

Automated Brick Pattern Generator for Robotic Assembly using Machine Learning and Images

Bárbara Andrade Zandavali¹, Manuel Jiménez García²

¹University of East London ²University College London

¹b.zandavali@uel.ac.uk ²manuel.garcia@ucl.ac.uk

Brickwork is the oldest construction method still in use. Digital technologies, in turn, enabled new methods of representation and automation for bricklaying. While automation explored different approaches, representation was limited to declarative methods, as parametric filling algorithms. Alternatively, this work proposes a framework for automated brickwork using a machine learning model based on image-to-image translation (Conditional Generative Adversarial Networks). The framework consists of creating a dataset, training a model for each bond, and converting the output images into vectorial data for robotic assembly. Criteria such as: reaching wall boundary accuracy, avoidance of unsupported bricks, and brick's position accuracy were individually evaluated for each bond. The results demonstrate that the proposed framework fulfils boundary filling and respects overall bonding structural rules. Size accuracy demonstrated inferior performance for the scale tested. The association of this method with 'self-calibrating' robots could overcome this problem and be easily implemented for on-site.

INTRODUCTION

Robotic assembly for brickworks has been researched since the 1990s, (Anliker, 1988) yet in the last decade, a renewed interest for automatization has emerged. The introduction of robotic assembly in brickwork starts as a cost-effective alternative to traditional assembly. More recently, designer and builders have taken advantage of customization with minor cost impact using digital technologies. (Bonwetsch, 2015) While picking-and-placing is a trivial task for automation, the construction constraints is still a challenge. The impact of digital tools also influenced how architects represent brickwork. Hence, the design representation for the robotic bricklaying is also the 'instruction' for its automation.

Bonwetsch (2015) classifies representation for brickwork automation in two approaches: top-down

and bottom-up. While top-down representation subdivides a global and well-defined structure (wall boundary) into smaller units (bricks), bottom-up representation concentrates on the units' arrangement and its local relationship ignoring overall constraints. Top-down approaches are excessively technical, removing the opportunity for innovation, in turn disregarding the boundary constraint reducing the bottom-up applicability to real life scenarios. A more appropriate representation would combine both approaches, addressing a variety of shapes but still following a boundary.

Less declarative methods, such as machine learning image-to-image translation models, are suitable solutions. The machine learning models are able to generalize the content of the filling for any boundary condition based on image representations. (Isola et

al., 2016) In parallel, machine vision technologies enlarge the robot's ability to handle variations on an ongoing task, enhancing its flexibility. (Automata, 2018) For these reasons, this paper investigates an alternative method of representation for automated brickwork using images. This research suggests that using images for representation methods would benefit the integration of construction tasks to automation. Moreover, the application of less declarative methods as artificial neural networks is an alternative to be generic (different boundaries) while maintaining construction logic and pattern aesthetics. This paper, in turn, investigates whether image-to-image translation models' outcomes are satisfactorily accurate and generalized to be integrated to on-site robotic assembly?

BACKGROUND: MASONRY AUTOMATION AND MACHINE LEARNING

Masonry is an additive construction method consisting of layering units in courses from the lowest to the highest level. Although brickworks automation has been available for over 30 years, researches started to explore programmability potential shifting from engineering-oriented practices towards a design-oriented approach. The major downsides of brickwork automation were related to their lack of adaptation to the site environment and handling variations in the predefined task.

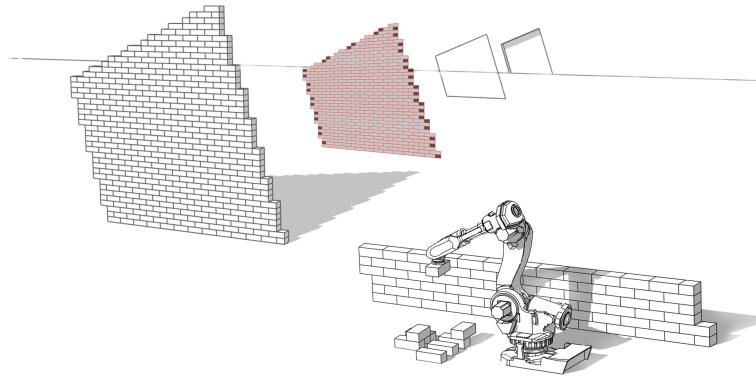
To enlarge the robot's adaptation to site conditions, previous research integrated mobility features and feedback systems to industrial robots. Alternatives for mobility include (a) developing a container to move the robot on site (Bärtschi et al., 2010), (b) adapting tracks in the robot (Arch Union, 2018), and (c) using an aerial structure to 'suspend' the robots (Gramazio & Kohler, 2016). Helm et al. (2012) and Dörfler et al. (2016) tackled both mobility and capturing information by equipping industrial robots with a mobile track system and 2D and 3D scanners that allowed the unit to detect objects or obstacles. Collaborative robots with mobility systems are also an alternative to enlarge robot's site adaptation. Will-

