud-based approaches [24,25]. Point-based voxelisation of point-cloud data offers a possible solution by allowing the generation of discretised models [17]. Even so, voxelisation methods do not consider the effect of the masonry-unit geometry or physical defects on the numerical model.

Image-processing approaches can offer various solutions to the problem of creating simplified records of masonry structures suitable for meso-scale modelling via the use of feature-detection [2,5,7,23] and segmentation techniques [2,4,21]. Research in automated detection and segmentation of masonry elements has drawn much attention by the scientific community [6,30]. Additionally, research in defect localisation using artificial intelligence also offers alternative methods to identify the extent of damage present on masonry structures, with high-level of automation [8,11,37]. Feature-detection of masonry elements presents a challenging task due to the anisotropic radiometric characteristics of the imagery or ortho-imagery involved. Raw images or photogrammetric derivatives, depending on the method of acquisition, may be of low quality for the automated detection of features. That includes the on-image sharpness of the edges of masonry units and defects, efficient contrast between masonry units and interfaces, and the effect of surface degradation on capturing the necessary radiometric data.

However, recent studies have demonstrated the efficient application of image-enhancing algorithmic implementations, with the purpose to improve the radiometric quality of digital records of masonry, facilitating the extraction of geometric features of their structural elements [18,36]. Advanced solutions have also considered the use of infrared thermography as a primary sensing technique for the interpretation of the structure of masonry [9]. Those recent developments establish the use of image-based applications as a viable solution for the mechanical evaluation of masonry elements. A similar notion is presented in [33–35], where it contemplates the use of binarised-images for the automatic construction of a voxel/pixel heterogenous pattern for the limit-analysis of irregular masonry. However, despite the rationale of identifying the structural composition characteristics of masonry in a cost and time-effective way, the practical use of feature and defect-detection is rarely used for the automated generation of discreet numerical models.

To resolve the commonly discussed topic of masonry evaluation using image-obtained data, this study proposes an automated methodological approach for numerical model generation using the output of image-processing applications. The main objective is the geometric feature-extraction from masonry structures (e.g. masonry units, mortar, and damage pathologies) using an innovative watershed segmentation approach. The methodology proposed in this document is part of a holistic framework that aims to automate fully the generation of masonry models from point-cloud data (PCD) and imagery data (Fig. 1). Although the approach is demonstrated here for use in discrete element modelling, it can be applied to other computational approaches for evaluating the structural behaviour of masonry structures.

2. Segmentation adjustments

The purpose of this section is to describe the refinement procedure followed to correct spatially the characteristics of masonry segmentation (Fig. 2), which can run as input for the accurate description of the masonry geometry in the structural analysis model. The procedure commences with a watershed segmentation-derived input and considers the actual geometric characteristics of both mortars and cracks in masonry to correct segmentation issues caused by geometric irregularities (Fig. 2: Steps 3 and 4).

Segmentation is often used in image processing to reduce the amount of available data on an image from pixels to regions. The segmentation technique considered is the marker-based watershed-transform, due to its innate ability to produce closed-regions. The methodology proposed is aimed to be used in combination with algorithms that can produce good binarisation. However, if the contrast between building blocks and their interfaces on the source imagery is adequate, typical image processing techniques can be applied (Fig. 3(d)). Additionally, If the input includes background information, it should be removed during the pre-processing stage. Moreover, a background of uniform colour can be used to limit the segmentation by generating the background-mask (Fig. 3 (c)). After an appropriate binarised image is applied, a morphological operation can be used to detect the local minima (Fig. 3(e)). In which case, H-minima transform is often used to remove false local-minima and prevent over-segmentation (Fig. 3(f)). The local-minima of a

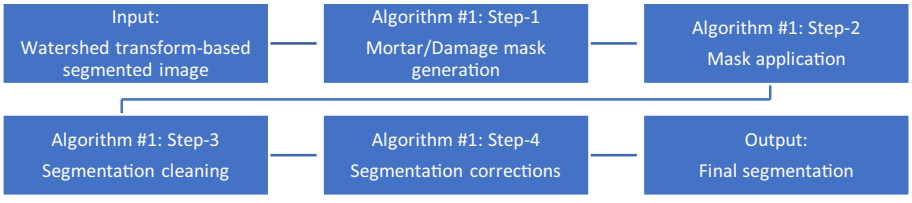

Fig. 2. Workflow of the algorithm responsible for the segmentation modifications (Algorithm #1).

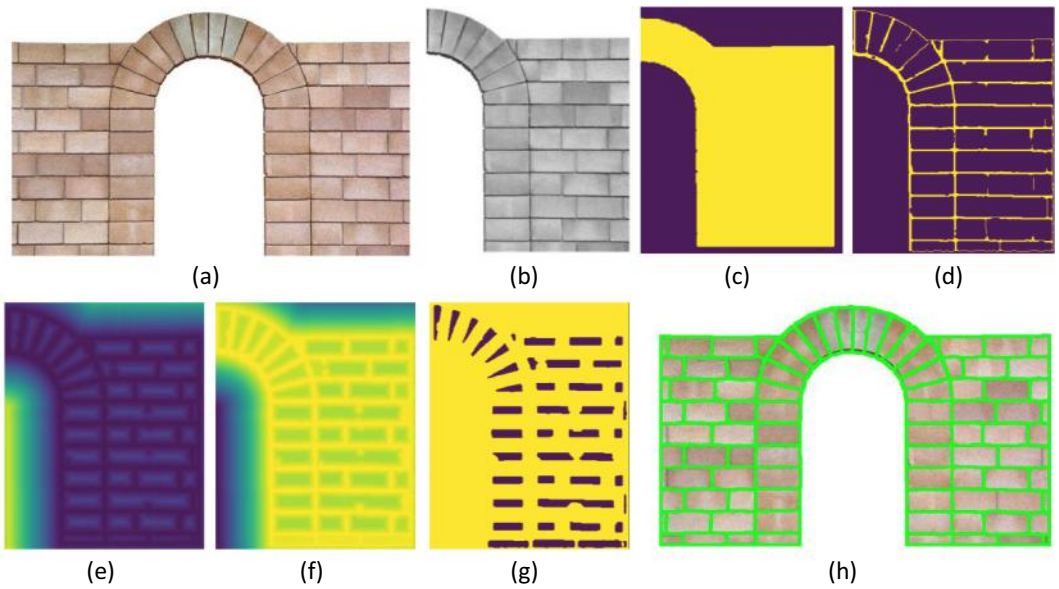

Fig. 3. Marker-based watershed segmentation, (a): Image source; (b): Bilateral blurring on greyscale to reduce noise and retain edges; (c): Background mask by global-thresholding; (d): Canny edge-detector applied on the filtered image (after erosion/dilation); (e): Inverse distance-transform for the markers; (f): H-Minima transform to remove false minima; (g): Local-minima for watershed-markers (after dilation/erosion); (h): Segmentation lines.

modified distance-transform provides the markers of the watershed (Fig. 3(g)). Fig. 3 shows a typical procedure of watershed-segmentation.

2.1. Mortar and damage mask generation

Considering the labelling convention of the watershed segmentation, the proposed algorithm generates a mortar-mask based on the numerical values of the watershed array, where the background label is marked with zero values, while the segmented areas are positive. The input for the mask generation process is the initial watershed array, padded by 1-pixel with zero values in all four directions. That allows to scan the image using a 2 × 2 Region of Interest (ROI). The size of 2 × 2 ROI corresponds to the minimum size required to detect an area where multiple labels are present. The following are the steps to define the network of mortar interfaces:

1. *Generated Mortar Mask (GMM)*: The mortar mask is generated using a 2 × 2 ROI to scan the image for inner interfaces (i.e. where the 2 × 2 ROI has two or more unique values, and all values are larger than zero) (Fig. 4(b)).
2. *Imported Mortar Mask (IMM)*: Optionally, the original rasterised output of the feature-detection, can be included to provide with minor corrections to the GMM (Fig. 4(c)). However, the rasterised image will contain perimetral edges that should be removed.
   2.1. *Generated Perimeter Mask (GPM)*: When the original raster image is used, the perimeter mask is generated (i.e. where the 2 × 2 ROI has two or more unique values and contains at least one zero) which aims to remove the perimetral edges from the IMM (Fig. 4(d)).
3. *Generated Background Mask (GBM)*: Excessive mortar caused by the dilation of GMM and IMM is removed by the background mask (i.e. where the padded watershed array has zero values), the GBM is applied to the final segmentation (Fig. 4(e)).

The padding of all masks contains values of ones, except the GBM that contains zeros (Fig. 4). This allows the background to reduce the segmentation, even if it envelopes the entire image. The initial line-thickness of each mask that uses a 2 × 2 ROI is at least 2-pixels, since the entire ROI is transferred to the mask. The 2-pixel thickness is required due to the segmentations being connected, and a mortar of 1-pixel thickness would reduce the size of a masonry unit in one side. The size of each mask is controlled by erosion/dilation, which effectively adjusts their effect. The GMM is adjusted manually to represent the average mortar thickness. The GBM and IMM are generally used as generated without modifications to their thickness. However, the GPM is given an excessive value to remove the perimeter of the imported-mortar-mask, given that it does not override succeeding mortar layers. The option to adjust each mask using morphological erosion and dilation allows the fine-tuning of the final result if it is required.

If the damage (i.e. cracking) in masonry is provided as a raster image (IDM, Fig. 5(d)), it can be applied to the final segmentation before the background mask. However, the use of external masks should be

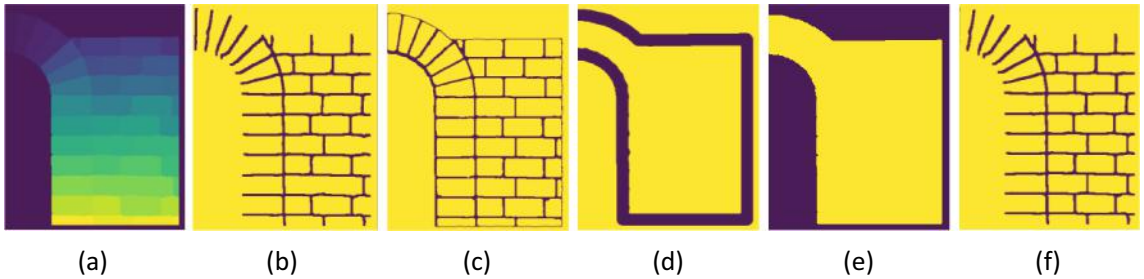

Fig. 4. Raster masks; (a): Initial watershed segmentation; (b): Generated-mortar (GMM); (c): Imported-mortar (IMM); (d): Generated-perimeter (GPM); (e): Generated-background (GBM); (f): Final mortar (FMM = IMM + GMM).

