Collapsing Complexities: *Encoding Multidimensional Architecture Models into Images*

Viktória Sándor(✉), Mathias Bank(✉), Kristina Schinegger, and Stefan Rutzinger

Department of Design, University Innsbruck, i.sd, Technikerstr. 21, 6020 Innsbruck, Austria
{Viktoria.Sandor,Mathias.Bank-Stigsen,Kristina.Schinegger, Stefan.Rutzinger}@uibk.ac.at

Abstract. The paper details a 3D to 2D encoding method, which can store complex digital 3D models of architecture within a single image. The proposed encoding works in combination with a point cloud notation and a sequential slicing operation where each slice of points is stored as a single row of pixels in the UV space of a 1024 × 1024 image. The performance of the notation system is compared between a StyleGan2 and existing image editing methods and evaluated through the production of new 3D models of houses with material attributes. The uncovered findings maintain the relatively high level of detail stored through the encoding while allowing for innovative ways of form-finding—producing new and unseen 3d models of architectural houses.

Keywords: Form finding · 3D · Encoding · Point cloud · Machine learning · Architectural design

1 Introduction

Architectural design processes are increasingly situated in the digital space, resulting in a large amount of architectural 3D models. During all phases of an architectural design process, from concept to completion, digital 3D models increasingly act as the link between what we can think, and what we can build [4]. As a result, 3D models have become the core method for communicating and creating an architectural design. Compared with traditional 2D representations, the 3D model provides a more holistic representation of the spatial relationships that lay behind a given design [4], while offering an accurate and adequate representation of the architectural space and its proportions [7]. The transition from 2 to 3D has inevitably increased the complexity for architects to visualise and explore new ideas. This is particularly true early in the design process, where any promising design intent needs to be translated into 3D to be properly communicated. To overcome this new complexity, there is growing interest in working with artificial neural networks in digital design processes. Although artificial neural networks have the potential to address architectural complexities, training networks directly on 3D models remains a challenge.

To establish usable artificial neural networks for design collaboration, we need to develop notations that can transmit the spatial qualities and complexities of architecture. Current research shows several attempts to train artificial neural networks on 3D models. Although some utilise a three-dimensional medium such as point clouds [1] or voxels, the development of 3D machine learning is still difficult for architectural purposes, due to availability, speed, and resolution. Therefore, our research focuses on exploring a 2D notation of digital 3D models, which can interact with a 2D machine learning framework to assist architects in the early stages of the design process.

1.1 Relative Work

Artificial neural networks trained on 2D data have shown a remarkable "talent" for grasping patterns and concepts within complex datasets; but how can we best encode 3D models into 2D mediums to benefit from this? Tomographic data is one such approach and is an inspiration for the work by Kench and Cooper. Their developed SliceGan can synthesise high fidelity 3D datasets using a single representative image. Their approach was successfully demonstrated on material microstructures [6]. Along a similar vein, with a more architectural agenda, Zhang and Huang presented their solution for a machine learning aided 2D–3D architectural form-finding, by introducing a sequential slicing of a given 3D model. In their method, a sequence of images, describing the 3D model, are stitched into a single image for compatibility with a 2D neural network [12]. With The Spire of AI project, Zheng and Ren introduced a method for voxel-based 3D neural style transfer using 2D slicing, in which pixel points of stylized 2D slice images are extracted and mapped into a 3D domain according to the slice order [9].

In contrast to the slicing approach Miguel et al., introduced a method for notating 3D models into a connectivity map utilising voxelated wireframes. Through variational autoencoders, this notation facilitated the generation, manipulation and form-finding of structural typologies [8]. Another example of an abstract notation takes place when storing data in pixels by plotting pixel plots. Although pixel plots are a popular format in data visualisation, they are also used for vertex animation textures. These textures are used to control morphing animation by mapping vectors to colours and storing them on the row corresponding to the animation frame [10]. This approach is often applied within the gaming industry since it is an efficient way to store a lot of data within an image while seamlessly connecting to the shader pipelines.

The solutions for compressing multidimensional data into 2D images are continuously evolving, nevertheless, it is still hard to properly store a complex architectural 3d model in a single 2D image.

1.2 Objectives

With these developments in mind, we want to present a new 3D to 2D notation system that through the manipulation of encoded 2D images can perform design operations on 3D models of architecture. The approach is combining a sequential slicing with a point cloud notation, where additional information such as the materiality is assigned to colours. The coloured point slices are then stored in a single image, encoding the points and their additional information to pixels. By testing the notation with artificial neural networks and traditional image editing techniques, we aim to explore new ways of form-finding in the early stages of a design process, while simultaneously evaluating the encoding methods' ability to transmit spatial qualities and complexities of 3D architecture models.