mann et al. (2014) publicised a complete workflow for Aerial Robotic Construction tested with blocks assembly. One of the challenges faced concerned the accuracy and reliability to place a brick in the 'correct position'. The inconstancy of 'flying' conditions reflected in system precision. These attempts focused on incorporating hardware devices (wheels, drones, cameras, scanners) to enlarge the robot's site adaptation with minor efforts to increase its flexibility in the bricklaying task.

Two different trials focused on solving task adaptability with smart algorithm solutions for a different purpose. These robots use systems to capture information from the environment and calibrate their accuracy based on the task to be performed. Nair et al. (2017) trained a robot to manipulate a rope using self-supervised machine learning model and imitation. Robot EVA (Automata, 2018), in turn, is a small and unexpansive robot developed to be trained based on imitation. The user first demonstrates the move and positioning, then the robot replicates the action calibrating the system according to the environment. This paper suggests that the embodiment of automation can go beyond sensing the environment and concentrates on creating a system that interprets the environment. It supports that the process of capturing the area to be filled, defining the boundaries and calculating each brick position is an intuitive process that can be replicated in automation with the support of artificial intelligence.

Machine learning (ML) is a subset of artificial intelligence that uses statistical techniques to enable computers program's to progressively improve performance on a specific task via experience ('learning'), without being explicitly programmed (Samuel, 1959). According to Duda & Stork (1976), learning refers to how the algorithm reduces the error on a training set by adjusting its weights. In the paper by Goodfellow, Bengio & Courville (2016), the task was to generate new examples based on the images in the training data, defined by them as synthesis and sampling. As data representation is a key factor for machine learning efficiency (Goodfellow, Bengio

Figure 1
Framework
diagram.



& Courville, 2016), using images for machine learning has its challenges and specificities. ML models trained with images datasets calibrate its weights according to every pixel values (RGB).

Inspired by the visual cortex behaviour, Convolution Neural Networks (CNN) is a powerful network for image processing tasks, where each convolution layer extracts a specific feature (filter) from the input image. Gatys, Ecker & Bethge (2015) investigated CNN potential for artistic purposes. The model undertakes that content and style representations are well separable in CNN. As a result, it produces a new image combining content (photo) and style (artwork) from two sources. Although the network results are innovative, the input image constrains the content of the outcome, which limits the generalization to other images.

Along these lines, Goodfellow et al. (2014) supported that methods for generative models are less disseminated than discriminative due to its complexity to 'converge'. They overcome this drawback by proposing a Generative Adversarial Network (GAN), whose architecture combines two models - generative and discriminative - in a competitive method to create images undistinguished from the genuine dataset. (Goodfellow et al., 2014). Isola et al. (2016) proposed an intermediary approach, the Conditional Generative Adversarial Networks (cGAN), a

general-purpose solution to image-to-image translation problems, publicly available as 'pix2pix'. (Isola et al., 2016) Like GAN architecture, Pix2Pix trains generative and discriminative networks simultaneously. Alternatively, it uses a pair of images for training, the input and the target. Pix2pix model learns the objective throughout training, by comparing the defined inputs and outputs during and inferring the objective. cGAN was chosen to be part of this paper due to its flexibility, the absence of labelling, and reliability even with small datasets. (Isola et al., 2016)

MATERIALS AND METHODS

This paper proposed a framework for automated brickwork using a machine learning model based on image-to-image translation (cGAN) able to replicate traditional bonding to every possible flat shape. The framework development requires (a) the development of a large dataset of brick walls, (b) the network training and (c) the translation of images outcomes into instructions for assembly. To create the dataset, a filling algorithm describes the brick patterns for a variety of wall geometries. The second step comprises training cGAN (Isola et al., 2016) to generate brick patterns based on each bond dataset using image pairs with the input (wall boundary) and target image (brick wall). Then, an image processing algorithm converts the images into spatial positions for

automatic assembly simulation. Fig 01 illustrates the frameworks' major steps: recognizing a boundary, returning an image of the wall with a bond pattern and, finally, converting its positions in space to be built.