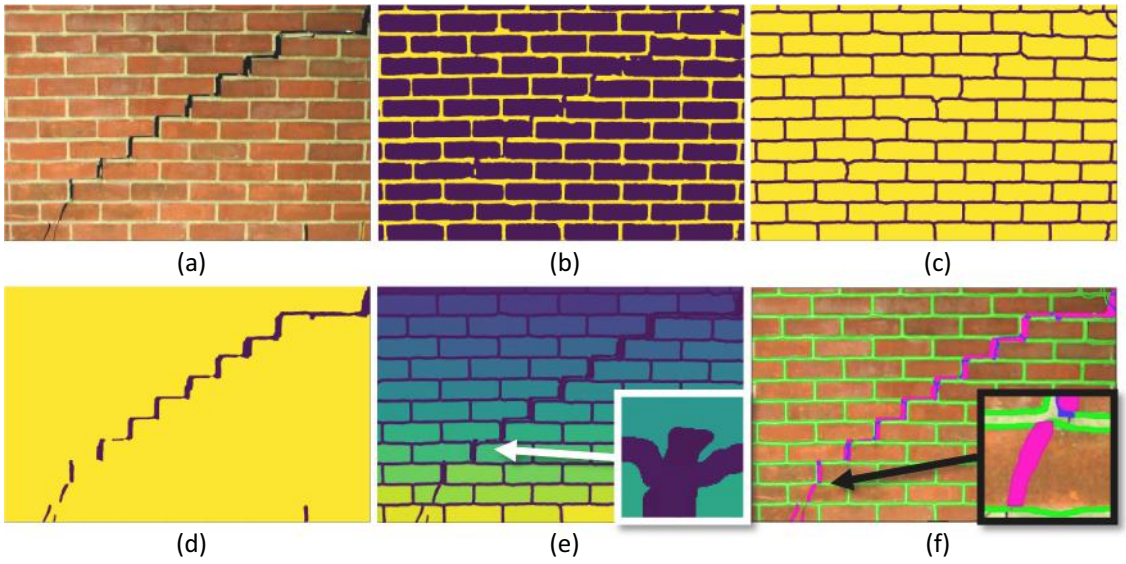

Fig. 5. Watershed segmentation demonstrating the name convention (GMM & IDM); (a): Source image; (b): Initial rasterised image; (c): GMM; (d): IDM; (e): Watershed lines with damage; (f): Watershed lines with damage states (Blue: Damage on mortar pixels, Magenta: Damage on brick pixels).

adjusted or avoided; if they are inaccurate (i.e. excessive noise, false detection of damage, etc.). If not, they may create inappropriate discontinuities on the material (Fig. 5(e)) and may cause incorrect estimation of collapse loads during the numerical analysis. Finally, a unique label is applied by each mask, on the final segmentation, used by the feature extraction method to identify different locations (i.e. *mortar* = − 1; *damage in mortar* = − 2; *damage in masonry unit* = − 3). Assuming that the accurate mortar (GMM, IMM) and damage (IDM) location is provided, the different damage states will indicate different damage types (i.e. crack on mortar, crack on brick, loss of material due to spalling or excessive cracking, etc). If not, they will only indicate the prior label before the damage is assigned to the affected location (Fig. 5 (f)).

2.2. Segmentation cleaning and correction

The application of external masks, on the watershed array, may cause the isolation of individual pixels or separation of a segmentation into multiple objects (i.e. Fig. 5(d)). Their existence must be corrected before the feature extraction as it may cause issues with the definition of each block.

The first step, towards the correction of the segmentation, is the elimination of isolated pixels (Fig. 6). Pixels that do not have a 4-connectivity (i.e., no diagonal connectivity) with a label are considered "isolated". An "isolated" pixel may cause the erroneous description of a masonry units' perimeter. For that reason, they are replaced with the most common label of its neighbour pixels (Fig. 6 (c) & d). All values of the 3 × 3 ROI are considered, but the pixel is replaced only by a label that has 4-connectivity with it.

The second and final step, for the correction of the segmentation, is the separation of duplicate objects (Fig. 7). If two or more segmentations have the same label, they are considered duplicates. Duplicate objects may cause conflicts during the feature-extraction. For that reason, duplicate objects are provided with a new label. Initially, each zero/positive label is isolated on a new-array (Fig. 7(b)) with a size equal to the original watershed padded by 1-pixel of zero values. The new-array contains zeros and ones, where one is the segmentation label examined. The watershed segmentation is then applied using as a mask and source

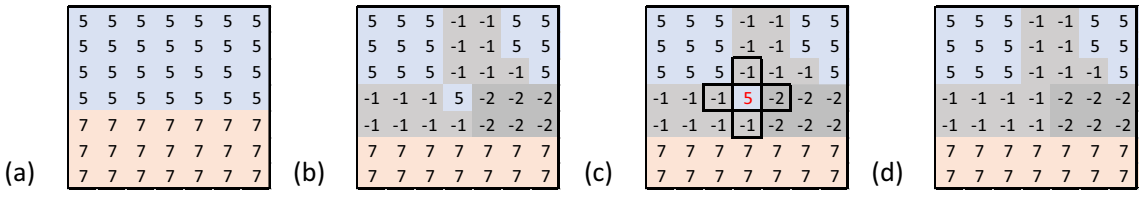

Fig. 6. Segmentation-cleaning; (a): Initial segmentation; (b): After mask application; (c): Detection of isolated pixels; (d): Replacement of isolated pixels with the most common value of the 3 × 3 ROI (limited to 4-connectivity labels).

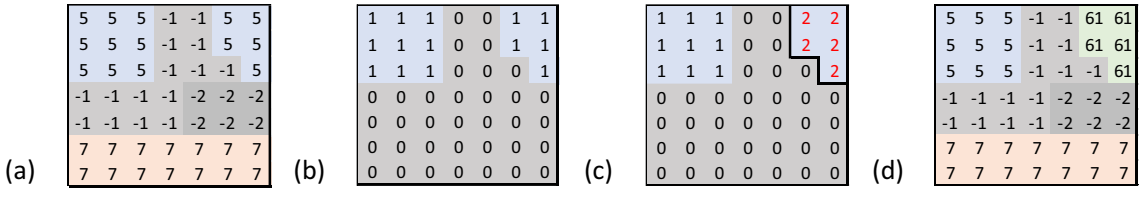

Fig. 7. Segmentation-corrections; (a): Masked segmentation; (b): Isolation of label #5 in a new array; (c): Watershed segmentation and detection of additional labels; (d): *Re*-labelling of additional segmentation.

the new-array and the inverse new-array to acquire the local-minima for the markers (Fig. 7(c)). After the first segmentation, any subsequent labels are assigned a new value equal to the existing maximum plus one (Fig. 7(d)). The command used for the markers should consider only pixels with 4-connectivity to separate segmentations that are not connected vertically or horizontally. Doing so also solves the issue where two isolated pixels are located side-by-side and thus not detected by the segmentation-cleaning process demonstrated above (Fig. 6). Only zero and positive labels are verified during this step. Furthermore, any modified label is stored in the Changed-Contour-List (CCL), with its prior label to retain the previous state/type of the segmentation (i.e. Blocks, Background, etc.). The structure of the Changed-Contour-List is provided below:

$$CCL_n = [New\ Label, Old\ Label] \tag{1}$$

3. Feature extraction

The second part of the proposed framework is the extraction of a network of lines and nodes, which will represent the simplified geometry of the interfaces. The final output is a collection of polylines that will be used for the numerical model generation. The input of this section is the modified segmentation acquired previously (Algorithm #2, see Fig. 1). It initiates by scanning the watershed array to extract the coordinates in-between segmentations using a novel approach (Fig. 8: Step-1). The correct order of the extracted pixels is provided by using a border-following algorithm (Fig. 8: Step-2). Additionally, it uses a generalisation algorithm to reduce the number of vertices that describe an interface (Fig. 8: Step-3). Furthermore, it includes geometric operations to adjust the location of selected vertices and produce more accurate results (Fig. 8: Step-4).

3.1. Point detection and contour definition

The input of the point-detection algorithm is the watershed array padded by 1-pixel of zero values in all directions (i.e. from 100 × 100 to 102 × 102, (Fig. 9(a)), to allow scanning using a 2 × 2 ROI. Moreover, the geometric definition of damage is excluded at this stage; since the same labels have been assigned to multiple segmentations and may cause false detection of the perimetrical characteristics of masonry units. The simplest solution is to apply the point detection on a padded watershed array (PWS), where all damage labels are replaced with mortar (i.e. PWS(< −1) = −1). Initially, the watershed array is scanned using a 2 × 2 ROI under the following conditions:

1. *Interface-Point*: If the 2 × 2 ROI contains two or more unique values, it is considered an interface-point (Fig. 9(a)). In which case, it is saved on the interface-point-list (IPL), with its ID and location.
2. *End-Point*: If the 2 × 2 ROI contains three or more unique values, it is considered an end-point (Fig. 9(c)). In which case it is saved on the end-point-list (EPL), with its ID and location.