2 Methodology

The focal point of the research is the development of an encoding framework that allows the compression of a complex 3D model to a single, pixel image. In the proposed encoding method, point clouds play a significantly important role. While point clouds in architecture are mostly utilised in the digitization process of real-world objects, we chose them for their nature of describing digital surfaces and volumes in discrete format. Due to their multidimensional data-storage capacity, they are an ideal medium for representing a large number of (spatial) attributes. The proposed 3D to 2D encoding method in the research relies on the translation of 3D solids to 3D pixels and 3D pixels to 2D pixels that emerge into images. Since at the stage of 3D pixelization of models, it is advantageous for the resolution to remain adaptive and flexible, while irregularities do not interfere with the encoding method, 3D point clouds were chosen over voxels for the translation of 3D geometries.

Based on the desired image resolution of the encodings, the representative clouds are sampled. In this research we aimed for 1024×1024 pixel-sized image encodings, it being currently the most convenient image size for the training of StyleGans. The resolution of the encodings can be adapted for other purposes.

At this stage, the proposed method focuses only on the encoding of 3D solid compositions, where solids represent materials as spatially closed volumes to which colours are assigned. The resulting coloured volume compositions describe spatial concepts through labelled material distributions.

2.1 Encoding

The encoding consists of three main parts, that combine point cloud notation and sequential slicing operations on 3D models: Model Discretization, Colour Mapping and Legend construction.

2.1.1 Model Discretization

As the first step, we start with the discretization of the 3D model dataset. The discretization consists of four different steps. While each step differs in its resolution and sequence, they all preserve the spatial attributes of the original models (material, position, etc.) (Fig. 1).

Through slicing along a pre-defined axis, we divide the solid 3D model into groups of planar surfaces (sections). Each of the resulting surface sets, belonging to the same plane, is then split into 4 areas of similar size. The division can be defined by the dimensions of the whole model or by the local dimensions of each slice. In addition, the size and proportion of the segments can be uniform or customised based on the varying mass distribution of the model (Fig. 2). The four segments are further discretized by point cloud scattering (populating) on the corresponding surfaces.

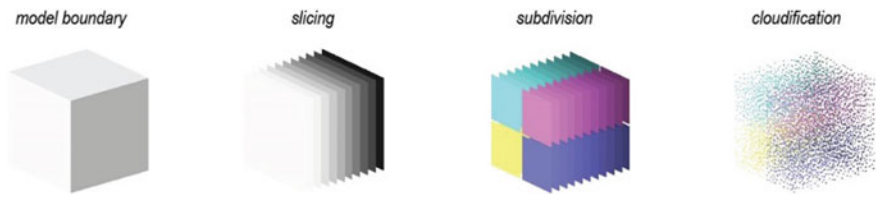

Fig. 1. Discretization Steps. Four model discretization steps of different resolutions and scales: model boundary, slicing, subdivision, cloudification.

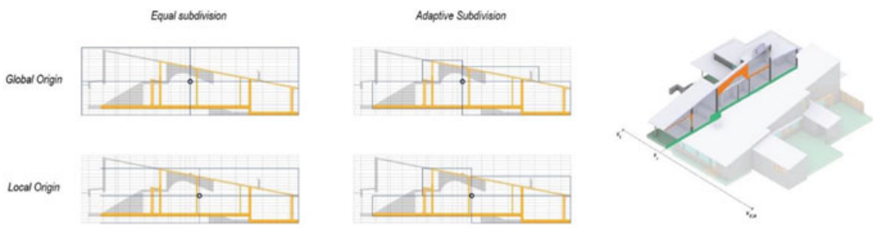

Fig. 2. Global versus local subdivision. Four distinct strategies for slice-subdivision. Top left: Equal slice-subdivision with global origin. Top right: Adaptive slice-subdivision with global origin. Bottom left: Equal slice-subdivision with Local Origin. Bottom-right: Adaptive slice-subdivision with local origin.

The discretization method, detailed above, provide us with precisely structured multi-dimensional 3D point clouds. Other than their densities, which represent the mass-void characteristics of the model, they also visualise the material attributes of space, based on the original materials, assigned to the 3D volumes.

