2025 European Conference on Computing in Construction

CIB W78 Conference on IT in Construction

Porto, Portugal
July 14-17, 2025

A CONVOLUTIONAL NEURAL NETWORK BASED PIPELINE FOR THE STREAMLINING OF THE MASONRY QUALITY INDEX ANALYSIS

Andrei Farcasiu and Bora Pulatsu
Carleton University, Ottawa, Canada

Abstract

YOLOv11, a CNN-based object detection and instance segmentation algorithm, is used to automatically capture Masonry Quality Index (MQI) parameters for existing masonry structures. Training is performed using a suitable dataset for detecting bricks, and its hyperparameters are adjusted systematically for optimal accuracy. A workflow is proposed in which models are trained on the "MCrack1300" dataset and evaluated using orthomosaics of an unreinforced masonry building. Optimal hyperparameters are determined iteratively, and their impact on minimizing loss is compared. The proposed model captures block size distributions and staggering ratios associated with the construction quality of masonry walls.

Introduction

The documentation of unreinforced masonry buildings has the potential to be streamlined alongside the ongoing evolution of information technology. Given the impact of the construction quality on the structural performance of unreinforced masonry buildings, the development of fast, intuitive and reliable workflows to document and digitize their morphologies offers excellent advantages for seismic retrofitting, forensic engineering, conservation, and adaptive reuse projects. The introduction of machine learning and computer vision algorithms has opened the door to automated workflows where one can reproduce entire structures with minimal manual intervention. As such, convolutional neural networks (CNNs) serve as the basis for this task as they are designed to recognize patterns within vision-based data such as images and videos. CNNs perform various tasks, such as object detection, where bounding boxes are drawn over desired objects, and instance segmentation, where polygon masks are traced around them. Many open-source CNNs are more than capable of object detection and instance segmentation. However, their success depends on their adopted neural network architecture, user accessibility, and computational demand. In addition, they must be trained on extensive datasets of ground truth data in order to direct the neural network to detect specific objects. Furthermore, the performance of the models can be further enhanced by adjusting the hyperparameters of the model, which are high-level settings initialized prior to training that govern how the dataset is processed. While there are various hyperparameters utilized by different CNNs, the most common ones that are adjusted to increase accuracy are the number of epochs, weight decay, learning rate, and batch size (Nielsen, 2015).

Once a model is trained on a suitable dataset, it can be used to digitize masonry facades in a format that allows for generating CAD drawings or analytical models for structural analyses. This study provides an overview of a pipeline in which orthophotos of masonry facades are processed through object detection and instance segmentation models to detect and generate masks of masonry units. The primary purpose of this algorithm is to provide qualitative estimations of the masonry façade's staggering ratios and block size distributions, parameters used by the Masonry Quality Index analysis (Borri et al., 2015). The key component of this pipeline is "You-Only-Look-Once" (YOLOv11), a real-time object detection and instance segmentation model maintained by Ultralytics, which provides an extensive, user-friendly workflow for training, validating, and evaluating models. A model is trained using the "MCrack1300" dataset, a sizeable collection of annotated images of masonry units. The trained model is then assessed on a set of orthomosaics generated from photogrammetric models of a clay masonry building in Kemptville, Ontario, Canada.

YOLOv11 Architecture Summary

YOLOv11 is a real-time CNN designed to detect and trace masks of objects efficiently by processing images in entire pass-throughs, directly predicting bounding boxes, estimating class probabilities, and generating segmentation masks (Redmon et al., 2016). The architecture of YOLOv11 consists of backbone, head, and neck layers, which execute the feature extraction and classification work. The backbone layer employs convolutional layers and blocks to process input images and extract feature maps that capture specific geometric patterns of interest (Khanam and Hussain, 2024). The neck layer receives the feature maps generated from the backbone layer and prepares them to be transferred to the head layer. Lastly, the head layer facilitates the

predictions, generating the object detection outputs based on the refined feature maps provided by the neck layer.