The ID is the ordered list of unique values from the ROI. The location saved for each point detected is the top-left location of the 2 × 2 ROI. Each collection of points of a unique ID (of two components), makes an interface with known end-points, and their location is saved on the interface list (IL). End-points have three or more ID values and are stored to all appropriate interfaces (where they have two common values). Practically, the point-detection does not consider the location of pixels, but the gridlines instead. This method was selected to avoid the separation between segmentations. For that reason, the coordinates include an additional unit in the two directions (i.e. from 100 × 100 to 101 × 101, Fig. 9(b)). The structure of the lists is provided below:

$$ROI = PWS[\,i, i+1 : j, j+1] => Loc = [x, y] = [j, i] \tag{2}$$

$$\mathrm{IPL_n} = [\mathrm{ID}, Loc]\ (\mathrm{For}\ 2 + \mathrm{unique\ labels}) \tag{3}$$

$$\mathrm{EPL_n} = [\mathrm{ID}, Loc]\ (\mathrm{For}\ 3 + \mathrm{unique\ labels}) \tag{4}$$

$$IL_n = [ID, End\,Points, Interface\ Points] \tag{5}$$

At this stage, all nodes that define the geometry of the interfaces have been identified. However, their order is still unknown. Their sequence is calculated by applying the border-following algorithm developed by [31]. The algorithm is applied to a new 2D-array for each unique ID with all its points marked (Fig. 10(b)). Moreover, only the parent-contours are stored (outer perimeter of a closed object), using the algorithm's hierarchy. As mentioned previously, the algorithm returns the perimeter that contains duplicate points if it is an open shape (Fig. 10(c)). Additionally, it does not return the last point of closed shapes and thus, complicates the determination of closed objects. The aforementioned are addressed by the following framework:

1. Create an array of zeros where ones mark the examined interface.
2. Apply the border following algorithm on the new array and extract the outer-contours. In rare cases, multiple contours may be extracted

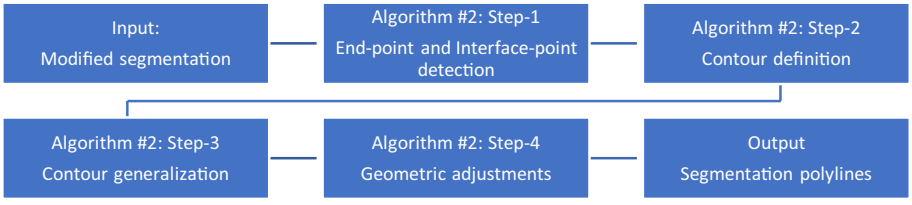

Fig. 8. Workflow of the algorithm responsible for the feature extraction (Algorithm #2).

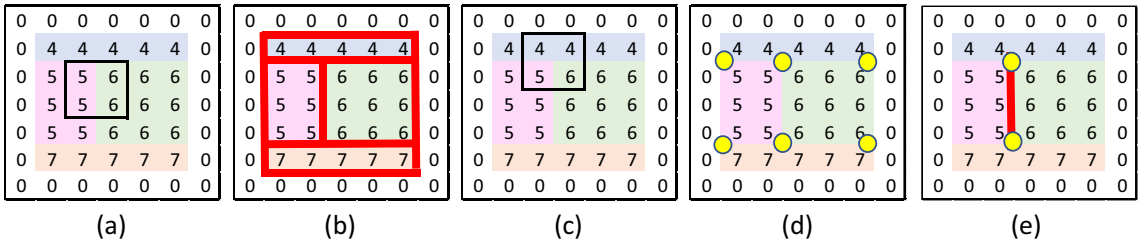

Fig. 9. Point detection on the padded array; (a): interface-point with 2 unique labels; (b): All detectable interface-points; (c): interface-point and end-point with 3 unique labels; (d): All end-points; (e): interface with ID = [5,6] (gridline).

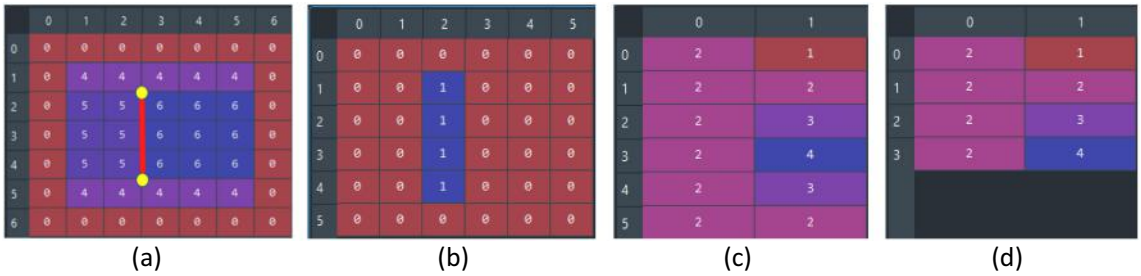

Fig. 10. Ordering of interface points; (a): Marked interface with ID = [5,6], end-point #1 = [2,1], and end-point #2 = [2,4]; (b): New array of examined interface (counting gridlines); (c): Border-following output (perimeter); (d): Modified output (polyline).

with the same interface ID (i.e. when the mortar label is in contact with a specific block at two separated locations).

2.1. Scan each contour detected to find if it contains any end-points of the same interface. Each end-point is only considered once.
   2.1.1. If less than two end-points are detected, consider the object a closed shape.
   2.1.2. If two are detected, store only the first range between the end-points.
   2.1.3. If three or more are detected, find the first location of every end-point and store all ranges between end-points separately. Duplicate end-points are not considered.
2.2. If the contour is considered a closed shape, and the first and last points are not equal, append the first point to the end of the *xy*-array.
2.3. Finally, store each range separately to the contour-list (CL), including the interface ID.

The structure of the contour-list is the following:

$$CL_n = [ID,\ End\text{-}Points,\ xy\text{-}Array] \tag{6}$$

3.2. Contour generalisation

Contour generalisation is the process of reducing the number of vertices that define a contour. This process will dramatically reduce the time-cost of the structural analysis. Additionally, it will provide general corrections to the shape by smoothing the interface. It is also the reason why the border-following algorithm was not used to extract the shape of each block, but was used instead on each interface. Using the border-following algorithm on the blocks, the generalisation of contours would produce multiple generalisations for the same interface (since an interface is common between two segmentations). Moreover, a line generalisation algorithm is developed, based on the "*Ramer–Douglas–Peucker*" algorithm [12,26], with conditions specialised to the needs of the research.

Considering an original contour $C = [p_0,\ldots,p_n]$, with n subset of vertices p, a generalised contour can be defined as $C' = [\ldots,p'_{k-1},p'_k,\ldots] \subseteq C$, with the minimum number of elements that satisfies the condition $f(S_k) \le t$. Each sequential pair of vertices of C', divides C into sections $S_k = \{p_{k-1}',\ldots,p_i,\ldots,p'_k\} \in C$. The examined vertex $p_i \in S_k$ is tested with regards to the straight-line segment $\overline{p'_{k-1}p'_k}$. If the condition is not satisfied, the vertex p_i with the highest vertical distance (vd), is stored in C'. The (simplified) original condition in [26], is the following:

$$\mathrm{f}(S_k) = max(vd(S_k)) \le t \tag{7}$$

where $vd(S_k)$ is the vertical distance of all elements in S_k and the line segment $\overline{p'_{k-1}p'_k}$, and t is the constant-threshold value. Compared to the original, the following changes were implemented:

Transformation: Before the first iteration and only if the contour is closed (i.e. $p_0 = p_n$), the contour is reformed to start/end from the point $p_i \in C$, with the largest distance from the initial point p_0 (similar to Eq. (8)).

Condition #1: For the first iteration and only if the contour is closed (i.e. $p_0 = p_n$), add the point $p_i \in S_k$ with the largest distance from the initial point p_0, to the generalised contour. Where p_i is defined as:

$$f(i) = f(S_k) = max(ld(S_k)) = max(ld1(S_k), ld2(S_k)) \tag{8}$$

Condition #2: If the previous condition did not activate, then the second condition is considered. Add the point $p_i \in S_k$ with the largest distance from either the first or last point of the line segment, if it satisfies either: The vertical and length, or the horizontal and length thresholds. Using the minimum of the horizontal and length values of the pair. Where p_i is defined as:

$$f(i) = f(S_k) = max(ld(S_k)) = max(ld1(S_k), ld2(S_k)) \tag{9}$$

$$\text{Where the condition is disregarded if : } \mathrm{vd(i)} \le \mathrm{t_v} \text{ or } min(\mathrm{ld1(i)}, \mathrm{ld2(i)}) \le \mathrm{t_l} \tag{10}$$

$$\text{and : } min(\mathrm{hd1(i)}, \mathrm{hd2(i)}) \le \mathrm{t_h} \text{ or } min(\mathrm{ld1(i)}, \mathrm{ld2(i)}) \le \mathrm{t_l} \tag{11}$$

Condition #3: If the previous condition did not activate, then the third condition is considered. Add the point $p_i \in S_k$ with the largest vertical distance from the line segment, if it satisfies both: the vertical and length thresholds. Using the minimum of the length values of the pair. Where p_i is defined as:

$$f(i) = f(S_k) = max(vd(S_k)) \tag{12}$$

$$\text{Where the condition is disregarded if : } \mathrm{vd(i)} \le \mathrm{t_v} \text{ or } min(\mathrm{ld1(i)}, \mathrm{ld2(i)}) \le \mathrm{t_l} \tag{13}$$

Threshold: The threshold values are: Vertical (t_v), Horizontal (t_h), and Length (t_l). If the value of any threshold is below one, then it is considered the ratio of the threshold over the length of the line segment. Else, if it is above one, it is the actual threshold.

$$\text{If } \mathrm{t} < 1 : \mathrm{t} = \mathrm{t} \cdot \mathrm{length}\left(\overline{\mathrm{p'_{k-1}p'_k}}\right); \text{else : } \mathrm{t} = \mathrm{t} \tag{14}$$

where f(i) is a simplification of: $f(i) = f([p_{k-1}',p_i,p_k'])$. Each $\mathrm{p_i}$ point considered creates two horizontal (hd) and two length (ld) values (Fig. 11(a)), but only the minimum of each pair is tested (Eq. (10), (11), (13)). The "*Transformation*" ensures that the first point of a closed shape is essential to describe the general shape (i.e. Fig. 12(a1) as opposed to Fig. 12(b1)). While "*Condition #2*" (Eq. (11)), ensures that a vertex with a vertical point outside the range of the line segment and small vertical distance, is considered with the horizontal threshold instead (Fig. 11(a)

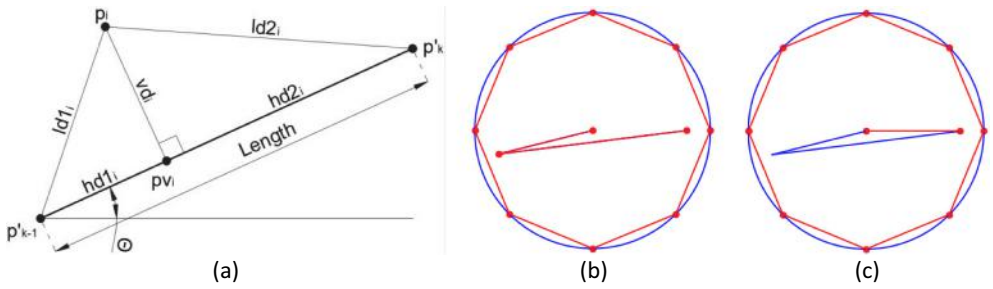

Fig. 11. (a): Variables of proposed algorithm; (b) Proposed algorithm (with: "th"); (c): Original algorithm (without: "th").