Generating the Dataset

The goal of this algorithm is to fill a variety of wall shapes with predefined bond patterns. Each bond pattern has a rule set to describe the brick arrangement. The chosen bonds use a single standard brick shape in two positions: stretcher and headers. The bond differed themselves by the number of stretcher and headers units, the brick's arrangement, and how each row is aligned with the one below and above. The algorithm scans the polygon area (wall) checking whether the brick is inside the boundary. The screen size used in the framework (500 x 500 pixels) refers to 500 x 500 cm. The filling algorithm developed for the bricks' patterns comprise four main methods: (a) wall shape generation, (b) brick patterns filling, (c) exporting methods for images (png format - pixels) and positions (csv format - vectorial).

Firstly, the algorithm creates a bi-dimensional shape that represents wall shapes. Both elements are described by a quad and its four vertices, located into one of four screen quadrants. The lower vertices are constrained to the same vertical value to guarantee that the first row is parallel to the floor. After the wall boundary is set, the algorithm calculates the vertical and horizontal minimum and maximum values defining a bounding box outside the polygon. (Fig. 2) To describe the brick bond within the wall boundary, this paper uses a 'Scan Line Polygon Filling algorithm'. (Kubitz, 1968; Reynolds 1997; Lieberman 1978).

The algorithm goal is to define the polygon edges and scan its area pixel by pixel in the screen using the edges as starting and stopping points. The brick pattern algorithm, in turn, iterates respecting the brick size instead of checking every pixel. The minimum and maximum method from wall shape creation defines the scanning area and the starting point for scanning area combines the 'back' value for the 'x' coordinate and the bottom value for the 'y' co-

ordinate. The algorithm, then, checks the pixel colour in the brick centre and its four corners to assure that the whole brick will be comprised within the boundary (Fig. 2). The filling pattern varies for each bonding, this paper uses four traditional bonding styles: (a) Stretcher, (b) Header, (c) English, (d) Flemish. After creating the brick pattern, the algorithm places sliced bricks to fill in the remaining gap spaces. The filling algorithm outputs comprise two images, the wall boundary and the bonding, and a table. The table summarises every brick position (coordinate X, Y and Z), dimensions and type.

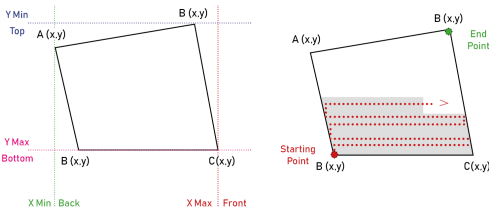


Figure 2
Scan line Filling
Algorithm Diagram.

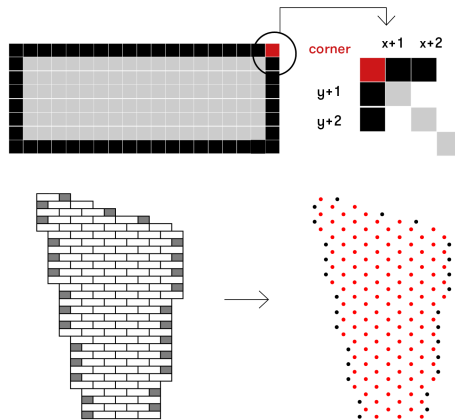
Machine Learning

The machine learning step comprises the data set pre-processing, the model training and testing. Dataset creation is a key factor for training the network. The chosen network cGAN requires two images to train the network. One image is the network input (A) and the other is the target image (B). In this research, input images describe the wall boundaries while the target images describe the same shape filled with the brick patterns. The data set is split into training and testing groups, with a proportion of 70 % training - 30 % testing. Training the model requires the upload of the training portion of the dataset and to specify training parameters. The training method requires image pairs as inputs, the direction of training (from input to target) and the number of iterations (epochs) per image pairs. Since the training is based on the relation of the data set pairs, each bond has a unique model. After training the model, its evaluation uses the testing group to test the model's ability to generalize the solution.

