

Towards rapid and automated vulnerability classification of concrete buildings

Lissette Iturburu^{1†}, Jean Kwannandar^{1‡}, Shirley J. Dyke^{1,2§}, Xiaoyu Liu^{2§}, Xin Zhang^{1†} and Julio Ramirez^{1§}

1. *Lyles School of Civil Engineering, Purdue University, West Lafayette, IN 47907, United States*

2. *School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, United States*

Abstract: With the overwhelming number of older reinforced concrete buildings that need to be assessed for seismic vulnerability in a city, local governments face the question of how to assess their building inventory. By leveraging engineering drawings that are stored in a digital format, a well-established method for classification reinforced concrete buildings with respect to seismic vulnerability, and machine learning techniques, we have developed a technique to automatically extract quantitative information from the drawings to classify vulnerability. Using this technique, stakeholders will be able to rapidly classify buildings according to their seismic vulnerability and have access to information they need to prioritize a large building inventory. The approach has the potential to have significant impact on our ability to rapidly make decisions related to retrofit and improvements in our communities. In the Los Angeles County alone it is estimated that several thousand buildings of this type exist. The Hassan index is adopted here as the method for automation due to its simple application during the classification of the vulnerable reinforced concrete buildings. This paper will present the technique used for automating information extraction to compute the Hassan index for a large building inventory.

Keywords: object detection; segmentation; U-Net; vulnerability; hazards; earthquakes; reinforced concrete building; inventories and databases

1 Introduction

Reducing the loss of life and economic and social impact in the aftermath of an earthquake goes hand in hand with reducing the vulnerability of buildings to the hazard (Loos *et al.*, 2020). Seismically vulnerable buildings include non-engineered construction and buildings with design deficiencies such as soft-story and vertical stiffness irregularities. Seismic assessment methods that can capture such deficiencies range from empirical assessment approaches such as on-site screenings and scoring methods to more accurate and detailed methods such as non-linear time history analysis (Kassem *et al.*, 2020). But the high quantity of buildings in a given city can make it challenging to allocate sufficient funds for retrofit programs in buildings where it is more necessary (Loos *et al.*, 2020). Cities with large building inventories will appreciate the potential benefits of automation and

have access to an automated ranking system that will expose those buildings in their inventory that need to be carefully studied and, if necessary, retrofitted before a major seismic event tests the built environment.

Some researchers have explored methods to assist with building vulnerability classification. Yeum *et al.* (2016, 2018) designed definitions and corresponding image classifiers to classify images of buildings into “collapsed” and “non-collapsed” based on images containing a building overview. As categorizing the buildings from the information generated through a single image could cause bias, researchers have also developed methods to fuse the information from more than one single building image. For example, with data from hurricane surveys conducted in 2020, Lenjani *et al.* (2020) developed a Bayesian-based method to classify the post-event condition of a house. Later, Liu *et al.* (2022) enhanced this method to classify the damage state of a set of building images. An example of recently developed machine learning assisted technology to help manage city-scale building inventories is BRAILS (Wang *et al.*, 2019), where building properties are extracted from Google imagery and local government agencies to develop building information models (BIM) and store information relevant to risk management. Another example is SURF (Yu *et al.*, 2019), a building classifier, that uses Google imagery to sweep through cities to obtain geometric information such as building

Correspondence to: Lissette Iturburu, Lyles School of Civil Engineering, Purdue University, West Lafayette, IN 47907, United States

Tel: +1-765-430-7853

E-mail: liturbur@purdue.edu

[†]PhD Student; [‡]M.Sc; [§]PhD

Supported by: US National Science Foundation under Grant No. NSF- OAC-1835473

Received August 5, 2022; Accepted March 13, 2023

roof shapes and buildings with a potential soft-story mechanism, which are then used to populate building characteristics in BRAILS (Wang *et al.*, 2019). Other efforts include the use of information fusion and machine learning techniques to combine data from multiple sources and create a building inventory (Wang *et al.*, 2021), and be able to analyze the several social dimensions that affect a city after a natural hazard (Szczyrba *et al.*, 2021).

The Hassan index (Hassan and Sozen, 1997) is selected for this study as the basis for classifying a building with respect to seismic vulnerability. The method is simple, widely studied and used, and only requires the floor plan area, the area of columns, and the area of concrete and/or masonry walls. We now aim to empower communities to make use of it through automation and artificial intelligence (AI), while scaling it to the city-level. The computer vision-based method is developed to extract information from the building structural drawings and such information can be then analyzed to determine the buildings' vulnerability. Currently, the Hassan index calculation process requires considerable human effort. Based on the authors' experience, manually generating the information from drawings takes around 1 hour per drawing, which could be made significantly more efficient with machine learning methods.

Now, let us examine the enabling role of computer vision to extract information from building drawings. Most of the popular applications of computer vision centered on the analysis of the content of natural scene images (Figs. 1(a) and 1(b)). Structural framing plan images (Fig. 1(c)) belong to the category of non-natural images along with graphic images, synthetic scene statistics, screen content images (Zheng *et al.*, 2019) and images with text such as those of book pages (Xu *et al.*, 2016; Li *et al.*, 2018). This distinction from natural scene images is of relevance because the semantic information that define the content of these images such as local and global features show different characteristics than those of natural scene images, for which examples can be found in the work of Min *et al.* (2021).

The capability to read structural drawings using classical computer vision methods was very limited in the

years before the boom of AI deep learning. Researchers had to hard code the rules to make the algorithm identify elements such as walls, columns, and dimension lines in the different framing plan configurations and text styles, e.g., So *et al.* (1998), Gimenez *et al.* (2015), Lu and Lee (2017). Nowadays, deep learning neural networks are used to identify these patterns in structural and architectural framing plans to build 3D BIM models (Zhao *et al.*, 2021; Lv *et al.*, 2021), for recognition of special objects such as in piping drawings (Zhang *et al.*, 2020) or structural components (Zhao *et al.*, 2020), and for simplifying extensive complicated architectural drawings into simple floor plans (Kim *et al.*, 2021). The analysis of structural framing plans has remained limited. The work of Lv *et al.* (2021) uses semantic segmentation to identify architectural elements such as partition walls, doors, windows along with spaces such as living room, balconies, and text detection to understand the space use in a building. Zhao *et al.* (2021) uses a combination of object detection to identify location of structural drawings and classical computer vision techniques to extract coordinates of the different structural elements in a building and then build a 3D computational structural model. All of these works have been centered around the general characteristics of architectural and structural drawings. Our study focuses only on the characteristics present in framing plans of concrete buildings. Next, let's examine the Hassan index and how to empower communities to classify large inventories of reinforced concrete buildings for the purpose of performing seismic vulnerability classification in an automated way.

The Hassan index (Hassan and Sozen, 1997) is a widely studied method for the classification of seismic vulnerability of concrete buildings. The index is computed by calculating the areas of the load-bearing structural elements in a building: columns and walls with respect to the total area of the floors above the critical story. For the columns, the column index (CI) is calculated as presented in Eq. (1), where $\sum A_c$ is the sum of the cross-sectional areas of columns on the first floor. For the wall index (WI) Eq. (2), $\sum A_w$ is the sum of cross-sectional areas of reinforced concrete structural walls on the first floor, $\sum A_{mw}$ is the sum of cross-sectional areas of infill masonry walls in the first level. In both equations,

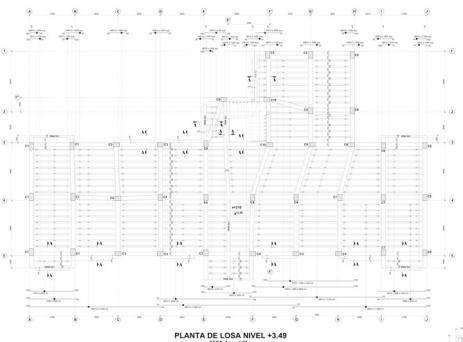


Fig. 1 (a) Bridge image from the ImageNet repository (Deng *et al.*, 2009), (b) building image from Datacenter Hub (Chungwook *et al.*, 2018), and (c) a structural framing plan from our inventory

and $\sum A_F$ is the sum of floor areas from the first floor and up, without including the ground-floor. The Hassan index (often called, the priority index) is calculated using Eq. (3), by adding the column index with the minimum of the wall indexes in each horizontal direction. By using Eq. (4), the curve classifies vulnerable and non-vulnerable buildings, as these are defined by Hassan and Sozen (1997). It is to be noted that Eq. (4) needs to be calibrated for each city and the coefficients set in this linear inequality should change accordingly as it is discussed in the work by Pujol *et al.* (2020).

$$CI = \frac{\frac{1}{2} \sum A_c}{\sum A_F} \quad (1)$$

$$WI = \frac{\sum A_w + \frac{1}{10} \sum A_{mw}}{\sum A_F} \quad (2)$$

$$HI = CI + \min (WI_x, WI_y) \quad (3)$$

$$WI + CI < 0.25\% \quad (4)$$

Similar scoring indexes like the Hassan index exist in the literature (Lv *et al.*, 2021), and examples include the “wall index” in Europe for load-bearing masonry and RC shear walls (Yeum *et al.*, 2016, 2018), and the “capacity index” (Kassem *et al.*, 2020) that considers plan irregularities, column length and workmanship. The “performance score index” (Kassem *et al.*, 2020) considers material quality, steel corrosion, vertical and horizontal irregularities, ground conditions, depth of foundation, seismicity, and other parameters (Kassem *et al.*, 2020). These methods require the review of architectural, structural, and geotechnical information, resulting in increased effort. The Hassan index’s simplicity has attracted attention from numerous researchers throughout the years (Liu *et al.*, 2022; Wang *et al.*, 2019). Many researchers that have further investigated the index by (1) adding additional structural parameters and parameters related to the seismic hazard such as the peak ground acceleration (Pujol *et al.*, 2020), or by (2) adding weights to the CI and WI parameters to modify the classification model and make the threshold more robust. Pujol *et al.*, (2020) explains that such parameters defining the slope that separates vulnerable buildings are less critical than the local quality of the buildings and suggests that cities define a local threshold for a retrofit program. It must be noted that the Hassan index is meant to be calibrated with site-specific building data to be used as a classification tool one city at a time, reflecting the view expressed in Pujol *et al.* (2020). Moreover, evidence from past studies (Pujol *et al.*, 2020; Brzev *et al.*, 2017; Balbirnie *et al.*, 2021) clearly shows

that severe damage can be expected in buildings with lower CI and WI values.

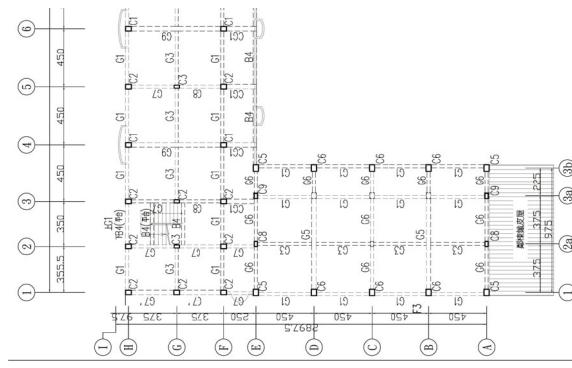
An example on the effect of seismic vulnerability boundaries in the building environment is discussed by Jury and Ferner (2015), for New Zealand. The new building standard index (%NBS) is an index used in New Zealand for classifying a building as seismically vulnerable. The New Zealand Code defines a building as earthquake prone when the building presents a %NBS of 34% or lower. After the 2010-2011 Canterbury earthquakes and with thousands of buildings with moderate to severe damage, the %NBS had a significant impact on the built environment in New Zealand as some insurance companies would insure buildings based on the %NBS limits (Marquis *et al.*, 2017). A discussion of the comparison between the %NBS and the Hassan index is offered by Balbirnie *et al.* (2021). This study mentions that when using the %NBS for the reinforced concrete buildings in Christchurch that collapsed in the 2010-2011 Canterbury earthquakes (buildings such as the Pyne Gould Corporation (PGC) and Canterbury Television (CTV) buildings), were not classified as “earthquake prone” and would have not been examined for retrofit, while if they were instead assessed using the Hassan index as classifier for that city, the buildings would have been classified as vulnerable, meaning the building would have undergone a structural evaluation. This comparison does not reach any conclusion regarding the superiority of an index over another, but rather signals the importance of calibration when defining a boundary for vulnerable buildings. The Hassan index is not a fit-for-all index used in classification, as Hassan and Sozen (1997) explain in the original paper. The index leaves more information out than it takes in and is no replacement for expert knowledge when assessing the vulnerability of a building. However, it is useful in the classification of those buildings that may need a post-event structural engineering inspection.

The AI offers the opportunity to automate the calculation of the Hassan index and extend its application to larger inventories of reinforced concrete buildings. The Hassan index was developed to rank reinforced concrete building vulnerability, and that is the vision of the authors of this paper when coupled with the technique presented herein. As more data is collected for more regions, the decision boundary can become more robust and adaptive for local conditions. One of the main challenges that remain when using available AI-based models involves the curation of a sufficiently large database. Open databases exist for architectural drawings. These include the Cubi-Casa5K (So *et al.*, 1998) dataset that contains 5,000 floor plan images of framing plans from the Finland architecture and the Rent3D (R3D) (Gimenez *et al.*, 2015) dataset with 215 floor plans. These are architectural framing plans centered on spatial distribution and lack structural elements. Currently, there is no openly available dataset for structural framing plans and researchers must build

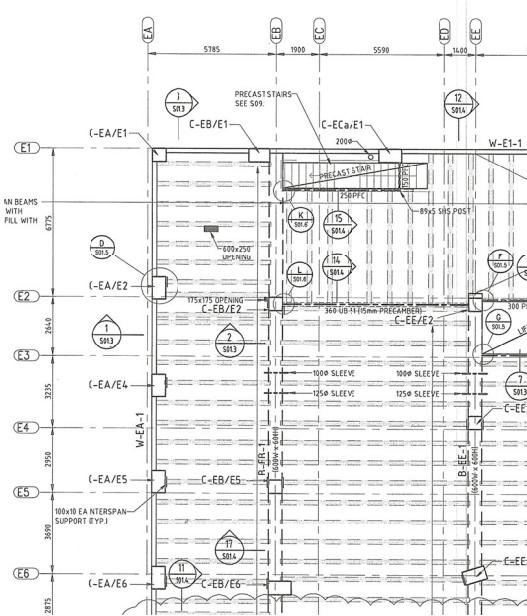
their own curated database (Zhao *et al.*, 2021).