The implementation of YOLOv11 in this study incorporates two main Python libraries, one for the training of YOLOv11 models to detect bricks and the other for the execution of instance segmentation tasks on high-resolution input images via a tile-based approach. The Ultralytics library provides an extensive toolset to train, validate, and evaluate YOLOv11 models using customized datasets. New model checkpoints are trained from generalized boilerplate models, such as "YOLOv11-seg", under a prescribed number of epochs, which are individual complete pass-throughs of the dataset. The objective of the training process is to adjust the model weights between each subsequent epoch to minimize the loss function, which measures the difference between the predicted and expected outputs of the model (Terven et al., 2023). YOLO datasets contain extensive collections of images with manually drawn annotations over the objects to be detected. Datasets are usually split into training and validation data, where the former are fed to the model during the training process. At the same time, the latter are used to evaluate the model's accuracy. The number of epochs is an important consideration while training an object detection and instance segmentation model, as an insufficient number will lead to a model that fails to detect the target objects with a high degree of confidence. In contrast, an excessive number of epochs will lead to overfitting, where the model's performance degrades despite more thorough training.

Another driving factor in the accuracy of YOLO models is the optimizer and configuration of hyperparameters used to train them. Ultralytics automatically selects an optimizer based on the size and complexity of the dataset, which in this case is AdamW, a variation of the Adam stochastic optimization algorithm. AdamW updates the weights and biases of models using learning rates that adapt based on the means and variances between gradients (Kingma and Ba, 2014). This optimizer utilizes weight decay to introduce regularization when calculating the loss during each training step (Nielsen, 2015). AdamW expands on the impact of weight decay by decoupling it from the optimization steps taken when calculating the loss (Loshchilov and Hutter, 2017). Additionally, Adam considers the learning rate, which determines the size of increments in which gradient descent algorithms minimize loss. As AdamW incorporates stochastic gradient descent (SGD), adjusting this hyperparameter may introduce more stability during training, albeit with a risk of slower convergence (Senior et al., 2013). While not specific to the AdamW optimizer, the batch size hyperparameter controls the number of training images simultaneously processed during training. Adjusting this hyperparameter can positively or negatively affect the training speed depending on computational limitations (Kandel and Castelli, 2020a).

MCrack1300 Dataset

The models generated for this study are achieved using the "MCrack1300" dataset, which was initially developed by Ye et al. for their structural damage detection framework based on the Segment Anything (SAM) Instance segmentation model (Kirillov et al., 2023). The dataset comprises 1300 annotated images of masonry facades compiled from publicly available images from the internet, a dataset designed to detect cracks in masonry assemblies, and photographs taken of masonry buildings located within the vicinity of the University of Birmingham Campus in the United Kingdom (Ye et al., 2024). The dataset provides annotations for bricks, broken bricks, and cracks, completed using Roboflow, a web-based toolset that allows for the organization, preparation, and dissemination of CNN datasets. The masonry facades found within the dataset, with several examples shown in Figure 1, are of varying colours, textures, and spatial configurations. Of the images provided, 1000 are set aside for training, while the rest are allocated for validation and testing. While the dataset is initially designed to fine-tune the SAM model to improve its generation of masonry segmentation masks, Roboflow can export annotated images for various CNNs, including YOLOv11.

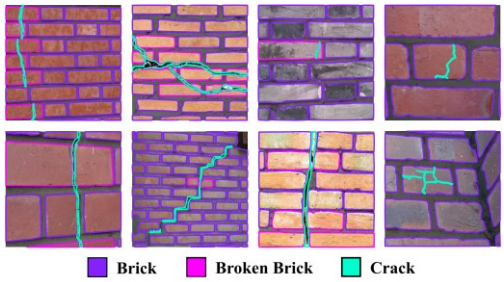


Figure 1: MCrack1300 Training Data Examples

Case Study

The input data used to evaluate the performance and accuracy of the models trained in this study originate from a photogrammetric point cloud generated by an unreinforced masonry house in Kemptville, Ontario, Canada. This clay masonry residential building was constructed in the late 19th century, with an extension on the southwestern elevation built sometime in the 20th century. The point cloud was created by obtaining aerial and terrestrial photographs and processing them using photogrammetry software. For this building, 629 images were obtained using a DJI Mini 3 Pro drone and an iPhone 15 Pro Max. The images are imported into Agisoft Metashape, which detects and matches key points across the photographs to determine their spatial locations and orientations (Over et al., 2021). The dense cloud generated using this program is used to render two orthomosaics, shown in Figures 2 a) and b), representing the building's southwestern and northwestern elevations, respectively. To obtain accurate block size distributions, reference measurements are taken in order to scale the orthophotos. The orthophotos provide clear views of the building's masonry topology with minimal obstacles. The

visual characteristic of the masonry is ideal for evaluating the trained models, as the colours of the brick and the mortar are visually distinct.