Converting Images into Instructions

As a final step, the framework converts the information from the network outcomes (images composed by pixels) into positions in the space (x, y, z coordinates). The proposed method scans every pixel in the screen to find the bricks extremities. The corner pixel differs from the other pixels due to its neighbourhood pixels colours. (Fig. 3) The algorithm scans the entire screen, finds the upper left corner. The positions are then exported as values in a table, which can be used as instructions.

Figure 3
Diagram of the method to convert the output into instructions.



EXPERIMENTS AND RESULTS

The experiments are divided into two approaches aiming to (a) set up and ideal model for the network (offline), and (b) simulate hypothetical on-site situations (online). Offline experiments tested a variety of dataset representation style, the dataset size, number of iterations (epochs), and including information in the target. (Zandavali, 2018) Since dataset style and size were more representatives, the experiments and results are explained as follows. Online experiments compared the four bonding for the same testing sample. The chosen criteria to evaluate the mod-

els' performance comprised of quantitative and qualitative features.

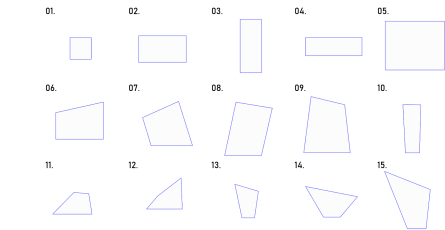
Evaluation Criteria and Sample

The quantitative aspect focuses on the performance indicators calculated by the cGAN model for its networks and performance rate. Each model comprises three networks; one for the discriminator, and two for the generator (GAN and L1), whereas the ideal trade-off is to compensate discriminator and generator simultaneously. The performance rate evaluates the resemblance between the output and the target comparing the value of every pixel in both images. Alternatively, qualitative criteria focus on automated construction constraints as (a) filling accuracy of the input boundary ('Boundary index'), (b) avoiding unsupported bricks (overhangs larger than half of its size), (c) replicating the brick size.

The best model performs the lowest discriminator and L1 loss and highest GAN loss for its quantitative results. Performance rates range from 0 for a discrepant copy, to 1 for a perfect copy. Qualitative indicators take target images as a baseline, the closer the output is to the target, the better its representation and the network performance. Boundary 'index' is the percentual difference area coverage of the filling for a target image (target percental = target bricks area (/) boundary area (*) 100) and the output (output percental = output bricks area (/) boundary area (*) 100). Therefore, the smaller the boundary index, the more accurate is the area coverage. Unsupported bricks are counted for each evaluation shape with obtuse base angles. Since target images do not display unsupported bricks, the fewer unsupported bricks in the outputs, the better. Brick size 'index' calculates a percentual based on the difference in the number of rows from the target and output image (difference = target number of rows - output number of rows). This value is then divided by the baseline and multiplied by 100 (difference (/) target number of rows (*) 100).

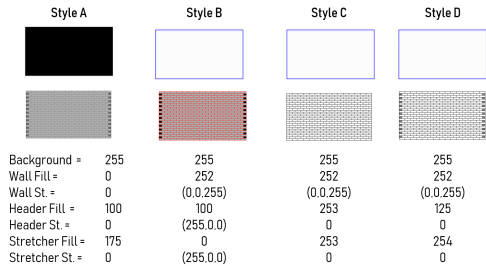
15 images were used as an evaluation sample for qualitative criteria. The shapes vary in size (small, medium and large) and interior angles (square, acute,

and obtuse angles - Fig. 4). The shapes are part of the testing set from the dataset processed after the network training. The variety of shapes aims to evaluate discrepancies for different proportions, sizes and angles. Shapes with obtuse internal angles on the bottom vertices enable the evaluation of unsupported bricks criteria.