2.1.2 Colour Mapping

The colour mapping of discrete 3D models (coloured point clouds) is based on 8-bit colour mapping of discrete positions in space. Using a similar technique to the previously mentioned vertex animation textures, the X and Y coordinates of each point in each slice segment are mapped to the green and blue channels of the RGB space. The original colour of the point, representing the material property in this dataset, is assigned to a predefined red channel value. As a result, we can express 255 $\times$ 255 positions in each slice segment with up to 255 different materials. Using this technique, we can represent 260,100 different positions on each slice of the model.

To bring the points of the 3D point cloud into a single 1024 $\times$ 1024 pixel image—pixel plot—we keep the discretization structure of the model and use it as the layout strategy for the image. The pixel plot of the 3D model is divided into 1024-pixel rows, four 254-pixel wide columns and an 8-pixel wide legend. The rows represent the slices, the columns their normalised segments, and the legend the position and scale of each segment (Fig. 3). Such structure provides us with the visualisation capacity of 1024 model slices each with a maximum of 1016 material labelled points. The resulting pixel plot stores a point cloud with the size of 1.040.384.

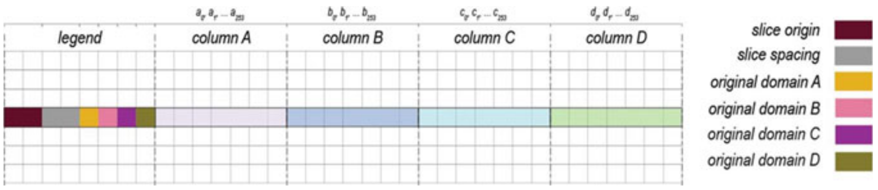

Fig. 3. Pixel plot structure, legend and columns. The four columns A, B, C and D, represent all slice divisions of the model. The legend on the far left of the pixel plot is responsible for controlling the model scale and overall geometry. The first two pixels contain the origins of each slice, 3–4 pixels store the slice spacing values, pixels 5–8 contain the original subdivision proportions of slices.

Since the number of expressible point positions in a slice segment is about 256 times larger than the number of storable positions in a 1024 × 1024 pixel plot, we had to sample the point cloud slices in many cases. After sampling, we tested four different strategies to sort the pixels representing points in the plots (Fig. 4).

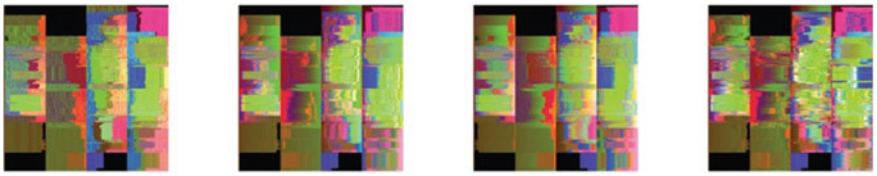

Fig. 4. Pixel sorting, material and position. 4 different strategies to sort the pixels of the plots. The examples show the 4 sorting results of Alvar Aalto's Louise Carre pixel plots. Left to Right: sorting based on materials (Red Channel); sorting based on position distance from Origin (0-GB); sorting based on position and material distance from Origin (0-RGB); sorting based on "travelling salesman problem" 2D

2.1.3 Legend

The first eight pixels of each row of the pixel plot is saved for the so-called legend. While the four 254-pixel-wide columns store the position of the model points mapped to a normalised domain, the legend stores the location of the origins and scales of the original cloud segments. To provide flexibility in the manipulation of 3D models, the legend is split into 6 sections. Pixels 1–2 store the local origin of each slice, pixels 3–4 store the spacing, and pixels 5–8 the original domain sizes of each slice segment (Fig. 3).

2.2 Decoding

The decoding of the pixel plot starts with the reading and structuring of the .raw image file. The next step is the extraction and decoding of the legend pixels to the slice origin, slice spacing and the four domain dimensions. The pixel channels of the remaining 4 columns, A, B, C and D are split and used to define the position and materiality of the model points. The green and blue channels are read as the x and y coordinates of the points while the red channel defines their materiality. To place the points of the columns to the right area of the slices, all green values of A and C and all blue values of C and D columns are inverted. At this stage, all decoded points lie in a single plane around the origin. To introduce the third dimension of the model, we ideally use the spacing values decoded from pixels 3–4 of the legend. To avoid segment distortion, the point coordinates are mapped to the original domain dimension, decoded from pixels 5–8 of the legend. In the case where individual origins are used for the segments, each segment is moved by the decoded vectors of legend pixels 1–2. If suitable, the legend can be replaced by other numeric manipulators.