The development of a database of structural drawings consists of first collecting structural framing plans from around the world. While structural framing plans have certain level of standardization, each company decides how to ultimately present their structural drawings in their own predefined style. The database used for training and validation of the model must contain as many of these styles as possible to be able to develop robust models against unseen styles (Zhao *et al.*, 2021). Examples of the framing plans in our inventory are presented in Fig. 2 and the general information of the dataset is listed in Table 1. To date, our drawing dataset consists of 245 structural framing plans. Fifteen percent of these framing plans belong to structural drawings that have been collected during reconnaissance missions after a major earthquake. Thirty five percent belong to drawings built before the 1990's, and fifty five percent of the 245 framing plans belong to buildings built within the last 20 years. Our dataset is divided into 4 different collections

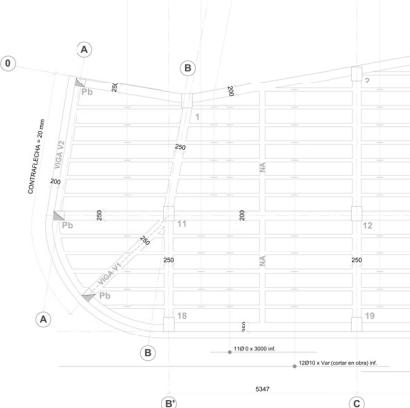
of structural drawing images, each from a different region of the world. A summary of the description of the contents of the dataset is shown in Table 1 and Table 2 along with some of the typical details found for columns, in Fig. 3, and walls, in Fig. 4. The dataset is intentionally built to contain a diverse arrange of patterns and styles for the details contained in a structural drawing. This dataset includes 2 types of floor plans: regular and irregular. Regular floor plans are defined by their shapes, which are nearly square or rectangular, and irregular floor plans include those that are non-rectangular such as L-shaped buildings or rectangular floor plans with re-entrant corners. The dataset also includes two types of drawing methods, using computer software and hand drawn. It also includes two types of sources; the first source type is scanned structural drawings as PDF files and the second type is camera-captured structural drawings as JPG files. The camera captured structural drawings are of varied quality, including images similar to those that might be obtained in the field with low contrast, folding lines,



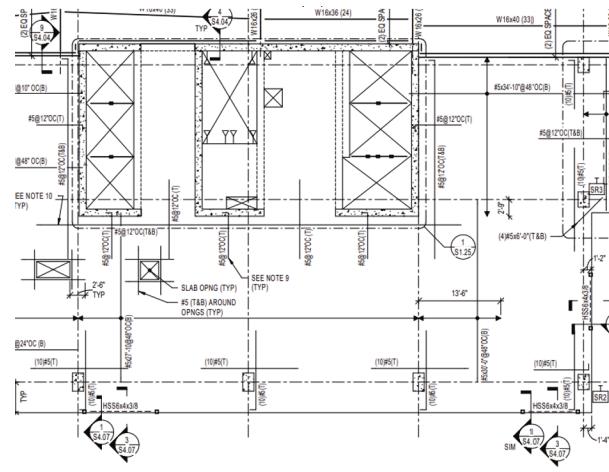
(a)



(c)



(b)



(d)

Fig. 2 Examples of sections from the structural drawings in our inventory examples from (a) Taiwan, China (b) Ecuador (c) New Zealand (d) United States

blurred, or ink stains.

In Dataset 1 (D1), about 60 % of the structural elements in drawings have no infill pattern, 30% have a solid fill pattern, and 10% have customized infill patterns. Also, 80% of the drawings in this dataset do not have any walls. The floor plans for type of this dataset are 60% regular. In addition, 70% of the drawings in this dataset have high resolution. All these drawings are computer-aided, and the source of the input is either a camera and scanned documents, representing 30% and 70%, respectively. In Dataset 2 (D2), about 55% of the drawings have customized fill patterns, 40% of the

drawings have no fill pattern, and 5% of the drawings have full block pattern. For this dataset, 5% of the drawings have no columns and only contain walls, 92% contain only columns and 3% contain columns and walls. Moreover, the drawings in this dataset have high resolution. In Dataset 3 (D3), about 60% of the drawings have customized fill pattern. Seventy percent of the drawings in this dataset D3, are hand drawn. In Dataset 4 (D4), about 80% of the drawings have fill pattern column features and customized fill pattern wall features. Furthermore, 80% of the drawings in this dataset have a high resolution. All drawings are 50% computer-aided and 50% manually drawn by hand.

Table 1 General information of the inventory

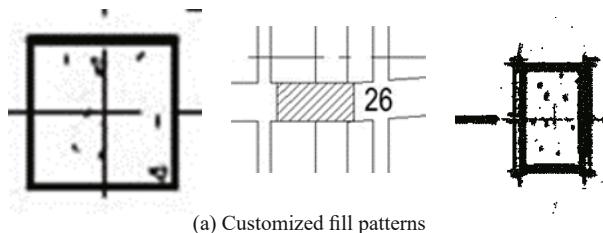
Label	Number of framing plans	Place of Origin
D1	57	Taiwan, China
D2	128	Ecuador; Indonesia
D3	54	New Zealand
D4	6	USA

Table 2 Drawing scale variation of the inventory

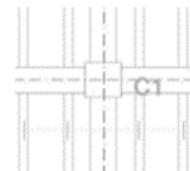
Drawing scale	Framing plans (%)
1:40	2
1:50	23
1:75	7
1:100	37
1:150	5
1:200	3
1/4":1'-0"	3
1/8":1'-0"	10
Not mentioned	10

2 Automation of the Hassan index: technical approach

The general outline of the automation of the calculation of the Hassan index for a given set of structural drawings is shown in Fig. 5. The process starts with the identification of an inventory of N number of buildings for the calculation of the Hassan index (Step 1). Next, the user needs to locate the critical framing plan from the structural drawing set and manually delineate the area of the structural drawing that contains the framing plan (Step 2). Then, by using deep learning segmentation models, columns and walls (Step 3) are automatically identified within the framing plan and printed on an image mask for the columns and an image mask for the walls. The next step is the generation of the framing plan based on the patches' prediction from the previous step and because segmentation predictions are often accompanied by noise in the prediction, we adopt automated standard computer vision techniques for denoising to remove the noise from the images (Step

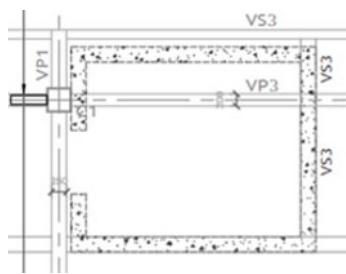


(a) Customized fill patterns



(b) No infill pattern

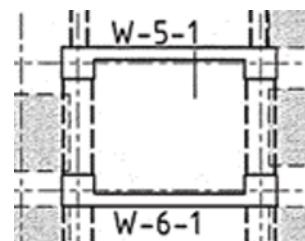
Fig. 3 Examples of fill patterns found in columns



(a) Customized fill pattern



(b) Full block pattern inside wall



(c) No infill pattern

Fig. 4 Fill patterns found in walls

4). The next step is to find the framing plan area, the structural elements are then enclosed in a perimeter using the concave hull algorithm developed by Kalinina *et al.* (2018), which then serves to calculate the area of the floor plan (Step 5). The floor plan area only includes the area that is confined by columns, and areas of cantilevers are neglected. With the area ratios of columns and walls calculated, we obtain the column and wall index pair for the building (Step 5). The user still needs to manually set the number of stories in each building, as this information can only be obtained from an elevation plan. Until this moment the user has given the relevant portion of framing plan as an input, and obtained the Hassan index parameters, the Hassan index and a mask of the columns and walls overlayed on the framing plan. Using a quick visual inspection, the user needs to accept or reject the model based on the overlay of the prediction mask on the framing plan. Then, we move on to the next building and start over from Step 2, until all available buildings in the city inventory have been read to define a decision boundary (Step 6). Once the decision boundary is determined, one can readily prioritize among the inventory of buildings being considered and generate a spatial vulnerability map (Step 7).

Because the Hassan index is calculated entirely using area ratios, calculating the true dimensions and areas of columns or walls is not necessary (i.e., the dimensions are not needed). Using segmentation, the pixels enclosed by the boundary of the columns and walls can be calculated. This area of pixels is divided by the area of the perimeter enclosed by the most exterior columns and walls. As a preprocessing step, all framing plans in the dataset were denoised using the Sauvola and Pietikäinen (2000) and NiBlack (1985) filters available in the Scikit Image (van der Walt *et al.*, 2014) library to remove salt-paper noise and shading present in some drawings. Then, 25 framing plans were set aside for the evaluation of the technique, and the remaining 220 framing plan were used for training the models.

2.1 Training of the deep learning models

Two prediction models were trained, one model for columns and one model for walls. The models were trained using the U-Net model developed by Ronneberger *et al.* (2015), and the focal Tversky loss (Abraham and Khan, 2019), which is popular in the literature for training on datasets with a relatively high class imbalance for segmentation problems (Yeung *et al.*, 2021). Hyperparameter tuning was performed for selecting the following parameters: the loss function coefficients, the learning rate, the images numbers in the batches and the number of layers in the neural network. For the training models, we used the focal Tversky loss with the delta factor of 0.10, which controls the false positives and negatives weighting, and the gamma factor of 0.75 which controls the degree of down-weighting of easy examples. Both parameters (alpha and gamma) were found to be the same for the columns model and for the walls model. An initial leaning rate of 1×10^{-3} with exponential decay of 0.90 was used for both models, decaying every 10,000 steps for the columns model and every 2,000 steps for the walls model. The columns dataset for training had a total of 65,000 images patches and the walls model had a total of 35,000 images patches. Each model had 3 layers for the encoder and the decoder architecture. It was found that more layers did not significantly affect the model accuracy. The model for the columns were trained on 50 epochs with a batch size of 10 grayscale images. For the walls model, the training was on 100 epochs, with a batch size of 5 grayscale images. The WI and the CI depend on the areas of columns, walls, and floor area. The Dice coefficient is selected as the metric for evaluating the prediction accuracy for the wall and column areas. The equation for this metric is given in Eq. (5), where true positives (TP), false positives (FP) and false negatives (FN) refer to these values at a pixel level.

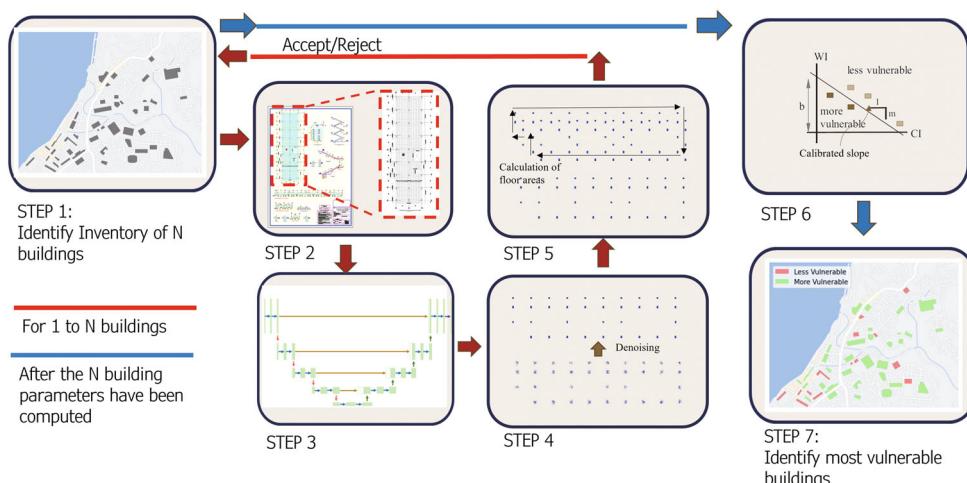


Fig. 5 Overview of the technical approach

$$\text{DICE} = \frac{2\text{TP}}{(2\text{TP} + \text{FP} + \text{FN})} \quad (5)$$

Drawings of the framing plan have an average size of approximately $10,000 \times 8,000$ pixels, ranging from $2,000 \times 4,000$ pixels to $15,000 \times 10,000$ pixels. Drafting details (when available) such as section cuts, column details and building information were removed from the sheet by manually cutting the sheet so that the final drawing input to the algorithm only included the gridlines and the framing plan. The structural drawings need to be divided into smaller pieces before training and smaller portions of the aforementioned section cuts of structural elements and floor systems can appear on an image and perhaps be mistaken for a column or a wall, if not removed. The result of this prediction could result in the prediction of a non-existent column or wall that will produce a floor plan different from the true floor plan. Columns and walls were annotated using standard segmentation tools (Labelbox, 2022). Examples of segmentation masks of the complete framing plan for two structural framing plans are shown in Fig. 6, columns are shown in blue and walls in green.

For training, the 220 framing plans were divided into smaller image patches as seen in Fig. 7. As seen in these figures, the colors of the framing plans have been inverted for training. The data was then augmented by flipping and rotating the images to generalize features for columns and walls and by applying filters that changed the image intensities (Buslaev *et al.*, 2020). It was found in our early trials that the tone of lines in a framing plan would affect the result predictions. The columns model used patches with a size of 512×512 pixels and an overlap of 20% among image patches, and the wall model

uses patches with 640×640 pixels and an overlap of 10% among image patches. Because a structural framing plan often has empty spaces in between structural elements, partitioning of the framing plans into smaller images results in a large quantity of images that contain no information. Thus, images having zero RGB intensity (0,0,0) were filtered out of the training in advance, as they have a detrimental effect on the training model. We found that such images comprise approximately 25% of the images for the columns model and approximately 15% of the images for the walls model, based on the total image patches. Including these images during training would cause the training to experience complete batches of images with no information to learn, and ultimately slow down the training. At this stage, the 220 framing plans patches are then randomly divided into training, validation and testing sets using a split of 70%, 20% and 10%, respectively.