Offline Experiments

Comparing Dataset Styles. The definition of style was tested concerning two main aspects: (a) if the model could perform well the ‘filling task’ and (b) if the outcome could be easily transformed into vector data for the assembly. The tested datasets used 200 images for training and iterated 100 times (epochs) for the stretcher bond. The styles varied the filling and colour strokes for the wall boundary in the input image and for the header and stretcher bonds in the target. The four styles are illustrated in Fig. 5 with the respective RGB values for (a) background, (b) wall stroke, (c) wall filling, (d) header brick fill, (e) header brick stroke, (f) stretcher brick fill, (e) stretcher brick stroke. Greyscale values use one value varying from 0 (black) to 255 (white).



The network was able to represent the overall stretcher bonding logic for every style. (Fig. 6) The area coverage for the boundary was similar within the scenarios. However, style A and C show a weaker boundary definition. Unsupported bricks index was the most representative result and demonstrated that using a distinct colour for the bricks (style A and B) coped with the network to ‘figure out’ the structural logic. (Table 1) The bricks’ size representation discrepancy is similar for every dataset. The outputs represented the standard brick (24 x 8 cm) with 26 x 9 cm. Style B displayed a superior quantitative performance since it has lower discriminator and L1 loss results and higher GAN loss results.

Parameters	St. A	St. B	St. C	St. D
Discriminator Loss	0.586	0.214	0.711	0.569
Generator GAN Loss	2.670	4.420	2.236	2.649
Generator L1 Loss	0.037	0.073	0.306	0.088
Performance Rate	0.482	0.533	0.461	0.470
Boundary Coverage (%)	93.66	93.97	92.24	93.78
Un-supported bricks (un.)	4	6	19	23
Brick Size (%)	158.38	155.92	159.14	155.92
	10.56	10.39	10.61	10.39

Comparing Dataset Size. The experiment concerning dataset size compared four scenarios, with 200, 500, 1000, and 2000 dataset. The dataset ratio (training/testing and regular/irregular shapes) is identical for every scenario. (Table 02). The experiment compares each dataset size trained for 100 epochs and style D. Table 02 summarizes the results for each dataset size scenario. Quantitative results demonstrated a significant increase for GAN loss and decrease for the discriminator loss from scenario A' to B' (200 to 500 images training dataset). In turn, from 500 to 1000 and from 1000 to 2000, discriminator loss decreased only 0,028 and 0,033 respectively.

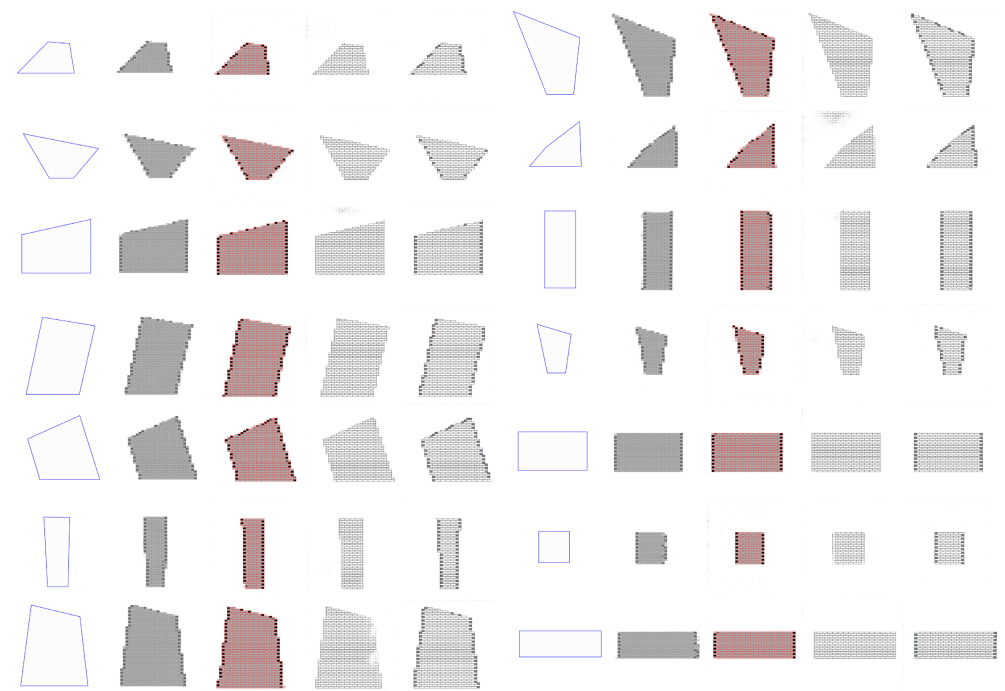
Similarly, GAN loss had a minor growth from 500 to 2000. Qualitative results were less expressive, except for the ‘un-supported’ bricks. The major impact of enlarging the dataset was the decrease of unsupported bricks. While 1000 image training dataset evaluation shapes displayed 24 unsupported bricks, 2000 images dataset reduced this value to 4. This re-