& (c)). Additionally, the length threshold (Eq. (10), (11), (13)) ensures that both line segments created have adequate length. Modifying the threshold values (t_v, t_h, t_l) allows to control the level of detail of the final output by reducing the number of vertices accordingly. Lastly, the threshold ratio for values below one, ensures that the preferred accuracy is preserved by dynamically adjusting the applied threshold (Eq. (14)). Every identified polyline is stored in the line-list (LL). The structure of the line-list is provided below:

$$LL_n = [ID, Original\ Contour, Generalised\ Polyline] \tag{15}$$

The resulting generalisations of the original "*Ramer–Douglas–Peucker*" (RDP) and the proposed algorithm (PROP) are compared at two levels of detail (Fig. 12). The level of detail is measured in the number of line segments (Lines) of the total output. Group [a] demonstrates the original algorithm using a static threshold (RDP), group [b] the proposed algorithm using static threshold (PROP-S), and [c] the proposed algorithm using a dynamic threshold (PROP-D). The results displayed below include only the geometry with mortar, since without mortar, almost every main vertex is pre-determined by the end-points between the interfaces. Regarding the results, the RDP algorithm detects a false point as initial (p'_0) on top of the closed contour in (i.e. [a1]), which is eliminated by the proposed algorithm using the "*Transformation*" (i.e. [b1, c1]). When the dynamic threshold with high ratio values is used (not shown below), the proposed algorithm may omit a second vertex to describe the small curvature near the corners better. Additionally, if a very small threshold-ratio is used (Eq. (14)), it may create an excessive number of vertices near the edges, where the threshold becomes smaller due to limited distance. Finally, all cases could be used for the development of the geometry for numerical analysis except [a2, b2], where it is possible that their highly uneven and irregular interface could cause overestimation of collapse loads due to local hinging phenomena.

3.3. Geometric adjustments

An issue arising from segmenting the masonry imagery via a watershed-based transform is that the location where two segmentations meet is not placed at the centre of gravity of the mortar area of the binarised image (Fig. 13(a1)). This is because the segmentation is growing further from the marker provided. Thus, any pixel located in the bright section will be labelled from the closest marker, creating a triangular shape (Fig. 13(a2)). This issue is amended partially when using the distance transform of the raster image as a source for the segmentation (Fig. 13(b)). Doing so creates a boundary located in the middle of the bright section, forcing the assignment of half the mortar thickness to each block. Another solution suggested, only when the final model includes mortar, is to use a large erosion value on the GMM and then dilation to restore its average size, covering the small cavities near the end-points (Fig. 13(c)). However, that will also introduce a small curvature to the edges of the block. Nonetheless, the erosion/dilation solution is often unnecessary since the generalisation of the contour will correct this, assuming that the threshold is not excessively low. However, it is required to have implemented the transformation of the previous section; otherwise, the top-most point will be included, since it is the first point detected by the border following algorithm (Fig. 12(a1)).

An additional processing step is introduced that adjusts only the affected vertices when the mortar is not modelled (Fig. 15(a)). The issue develops where two segmentations are connected, and two of the three interface lines form a near ~180° angle. Thus, the algorithm must target end-points that have three ID values (three unique values in the 2 × 2 ROI), and all are larger than zero. The location where the ID of end-points have zero or negative values are not affected since they do not have a second marker. Any end-point that has precisely three unique labels (i.e. end-points connected to three polylines), and all its values are larger than zero (i.e. not mortar, damage or background) is considered a possible candidate. Knowing the location and the unique ID, makes possible to identify the three polylines that are connected to the end-point. For every polyline detected, the line-segment is formed between the end-point and the previous on the polyline's *xy*-array. That may be either the first two or the last two points. If the angular difference between any combination of two line-segments is ~180°, then the end-point is relocated so that the angular difference between the two lines

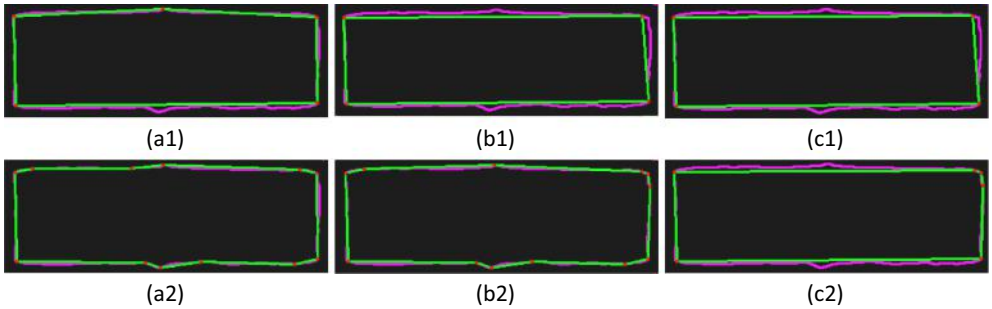

Fig. 12. Comparison of generalisation (Purple: Contour, Green: Polyline); (a1): RDP (Lines = 443); (b1): PROP-S (Lines = 438); (c1): PROP-D (Lines = 431); (a2): RDP (Lines = 1049); (b2): PROP-S (Lines = 1048); (c2): PROP-D (Lines = 1075).

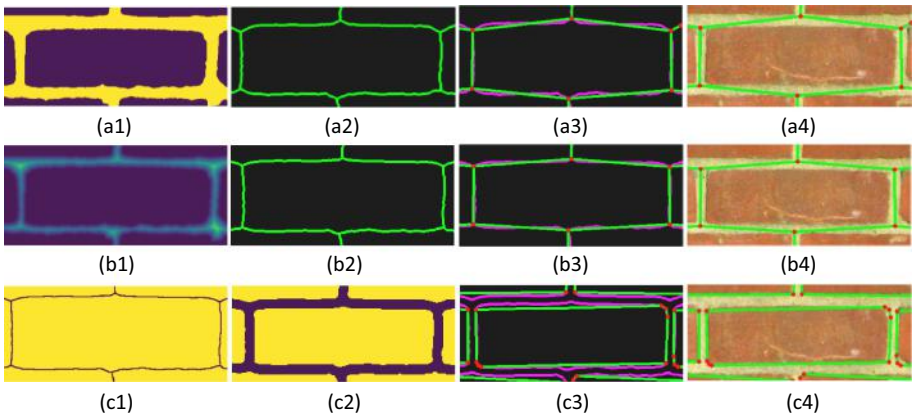

Fig. 13. Simple corrections to segmentation: (a): Contours and polylines using the original rasterised image; (b): Contours and polylines using the distance transform as a source; (c): Correcting generated mortar-mask using erosion/dilation.

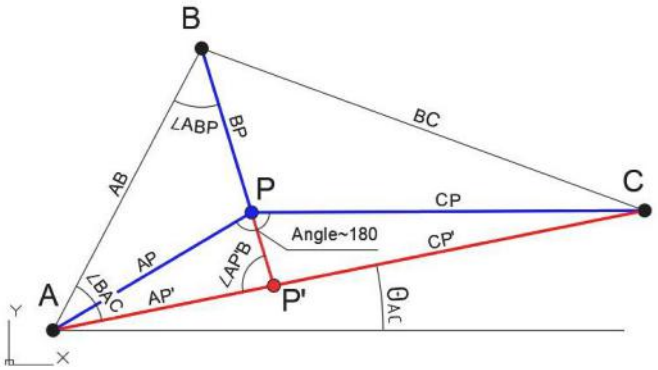

Fig. 14. Variables used in the geometric corrections (P: End-point, Blue: Old lines, Red: Adjusted lines).

is exactly 180°. However, the global angle of the remaining line must remain the same to retain its shape (Fig. 14: $\overline{BP}$). The extension of the line that does not form ~180° splits the geometry into two triangles (Fig. 14: $\overline{BP'}$), where the law of sines can be used to determine its length. Then, simple geometric functions are applied to calculate the co-ordinates of the adjusted end-point (Fig. 14: *P'*). A threshold value (t_a) is used to determine the tolerance of the angular difference. Furthermore, three iterations of the algorithm are required to correct all vertices. The formal description is provided below:

1. Create a copy of the "End-Point-List" and "Line-List" (i.e. EPL2 and LL2), to avoid modification of the original lists. Important: This may be required later if a closed-block is incorrectly defined.
2. Create the first iterative loop to repeat the process three times.
3. Create the second iterative loop to scan through the copied end-Point-list" (EPL2).
4. If the *ID* of the end-point contains exactly three labels, and all are higher than zero, consider the end-point (*P*) a possible candidate (i.e. $ID = [Lbl_1, Lbl_2, Lbl_3]$; and: $min(ID) > 0$).
   - 4.1. Identify the location index of the three polylines in the line-list (LL), by verifying that the first or last point of the polyline is equal to the location of the end-point saved on EPL2. The *ID* label of the end-point may also be used to locate the index.
   - 4.2. Create the line-segments by taking either the first two, or the last two points of the generalised line, based on the location of the end-point (i.e. if: $xy_P = xy_0$, then: $Line = [xy_0, xy_1]$; else if: $xy_P = xy_n$, then: $Line = [xy_n, xy_{n-1}]$).
   - 4.3. Calculate the angular difference between all combinations of the three line-segments to identify if two of the line segments form ~180° angle (i.e. $Ad_1 = \angle APC$; $Ad_2 = \angle CPB$; $Ad_3 = \angle BPA$). If the optional condition is used (Eq. (17)), calculate the global angle of the two line-segments that form ~180°, excluding the end-point (i.e. $A_1 = \Theta_{AC}$).
   - 4.4. If the conditions are satisfied (Eq. (16), Eq. (17), Eq. (18)), relocate the end-point (*P*), such that the lines that form ~180° become parallel (i.e. $\overline{AP'} \parallel \overline{P'C}$, with $P' \in \overline{AC}$), while retaining the global angle of the remaining line (i.e. $\Theta_{BP} = \Theta_{BP'}$).
   - 4.5. Update the value of the modified end-point to every list used (i.e. EPL2, LL2)
5. Repeat the process for all end-points during three iterations.