During training, the model had to overcome two types of class imbalances. The first type of class imbalance is related to the fact that the background occupies more than 95% of the image. As evidenced by the column index, even in small framing plans with big columns, the columns only occupy about 2% of an image containing the complete framing plan. The second type of class imbalance is that walls appear much less often in the training set than columns. For training the wall's prediction model, the number of columns was under-sampled by selecting one patch from every two patches, to obtain a more uniform composition of patches of walls, columns, and background information during training for this model. The dataset for this training was assembled as follows: 50% of patches contain structural walls and augmented walls, 30% of patches of images that contain columns and 20% of image patches that contain details

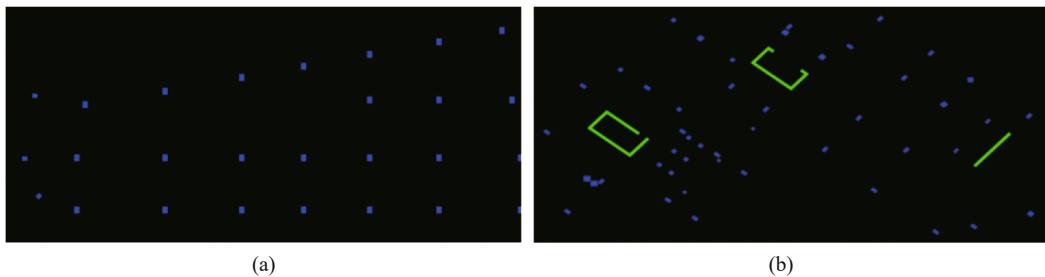


Fig. 6 Typical ground truth of a structural drawing containing (a) columns and (b) columns and walls

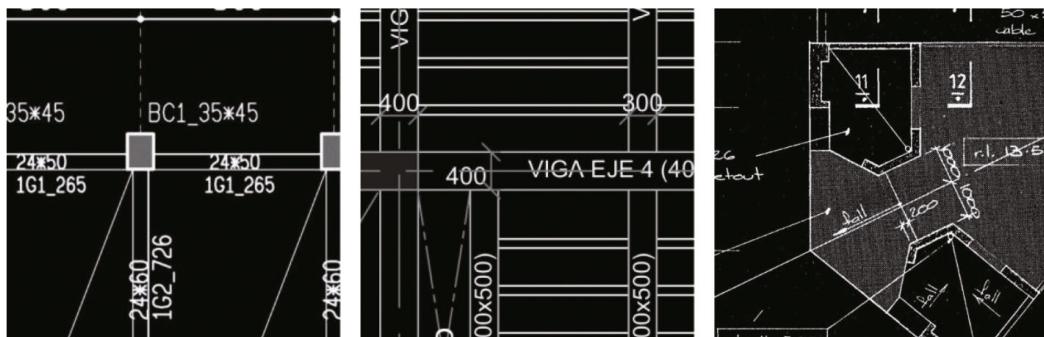


Fig. 7 Typical image patch

like stairs, gridlines, section lines, and floor openings. For a rectangular element to be considered a wall and not a column, a shape factor (defined as height over width) of 5 was used during the preprocessing.

Next, results of the column prediction model and the wall prediction model are evaluated for purposes of establishing possible limitations and how they could be

improved. The training and validations plots for the Dice coefficient metric are shown in the appendix Fig. A1 for both models. After 50 epochs, the validation accuracy is 60% for the column's prediction model. After 100 epochs, the validation accuracy is 58% for the wall's prediction model. Both models show to have a Dice coefficient close to 60% which can be explained by the

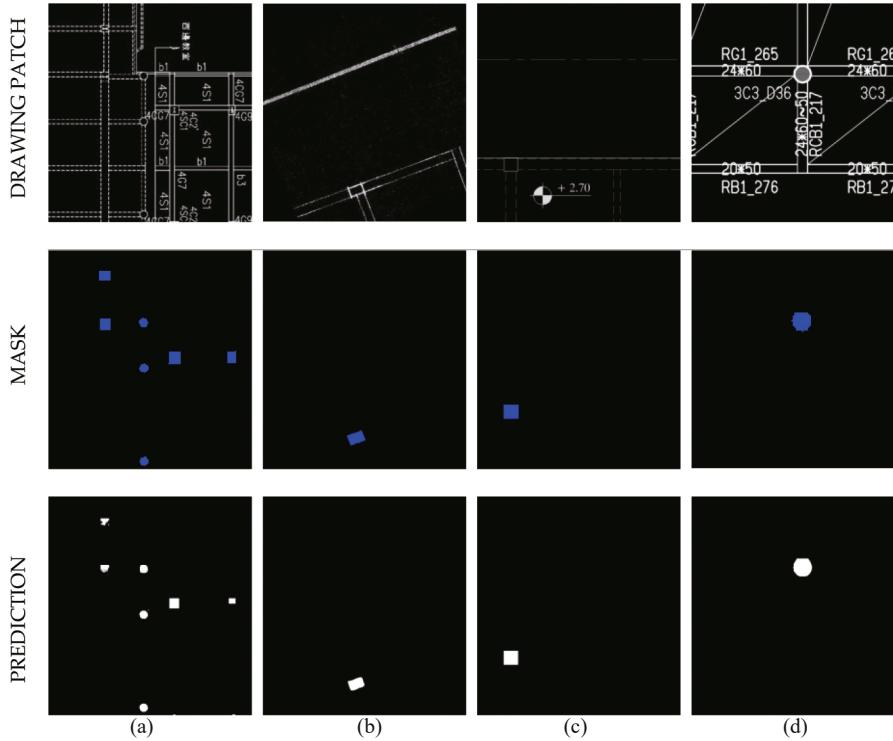


Fig. 8 Examples of patches, ground truth, predictions for unseen images of the testing set for examples (a), (b), (c), and (d). In blue: columns, in white: the column prediction

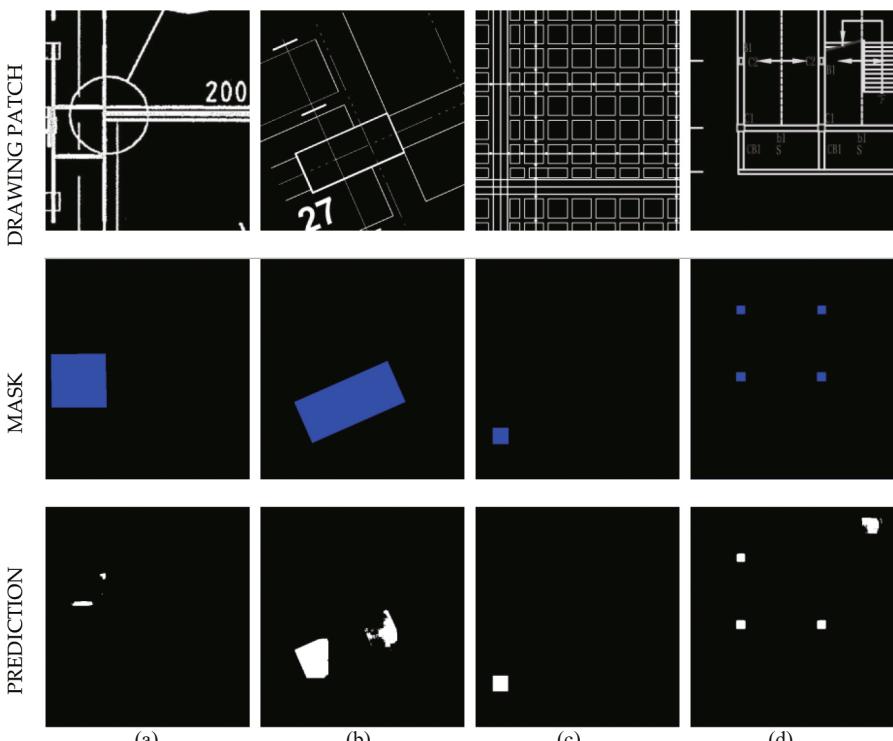


Fig. 9 Examples of patches, ground truth, predictions for images of the testing set for examples (a), (b), (c), and (d). In blue: the column's mask, in white: the column prediction

lack of semantic information around the edges of the patches and by the scale of the walls and columns with respect to the patch size as evidenced in the examples shown in Figs. 8 to 11.

In Figs. 8 to 11, a representative set of the segmentation results are shown for both the columns model and walls model. The ground truth (GT) is in blue color

for the columns and green color for the walls. Figure 8 shows that the model can predict columns in a variety of framing plan styles and with columns with different shapes and boundary conditions. Additionally, the model recognizes that symbols, Fig. 8(c), and text, Figs. 8(a) to 8(d), are not columns. On the other hand, Fig. 9 shows where the model has failed. Figures 9(a) and 9(b) shows

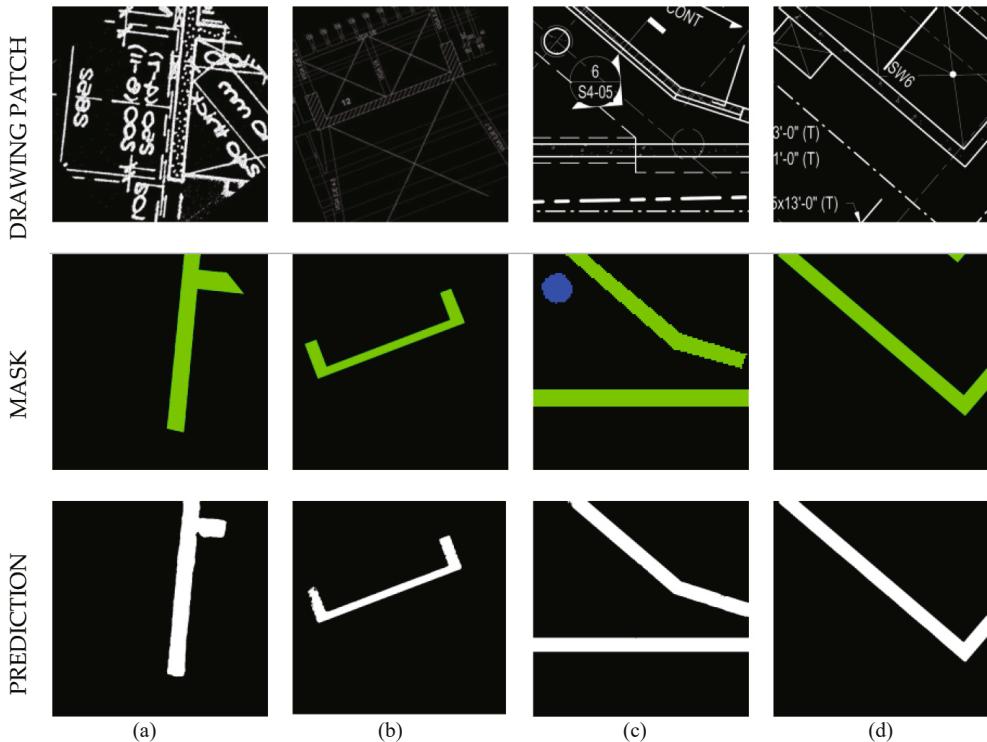


Fig. 10 Examples of patches, ground truth, predictions for images of the testing set for examples (a), (b), (c), and (d). In blue: the column's mask, in green: the wall's mask, in white: the wall prediction

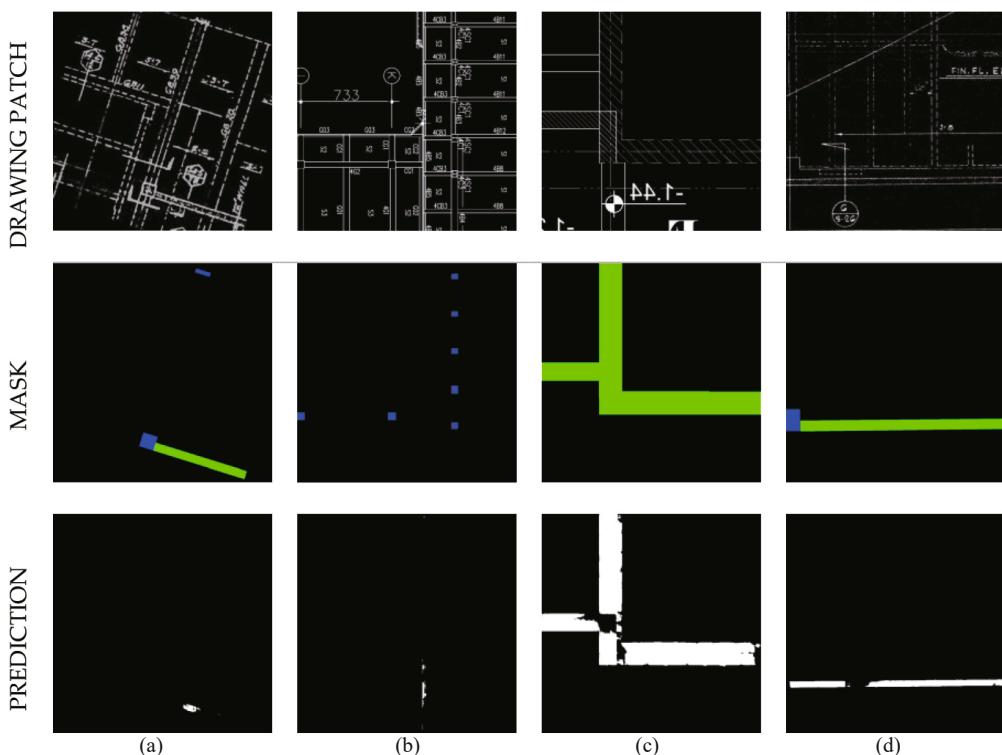


Fig. 11 Examples of patches, ground truth, predictions for unseen images of the testing set for examples (a), (b), (c), and (d). In blue: the column's mask, in green: the wall's mask, in white: the wall prediction

that the model does not recognize as columns the cases where there are non-concentric lines intersecting the columns. Figure 9(d) shows that near the stairs detail, a line that indicates the direction of the stairs forms a column-like shape, which was recognized as columns. Figures 10 and 11 show the prediction results for images from the testing set within the pool of drawings selected for training. Images in Fig. 10 show that walls are predicted in all cases where there is an infill pattern in rectangular regions. Figure 11 shows that the model has some problems predicting walls in images having lower resolution Fig. 11(a), and walls having lines intersecting in non-concentric regions Figs. 11(c) and 11(d). The result also shows in Fig. 11(b), that a complete wall predicted in a joint region.