Figure 4
Evaluation shapes.

Table 1
Dataset style
comparison results.

Figure 5
Dataset style of
input and target
images and its
corresponding
filling and stroke
colours.

Figure 6
Input and output
images for the four
dataset styles.



sult suggests that the network exposition to a large dataset contributes to replicate an implicit feature, which is not proportionally related to quantitative indicators.

Table 2
Dataset size
comparison results.

Parameters	A' 200	B' 500	C' 1000	D' 2000
Discriminator Loss	0.569	0.147	0.119	0.086
Generator GAN Loss	2.649	7.437	7.747	8.466
Generator L1 Loss	0.088	0.096	0.100	0.087
Performance Rate	0.47	0.51	0.56	0.49
Boundary Coverage (%)	93.78	94.50	95.24	95.28
Un-supported bricks (un.)	23	18	24	5
Brick Size (%)	158.38	163.49	157.65	158.37
	10.56	10.90	10.51	10.56

Final Model Set-up. To define a final model, this paper compared 8 eight trained models varying style

and dataset size using 100 epochs for training. Their performances are plotted in Fig. 7 comparing quantitative and qualitative criteria. Each model is referred in the graph for its style (A - D) and the number of images used in its training. Except for GAN loss, the model with a lower value has a superior performance. Two models displayed a superior performance (style B - 500 and style D - 2000) are highlighted with a darker grey in the graphs. Except for GAN loss, both models performed similarly. However, the model in style B used one-quarter of the images for its training. This fact reflects proportionally on the time expenditure of its training, from 1 to 4 hours in a standard GPU computer. Another remarkable factor is the efficiency of style B to avoid unsupported bricks. Moreover, the contrast within the colours in style B faci-

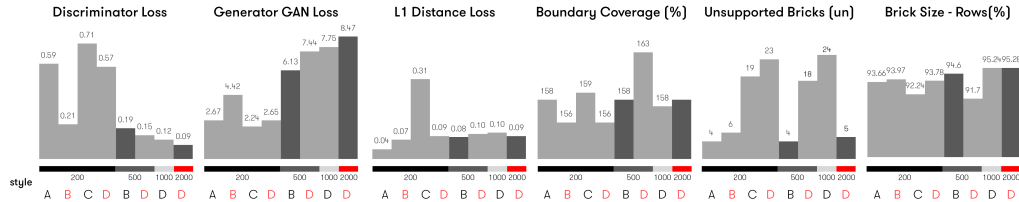


Figure 7
Graph comparing the performance of the tested networks.

tates post-processing the images. For this reason, the set up combining style B, 500 images for training was used for more bonding tests, presented as follows.

Online Experiments

Comparing bonding. Since the goal was to adapt to a task variation, this experiment aims to generate different bonding patterns for identical shapes, evaluate how each populates the boundary, and how the network performs for each pattern. Each bonding has a specific model to train and test its capacity to fill the wall shapes. English and Flemish bonding configuration prevents the filling for polygons with obtuse angles in the base. For this reason, the wall shapes used to train and test the model have base angles within 90 degrees.

The results have shown (Table 3) that the network was able to describe each bonding maintaining the bricks' local relationship. The number of unsupported bricks was higher in the outcomes of Header bonding than in Stretcher bonding. The discrepancy of the size of the bricks was similar in every bonding, reinforcing that this might be a limitation of the network architecture. While Header and Flemish bonds outcomes replicated the edges and bricks stroke with a significant resolution, the strokes within the darker bricks were poorly defined in the English bond (Fig. 8). For a matter of curiosity, although English and Flemish models were trained with constrained shapes, this paper also tested shapes with obtuse base angles and the outcomes are displayed in Fig. 8.