The main-condition is to verify if any combination of angular difference is ~180° (Eq. (16)). An optional-condition to automatically avoid rubble or arch-lines is to target cases where both ~180° and ~ 90° angular-difference are detected (Eq. (17)). An alternative optional-condition, that has the same purpose, is to verify that the global-angle of the adjusted line that forms near ~180° local-angle, is either: ~0° or multiples of ~90°; before applying the geometric corrections (Eq. (18)). All proposed conditions of the formal description (Sub-list: [4.4]), are provided below:

$$\text{Condition\#1 (Main)} : [180^\circ - t_a \le Ad_i \le 180^\circ + t_a] \tag{16}$$

$$\text{Condition\#2 (Optional)} : [90^\circ - t_a \le Ad_j \le 90^\circ + t_a]; i \ne j; \tag{17}$$

$$\text{Condition\#3 (Optional)} : [A_t - t_a \le A_i \le A_t + t_a]; \quad A_t = [0^\circ \vee 90^\circ \vee \ldots \vee 360^\circ] \tag{18}$$

Applying the geometric corrections to the interface between structural units allows for a more representative visualisation of masonry characteristics (Fig. 15). Regarding the examples demonstrated below, only the main condition was used (Eq. (16)). The arch-lines (Fig. 15(e)) were automatically excluded from the corrections since one of the three labels is zero (i.e. background). However, if the arch would contain more layers, an additional condition would be necessary to avoid the geometric adjustments on the inner interfaces (i.e. Eq. (17), Eq. (18)).

3.4. Producing closed-shapes

Specific numerical analysis software require closed-objects to define a shape (i.e. ABAQUS, 3DEC, LS-DIANA, etc.). For that purpose, it is necessary to generate the closed-block from the open-polylines. An interface ID contains two labels that are equal to the two objects that are in contact; where the label is the value of any segmentation. Thus, all polylines are compared, and those that contain a specific label are

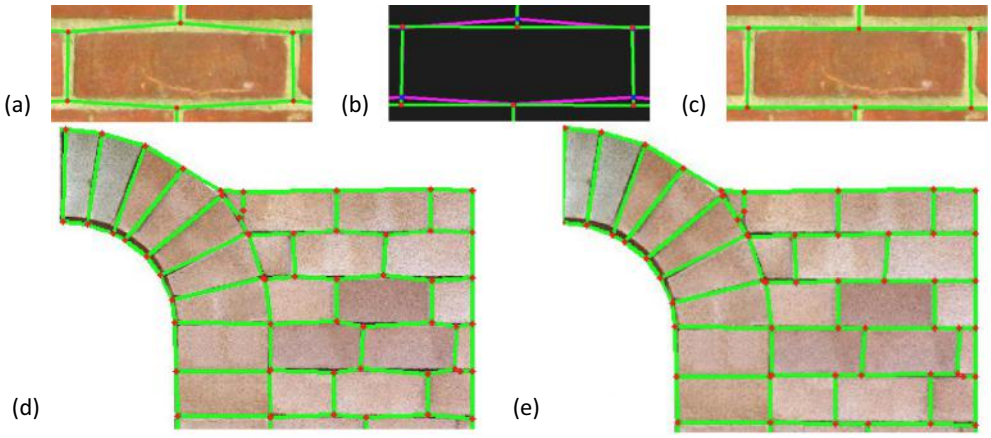

Fig. 15. Correcting geometrical inaccuracies ($t_a = 20°$): (a): Initial generalised lines; (b): Original and modified lines comparison; (c): Adjusted generalised lines; (d): Original lines on arched-door; (e): Adjusted lines on arched-door.

assigned to the block of the same value (i.e. Interface $ID = [2,3]$ is assigned to blocks #2 and #3). The collection of the polylines is added to a temporary-*block*-list (TBL) with its label.

The polylines are then combined by comparing the initial and final vertices of each entry in the collection (using the first entry as a temporary combined-polyline). The process is repeated until no connection is found with the temporary combined-polyline. Every entry used is removed from the polyline-collection to ensure that it is not used repetitively. If the combined-polyline is closed (i.e. equal first and last vertices), it is stored to the final block-list (BL) with its original label (Eq. (1)) to retain its type. Alternatively, if no connection is found and the combined polyline remains open, it is rejected. Furthermore, the process must repeat again for the same collection if there are remaining entries, which is possible when multiple segmentations have the same label. However, this will not be the case if the "*Segmentation Corrections*" have been applied. The original contour (the line with all points included), can also be stored to allow the use of the preferred accuracy for CAD model generation. Negative labels (damage and mortar) are excluded from this process, but the zero label is included to describe the overall perimeter. The structure of the block-lists is provided below:

$$TBL = [Label, Contour\ Collection, Polyline\ Collection] \tag{19}$$

$$\text{If Label} \geq 0 : \text{BL}_\text{n} = [\text{Old Label}, \text{Closed Contour}, \text{Closed Polyline}] \tag{20}$$

Moreover, if the generalisation and generation of closed shapes for the damage is required (i.e. for CAD design), each negatively-labelled segmentation must be assigned a unique label during the "*Segmentation Corrections*" (Eq. (1)), which aims to avoid conflicts with the contour definition.

Although a rare occurrence, a block label may fail to regenerate or provide the correct shape when an external mask is supplied. This is because interfaces of equal ID value are marked on the same array to apply the border-following algorithm. When 1-pixel segmentation separates two interfaces of equal ID, they are drawn connected since the method proposed considers the gridlines rather than the pixels. In this case, the algorithm may return incorrect ranges by omitting an end-point that is blocked by connected pixels (case #1), or return a single perimeter of the combined shape instead of two separated interfaces (case #2). The aforementioned refers only to individual outputs of the border-following algorithm that contain three or more end-points and is considered an open-shape. Additionally, it affects only watershed-segmentations that were modified using imported-masks (i.e. mortar, damage).

The simplest solution to this is to increase the size of the watershed segmentation two-fold just before the point-detection algorithm, which will effectively increase segmentations of 1-pixel thickness to 2-pixel. In this way, the unification of individual sections that are separated by 1-pixel is avoided. Alternatively, a programmable solution is also possible by forcing the algorithm to follow only the outer-perimeter of an affected interface. This is accomplished by applying the border-following algorithm on the gridlines of a segmentation instead, which will provide the outer perimeter of the affected block. The perimeter can be divided into interfaces by comparing the ID values of each point it contains. All contours/blocks of the affected cases must be removed and replaced with the adjusted items. However, the contour-generalisation and geometric adjustments require repetition for every affected item and interfaces/blocks in contact with the affected case, since modified interfaces may be common to two different segmentations. The programmable solution should be applied after combining interfaces into closed-blocks to ensure that all detection methods were used. Affected contours/blocks can be identified by:

1) *Detection method #1*: When an individual and open contour-output, of the border-following algorithm, contains duplicate inner-points.
2) *Detection method #2*: When the combined closed-shape includes duplicate inner-points (excluding the first since it is a closed shape).
3) *Detection method #3*: When failing to combine the interfaces of a segmentation into a closed-shape (i.e. unequal first and last vertices).

3.5. Data scaling

Before any further adjustments are made, the block and line list (BL and LL) are scaled to the preferred size by application of a scale factor to *xy*-coordinates of each element (Eq. (21)).

$$\begin{aligned} x' &= x \times scale \\ y' &= (ymax - y) \times scale \\ ymax &= max(y) \end{aligned} \tag{21}$$

The final geometry may include small objects that are inappropriate for use in CAD geometry or numerical model generation. Furthermore, the application of the line-generalisation algorithm may cause the generation of blocks with zero area due to inadequate space between essential vertices. Thus, the area verification of each object is required to provide proper input for the numerical analysis. The area of a closed polygon can be calculated using the Surveyor's Formula provided below (Eq. (22):

$$Area = \frac{1}{2} \times \left| \sum_{i=1}^{n-1} (x_i \times y_{i+1}) + (x_n \times y_1) - \sum_{i=1}^{n-1} (x_{i+1} \times y_i) - (x_1 \times y_n) \right| \tag{22}$$

Acquiring the area of either the scaled closed-contour or the closed-polyline allows the removal of small or zero-area objects from the block-

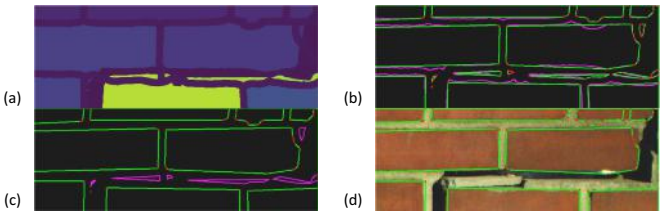

Fig. 16. Removing small objects; (a): Initial segmentation; (b): Original contours and generalised lines; (c): Removed small objects (in purple); (d): Remaining objects on source image.

Table 1
Macro properties of the brick elements- Block properties: Sandstone [20].

Density (kg/m^3)	Young's modulus (N/m^2)	Poisson's ratio
2350	$26364x10^6$	0.2

Table 2
Joint properties of the zero-thickness interfaces - Join contact properties of mortar [28].

Normal stiffness (N/m^3)	Shear stiffness (N/m^3)	Friction (*deg*)	Cohesive strength (N/m^2)	Tensile strength (N/m^2)	Dilation (*deg*)
4×10^{11}	2×10^{11}	38	0.6×10^6	0.6×10^6	4

Table 3
Macro properties of the brick elements [28].

Brick elements			Mortar elements		
Density (kg/m^3)	Young's modulus (N/m^2)	Poisson's ratio	Density (kg/m^3)	Young's modulus (N/m^2)	Poisson's ratio
1900	$19700x10^6$	0.2	1200	$2974x10^6$	0.2

list (BL) (Fig. 16). This can extend to lines by removing entries from the line-list (LL), that do not have at least one ID label equal to the labels of the remaining blocks (i.e. if both labels 10 and 11 do not exist in the adjusted block-list, then the polyline with $ID = [10, 11]$ is removed). Furthermore, the samples below exclude damage from the "*point detection*" section to avoid common issues (pg. 11). However, the damage will be included during the numerical analysis (Tables 1–3).