3 Implementation of the approach

This section focuses on the computer vision techniques and algorithms used for the calculation of the Hassan index. These steps are generation of the framing plan from the prediction models, denoising of the generated framing plan, and calculation of areas for the columns, walls, and floor plan area. These steps have been previously briefly introduced in Fig. 5 from Steps 4 and 5.

3.1 Regenerating the framing plan

Once the selection of a framing plan is carried out, a sliding window with the same patch size used during training is passed through the full image every 100 pixels as input for the prediction model. The model will predict, for each pixel, a probability that is between 0 and 1. This value represents the probability that a pixel belongs to the column class for the column prediction model, or, similarly, to the wall class for the wall prediction model. When the probability for a pixel is higher than or equal to 0.5, the pixel is more likely to belong to the corresponding class. The closer this probability is to 1, the more confident this prediction is. We have selected a threshold of 0.70. Thus, when the probability is 0.70 or higher, then the result for that pixel is rounded up to 1, indicating it falls into the corresponding class. Otherwise, the result is set to 0, indicating it is not in the corresponding class. Cropping the framing plans in small patches is the standard practice for deep learning-based methods for semantic segmentation because of the limitations of the GPU memory for accommodating deep learning operations. Therefore, overlapping the predictions patches (Messaoud *et al.*, 2022), averaging (Müller *et al.*, 2020), clipping (Huang *et al.*, 2018) and eliminating the smaller areas (Kestur *et al.*, 2019) in the prediction is used herein during this stage for improving the final prediction. The prediction for the columns model gives an image of 512×512 pixels and then the image is clipped 100 pixels along the four borders of the image, with a final size of 312×312 pixels. The

prediction for the walls model gives an image of 640×640 pixels, that is clipped around the edges resulting in an image of 440×440 pixels for the walls model. The patches are then overlapped, when at least 50% of these patches are overlapped for the columns model of the overlapped patches and walls prediction model have the same prediction (appendix Fig. A2(a)), the pixel wise prediction is deemed as truth (see appendix, Fig. A2(b)). These two boundaries were found by comparing the minimum number of patches needed to obtain less noise and a better prediction. When cropped images may or may not contain an entire wall (or an entire column), the pattern that characterizes walls might become more difficult to identify.

Using semantic segmentation results for calculating the area ratios for columns and walls makes the result sensitive to the noise in the prediction. The intensity of each pixel in an image is binarized using a threshold of 0.70 (Messaoud *et al.*, 2022). Then, a dilatation kernel is slid over the generated framing plan with a size of ten-by-ten pixels, (a kernel size chosen based on experience in this dataset) to fill the pixel with pixel value of zero inside the elements (columns, and walls) that did not get filled through the pixel-wise prediction. This step is performed for both the columns and walls. Then, the elements geometric characteristics such as area and centroids are obtained using the SciPy module (Virtanen *et al.*, 2020).

Columns tend to have similar sizes within a framing plan. We found that walls do not share this feature and thus this denoising approach of removing the smaller elements is only applied to the column's prediction mask. To take advantage of this fact on the columns' mask, the denoising approach used here depends on the average pixel area of columns within three standard deviations of the predicted elements in the framing plan. Then, a condition with two thresholds is established to remove columns from the framing plan. The average length of the columns for a given framing plan is multiplied by 0.1 to obtain the minimum limit and by 5 to obtain the maximum limit. For example, if the average length the columns is around 30 cm (12 in), then all areas smaller than 5 by 5 cm (2 by 2 inches) and greater than 150 by 150 cm (60 by 60 inches) will be deleted from the image. These thresholds were selected based on the type of errors observed in the predictions.

Noise removal using the techniques mentioned herein has been performed for the evaluation set, using the ratio of the false positives (FPR). These techniques include overlapping the prediction patches, averaging the patches, clipping and eliminating the smaller areas (the latter, only applicable in columns). This ratio is then calculated using the total pixel area predicted by the segmentation model ($PA_{\text{FalsePositives}}$) divided by the true floor plan area. The equation can be found in the appendix, Eq. (A1). The effect of these techniques can be found in the Fig. A3 for the complete evaluation set. This figure shows that the techniques are effective in

removing noise in 24 of the 25 cases. Because of the hard-coded nature of these denoising techniques, several patches can present the same wrong prediction on a difficult example resulting in the enforcement of a false positive. These denoising techniques also hint that there is a logical path for removal of false positives, which can also be tackled by increasing the size of the training set. Finally, Fig. A4 demonstrates the effect on the column predictions with and without the denoising techniques on one framing plan.

3.2 Finding the pixel area of columns and walls

The calculation of the Hassan index begins by obtaining the area of pixels of walls and columns through the segmentation prediction model developed and tested in the previous section. The calculation of the areas and centers of columns is simple. Columns present themselves as isolated and can either be rectangular or circular. The area is calculated by counting the non-zero pixels in the generated framing plan mask for columns.

Morphological operations were used to calculate the area and direction of the walls that are predicted by the wall model. The process for calculating the WI in both the X - and Y -directions is shown Algorithm 1. The process starts with calculating the medial axis (Tam and Heidrich, 2003), as shown in Fig. 12(b) and the distance transform as shown in Fig. 12(c) for a walls mask as shown in Fig. 12(a). The medial axis is sensitive to

noise and small branches will appear even on denoised images, because the borders in this image can still present some small imperfections, especially in a prediction mask. With the medial axis we have the main skeleton of a wall, but this wall is one body and thus the wall is trimmed at intersections through automatic morphological operations of the OpenCV (Bradski and Kaehler, 2000) implementation of the PlantCV library (Gehan *et al.*, 2017). This results in the generation of N segments. The points generated from each segment are then fitted with N regression lines that define the walls. These regression lines allow us to calculate the directionality of each of the walls and also determine whether they are perpendicular or parallel to each other, regardless to the true direction of the wall (Fig. 6(b)). Here we also take notice of a common domain-specific feature known to apply to structural framing plans: walls tend to be designed to be perpendicular to each other (Fig. 6(b)). We then take the dot product among the N lines vectors and separate those walls that are parallel to each other and the walls that are perpendicular, resulting in a symmetric $N \times N$ matrix containing zeros and ones. If two vectors are parallel, the result will be one; if they are perpendicular then the result will be zero. Because some borders might be imperfect, a threshold for 2 vectors to be parallel or perpendicular is set to 15 degrees. This value was selected to accommodate imperfections of the segmentation prediction and the subsequent branches generated by the medial axis method. Then, we multiply

Algorithm 1 Calculation of the areas of walls

Algorithm 1

Input: mask from the wall prediction model
Output: areas of walls and angles in W_x and W_y direction

- 1 [Skeleton], [Thickness]=Medial_axis (Prediction mask)
- 2 [Segments]_N=Trim_Intersections (skeleton) #Segments coordinates for each segment of walls
- 4 Vectors, angles=Linear_Regression[Segments] #Obtain the vector equation for each line and its angle
- 5 Orthogonality[threshold=15] = Vectors (.)Vectors #Take the dot product among vectors
- 6 W = Orthogonality*[Thickness]*[Length] #Add all areas in parallel directions
- 7 [W,angles]=Kmeans(angles,W,Sillouethe) #Obtain areas and angles of walls in each direction
- 8 [W_x, angle_x, W_y, angle_y]=sort[W,angles] #Sort cluster centers for areas and angles and find W_x, and W_y

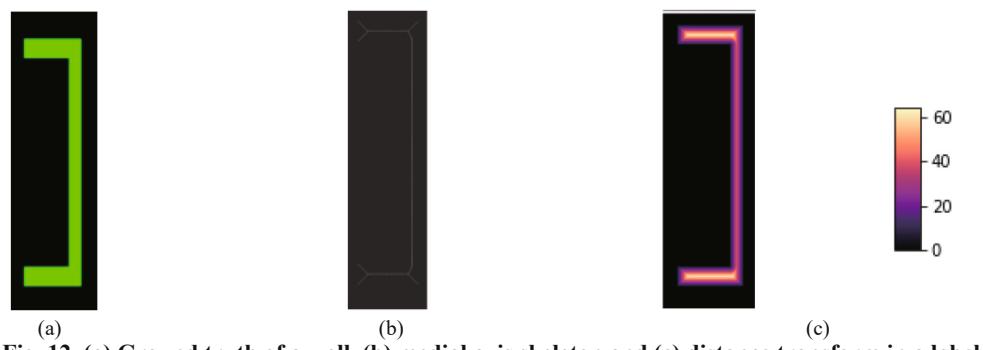


Fig. 12 (a) Ground truth of a wall, (b) medial axis skeleton and (c) distance transform in a label

this orthogonality matrix, by the vector of width of the segments, resulting in a vector with length $N \times 1$. These are finally divided using a K-means algorithm (Vassilvitskii and Arthur, 2006; Pedregosa *et al.*, 2011) by using the angle of each vectorized line segment. This angle was calculated for each segment in the skeleton with respect to the horizontal. The K-means algorithm uses then the area and angles of each of the segments to separate the wall area and angles pairs into clusters. The Silhouette score (Rousseeuw, 1987) is used to select the optimum number of clusters to separate these pairs. The result of K-means algorithm is the cluster centroid pairs. The output of Algorithm 1 is the area of the walls in each direction with its respective angles. These are obtained by sorting the clusters in order and by obtaining the two highest area of walls if walls exist in two different directions, or obtaining the maximum area of walls if walls exist only in one direction. For the calculation of the Hassan index only the areas are used, but the engineer can verify the results by obtaining the same expected angle for each wall direction from a visual inspection of the framing plan. Because of the presence of some false positive in some predictions, WI value less than 0.05% are not considered in this calculation and are set to zero.

3.3 Finding the floor plan area

The next step is to calculate the floor area in terms of the number of pixels. The concave hull method developed by Kalinina *et al.* (2018) was selected to derive an approximate contour of the perimeter of the framing plan. This algorithm uses the Delaunay method which connects the coordinates of elements in

a drawing and generates triangles with sides that depend on a length “ L ”. The value of L depends on the average distance among the coordinates and a given parameter α . This algorithm uses interpolation to give curved lines along the contour because its original application was for geospatial applications. Here the algorithm is modified to skip this interpolation and simply use straight lines, as shown in the representative results shown in Fig. 13. As shown in Figs. 13(a)–13(c), using the concave hull method the column coordinates (represented as points) are all enclosed within an area. The shape of this area converges from a concave hull ($\alpha = 0.5$) to a convex hull ($\alpha = 1$). For this particular framing plan, the closest shape to the true framing plan is the concave hull with an α of 0.55. For the concave hull with an $\alpha = 1$, the error of the area is 15% in comparison to the optimized $\alpha = 0.55$. With an α of 0.7, the error is of 4%.

At this step both the pixel area of columns and the pixel area of walls have been calculated and denoised. Algorithm 2 is used for finding the perimeter and pixel area of the floor plan. The first step is to calculate the coordinates of both the columns and the walls within the framing plan. For the columns, we use the centroid as coordinates (CC), while for the walls coordinates (WC), we obtain the coordinates from the vectorization process from Algorithm 1. Because the walls have several segments within an element, we set the minimum of coordinate pairs that come from a wall as 3 (start, mid and end). But ultimately, the number of coordinates pairs depend on the length of a wall segment and the thickness of a wall segment. A coordinate pair will be taken every 10 times the thickness of the wall in the segment being evaluated. Segments with lengths less than 1 times the

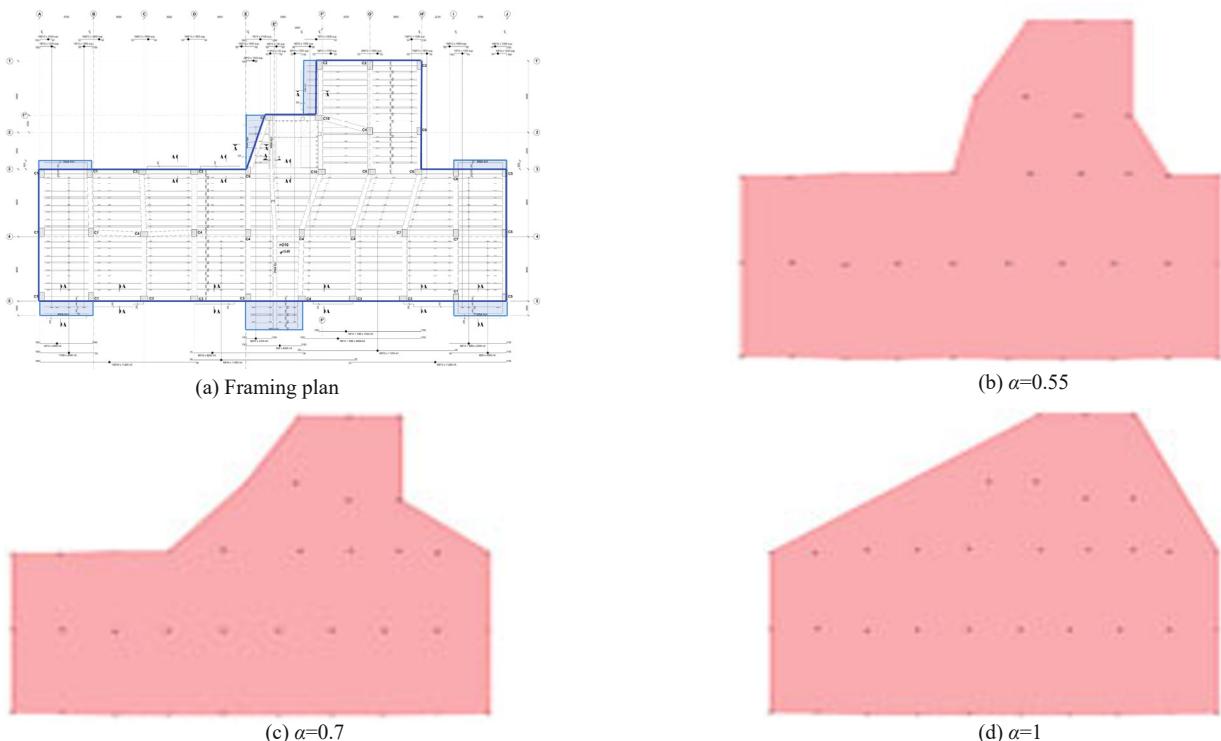


Fig. 13 Selection of alpha for concave hull

Algorithm 2 Shape finding algorithm for the framing plan**Algorithm 2**

Input: Framing plan mask of columns and walls	
Output: Area of the floor plan	

```

1 [CC]=coordinates (CM)
2 [WC]=discretize (WM, min=3, step=10*w, remove=1*w)
4 For  $\alpha$  = [0.7 to 1,  $\Delta$  = 0.5]:
5     Area, perimeter, Numpolygons = Delaunay ( $\alpha$ , CC, WC)
6     Numcontours = contour(area) #Find number of contours
8     If Numcontours = 1 and Numpolygons = 1:
9         Break

```

thickness of the wall segment evaluated are removed from this calculation. Once we obtain the coordinates of both structural elements, then we iterate from an alpha shape of 0.7 up to an alpha shape of 1, in increments Δ of 0.05. In each iteration we perform two checks. The first check is the number of openings within an area, and we perform this check by finding the number of contours inside the generated shape. Here we define an opening as an empty area within a framing plan perimeter. Because the triangulation depends on the alpha parameter and distance among element coordinates, empty areas can be generated inside a polygon, see more in Kalinina *et al.* (2018). Openings are not captured by the Hassan index and thus this does not impact the results when openings do exist within a framing plan. The second check is the number of polygons used to generate the shape that encloses the points. When these two conditions: the number of polygons and the number of contours is 1, then we break the loop with the minimum alpha shape possible, and thus the minimum area possible that can be generated with these points without having any openings inside the framing plans.