a)

b)

Figure 2: Case Study Orthophotos: a) Southwestern Façade, b) Northwestern Façade

Experiment Overview

The models' success thresholds do not depend on the complete coverage of all the masonry units in the input image. For this study, a model will be deemed sufficiently trained if it detects enough masonry units accurately to provide realistic block size distributions and staggering ratios. As such, to ensure that the output masks truly represent the geometry of the masonry units, the confidence ratio threshold, being the cut-off point of the model's confidence that the mask truly represents the target object, is set as 90%. As multiple iterations of the model are trained, this paper will provide an account of the hyperparameter adjustments made to increase the accuracy of each iteration.

The brick detection models are trained in an iterative approach in which the number of epochs, batch size, and learning rate are adjusted to obtain the maximum number of bricks detected. The models are trained using the boilerplate "yolo11n-seg" checkpoint file, which is a generalized file initially trained on the Microsoft Common Objects in Context (MS-COCO) dataset, a large, generalized dataset of annotated images of various objects (Lin et al., 2014). After testing, each model checkpoint is evaluated on both orthomosaics to obtain the total number of bricks detected. The objective of this systematic testing is to determine the point where the performance of the model reaches its peak and establish the basis for further improvement via each successive hyperparameter. The best combination is determined by training multiple models with numerous hyperparameter adjustments, and thorough insight is acquired into what hyperparameters impact the model's accuracy the most.

The evaluation of the trained models from both experiments on the orthomosaics is completed using an in-house program called "AutoBrick," shown in Figure 3, which provides a streamlined user interface for evaluating YOLO models on input images. The YOLO Patch-Based Inference library is used to detect objects in high-resolution images using a tiled-based approach. The accuracy of the inference is affected by the adjustment of the size of each tile in pixels, as well as the overlap between them. Introducing overlap between each tile helps to prevent the generation of detections with seams between tiles. In addition, masks can be overlaid on top input images to prevent the detection of false positives. The main output of this application is a .PNG image containing the segmentation masks in randomized colours. Once the output images are obtained, a Python script converts them into CAD drawings to obtain the staggering ratios manually. Afterward, using another Python script, the DXF files are processed to get the bounding boxes of each polygon, with their largest dimensions being outputted into a CSV file for use in obtaining block size distributions.

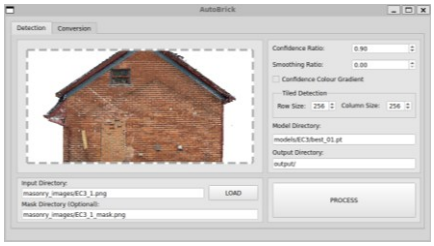


Figure 3: AutoBrick Interface

As previously mentioned, the main objective of the trained YOLOv11 model is to provide segmentation masks that allow for the efficient acquisition of parameters used in Masonry Quality Index (MQI) analyses. This preliminary analysis framework provides estimations of the material properties of masonry assemblies based on their qualitative and quantitative properties. Numerous parameters are analyzed, such as the shapes of the masonry units, mortar properties, continuity of the bed joints, and the staggering ratios (Ml) of the head joints and wall leaf connections. The parameters, depending on the criteria provided by the MQI procedure, are categorized as either fulfilled (F), partially fulfilled (PF), or not fulfilled (NF) (Hafner et al., 2021). Depending on their categorization, different values are assigned to each parameter and then used to estimate properties such as shear strengths, elastic moduli, compressive strengths, and shear moduli. Several MQI parameters are determined simply through visual assessments. For example, mortar quality is identified by observing its integrity across the façade. Upon reviewing the photography used to generate the orthomosaics, it is determined that the mortar is of medium quality and, therefore, the mortar mechanical properties (MM) parameter is partially fulfilled (PF). Other MQI parameters require quantitative analyses, in which convolutional neural networks can be used to fulfill them

efficiently. The staggering ratios' magnitude provides insight into how cracks propagate on the masonry under loading. As shown in Figure 4, the staggering ratio is determined by tracing a line from top to bottom following the shortest path between subsequent head joints. In the case of the staggering of the head joints, the magnitude of the Ml value and whether the wall is single, or double leaf affects whether or not the corresponding parameter, known as VJ, is categorized as F, PF, or NF.