4. Numerical model generation

The last part of the proposed framework is the numerical model generation, which includes mortar and damage depending on the user's preferences (Fig. 17: Output). The proposed framework can be used to simulate masonry with the simplified-micro-modelling (or meso-scale) and detailed-micro-modelling approach. For the development of the numerical simulations, the commercial software UDEC, developed by Itasca, has been used [19]. The formulation of the method was proposed initially by [10] to study jointed rock, modelled as an assemblage of rigid blocks. Later this approach was extended to other engineering fields requiring a detailed study of the contact between blocks or particles such as soil and other granular materials [16]. More recently, the approach was applied successfully to model historic masonry structures in which the collapse modes were typically governed by mechanisms in which the blocks' deformability plays little to no role at all.

4.1. Geometric model generation

The methodology used to develop the polylines from the previous sections has been used here to generate the model geometry in AutoCAD or directly into a structural analysis software (Fig. 18). The python library used to create the AutoCAD model was "*pyautocad*". The 3D model was extruded using a standard value for the depth (Fig. 18(b)). The 2D mesh was created in AutoCAD using the "*hatch*" command (Fig. 18(c)) and includes both mortar and damage. For the models developed using the detailed micro-modelling approach, the mesh generation was made externally when the mortar was modelled. Optimally, the mesh could be produced programmatically when the geometry is aimed for UDEC, to avoid the use of AutoCAD entirely. For cases where the model was initially generated in AutoCAD, the python library "*dxfgrabber*" was used to read DXF files. Lastly, each line was imported in UDEC using FISH (programming language embedded in ITASCA software) to generate the masonry units (Fig. 18(d)).

4.2. Mortar and damage group assignment

In the numerical model, the mortar was assigned by calculating the area of each element. So, if an element is smaller than a predefined value, then it is assigned to the "Mortar" group (Fig. 19(a)). The predefined value is equal to the square of the largest (vertical or horizontal) side of the mesh element, see (Eq. (23)).

$$\text{If } Area \leq (length)^2, \text{ then assign the element to the "}Mortar\text{" group} \tag{23}$$

Moreover, the mortar may be cracked, which needs to be reflected in the "*Mortar*" group. If this is the case, then the masked watershed segmentation (without padding) is used to extract the coordinates. Whenever a pixel with a damage label is detected, a FISH command is generated that re-assigns the "*Mortar*" element that contains the specified coordinates to the "*Damage*" group, see (Fig. 19(b)). It is essential to mention that the coordinates obtained require adjustment, since the interface-contours measure gridlines instead of pixels:

$$\text{If WS}(i,j) = -2 \text{ or WS}(i,j) = -3 : \text{Extract adjusted xy coordinates} \tag{24}$$

$$\begin{aligned} x &= (j + 0.5) \times scale \\ y &= (ymax - (i + 0.5)) \times scale \\ ymax &= max(y) \end{aligned} \tag{25}$$

The method mentioned earlier used to assign the damage, requires an excessive amount of time due to the large number of pixels detected (i.e. 315 s for 71,030 entries on a laptop with i7-9750h, Fig. 19(b)). For that reason, the accuracy of pixel extraction is limited to a preferred area (i.e. $Acc = 5px$). Thus, rejecting new entries if they are in proximity to an already extracted location, which reduces dramatically the computational time required (i.e. 18 s for 4386 entries, Fig. 20(a)). The method used to verify the distance is by creating a test-array of equal size to the original image where all coordinates extracted are marked by the

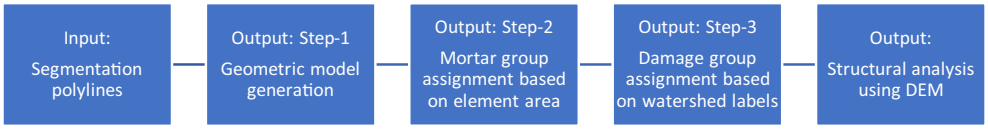

Fig. 17. Workflow of the numerical model generation (Output).

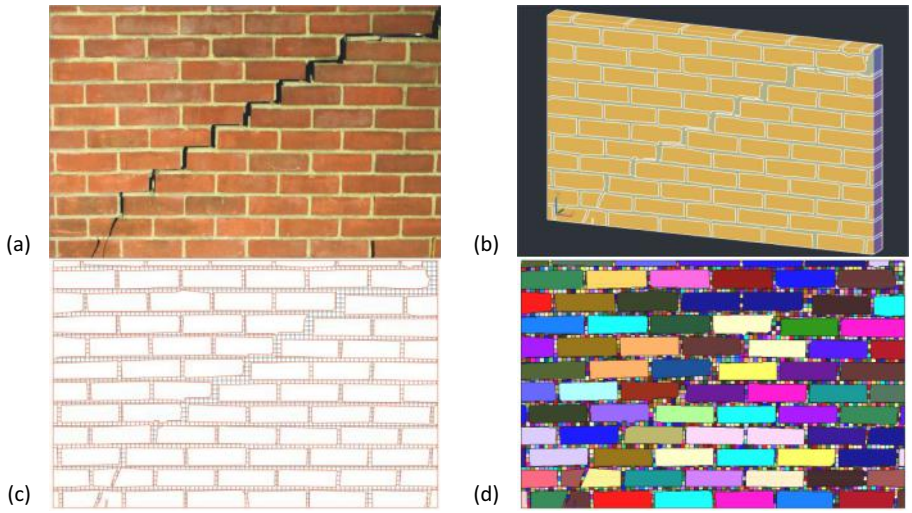

Fig. 18. Geometric model generation; (a): Image source; (b): AutoCAD 3D-model using blocks; (c): AutoCAD 2D-model using lines (detailed micro-modelling); (d): UDEC 2D-model using lines (detailed-micro-modelling).

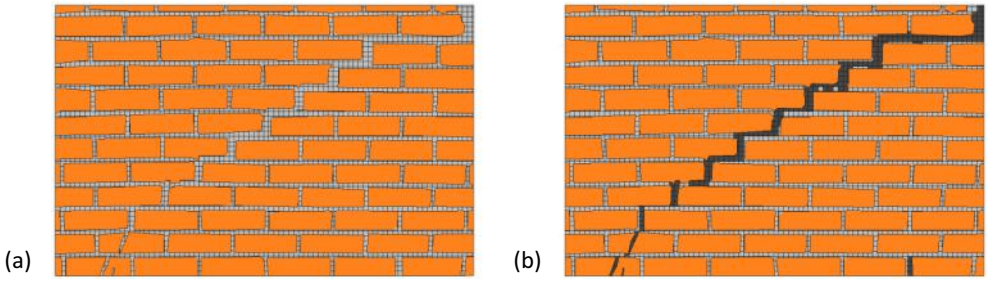

Fig. 19. Assigning mortar and damage (Orange: Brick, Grey: Mortar, Black: Damage); (a): Mortar comparing the element area; (b): Damage per pixel at each mortar element (71,030 entries – 315 s).

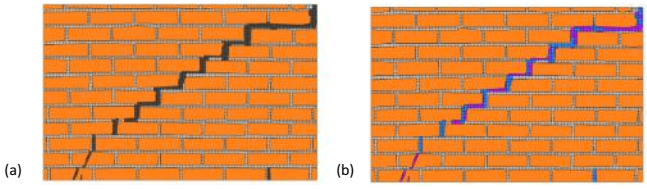

Fig. 20. Assigning damage using range (5-pixel range, Blue: Mortar damage, Purple: Block damage); (a): General damage (4386 entries, 18 s); (b): Multiple assignments giving priority to mortar-damage (4628 entries, 18 s).

preferred area of double the size of the accuracy (i.e. $TArray[i-5:i+5, j-5:j+5]=0$). Creating multiple test-arrays for each damage state allows to verify and assign multiple groups (Fig. 20(b)), which in turn can be used to assign different material and joint characteristics or to remove completely specified damage-groups. The assignment accuracy of multiple groups depends highly on the mesh size, the precision of the imported damage/mortar masks, and the applied order of the damage to the model (i.e. Fig. 20(b)), where the mortar-damage is given priority). For the tested accuracy, multiple damage assignments are not recommended since the mesh size is not sufficiently small, and due to inadequate accuracy of the imported (not generated) masks. The second method proposed is based on both; a range value targeting the centroid (Eqs. (26) & (27)) and the block that contains the specified coordinates (Eq. (25)), of mortar-elements only. The range of values used, in the numerical model, are provided below:

$$\begin{aligned} x1 &= (j+0.5-Acc)\times scale \\ x2 &= (j+0.5+Acc)\times scale \end{aligned} \tag{26}$$

$$\begin{aligned} y1 &= (ymax-(i+0.5-Acc))\times scale \\ y2 &= (ymax-(i+0.5+Acc))\times scale \end{aligned} \tag{27}$$

5. Numerical analysis of existing masonry walls

The geometry obtained using the proposed algorithm is compared with the idealised model to verify the proposed methodology's potential to be used in automated model generation. This section does not aim to predict the behaviour of a real structure. Instead, it tests the structure for similar behaviour, assuming a similar geometry is provided, while retaining the mechanical properties equal between all models. The numerical analysis also assumes an accurate model calibration, by following the typical modelling procedure and by using representative material characteristics, using information acquired from the literature. Moreover, the following generalisation values are suggested for the general definition of the geometric model aimed for numerical analysis:

$$Vertical\ Ratio = Horizontal\ Ratio = 0.1 \tag{28}$$

$$Min.Length = Mortar\ Thickness + 1\ (\text{in pixel size}) \tag{29}$$

The first model selected as a case study is the masonry arched-door (Fig. 21). The specific geometry was chosen as it introduces additional complexity to the analysis by including an arch and opening to the model. In this case, the mortar is represented as a zero-thickness interface since the simplified micro-modelling approach is typical for the numerical analysis of masonry structures. The vertical and horizontal generalisation-ratio used, to acquire the geometry, is equal to 0.1 with a 3-pixel minimum distance. Regarding the analysis, only deformable blocks are considered (although failure is expected to occur in the

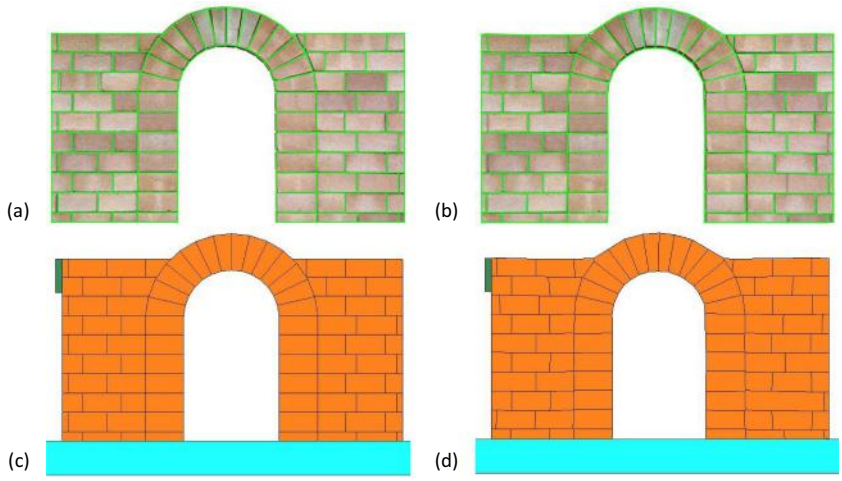

Fig. 21. Model geometry of Arched-Door; (a): Idealised model; (b): Generated model; (c): Idealised UDEC groups; (d): Generated UDEC groups.

mortar joints rather than in the masonry units). The source of the action affecting the structure is a horizontal velocity applied on a block pattern, see (Fig. 21(c) & (d)).