As explained by Kalinina *et al.* (2018), good approximations for a concave shape can typically be found with an α between 0.5 and 0.8. However, as the α is reduced, more points are needed to find a close-fitting shape. For this study, it was found that an α between 0.7 and 1 would be a better fit for most irregular framing plans. However, it was found that when the average error relative to the true area for irregular plans would be around 15% or more, this method could deeply impact the results for the Hassan index.

4 Evaluation of the approach

In this section, we assess the approach and the robustness of the technique from beginning to end. The twenty-five framing plans of the evaluation set from the dataset split (10% of the total framing plans in the inventory) were used in the evaluation of the approach. These framing plans were not part of the prediction models training, and they were separated from the four datasets proportionally using random selection. The true Hassan index was determined by going through each set

of structural drawings, annotating, and calculating the area of columns, walls, and the area of the critical floor. This process can take from 10 minutes to 1.5 hours with an average of 30 min per building. This evaluation set is composed of five framing plans from Dataset 1, thirteen framing plans from Dataset 2, four framing plans from Dataset 3 and three framing plans from Dataset 4. The automation of the Hassan index consists of obtaining column and wall ratios for each framing plan, and thus the metrics used in this section center around evaluating how the segmentation results alter the index calculation. The metrics used herein are defined in Eq. (6) to Eq. (15).

The equations for calculating the error in the column index, wall index, and the Hassan index (HI) are given in Eqs. (6) to (9). These equations seek to compare the true Hassan index and its components to the calculated using the approach developed herein. In these equations, the true values use the subscript: "GT" or ground truth. The indices calculated with this approach use the subscript: "predicted". The error calculations for Eqs. (9a) and (9b) take the minimum among the WI in both directions when there are walls in both directions. The Hassan index is calculated using Eq. (10a) and Eq. (10b). Equation (10a) calculates the HI using the CI and WI with ground truth framing plan area, and Eq. (10b), calculates the HI, including the calculated framing plan area. The predicted indices: $CI_{predicted}$, $WI_{x,predicted}$ and $WI_{y,predicted}$, in Eqs. (11) to (13), use the predicted column pixels ($CP_{predicted}$), predicted wall pixel in the X-direction ($WP_{x,predicted}$) and the predicted wall pixel in the Y-direction ($WP_{y,predicted}$) as calculated on the previous section. Note that for these equations, the floor plan area from the ground truth (GT) is used and not the floor plan area generated through the predictions. This distinction is performed to separate the error of the found framing plan from the Hassan index calculation. The error of the found framing plan, is calculated using Eq. (14). As previously mentioned, this method works well for regular framing plans. However, a great portion of the dataset contains irregular framing plans with strong re-entrant corners. The floor plan error (FPE) is calculated as the ratio of the found floor plan area through the algorithm to the floor plan area obtained from the mask. Finally, the vectorized area from the GT mask and from the prediction are compared with Eq. (15).

This metric intends to measure the total area of a wall that has been selected from the generated wall framing plan to be vectorized area.

$$CI_{\text{error}} = \frac{(CI_{\text{GT}} - CI_{\text{predicted}})}{CI_{\text{GT}}} \quad (6)$$

$$WI_{x_{\text{error}}} = \frac{(WI_{x,\text{GT}} - WI_{x,\text{predicted}})}{WI_{x,\text{GT}}} \quad (7)$$

$$WI_{y_{\text{error}}} = \frac{(WI_{y,\text{GT}} - WI_{y,\text{predicted}})}{WI_{y,\text{GT}}} \quad (8)$$

$$HI_{\text{error_predicted}} = \frac{(HI_{\text{GT}} - HI_{\text{predicted}})}{HI_{\text{GT}}} \quad (9a)$$

$$HI_{\text{error_calculated}} = \frac{(HI_{\text{GT}} - HI_{\text{calculated}})}{HI_{\text{GT}}} \quad (9b)$$

$$HI_{\text{predicted}} = CI_{\text{predicted}} + \min(W_{y,\text{predicted}}, W_{x,\text{predicted}}) \quad (10a)$$

$$HI_{\text{calculated}} = \frac{CP_{\text{predicted}}}{2 \times \text{Floor plan area}_{\text{predicted}}} + \min\left(\frac{WP_{x,\text{predicted}}}{\text{Floor plan area}_{\text{predicted}}}, \frac{WP_{y,\text{predicted}}}{\text{Floor plan area}_{\text{predicted}}}\right) \quad (10b)$$

$$CI_{\text{predicted}} = \frac{CP_{\text{predicted}}}{2 \times \text{Floor plan area}_{\text{GT}}} \quad (11)$$

$$WI_{x,\text{predicted}} = \frac{WP_{x,\text{predicted}}}{\text{Floor plan area}_{\text{GT}}} \quad (12)$$

$$WI_{y,\text{predicted}} = \frac{WP_{y,\text{predicted}}}{\text{Floor plan area}_{\text{GT}}} \quad (13)$$

$$FPE = \frac{\text{Floor plan area}_{\text{predicted}}}{\text{Floor plan area}_{\text{GT}}} \quad (14)$$

$$\text{Vectorized wall area} = \frac{W_{\text{Total_mask}}}{W_{\text{Total_prediction}}} \quad (15)$$

Error calculations are demonstrated with two examples from Dataset 3 for Example A and from Dataset 2 for Example B. Both structural drawings are rich in structural details to demonstrate the robustness of the prediction models on the framing for this inventory. The drawings have the masks overlayed with blue for the columns and green for the walls.

Example A is a framing plan from the New Zealand dataset (see framing plan in the appendix, Fig. A5). Both walls and columns presented a similar infill pattern as shown in the close-up details that can be found in the appendix, Fig. A6. As seen in this prediction results for this framing plan (Fig. 14), there are two columns near the stairs detail where both the columns and walls model had predicted columns and walls (respectively). This prediction is highlighted with red arrows in Fig. 14. With these prediction masks, we continue to Step 5 from Fig. 5 which is calculating the area of columns and walls. The area of columns can be readily obtained from the prediction mask. For the area of the walls, the walls need to be vectorized following the steps of Algorithm 2. The skeleton, before vectorization is shown in Fig. 15 for (a) GT and for (b) prediction mask. It can be seen that the prediction mask presents a few false positives. The percentage of the false positives is presented in Table 4, and is 0.26% with respect to the floor plan area of the GT. Some of these fragments will erroneously contribute to the calculation of the wall index. When compared to the label skeleton, this prediction will have 5% more area to distribute than the GT, as shown in Table 4. The next step is calculating the floor perimeter given the coordinates of the columns and the coordinates of the walls. Unlike the columns, the walls are discretized. The “form finding” for the floor perimeter is shown in Fig. 16. The final perimeter closely relates to the true shape of the building, but it skips a small area as seen in Fig. 16(b). The evaluation metrics are summarized in Table 3 and Table 4.

Example B is a framing plan from the Ecuador dataset. Both walls and columns presented characteristics like other walls and columns present in Dataset 3. The labelled framing plan along with the column and wall characteristics are shown in the appendix, Fig. A7 and Fig. A8. As seen in these prediction results for this framing plan (Fig. 17), one column is predicted (shown in cyan) in the walls area, and one column is missing from the prediction on the left to the wall. This behavior can be explained from the training. When they are present, the boundary elements would be labelled as columns and not as part of the walls. Now, with these prediction masks, we continue to the next step which is calculating the area of columns and walls. The area of columns can be readily obtained from the mask. For the walls calculation, the skeletons for (a) GT and for (b) prediction mask are shown in Fig. 18. It can be seen that the prediction mask presents a few false positives, less than what we saw in Example A. The percentage of FP is presented in Table 4, and is 0.06% with respect to the floor plan area of the mask. The next step is calculating the floor perimeter given the coordinates of the columns and the coordinates of the walls. The form finding for the floor perimeter is shown in Fig. 19. This is a regular plan, and thus the error is less likely to be large. The evaluation metrics are summarized in Tables 3 and 4. Results show that this framing plan had overall good

results and should be used to calculate the Hassan index.

The evaluation metrics for the complete evaluation set are shown in Fig. 20. There is a total of 25 drawings, of which all contained columns and 13 contained walls in at least one direction. Example A is framing plan number 11 and Example B is framing plan number 17. The results explained in the tables above can be also found in this figure with the corresponding framing number. Figure 20(a) divides the results by dataset because each dataset had special characteristics. The average errors

from all the framing plans in the calculation of the CI, WI_y, WI_x are 13.5%, 12.5% and 14%, respectively, for Figs. 20(b) to 20(d). Thus, a boundary of 15% was defined for these plots. Most of the framing plans with large errors (above the 15% error line) come from Dataset 1. The data in this dataset contains diverse architectural and structural details which can be difficult to capture during training. The form finding perimeter of the framing plan in Fig. 20(e) shows that the ratio of the framing plans is around 0.95, with one outlier. This



Fig. 14 Overlay of the prediction on the structural drawing

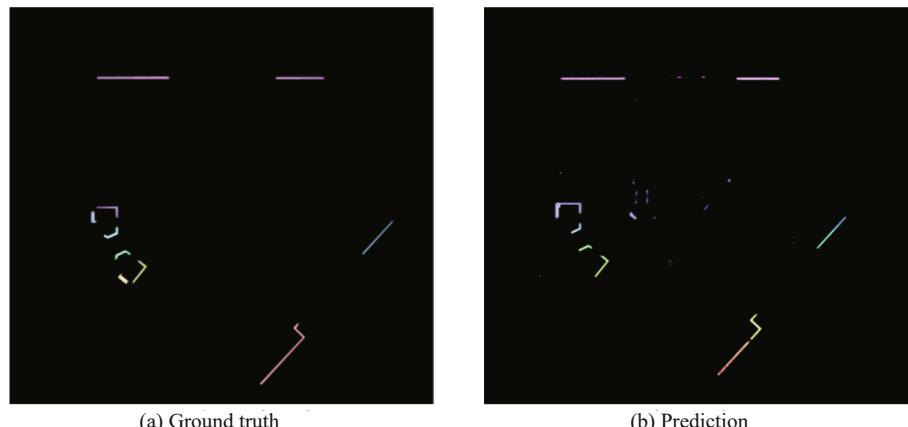


Fig. 15 Piece-wise segmentation of wall for (a) ground truth and (b) prediction mask

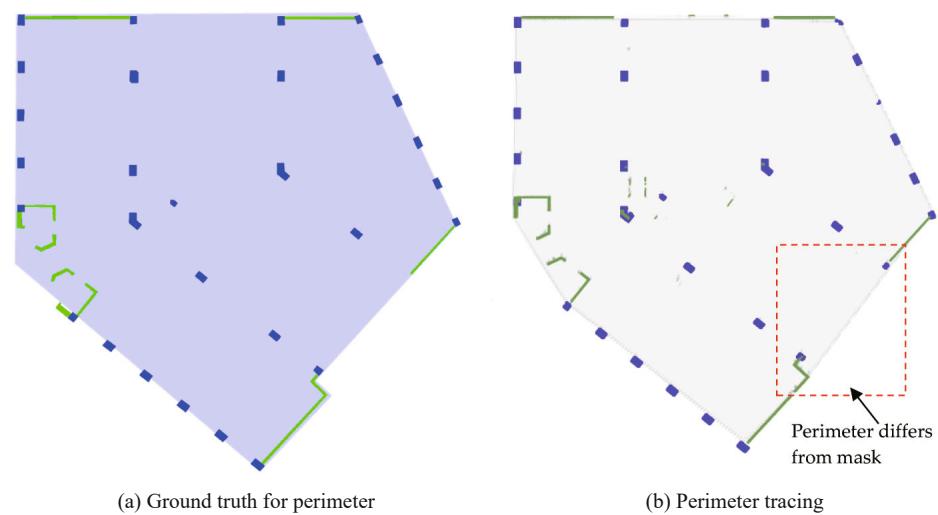


Fig. 16 (a) Floor plan for the ground truth and (b) the form finding results for the floor plan