3 Results

To evaluate the performance of the outlined encoding and decoding workflow, a dataset consisting of 50 labelled 3D models of famous architectural houses spanning 500 years of architectural history was used. All models were constructed from available published documentation and labelled according to the outlined method above. Each model furthermore has a similar level of detail, and a fully modelled interior (Fig. 5).

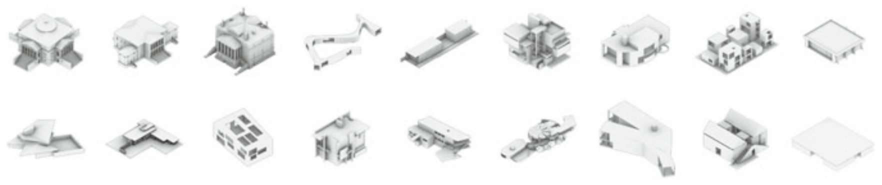

Fig. 5. Examples of 3D house models from the assembled dataset.

The form-finding performance of the proposed encoding–decoding workflow is compared in two different experiments. In the first experiment, we test conventional image editing methods, while in the second experiment, we use a StyleGan2 Ada [5] trained on pixel plots. In both experiments, the form-finding performance is evaluated through the production of new 3D models of houses, and their spatial segments with material attributes.

3.1 Image Editing

The encoding of a full 3D model into a single image offers a lot of new opportunities for manipulating the model through established image editing techniques. To properly test its potential, four housing models were encoded (Fig. 6), and image editing manipulations, such as Colour channel adjustments, Hue change, Saturation change, Blending, Collaging and Legend swapping were explored. Since the legend holds an enormous amount of power over the decoded results, it was excluded from the first five manipulations to better assess how the various editing operations perform.

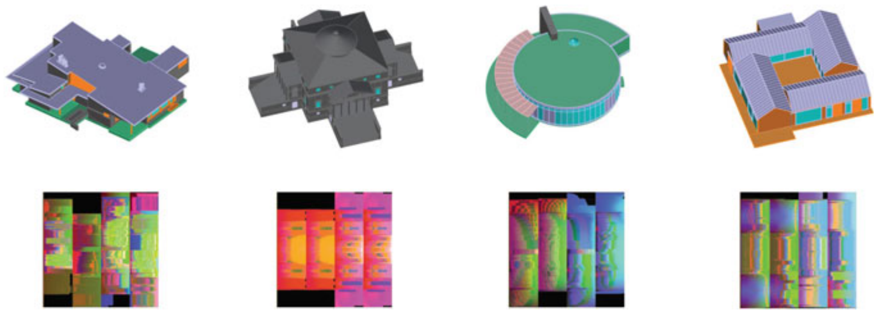

Fig. 6. The four 3D housing models and their corresponding pixel plots which is used for the six image editing manipulations. From left to right: Alvar Aalto, Maison Louise Carre. Andrea Palladio, La Rotonda. Arne Jacobsen, Leo Henriksen House. Lenshow Philmann, House at Mols Hills.

The results establish a small atlas of the potentialities in achieving new and varied 3D models through image editing techniques. Large colour adjustments through channels, hue or saturation easily end up distorting the initial 3D models beyond recognition, while smaller values, especially in the case with hue, show interesting deformations. The results (Fig. 7) from blending and collaging three different pixel plots produce exciting point cloud models, with architectural suggestions, highlighting the large and still mainly undiscovered repertoire in generating new shapes through these methods.

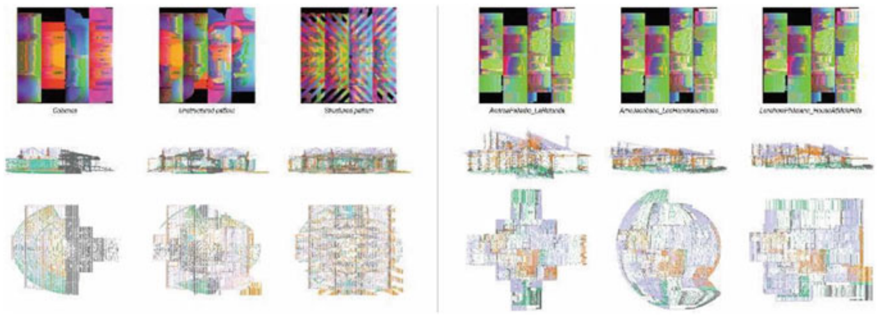

Fig. 7. Image editing results. Left: ***Collage***—Explores three different types of collages. The first exchanges full 256-pixel columns between three unique pixel plots. The second applies an unstructured circular brush pattern, while the third applies an organised repetitive pattern to collage three pixel plots together. The resulting models highlight the potential of using collaging techniques to create new models. Right: ***Legend***—The manipulation replaces the legend of a pixel plot with the legend from three other pixel plots (Fig. 6). As seen, this remaps the base model into the boundary shapes of the models, from which the replacement legends came. This is a powerful method to control and reconfigure complex models from one boundary representation into another