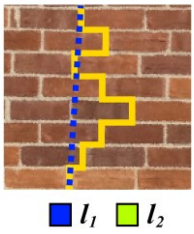


Figure 4: Staggering Ratio Lengths

The ratio, shown in Equation (1), is calculated by dividing the total length of the line by the distance from the line's starting and endpoints. The staggering ratio provides insight into how failure surfaces along the mortar joints develop. Since the material making up the mortar is usually weaker than the masonry unit material, cracking is most likely to occur within the mortar. The more staggered the joints are in each subsequent course of masonry, the more frictional reaction during horizontal loading increases, limiting the masonry's tensile strength.

$$M_l = {}^{l_2}/_{l_1} \quad (1)$$

The block size distribution provides an overview of the sizes of the bricks making up the façade. In summary, it provides the frequency of each brick residing within a specific distance range. This chart is helpful in MQI analyses, as it is used to categorize the stone/brick dimensions parameter (SD). This parameter is categorized based on the predominance of bricks within specific distance ranges, which the block size distribution provides. The dimensions of bricks contribute significantly to the seismic and static responsivity of masonry walls. Additionally, larger bricks weigh more, which causes confinement effects and affects the distribution of static and dynamic actions within the masonry system.

Results and Discussion

The first experiment is completed by training models for 20 epochs to 100 epochs, doubling the amount between each iteration. The output images from each inference are processed using the aforementioned Python script to obtain CSV spreadsheets containing the total number of blocks detected. As shown by the number of bricks detected for both orthophotos in Figure 5, it is evident that the effectiveness of the training reaches a plateau at 40 epochs for the northwestern orthomosaic and 60 epochs for the southwestern one. 40 epochs are chosen as the optimal value due to the latter amount causing the detection of false positives in the southwestern orthomosaic. The difference in the number of blocks detected between the two iterations is minuscule. Therefore, it is acceptable to use the 40-epoch iteration. Training models beyond this amount causes the brick detection performance to degrade as fewer masks are generated. However, the confidence ratio threshold has been high to eliminate false positives from the outputs. Therefore, it is evident that the more epochs a model is trained on, the more the confidence ratios of each detection decrease. This problem stems from overfitting, where the more the model is trained, the more it begins to overperform on the training data, causing the model's accuracy to degrade when evaluating it on external data. Furthermore, it is also important to observe the accuracy of the models by observing their inference results.

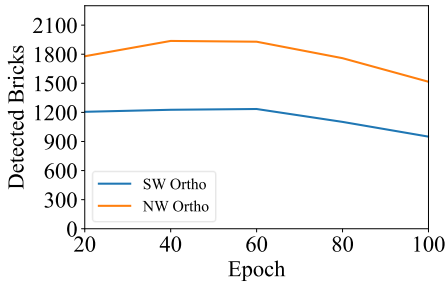


Figure 5: Epoch Adjustment Model Detection Results

Upon determining the optimal number of epochs, several new iterations are trained where the weight decay is adjusted. For smaller batch sizes, it is recommended that the weight decay remains at a low value for smaller batch sizes, as doing so for larger sizes yields poor results (Smith, 2018). The default batch size set by Ultralytics for training is 16, which means that 16 images from the training data set are processed during each forward pass during the training process. As such, several new models are trained with incremental weight decay values from 0.0005 to 0.001 to observe the effects of decreasing and increasing this hyperparameter. Figure 6 shows that increasing the weight decay to 0.007 improves the detection of bricks in both orthophotos the most. Despite the aforementioned recommendation, decreasing the weight decay yielded poorer results, most likely from insufficient regularization. The performance decays at values larger than 0.0007, which signifies that this is the threshold at which excessive regularization starts to occur, leading to underfitting (Wu et al., 2024).