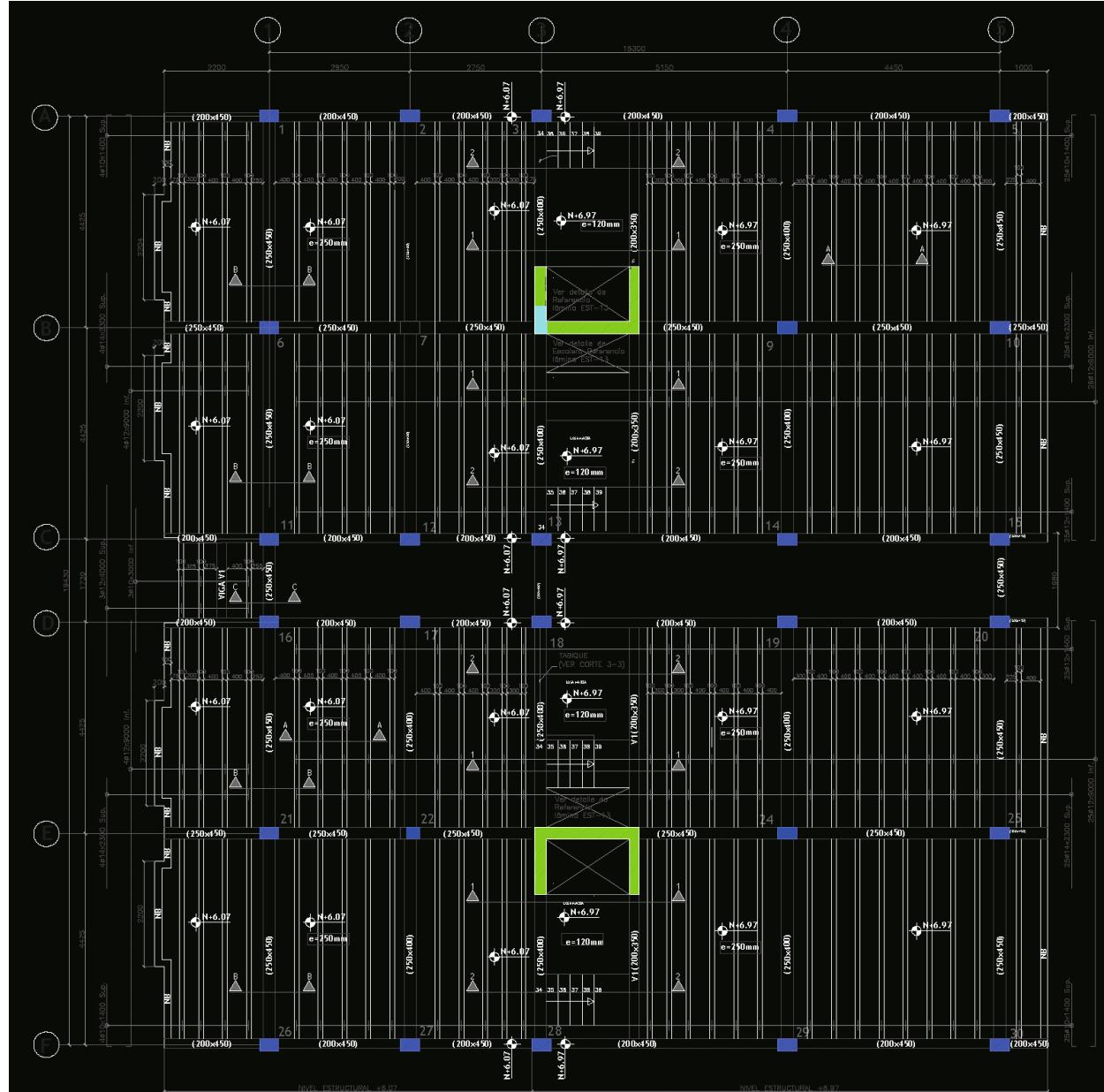


Fig. 17 Overlay of the prediction on the structural drawing

outlier, which almost doubled the framing plan area, had a highly irregular floor plan. Again, because this is a visual method, it would likely be necessary to make the calculation for this drawing by hand after detecting that the generated framing plan area is different from the true one. The vectorized wall ratio Fig. 20(f), shows more variability than other metrics. If the prediction model failed in detecting the walls, there would be more area

or less area to be accounted in the calculation and thus if the error in the wall index prediction is large so will be the vectorized area ratio. The Hassan index error calculated using the ground truth framing plan area is shown in Fig. 21, here the average error among all points is 14.35%. Note that for this calculation all the errors calculated previously accumulate, which can be observed by the number of points above the 15% line

Table 3 Evaluation of the Hassan index parameters

Metric	Example A			Example B		
	True value	Calculated	Relative error (%)	True value	Calculated	Relative error (%)
Column index (CI)	0.734%	0.81%	10.27	0.503%	0.509%	1.16
Wall index in $X(W_x)$	0.814%	0.776%	4.65	0.357%	0.346%	2.98
Wall index in $Y(W_y)$	0.767%	0.772%	0.53	0.321%	0.341%	6.18
Floor plan area (pixels)	10963024	10819524	1.3	1473701	1415330	3.96
$HI_{predicted}$ (%)	1.501%	1.582%	5.46	0.824%	0.850%	3.16
$HI_{calculated}$ (%)	1.501%	1.603%	6.80	0.824%	0.885%	7.4

Table 4 Evaluation of the segmentation

Metric	Example A	Example B
False positives in columns	0.26%	0.059%
False positives in walls	0.22%	0.01%
Vectorized wall area	1.048	0.853



Fig. 18 Piece-wise segmentation of wall for (a) mask and (b) prediction

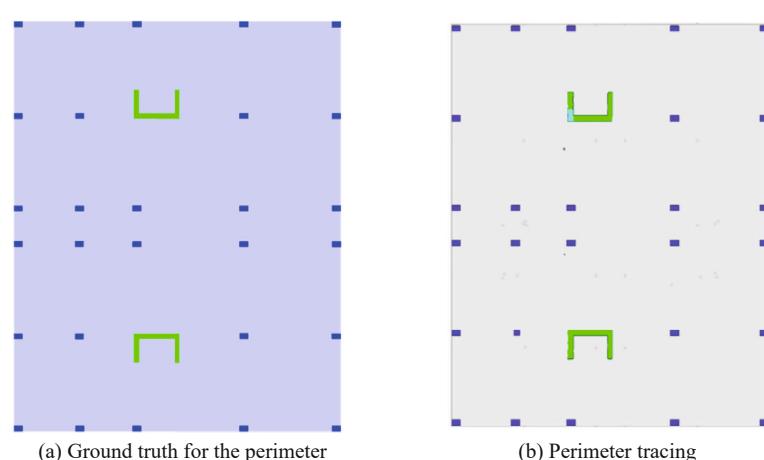


Fig. 19 Perimeter tracing for (a) ground truth and prediction (b) mask

and their distance above this line. Note that 30% of the drawings are above this line. When including only those framing plans that fall below the 15% limit, the average accuracy of the remaining 18 framing plans, is 94.96% (from 85.64%). The Hassan index error calculated using the complete approach developed herein is shown in Fig. 22. When comparing Figs. 21 to 22, we can visualize the effect of using the convex hull technique for irregular floor plans. When including only those framing plans that fall below the 15% limit, the average accuracy of the remaining 18 framing plans, is 93.43% (from 80.88%). These results show there is an opportunity to improve the method with the use of deep learning-based segmentation techniques to predict the framing plan area when enough structural drawings become available.

To summarize, our approach to automate the Hassan index and the Hassan index parameters show the average accuracy of the Hassan index estimation in the evaluation dataset is of 80.88%, making this a good approach for calculation of this index. Regarding the set of the framing plans where the approach failed, it can be explained by noting that this data was trained on a limited training images set of about 220 framing plans, which highlights the importance of the quantity and quality of data needed

for training neural networks. For the cases studied in this section, the predictions models failed at times to identify correctly columns or walls because specific features were available in the framing plan. For walls, these features include different infill patterns or no infill, and cluttered non-concentric lines within beams could be predicted as walls because the model can only extract a small patch to predict at a time. For columns, errors in the prediction result from cluttered lines inside columns, or columns without beams near them. The walls model was less susceptible to salt-and-pepper noise than the columns prediction model. These results indicate that the remaining challenges can be overcome with the addition of more framing plans in the training set. However, due to the large variety of drawing standards, even when a larger inventory is available for training, there still could be drawings with details that will not be recognized by the detector. For this evaluation set, the ratio of framing plans that had an error greater than 15% was 30% of the drawings in this set. In practice, we would leave this portion of drawings to the human. In this case, the machine learning model can process one drawing in less than 5 minutes, which extremely improves the working efficiency, compared to manually conduct drawings.

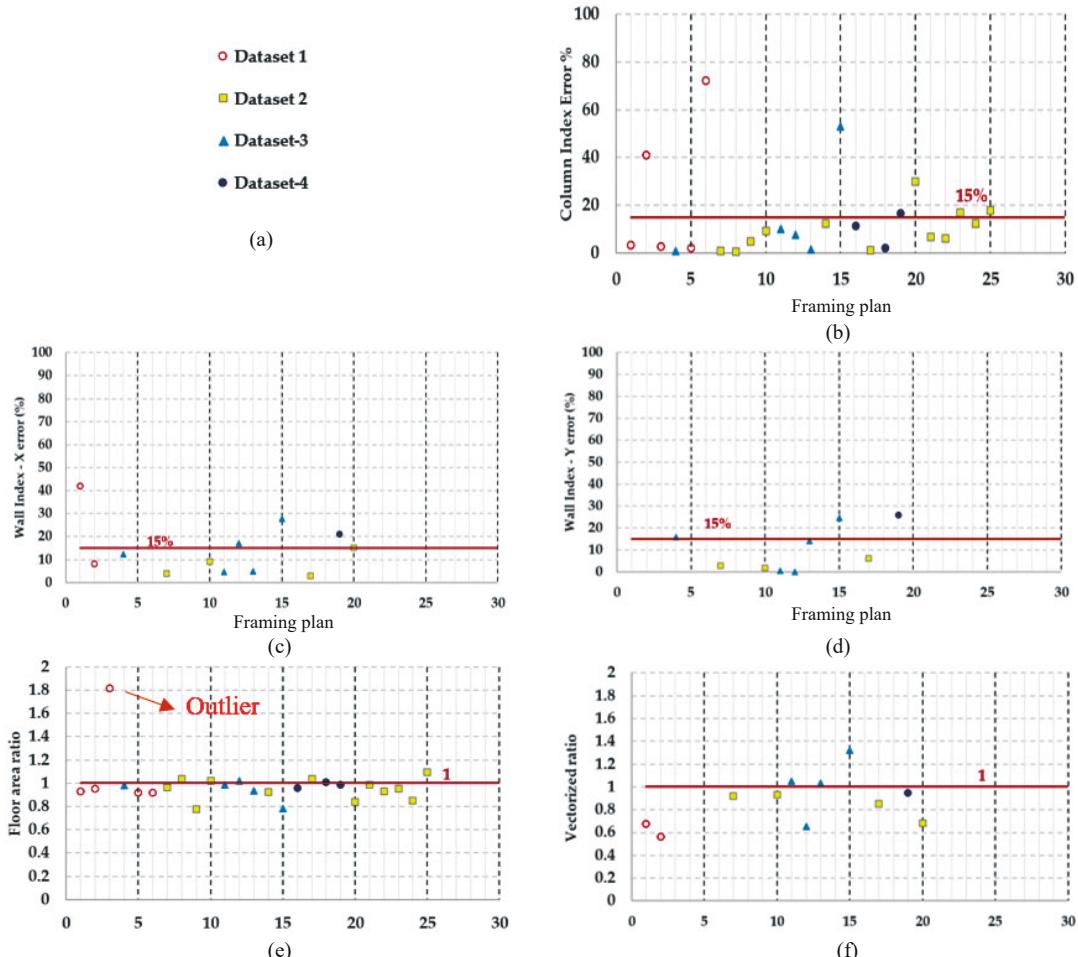


Fig. 20 Evaluation metrics for the 25 drawings in the evaluation set. (a) Dataset breakdown, (b) column index error, (c) wall index error in X, (d) wall index error in Y, (e) floor area ratio, and (f) vectorized ratio

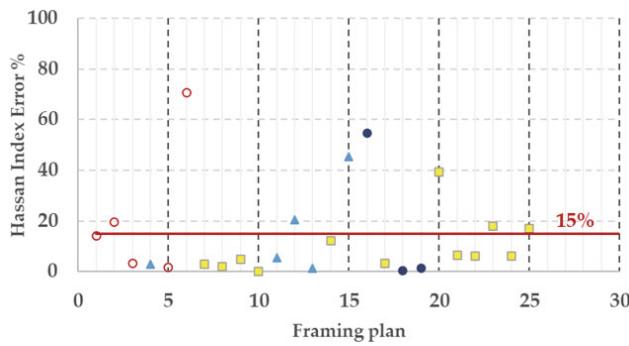


Fig. 21 Evaluation of error in the prediction of columns and walls for the Hassan index for the 25 drawings in the evaluation set

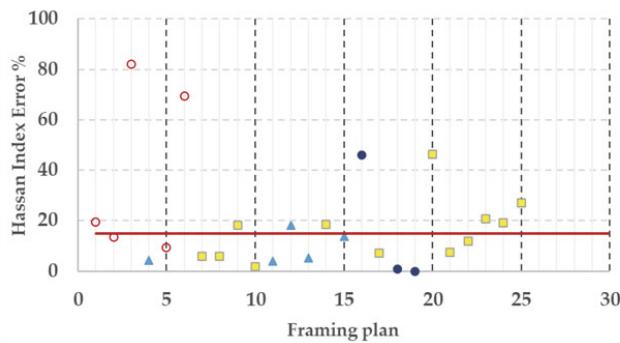


Fig. 22 Evaluation of error for the Hassan index using the complete approach for the 25 drawings in the evaluation set

From the training of the models to the inference, the denoising of the images and the calculation of the Hassan index parameters, a desktop computer was used with the following specifications: CPU - Intel i9-7820X @ 2.9Ghz, GPU - NVIDIA GeForce RTX 2080Ti - 11GB.

4.1 Remarks on limitations of the approach

The method presented in this paper can be applied to the calculation of the Hasan index provided that some conditions are met in the structural drawings: (1) the drawings need to be at scale, and (2) framing plans need to be more or less regular. For irregular plans, for best results the user should trace the perimeter of the framing plan. Alternatively, one may use a prediction model with an architecture such as the U-Net (Ronneberger *et al.*, 2015) to find the area of the framing plan. These limitations exist in the automation of the Hassan index calculation and offer opportunities for future research to improve the method.