The maximum loading of the idealised and generated model is 1240 kN and 1360 kN respectively (+9.7%, Fig 22(a)), assuming a 500 mm depth. The maximum-load difference is possibly caused by the interlocking/hinging mechanism between the top-left block-row and the arch in the generated model, not present in the idealised case. A more accurate binarised-source would provide overall better results. However, the failure-pattern at 10 mm displacement is similar, with only minor differences (Fig. 22(b) & (c)), although the geometry between the two models is not identical (i.e. merged blocks, additional edges, etc.).

The second model for comparison is the damaged brick-wall, including the physical modelling of mortar (Fig. 23(a) & (b)). The pixels corresponding to damaged areas, detected from the image-processing of the original image, were directly inserted on the idealised model. This allows the comparison of the extracted geometry without considering the success rate of the defect detection. Furthermore, the damaged locations were removed to imitate the separation of material on the original image. The vertical and horizontal generalisation-ratio used, to simplify the geometry, is equal to 0.1 with a 5-pixel minimum distance. The analysis considers only deformable blocks.

Moreover, the mechanical properties of the joints are the same as in the previous case. However, the brick/mortar units have different properties, which are provided below. In this particular case, the mortar is consisted of individual triangles of 20 mm sides to reduce the computational cost for the analysis. More importantly, the triangular mesh was selected since it permits diagonal separation. In the future, segmentation of the mortar can be done using Voronoi elements or

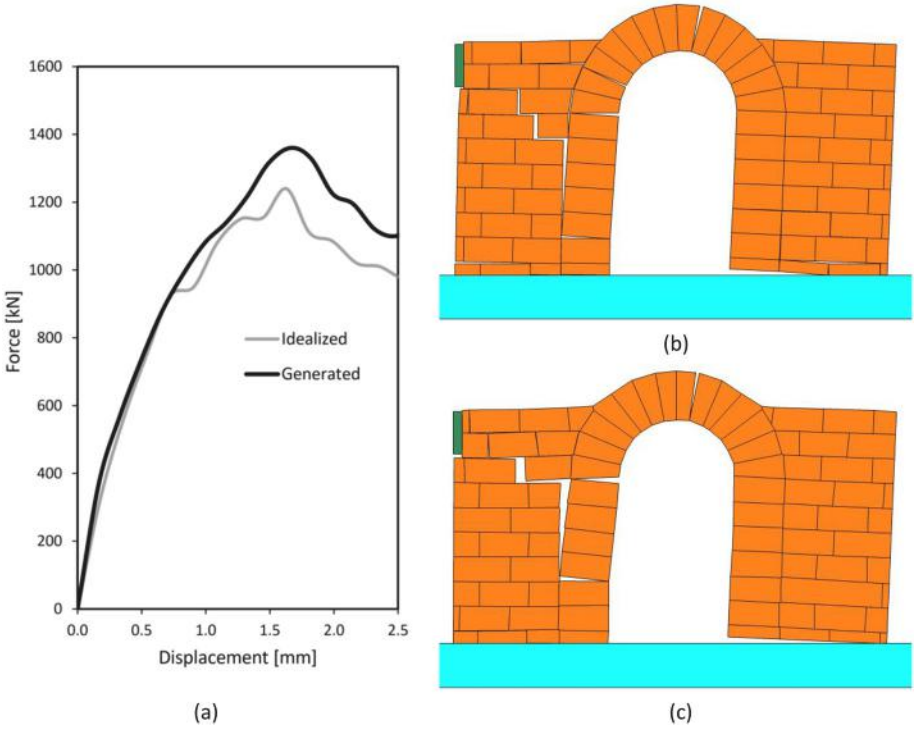

Fig. 22. (a): Force-Displacement graph of Arched-Door under horizontal loading; (b): Idealised model after 10 mm horizontal displacement; (c): Generated model after 10 mm horizontal displacement.

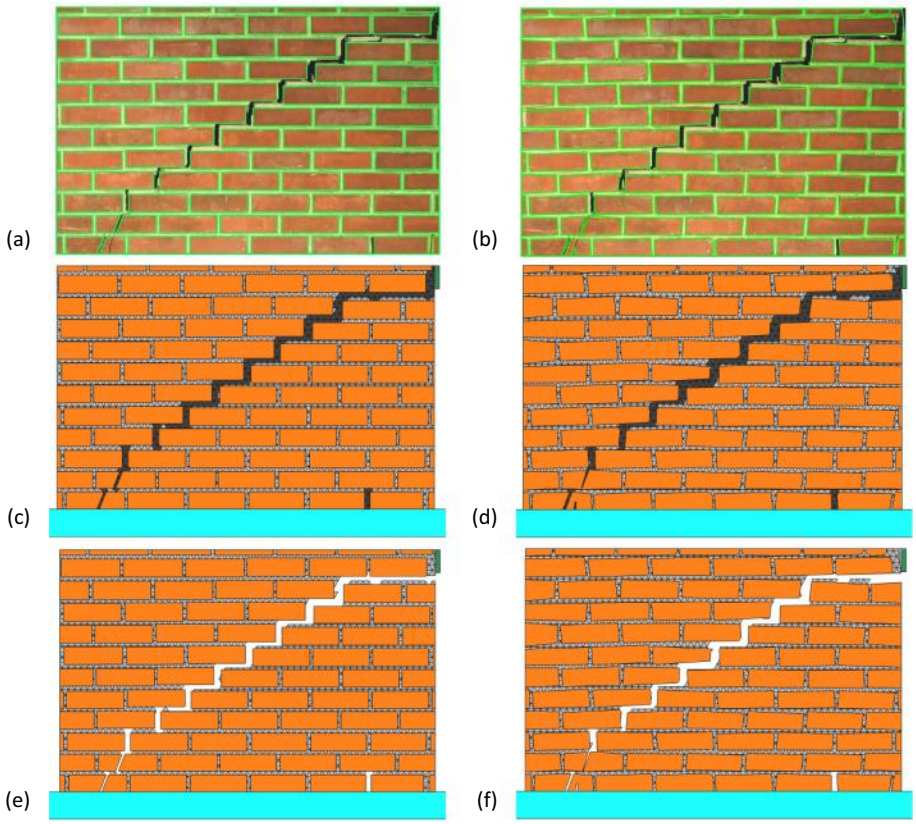

Fig. 23. Model geometry of Damaged-Wall; (a): Idealised model; (b): Generated model; (c): Idealised UDEC-groups; (d): Generated UDEC-groups; (e): Removed material of Idealised model; (f): Removed material of generated model.

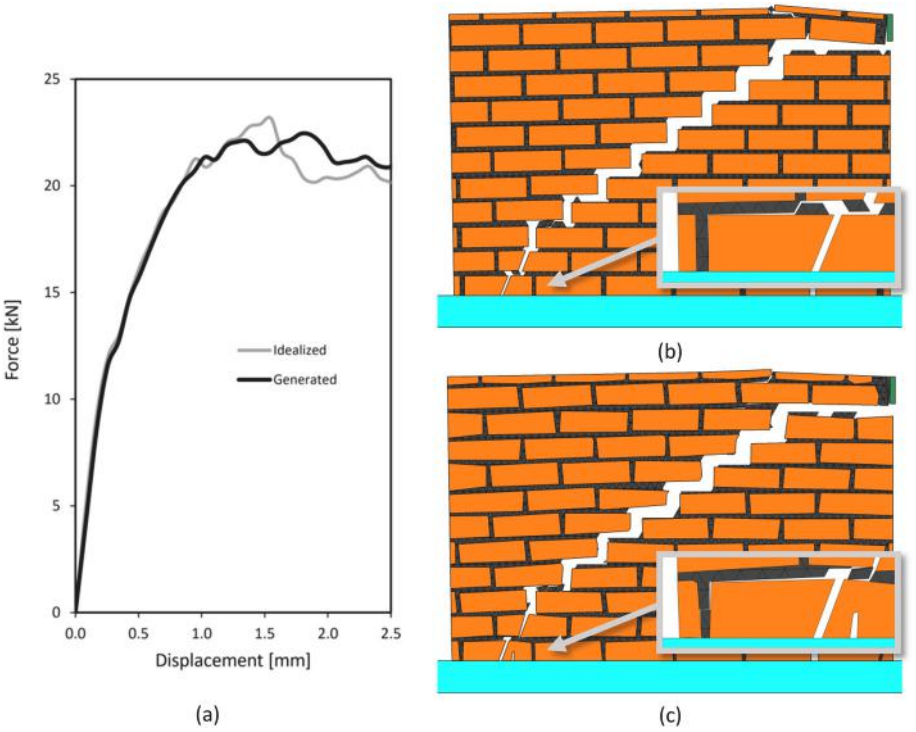

Fig. 24. (a): Force-Displacement graph of Damaged-Wall under horizontal loading; (b): Idealised model after 10 mm horizontal displacement; (c): Generated model after 10 mm horizontal displacement.

alternative shapes. The source of the action affecting the structure is a horizontal velocity applied on a block pattern, see (Fig. 23(c) & (d)).