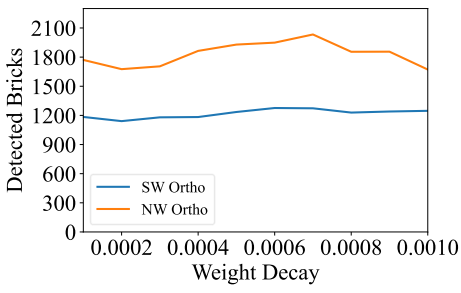


Figure 6: Weight Decay Adjustment Results Comparison

Several more iterations are trained where the learning rate is adjusted. It is also recommended that the learning rate must be lowered when training under smaller batch sizes. Once again, the learning rate is increased and decreased to observe their effects on the accuracy. Figure 7 shows that the default learning rate of 0.01 is optimal for detecting the most blocks for the northwestern orthomosaic, while a learning rate of 0.012 yields more bricks for the southwestern one. In this case, the decrease in accuracy when detecting blocks in the northwestern elevation is more significant than that of the adjacent one. The results show the importance of fine-tuning detection models to work for more than just one target image, as adjusting hyperparameters can quickly improve the detections for one particular subject but yield the opposite effect for others. When the learning rate is lowered, performance decreases, which may signify that underfitting is occurring, where the model had more difficulty learning during training, leading to suboptimal accuracy (Raitoharju, 2022). While increasing the learning rate improves performance for some input images, the lack of performance may signify that the subtle differences in the objects between each photo are significant enough to cause overfitting. Therefore, a learning rate of 0.01 is the most optimal for this case study, as there is a better balance between quantity and accuracy.

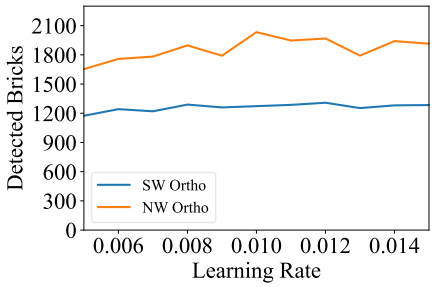


Figure 7: Learning Rate Adjustment Results Comparison

The optimal batch size remains as the default value, 16, due to the weight decay and learning rate adjusted specifically for this batch size. The results shown in Figure 8 reinforce the proportionality of the weight decay and learning rate to the batch size, as it shows that neither increasing nor decreasing the batch size provides any improvements. As there is increased regularization due to a larger weight decay, decreasing the batch size causes further regularization, which can decrease accuracy to excessiveness (Kandel and Castelli, 2020b). In addition, increasing the batch size requires larger learning rates to ensure stability. Therefore, in order to take advantage of the batch size to speed up the training process, it is necessary to adjust the other hyperparameters accordingly.

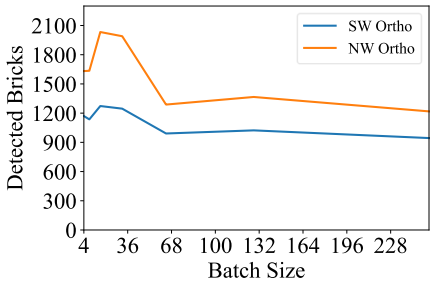
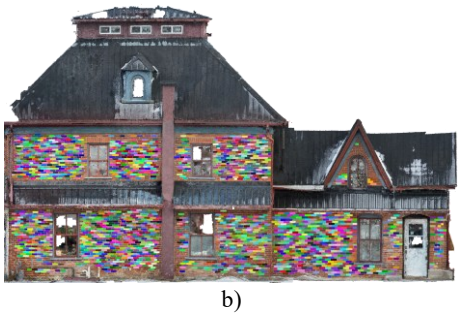


Figure 8: Batch Size Adjustment Results Comparison

The resulting masks from the most optimal model are provided in Figures 9 a) and b). The model detects sufficient bricks for plausible block size distributions and staggering ratios. While there are still noticeable gaps in the detections, such as at the lintels and various sections, it fails to detect false positives, which can distort the results. In addition, as this model is specifically tailored to help automate the Masonry Quality Index analysis, lintels are outside of its scope as they are not considered when determining the parameters. It is possible to improve the accuracy of the model further by training new iterations with changes in batch size and adjustments to the other hyperparameters accordingly. In addition, accuracy may improve by utilizing a larger dataset with annotations more similar to the target objects in the orthomosaics. Augmentation measures may also be employed to artificially expand the dataset by altering the existing images, such as rotations, reflections, and changes in brightness and contrast (Mumuni and Mumuni, 2022).