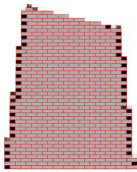
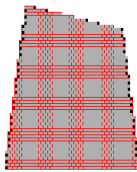
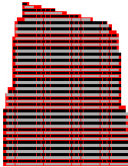
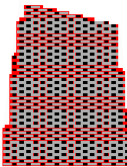
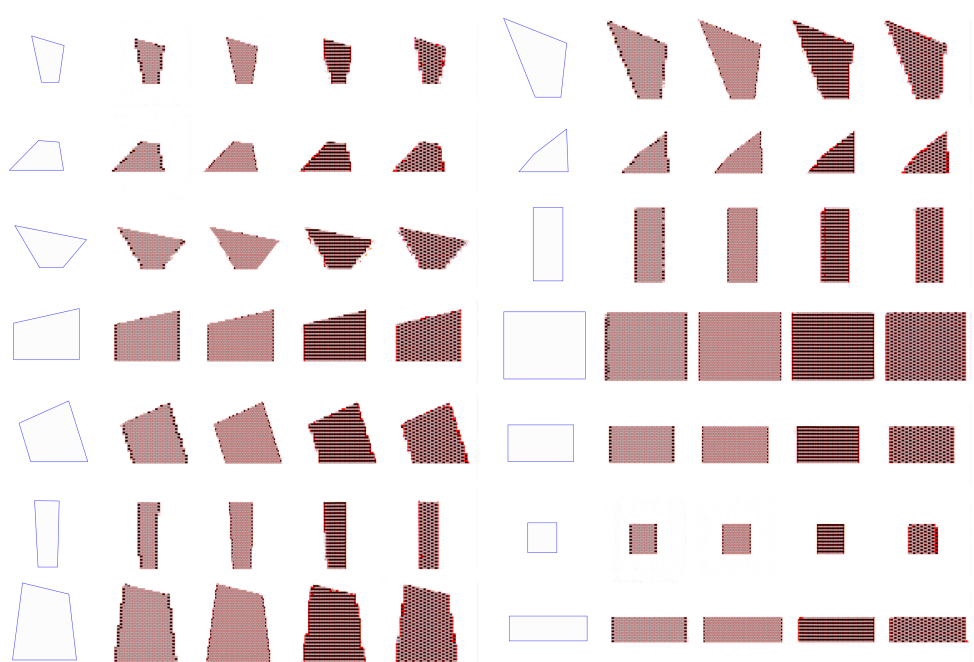
	Stretcher	Header
		
Discrim. Loss	0,190	0,216
GAN Loss	6,134	5,106
L1 Loss	0,083	0,079
Perf Rate	0,519	0,525
Boundary	91,70	92,54
Un-sup. bricks	4	7
Brick Size [%]	158,4 / 10,6	150,4 / 10,3
	English	Flemish
		
Discrim. Loss	0,055	0,109
GAN Loss	6,997	6,92
L1 Loss	0,211	0,078
Perf Rate	0,499	0,501
Boundary	94,61	95,30
Un-sup. bricks	n/a	n/a
Brick Size [%]	92,84 / 10,32	103,58 / 11,51

Table 3
Results for the bonding comparison.

CONCLUSION

The question posed at the beginning of this research addresses the extent to which machine learning models, using images as a method of represen-

Figure 8
Input and output
images for the four
bonding patterns:
Stretcher, Header,
English and
Flemish.



tation, could generate outputs satisfactorily accurate and generalized for automated brickwork construction. The outcomes demonstrated promising results to replicate the bonding pattern, in turn, the accuracy to describe brick size was questionable (10%). Moreover, even the final models ‘placed’ unsupported bricks occasionally. The relevance of dataset representation was evident and, at the same time, unpredictable. The inclusion of qualitative metrics was a key feature to evaluate the network performance for the specific task, while the losses and performance rates lack ‘meaningful’ information. Better outcomes could benefit from increasing or decreasing the number of layers in the network, opting for different optimizers, changing the activation functions and bias functions from the original network. Another limita-

tion faced in this work was the resolution of the network output which compromises the conversion of the bi-dimensional results into 3d models.