The maximum loading of the idealised and generated model is 23.15 kN and 22.45 kN respectively (−3.0%, Fig. 24(a)), assuming a 102.5 mm depth. Moreover, the failure-pattern at 10 mm displacement is similar, following the pre-existing damage and causing separation at the top-right and bottom-left corners (Fig. 24(b) & (c)). However, the shape of the mesh has not been optimised for numerical analysis and may interfere with the results due to local hinging phenomena. The investigation of the mesh type and size is outside the scope of this research. Nonetheless, the expected behaviour is observed in both cases (i.e. separation at damaged locations).

6. Conclusions

The proposed methodology proves to be a time-efficient and robust system of acquiring the geometric shape of a masonry structure for numerical analysis, especially in the case of complex structures where a significant effort is required to create the numerical model. Any type of masonry construction is supported, as long as an adequate source is provided (i.e. ashlar, rubble, dry-joint, mortared-joint, etc.). Another possible application of the proposed methodology is the automated assessment of numerical analysis results (by comparing the displacements of the elements with the coordinates of the objects acquired during the feature-extraction). Especially during evaluation of an inverse analysis or assessment of proposed models aimed to predict crack propagation [32]. Other uses of the algorithm may include the generation of precise CAD designs of real structures, in a timely-efficient manner. Additionally, it provides a use-case for state-of-the-art research in feature-detection and segmentation. However, a reliable method of feature-detection is required for optimal results.

From the analysis of results, the efficiency of the methodology depends highly on the accuracy of the geometry extracted and the number of lines used to describe the same geometry. The generated model has the potential to provide more accurate results, assuming that it resembles the shape of the structure in more detail than the idealised geometry. Nevertheless, the user must ensure that no interlocking between the blocks is present, where it is not anticipated, due to unnecessary complexity. This is resolved by the proposed generalisation-algorithm, which reduces the number of edges of the blocks in the model. For general use, the values proposed in the "*Numerical Analysis*" section (Eqs. (28) & (29)) demonstrate adequate results, which are taking advantage of the dynamic-adjustment to automate the assignment (Eq. (14)). If necessary, a larger generalisation-ratio can be used to simplify the model further. However, the physical modelling of mortar assists on that regard since there is no direct interaction between the masonry blocks.

The main limitation of the methodology is the accuracy of the original binarised-image used to produce the watershed segmentation. The proposed approach is aimed to be used in combination with state-of-the-art image-processing techniques to improve the overall precision of the feature-extraction. Additionally, the methodology is limited to 2D model generation. Although, simple 3D models can be generated by assuming a standard depth value across a single plane (i.e. Fig. 18(b)).

Regarding the discrete element analysis, the mortar mesh has not been optimised for the current use-case. Further investigation is required to optimise the mesh type and the size of each element. A different mesh type (i.e. Voronoi, [28]) and equal or smaller size than the mortar-thickness may prove more efficient in highly stressed areas. However, it may also increase the computational effort required. Moreover, the joint characteristics consider a single global assignment, where the mortar-to-mortar interface is equal to the block-to-mortar (only valid for the damaged-wall example). A more accurate representation of the joints could provide a more realistic outcome. Lastly, the material, joint, and damage characteristics require further investigation and should be calibrated to experimental data for a realistic depiction of the analysis results.

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

This work was partially funded by European Union's Framework Programme for Research and Innovation Horizon 2020, under the H2020-Marie–Skłodowska Curie Actions-COFUND scheme (Grant Agreement ID: 754511), and from the banking foundation Compagnia di San Paolo.

References

[1] C. Altuntas, S. Hezer, S. Kirli, Image based methods for surveying heritage of masonry arch bridge with the example of dokuzunhan in konya, Turkey 3 (2017) 13–20, https://doi.org/10.5281/zenodo.438183.
[2] 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.
[3] P.G. Asteris, V. Plevris, V. Sarhosis, L. Papaloizou, A. Mohebkhah, P. Komodromos, Numerical modeling of historic masonry structures, in: Austin De Marco, Kayla Wolfe, Christina Henning, B. Carbaugh (Eds.), Handbook of Research on Seismic Assessment and Rehabilitation of Historic Structures, Engineering Science Reference (an imprint of IGI Global), United States of America, 2015, pp. 213–256, https://doi.org/10.4018/978-1-4666-8286-3.ch007.
[4] 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.
[5] 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.
[6] D. Brackenbury, M. Dejong, Mapping Mortar Joints in Image Textured 3D Models to Enable Automatic Damage Detection of Masonry Arch Bridges, Tampere, Finland, 2018. URL, http://programme.exordo.com/icccbe2018/delegates/presentation/344/ (accessed 27.1.21).
[7] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-8 (1986) 679–698, https://doi.org/10.1109/TPAMI.1986.4767851.
[8] 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.
[9] F. Cluni, D. Costarelli, A.M. Minotti, G. Vinti, Enhancement of thermographic images as tool for structural analysis in earthquake engineering, NDT and E Int. 70 (2015) 60–72, https://doi.org/10.1016/j.ndteint.2014.10.001.
[10] P.A. Cundall, A computer model for simulating progressive large-scale movements in blocky rock systems, in: Proocedings of the Symposium of the International Society of Rock Mechanics, Nancy, France, 1971 p. Vol. 1., Paper No. II-8. URL https://www.scirp.org/(S(i43dyn45teexjx455qlt3d2q))/reference/ReferencesPapers.aspx?ReferenceID=1230197 (accessed 27.1.21).
[11] 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. (2021), https://doi.org/10.1016/j.autcon.2021.103606.
[12] D.H. Douglas, T.K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica, Int. J. Geogr. Inform. Geovisualiz. 10 (1973) 112–122, https://doi.org/10.3138/fm57-6770-u75u-7727.
[13] E. Erdogmus, B. Pulatsu, B. Can, K. Ozkan, Analysis of the Last Standing Arch of the Roman Aqueduct at Blaundos, in: P.B. Dillion, F.S. Fonseca (Eds.), 13th North American Masonry Conference. Salt Lake City, Utah, 2019, pp. 483–493. URL, https://www.researchgate.net/publication/334001391_Analysis_of_the_Last_Standing_Arch_of_the_Roman_Aqueduct_at_Blaundos (accessed 27.1.21).
[14] T. Forgács, V. Sarhosis, K. Bagi, Influence of construction method on the load bearing capacity of skew masonry arches, Eng. Struct. 168 (2018) 612–627, https://doi.org/10.1016/j.engstruct.2018.05.005.
[15] T. Forgács, V. Sarhosis, K. Bagi, Minimum thickness of semi-circular skewed masonry arches, Eng. Struct. 140 (2017) 317–336, https://doi.org/10.1016/j.engstruct.2017.02.036.
[16] J. Ghaboussi, R. Barbosa, Three-dimensional discrete element method for granular materials, Int. J. Numer. Anal. Methods Geomech. 14 (1990) 451–472, https://doi.org/10.1002/nag.1610140702.
[17] T. Hinks, H. Carr, L. Truong-Hong, D.F. Laefer, Point cloud data conversion into solid models via point-based Voxelization, J. Surv. Eng. 139 (2013) 72–83, https://doi.org/10.1061/(asce)su.1943-5428.0000097.

[18] 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 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.
[19] Itasca, ITASCA Consulting Limited [WWW Document], URL, https://www.itasca.co.uk/, 2019. accessed 22.9.20.
[20] A. Karagianni, G. Karoutzos, S. Ktena, N. Vagenas, I. Vlachopoulos, N. Sabatakakis, G. Koukis, Elastic properties of rocks, Bull. Geol. Soc. Greece 43 (2010) 1165–1168, https://doi.org/10.12681/bgsg.11291.
[21] 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.
[22] P.B. Lourenço, Computational strategies for masonry structures. PhD Thesis, Delft University Press, 1996. URL, http://www.narcis.nl/publication/RecordID/oai:tudelft.nl:uuid:4f5a2c6c-d5b7-4043-9d06-8c0b7b9f1f6f (accessed 27.1.21).
[23] 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.
[24] P. Morer, I. de Arteaga, A. Ortueta, A low-cost photogrammetric methodology to obtain geometrical data of masonry arch bridges, J. Archit. Conserv. 19 (2013) 246–264, https://doi.org/10.1080/13556207.2013.869974.
[25] 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.
[26] U. Ramer, An iterative procedure for the polygonal approximation of plane curves, Computer Graphics and Image Processing 1 (1972) 244–256, https://doi.org/10.1016/S0146-664X(72)80017-0.
[27] V. Sarhosis, S.W. Garrity, Y. Sheng, Influence of brick-mortar interface on the mechanical behaviour of low bond strength masonry brickwork lintels, Eng. Struct. 88 (2015) 1–11, https://doi.org/10.1016/j.engstruct.2014.12.014.
[28] V. Sarhosis, J.V. Lemos, A detailed micro-modelling approach for the structural analysis of masonry assemblages, Comput. Struct. 206 (2018) 66–81, https://doi.org/10.1016/j.compstruc.2018.06.003.
[29] V. Sarhosis, Y. Sheng, Identification of material parameters for low bond strength masonry, Eng. Struct. 60 (2014) 100–110, https://doi.org/10.1016/j.engstruct.2013.12.013.
[30] 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. URL, https://www.isprs.org/proceedings/XXXVII/congress/5_pdf/99.pdf (accessed 27.1.21).
[31] S. Suzuki, K. Abe, Topological structural analysis of digitized binary images by border following, Comp. Vision, Graphics Image Processing 30 (1985) 32–46, https://doi.org/10.1016/0734-189X(85)90016-7.
[32] 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.
[33] S. Tiberti, G. Milani, 3D voxel homogenized limit analysis of single-leaf non-periodic masonry, Comput. Struct. 229 (2020) 106186, https://doi.org/10.1016/j.compstruc.2019.106186.
[34] 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.
[35] 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.
[36] 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.
[37] 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.
[38] Y. Zhang, L. Macorini, B.A. Izzuddin, Numerical investigation of arches in brick-masonry bridges, Struct. Infrastruct. Eng. 14 (2018) 14–32, https://doi.org/10.1080/15732479.2017.1324883.