a)



b)

Figure 9: Case Study Mask Outputs: a) Southwestern Façade, b) Northwestern Façade

While it is important to analyze the performance of the models by comparing their ability to detect bricks within the ortho mosaics, observing the training metrics provides additional valuable insight into how powerful each

hyperparameter is in minimizing the loss function. Ultimately, the two primary hyperparameters that significantly improve accuracy are the number of epochs and the weight decay. As shown in Figure 10, a higher weight decay value does indeed decrease the loss when the model trained under 40 epochs, albeit by a small amount. Additionally, both increasing and decreasing the learning rate less reduces the loss even further, despite less bricks being detected. As a lower loss value signifies that the model performs accurate predictions, the decrease in detected bricks may further prove that overfitting is occurring, as the model performs well on the MCrack1300 dataset, but not the orthomosaics. It is evident that adjusting the learning rate improves the accuracy of the model more than the weight decay, but in the case of this study, adjusting the weight decay ensures more brick detections, as increasing it reduces overfitting, allowing for more detections to be made with higher confidence.

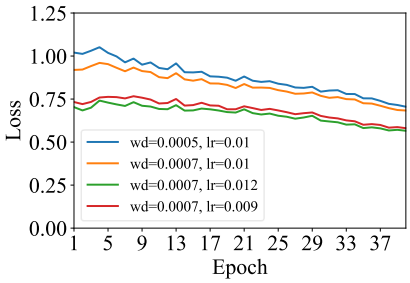
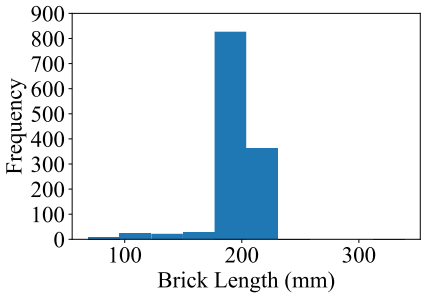
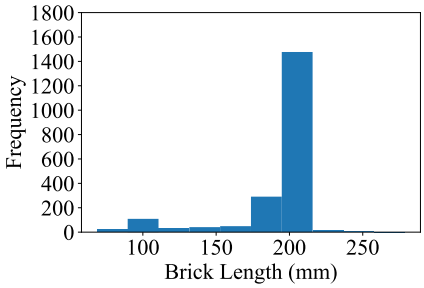


Figure 10: Comparison of Loss Reduction for Different Hyperparameter Adjustments

Now that the most optimized iteration of the brick detection model is obtained, its outputs can be used to obtain the block size distributions and staggering ratios. Once YOLOv11 generates all of the brick masks, they are processed by an additional Python function where the corresponding polygons are converted into bounding boxes to estimate each brick's heights and lengths. As the Masonry Quality Index considers the largest dimensions of each brick, the script outputs the width of each brick into a spreadsheet for use in providing the histogram shown in Figures 11 a) and b). The outputted dimensions are also scaled using a coefficient which converts the sizes of the bricks in pixels to their real-life dimensions in millimeters. The block size distributions for both ortho mosaics show that most bricks have dimensions more significant than 200 mm but less than 400 mm. Therefore, the associated MQI parameter for the masonry unit dimensions is to be considered partially fulfilled (PF). While it is evident in the result masks that the vast majority of bricks are of similar lengths, with a few slightly more significant, there are a small number of blocks with lengths less than 100 m. This is to be expected as the YOLOv11 model detected bricks perpendicularly placed to interlock with the adjacent facades. In addition, it is also apparent that the YOLOv11 model still struggles with a small number of bricks, as there are masks that do not fully encapsulate them. Regardless, enough bricks are detected accurately to provide a block size distribution, which helps fulfill the SD MQI criteria.



a)



b)

Figure 11: Block Size Distributions: a) Southwestern Façade, b) Northwestern Façade