5 Conclusions

A method for automating the calculation of a vulnerability index to empower cities to rank and classify reinforced concrete buildings is developed and validated herein. The Hassan index is used as the classification tool for seismic vulnerability because it has been used by

hundreds of researchers to correlate damage and seismic vulnerability since it was first proposed in 1997. The parameters included in this index are the area of columns and walls on the first floor, which play an important role in the capacity of the building to withstand seismic demands, and the area of the framing plan, which is used for calculating the seismic demand. The simplicity of this method lends itself to the ability to automate the calculations by integrating current computer vision techniques.

This study shows that deep learning based semantic segmentation and computer vision techniques can be used in the automation of the Hassan index calculation for one building at a time. The average accuracy of the approach adopted in this study is of 80.88%. This approach is human aided as it still needs human component to verify visually that the segmentation predictions have been done correctly per framing plan. Still, by using the approach explained herein, engineers can greatly reduce the man hours used to calculate the Hassan index for complex and large structural drawings. Moreover, this works opens up the ability to obtaining geometric information from framing plans such as plan torsional irregularity, shape ratios of columns and walls, among other characteristics useful for design and analysis of buildings. This new information could be beneficial for enriching the building inventories and calculating other indexes.

With an inventory of 245 unique framing plans of structural and architectural drawings, we were able to estimate the Hassan index for these concrete buildings. While some limitations still remain, this work illustrates progress towards the automation of the Hassan index for an inventory of concrete buildings.

References

- Abraham N and Khan NM (2019), “A Novel Focal Tversky Loss Function with Improved Attention U-Net for Lesion Segmentation,” *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pp. 683–687.
- Balbirnie K, Chandramohan R, Pujol S and Wright J (2021), “Identifying Vulnerable Reinforced Concrete Buildings in Wellington Using a Simple Assessment Methodology,” *Proceedings of the 2021 New Zealand Society for Earthquake Engineering Annual Technical Conference*. <http://13.237.132.70/handle/nzsee/2368>
- Bradski G and Kaehler A (2000), “OpenCV,” *Dr. Dobb's Journal of Software Tools*, 3: 2.
- Brzev S, Pandey B, Maharjan DK and Ventura C (2017), “Seismic Vulnerability Assessment of Low-Rise Reinforced Concrete Buildings Affected by the 2015 Gorkha, Nepal, Earthquake,” *Earthquake Spectra*, 33(S1): 275–298.
- Buslaev A, Iglovikov VI, Khvedchenya E, Parinov A, Druzhinin M and Kalinin AA (2020), “Albumentations:

Fast and Flexible Image Augmentations," *Information*, **11**(2): 125. doi: 10.3390/info11020125

Chungwook S, Laughery L, Chiou TC and Weng P (2018), *2017 Pohang Earthquake*. <https://datacenterhub.org/deedsdv/publications/view/296>

Code P (2005), *Eurocode 2: Design of Concrete Structures-Part 1-1: General Rules and Rules for Buildings*, British Standard Institution, London, UK.

Deng J, Dong W, Socher R, Li LJ, Li K and Li FF (2009), "Imagenet: A Large-Scale Hierarchical Image Database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255.

Gehan MA, Fahlgren N, Abbasi A, Berry JC and Sax T (2017), "PlantCV v2: Image Analysis Software for High-Throughput Plant Phenotyping," *PeerJ*, **5**: e4088.

Gimenez J, Hippolyte JL, Robert S, Suard F and Zreik K (2015), "Reconstruction of 3D Building Information Models from 2D Scanned Plans," *Journal of Building Engineering*, **2**: 24–35.

Hassan AF and Sozen MA (1997), "Seismic Vulnerability Assessment of Low-Rise Buildings in Regions with Infrequent Earthquakes," *ACI Struc*, **94**(1): 31–39.

Huang B, Reichman D, Collins LM, Bradbury K and Malof JM (2018), "Tiling and Stitching Segmentation Output for Remote Sensing: Basic Challenges and Recommendations," *arXiv preprint arXiv*, 1805.12219.

Jury R and Ferner H (2015), "Seismic Risk Management in the New Zealand Context," *Improving the Seismic Performance of Existing Buildings and Other Structures*, pp. 728–740.

Kalinina D, Ingilevich V, Lantseva A and Ivanov S (2018), "Computing Concave Hull with Closed Curve Smoothing: Performance, Concaveness Measure and Applications," *Procedia Comput Sci*, **136**: 479–488. doi: <https://doi.org/10.1016/j.procs.2018.08.258>

Kalinina D, Ingilevich V, Lantseva A and Ivanov S (2018), "Computing Concave Hull with Closed Curve Smoothing: Performance, Concaveness Measure and Applications," *Procedia Comput Sci*, **136**: 479–488.

Kannala J (2019), "CubiCasa5K: A Dataset and an Improved Multi-Task Model for Floorplan Image Analysis," *Image Analysis: 21st Scandinavian Conference, SCIA 2019*, Norrköping, Sweden, **11482**: 28.

Kassem MM, Nazri FM and Farsangi EN (2020), "The Seismic Vulnerability Assessment Methodologies: A State-of-the-Art Review," *Ain Shams Engineering Journal*, **11**(4): 849–864.

Kestur R, Meduri A and Narasipura O (2019), "MangoNet: A Deep Semantic Segmentation Architecture for a Method to Detect and Count Mangoes in an Open Orchard," *Eng Appl Artif Intell*, **77**: 59–69.

Kim S, Park S, Kim H and Yu K (2021), "Deep Floor Plan Analysis for Complicated Drawings Based on Style Transfer," *Journal of Computing in Civil Engineering*, **35**(2): 04020066.

Labelbox (2022), <https://labelbox.com>.

Lenjani A, Dyke SJ, Bilionis L, Yeum CM, Kamiya K, Choi J, Liu XY and Chowdhury AG (2020), "Towards Fully Automated Post-Event Data Collection and Analysis: Pre-Event and Post-Event Information Fusion," *Eng Struct*, **208**: 109884.

Li H, Zhu F and Qiu J (2018), "CG-DIQA: No-Reference Document Image Quality Assessment Based on Character Gradient," *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 3622–3626. doi: 10.1109/ICPR.2018.8545433

Liu C, Schwing AG, Kundu K, Urtasun R and Fidler S (2015), "Rent3d: Floor-Plan Priors for Monocular Layout Estimation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3413–3421.

Liu X, Iturburu L, Dyke SJ, Lenjani A, Ramirez J and Zhang X (2022), "Information Fusion to Automatically Classify Post-Event Building Damage State," *Eng Struct*, **253**: 113765.

Loos S, Lallemand D, Baker JW, McLaughrey J and Singh R (2020), "G-DIF: A Geospatial Data Integration Framework to Rapidly Estimate Post-Earthquake Damage," *Earthquake Spectra*, **36**(4): 1695–1718.

Lu Q and Lee S (2017), "A Semi-Automatic Approach to Detect Structural Components from CAD Drawings for Constructing As-Is BIM Objects," *Computing in Civil Engineering*, pp. 84–91.

Lv X, Zhao S, Yu X and Zhao B (2021), "Residential Floor Plan Recognition and Reconstruction," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16717–16726.

Marquis F, Kim JJ, Elwood KJ and Chang SE (2017), "Understanding Post-Earthquake Decisions on Multi-Storey Concrete Buildings in Christchurch, New Zealand," *Bulletin of Earthquake Engineering*, **15**(2): 731–758.

Messaoud NJ, Mansoura A, Aissi M, Ayaria R, Frih M, Abdallah AB and Bedoui MH (2022), "Automated Segmentation of Multiple Sclerosis Lesions Based on Convolutional Neural Networks," *Comput Methods Biomed Eng Imaging Vis*, pp. 1–19.

Min XK, Gu K, Zhai GT, Yang XK, Zhang WJ, Le CP and Chen CW (2021), "Screen Content Quality Assessment: Overview, Benchmark, and Beyond," *ACM Computing Surveys (CSUR)*, **54**(9): 1–36.

Müller D, Rey IS and Kramer K (2020), "Automated Chest CT Image Segmentation of Covid-19 Lung Infection Based on 3D U-Net," *arXiv preprint arXiv*, 2007.04774.

Niblack W (1985), *An Introduction to Digital Image Processing*, Strandberg Publishing Company, USA.

Pedregosa F, Varoquaux G, Gramfort A, et al. (2011), "Scikit-Learn: Machine Learning in Python," *the Journal of Machine Learning Research*, **12**: 2825–2830.

Pujol S, Laughery L, Puranam A, Hesam P, Cheng LH, Lund A and Irfanoglu A (2020), "Evaluation of Seismic Vulnerability Indices for Low-rise Reinforced Concrete Buildings Including Data from the 6 February 2016 Taiwan Earthquake," *Journal of Disaster Research*, **15**(1): 9–19.

Ronneberger O, Fischer P and Brox T (2015), "U-net: Convolutional Networks for Biomedical Image Segmentation," *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241.

Rousseeuw PJ (1987), "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," *J Comput Appl Math*, **20**: 53–65.

Sauvola J and Pietikäinen M (2000), "Adaptive Document Image Binarization," *Pattern Recognit*, **33**(2): 225–236.

So C, Baciu G and Sun H (1998), "Reconstruction of 3D Virtual Buildings from 2D Architectural Floor Plans," *Proceedings of the ACM symposium on Virtual Reality Software and Technology*, pp. 17–23.

Szczyrba J, Zhang Y, Pamukcu D, Eroglu DI and Weiss R (2021), "Quantifying the Role of Vulnerability in Hurricane Damage via a Machine Learning Case Study," *Nat Hazards Rev*, **22**(3): 04021028.

Tam R and Heidrich W (2003), "Shape Simplification Based on the Medial Axis Transform," *IEEE Visualization, VIS 2003*, pp. 481–488.

van der Walt S, Schenberger JL, Juan NI and Boulogne F (2014), "Scikit-Image: Image Processing in Python," *PeerJ*, **2**: e453.

Vassilvitskii S and Arthur D (2006), "K-Means++: The Advantages of Careful Seeding," *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035.

Virtanen P, Gommers R, Oliphant TE, *et al.* (2020), "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nat Methods*, **17**(3): 261–272.

Wang C, Yu Q, McKenna F, Cetiner B, Yu SX, Taciroglu E and Law KH (2019), *NHERI-SimCenter/BRAILS: v1.0.1*. <https://github.com/NHERI-SimCenter/BRAILS>

Wang CF, Yu Q, Law KH, McKenna F, Yu SX, Taciroglu E, Zsarnóczay A, Elhaddad W and Cetiner B (2021), "Machine Learning-Based Regional Scale Intelligent Modeling of Building Information for Natural Hazard Risk Management," *Autom Constr*, **122**: 103474. doi: <https://doi.org/10.1016/j.autcon.2020.103474>

Xu J, Ye P, Li Q, Liu Y and Doermann D (2016), "No-Reference Document Image Quality Assessment Based on High order Image Statistics," *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3289–3293.

Yeum CM and Dyke SJ (2016), "Big Visual Data Analytics for Damage Classification in Civil Engineering," *Transforming the Future of Infrastructure Through Smarter Information: Proceedings of the International Conference on Smart Infrastructure and Construction*, pp. 569–574.

Yeum CM, Dyke SJ and Ramirez J (2018), "Visual Data Classification in Post-Event Building Reconnaissance," *Eng Struct*, **155**: 16–24.

Yeung M, Sala E, Schönlieb CB and Rundo L (2021), "Unified Focal Loss: Generalising Dice and Cross Entropy-Based Losses to Handle Class Imbalanced Medical Image Segmentation," *Computerized Medical Imaging and Graphics*, 102026.

Yu Q, Wang CF, Cetiner B, Yu SX, McKenna F, Taciroglu E and Law KH (2019), "Building Information Modeling and Classification by Visual Learning at a City scale," *ArXiv preprint arXiv*, 1910.06391.

Zhang Y, Ca J and Cai H (2020), "CNN-Based Symbol Recognition in Piping Drawings," *Construction Research Congress 2020: Computer Applications*, pp. 576–584.

Zhao Y, Deng X and Lai H (2020), "A YOLO-Based Method to Recognize Structural Components from 2D Drawings," *Construction Research Congress 2020: Computer Applications*, pp. 753–762.

Zhao Y, Deng X and Lai H (2021), "Reconstructing BIM from 2D Structural Drawings for Existing Buildings," *Autom Constr*, **128**: 103750.

Zheng L, Shen L, Chen J, An P and Luo J (2019), "No-Reference Quality Assessment for Screen Content Images Based on Hybrid Region Features Fusion," *IEEE Trans Multimedia*, **21**(8): 2057–2070.