The method in this research combines top-down and bottom-up methods gathering control and flexibility to address the wall’s boundary. The outcomes demonstrated that the network was able to recognize the boundary and adapt to a variety of unknown wall shapes. Moreover, the outcomes demonstrated that the network also respected the local relationship, inherent to every bonding, establishing a negotiation between the overall shape and the emergent pattern. Although the results demonstrated a few limitations, this research contributes to a promising and growing field that associates ‘smart’ algorithms to automation. The potential of using machine learn-

ing resides in its level of abstraction, which could increase the system adaptability and enlarge the automation on site.

The results from this research unfolded possible paths for future work. One path envisions training the network for situations that go beyond panels such as including openings and exterior corners. A second attempt focus on testing the framework outcomes in physical experiments which require the adaptation of the solution to environment and machinery constraints. Finally, a third path comprises comparing the results presented in this research to other machine learning models in architecture, such as ones using further image-to-image translation architecture or using vector data information.

REFERENCES

- Anliker, F 1988 'Needs for robots and advanced machines at construction sites. Social aspects of robotics', *5th International Symposium on Automation and Robotics in Construction.*, Tokyo
- Bonwetsch, T 2015, *Robotically assembled brickwork, Manipulating assembly processes of discrete elements*, Ph.D. Thesis, ETH Zurich
- Duda, R, Hart, P and Stork, D 1976, 'Pattern Classification', *Journal of Classification*, 24(2), pp. 305-307
- Dörfler, K, Sandy, T, Giftthaler, M, Gramazio, F, Kohler, M and J, Buchli 2016 'Mobile Robotic Brickwork', *Robotic Fabrication in Architecture, Art and Design*, Robotic Fabrication in Architecture, Art and Design , pp. 204-217
- Gatys, L, Ecker, A and Bethge, M 2015 'A neural algorithm of artistic style', *IEEE Conference on Computer Vision and Pattern Recognition*
- Goodfellow, I, Bengio, Y and Courville, A 2016, *Deep Learning*, MIT
- Goodfellow, I, Pouget-Abadie, J, Mirza, M, Xu, B, Warde-Farley, D, Ozair, S, Courville, A and Bengio, Y 2014 'Generative adversarial nets', *Advances in neural information processing systems*
- Helm, V, Willmann, J, Gramazio, F and Kohler, M 2014 'In-Situ Robotic Fabrication: Advanced Digital Manufacturing Beyond the Laboratory', *Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe: Technology Transfer Experiments from the ECHORD Project*
- Isola, P, Zhu, J, Zhou, T and Efros, A 2017 'Computer Vision and Pattern Recognition Image-to-Image Translation with Conditional Adversarial Networks', *Conference on Computer Vision and Pattern Recognition - CVPR*
- Kubitz, W 1968, *A tricolor cartograph*, Ph.D. Thesis, University of Illinois
- Liberman, H 1978 'How to color in a coloring book', *ACM SIGGRAPH Computer Graphics*
- Nair, A, Chen, D, Agrawal, P, Isola, P, Abbeel, P, Malik, J and Levine, S 2017 'Combining Self-Supervised Learning and Imitation for Vision-Based Rope Manipulation', *International Conference on Robotics and Automation ICRA*
- Reynolds, C 1977 'Filling Polygons', *Architecture Machines*
- Samuel, A 1959, 'Some Studies in Machine Learning Using the Game of Checkers', *IBM Journal of Research and Development*, 3(3), pp. 210-229
- Willmann, J, Augugliaro, F, Cadalbert, T, D'Andrea, R, Gramazio, F and Kohler, F 2012, 'Aerial Robotic Construction towards a New Field of Architectural Research', *International Journal of Architectural Computing*, 10(3), pp. 439-459
- Zandavali, B 2018, *Automated Brick Pattern Generator for Robotic Assembly using Machine Learning and Images*, Master's Thesis, University College London
- [1] <https://automata.tech/product.html>
- [2] <http://www.archi-union.com/Homes/Projectshow/index/id/40>
- [3] <http://gramaziokohler.arch.ethz.ch/web/e/projekte/135.html>