The staggering ratios are obtained using an automated tool, shown in Figure 12, that samples and measures the horizontal offsets between head joints in the provided masks. The masks from the outputted PNG image are used as the tool's input data, where it superimposes the masks over a grid, where the cells are assigned a corresponding brick based on the extent of each mask occupying the majority of space in each cell (Griesbach et al., 2024). In the case a cell lies on top of the mortar, its detection expands to obtain the area of its nearest neighboring brick. This process results in a set of rectangular brick geometries with equal heights in each course, flush with each other with zero-thickness interfaces. Afterwards, the tool samples multiple head joints along the highest course of the façade and then scans for the closest head joint in the course under it. If the path following the head and bed joints intersects with the exterior of the wall, it is deemed as invalid and is discarded. This process continues iteratively for each subsequent course until it finds the shortest path from the highest to lowest courses along the head joints. The staggering ratio is then calculated as per equation (1), where the total length of the path is divided by the distance between the start point and end point.

For this study, multiple staggering ratios are provided for both orthomosaics, with distributions shown in Figures 13 a) and b). For the northwestern elevation, a total of 48 staggering paths are sampled, resulting in ratios ranging from 1.40 to 2.33, with the majority of ratios being above 1.70. Similar results are shown for the southwestern elevation, with 21 sampled ratios ranging from 1.68 to 2.28. The majority of the staggering ratios from both results are well above the threshold of 1.6 for both single

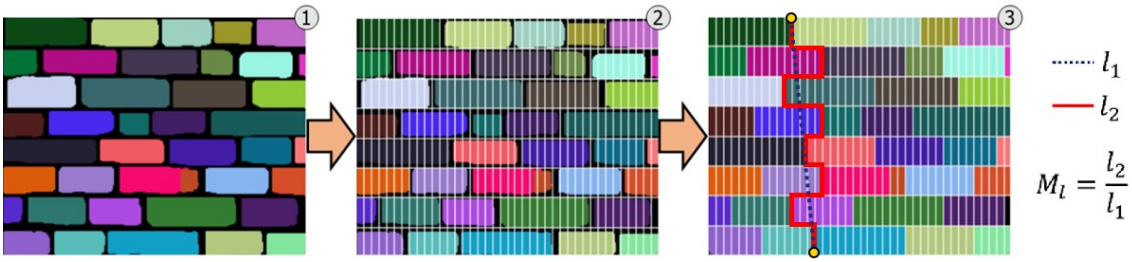


Figure 12: Staggering Ratio Calculation Workflow (Griesbach et al., 2024)

and double lead walls, which means that the corresponding MQI parameter, VJ, is considered to be fulfilled (F). There are several staggering ratios that are below 1.6, but they are disregarded due to being only a miniscule minority. The ratios vary greatly both orthomosaics as the masonry bond is not uniform but is rather irregular with bricks of different lengths found in each course. These irregularities lead to the detection of both larger and smaller offsets between head joints. The detection of smaller ratios, most notable in the northwestern façade, may also stem from paths drawn along head joints that are in remarkably close proximity from one another, they have almost negligible horizontal offsets. Furthermore, there may be inaccurate ratios that stem from paths drawn along head joints of incomplete segmentation masks. Regardless, the automated procedure is successful in providing detailed overviews of the building's masonry morphology.

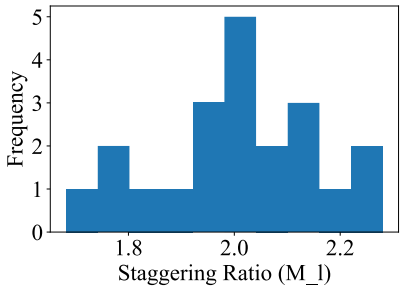


Figure 13: Staggering Ratio Distributions: a) Southwestern Façade, b) Northwestern Façade

Conclusion

This study shows that novel computer vision techniques can benefit those using the Masonry Quality Index, which can play a crucial role in the preliminary on-site assessment and computational modeling of masonry structures. By utilizing an extensive dataset and taking advantage of the accessibility of machine learning frameworks, the coupling of object detection and segmentation models with user-friendly interfaces allows practitioners to take advantage of its power to speed up their analysis workflows. While this study showed the importance of selecting a proper dataset to train YOLOv11 models, it also shows the importance of correct hyperparameters to ensure optimal performance. Moving forward, this case study will be improved further to include the identification and quantification of mortar joint thicknesses, and the delineation of structural defects in its scope, including mortar joint cracks, brick failures, and various other forms of damage.