Appendix

$$FPR = \frac{PA_{\text{FalsePositives}}}{\text{Floor plan area}_{\text{GT}}} \quad (A1)$$

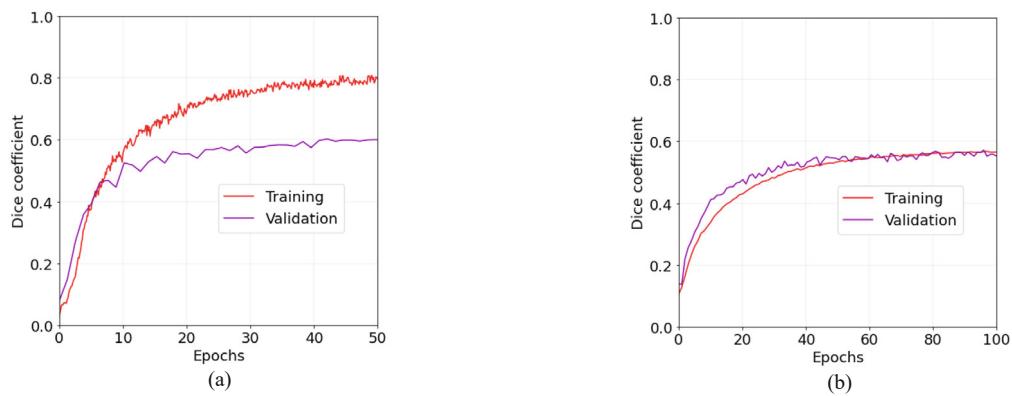


Fig. A1 Training and validation for (a) columns prediction model and (b) walls prediction model

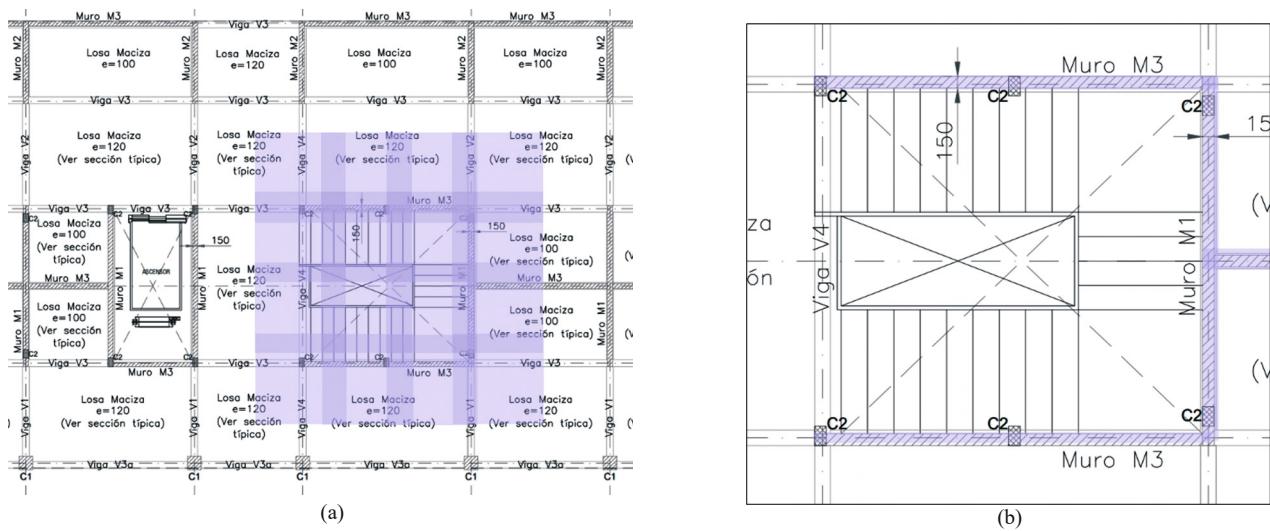


Fig. A2 (a) Sliding window producing overlap, (b) result of overlap

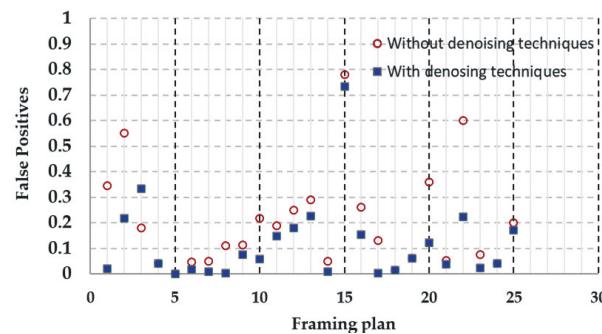


Fig. A3 Effect on the denoising techniques on the elimination of the false positives for the predictions

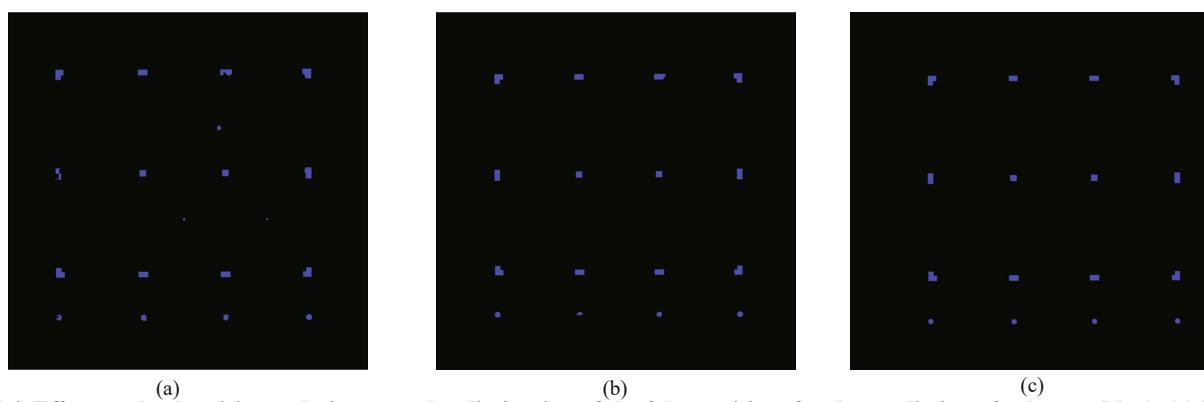


Fig. A4 Effect on the denoising techniques on the elimination of the false positives for the prediction of columns (blue). (a) Before denoising techniques, (b) after denoising techniques, (c) ground truth

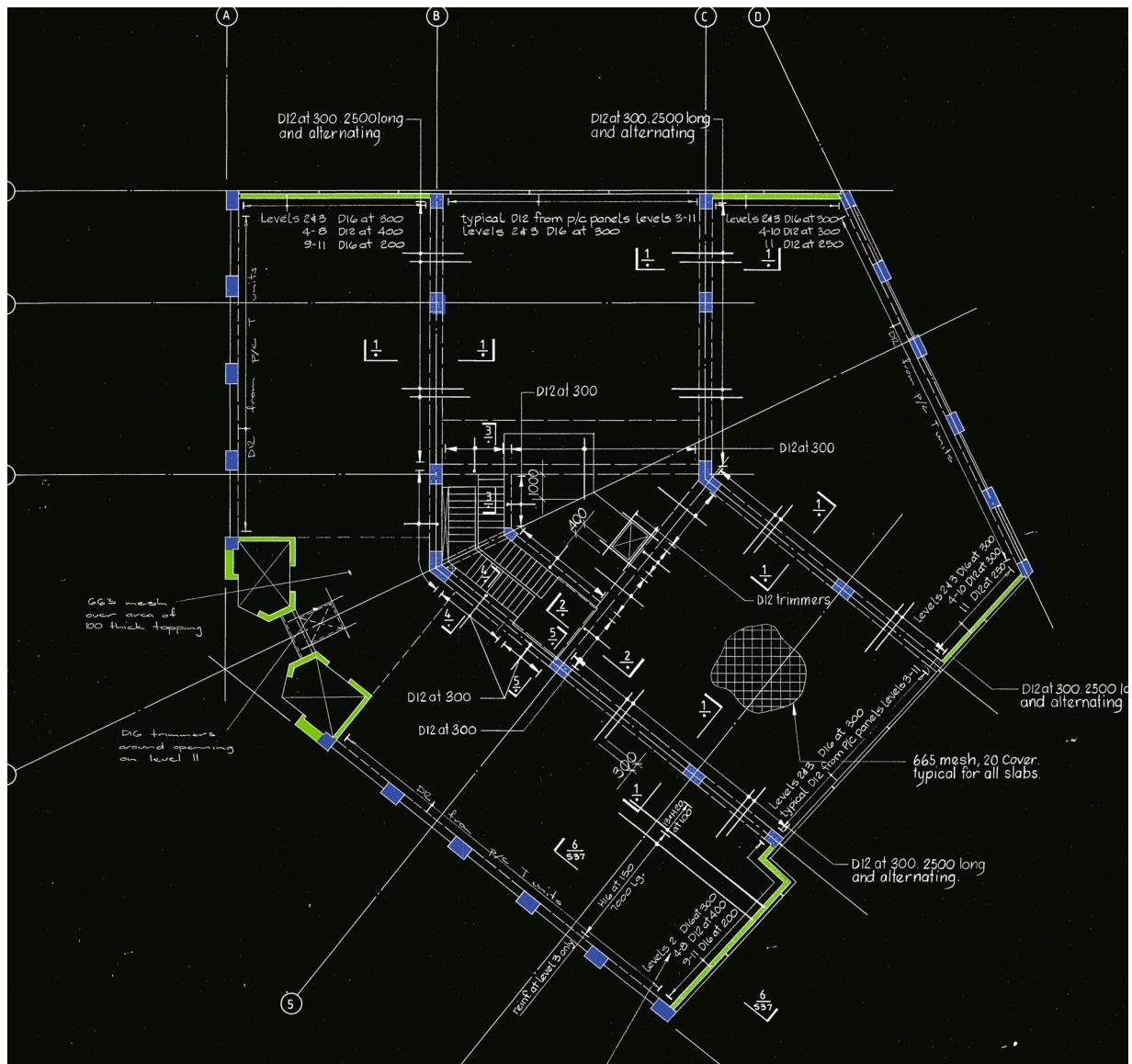
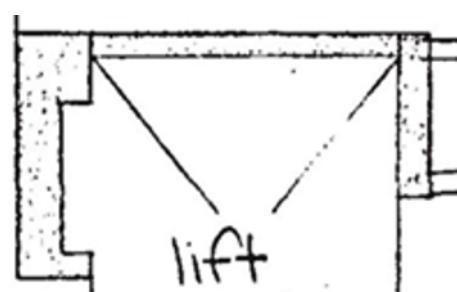


Fig. A5 Overlay of the structural drawing and the ground truth of columns (blue) and walls (green)



(a) Column



(b) Wall

Fig. A6 Close up of column and wall details in the framing plan

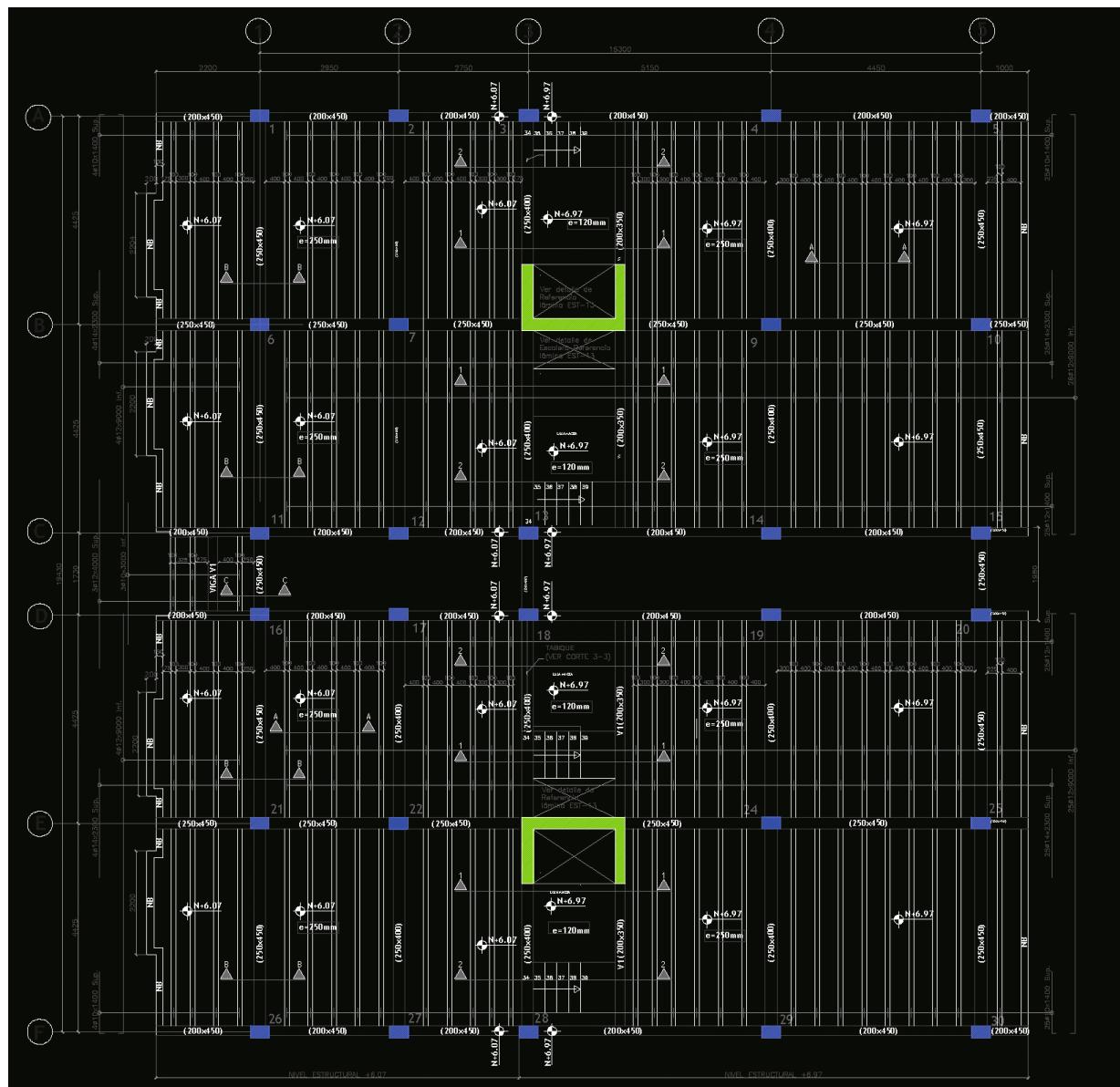


Fig. A7 Overlay of the structural drawing and the ground truth of columns (blue) and walls (green)

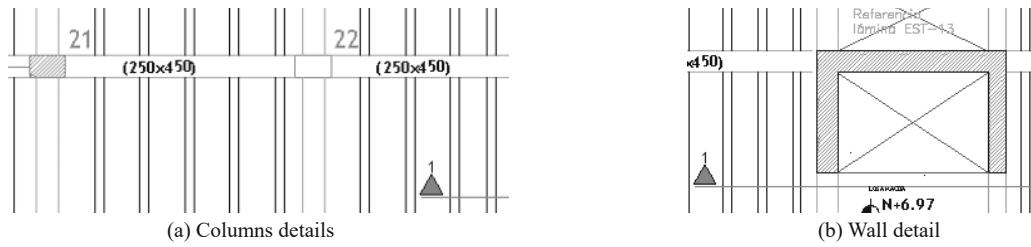


Fig. A8 Column and wall details in the structural drawing