Acknowledgments

The authors would like to acknowledge Peter Griesbach for providing the tool used to obtain the staggering ratio for the orthomosaic masks. Moreover, the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN- 2022-04938 is gratefully acknowledged.

References

Borri, A., Corradi, M., Castori, G., De Maria, A., 2015. A method for the analysis and classification of historic masonry. Bulletin of Earthquake Engineering 13. https://doi.org/10.1007/s10518-015-9731-4

Griesbach, P., Farcasiu, A., Pulatsu, B., 2024. Digital Replication of Unreinforced Masonry Walls with Irregular Openings via an Efficient Algorithm for Structural Analysis. CSCE Annual Conference 2024.

Hafner, I., Kišiček, T., Renić, T., Ožić, K., 2021. An insight into The Masonry Quality Index (MQI) method for the visual assessment of existing masonry structures, in: 1st Croatian Conference on Earthquake Engineering. University of Zagreb Faculty of Civil Engineering, pp. 643–654. https://doi.org/10.5592/CO/1CroCEE.2021.188

Jepkoech, J., Mugo, D.M., Kenduiywo, B.K., Too, E.C., 2021. The Effect of Adaptive Learning Rate on the Accuracy of Neural Networks. International Journal of Advanced Computer Science and Applications 12. https://doi.org/10.14569/IJACSA.2021.0120885

Kandel, I., Castelli, M., 2020a. The effect of batch size on the generalizability of the convolutional neural

networks on a histopathology dataset. ICT Express 6, 312–315. https://doi.org/10.1016/j.icte.2020.04.010

Kandel, I., Castelli, M., 2020b. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. ICT Express 6, 312–315. https://doi.org/https://doi.org/10.1016/j.icte.2020.04.010

Khanam, R., Hussain, M., 2024. YOLOv11: An Overview of the Key Architectural Enhancements.

Kingma, D.P., Ba, J., 2014. Adam: A Method for Stochastic Optimization.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R., 2023. Segment Anything. Proceedings of the IEEE International Conference on Computer Vision 3992–4003. https://doi.org/10.1109/ICCV51070.2023.00371

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P., 2014. Microsoft COCO: Common Objects in Context.

Loshchilov, I., Hutter, F., 2017. Decoupled Weight Decay Regularization.

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

Nielsen, A., 2015. Neural networks and deep learning.

Over, J.-S.R., Ritchie, A.C., Kranenburg, C.J., Brown, J.A., Buscombe, D.D., Noble, T., Sherwood, C.R., Warrick, J.A., Wernette, P.A., 2021. Processing coastal imagery with Agisoft Metashape Professional Edition, version 1.6—Structure from motion workflow documentation.

Raitoharju, J., 2022. Convolutional neural networks, in: Deep Learning for Robot Perception and Cognition. Elsevier, pp. 35–69. https://doi.org/10.1016/B978-0-32-385787-1.00008-7

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2016-Decem, 779–788. https://doi.org/10.1109/CVPR.2016.91

Smith, L.N., 2018. A disciplined approach to neural network hyper-parameters: Part 1 -- learning rate, batch size, momentum, and weight decay.

Terven, J., Cordova-Esparza, D.-M., Ramirez-Pedraza, A., Chávez Urbiola, E., 2023. Loss Functions and Metrics in Deep Learning. A Review. https://doi.org/10.48550/arXiv.2307.02694

Wu, H., Wang, W., Malepathirana, T., Senanayake, D., Oetomo, D., Halgamuge, S., 2024. When to Grow? A Fitting Risk-Aware Policy for Layer Growing in Deep Neural Networks. Proceedings of the AAAI Conference on Artificial Intelligence 38, 5994–6002. https://doi.org/10.1609/aaai.v38i6.28414

Ye, Z., Lovell, L., Faramarzi, A., Ninić, J., 2024. Sam-based instance segmentation models for the automation of structural damage detection. Advanced Engineering Informatics 62, 102826. https://doi.org/https://doi.org/10.1016/j.aei.2024.102826