3.2 StyleGan

2D StyleGans have shown a remarkable ability for producing synthetic results that appear eerily similar to the data on which they were trained [11]. There are a plethora of different StyleGans and approaches for this purpose. For our experiments, we use the base repository for StylGan2 Ada from Nvidia [3] and test how well a network can generate new images based on our encoding structure. For the evaluation, we observed the extent to which architectural forms are reproduced in the decoded point cloud models.

3.2.1 Training on Segments

To generate sufficient data for training an artificial neural network, the full dataset of 50 architectural 3D models of houses were used. Following the outlined encoding method, 50 pixel plots were obtained from the 50 architectural 3D models for each slicing direction (Fig. 10). To increase the size of the training dataset, all pixel plots were divided along the columns into sixteen 256 × 256 pixel segments (Fig. 8).

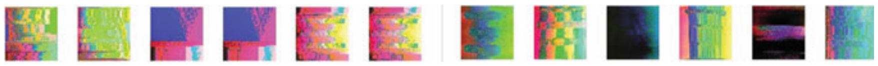

Fig. 8. Segment training data versus StyleGan outcome. To the left is a subset of the segments used to train the network, while the right displays a selection of synthetic images produced by the trained StyleGan2 network.

The StyleGan2 Ada network was trained with a dataset of around 2000 images at 256 × 256 resolution. Furthermore, it was utilising transfer learning, training on top of the FFhQ-10 k dataset at 256 × 256 [2]. The network was trained for 3000 kimg. The produced synthetic images (Fig. 8) appear similar in layout but are limited in content. Although the resulting architectural properties of the synthetic pixel plots segments were vague, some of them showed recognizable spatial properties of the training data when decoded into point cloud models (Fig. 9).

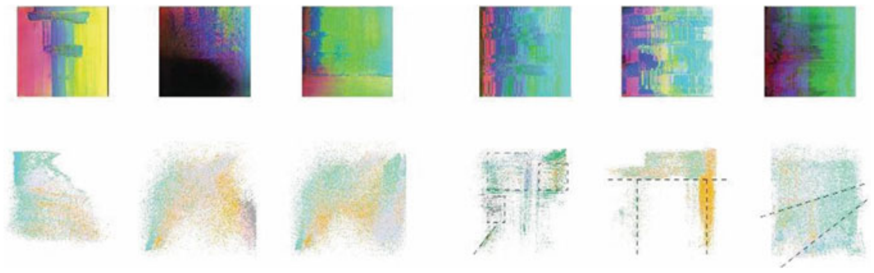

Fig. 9. Decoded results, synthetic segments. To the left are three common outcomes, while the right side displays three of the best, but also uncommon, outcomes

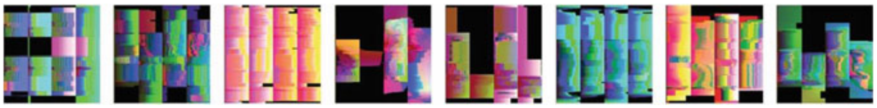

Fig. 10. Examples of pixel plots from 3D houses. A subset of the dataset used for training the StyleGan network on the full pixel plots.

3.2.2 Training on Entire Models

Similar to the segment training, a dataset consisting of the pixel plots depicting the 50 architectural 3D models was assembled (Fig. 10). This time, instead of splitting the pixel plots, the training dataset was multiplied by slicing each house in five directions, giving a total of 500 images after vertical mirroring. The legend stored the dimensions and the origin of the global domain in each pixel plot, which provided representative information about the dimensions of the model in the entire plot.

The StyleGan2 Ada network with the full pixel plots was trained with the same basic parameters as the segments but at a 1024 × 1024 resolution. Due to the relatively smaller dataset of only 500 images, the decoded results shown in Fig. 11, aren't very indicative of the feasibility of utilising a StyleGan to create new shapes. All the decoded images in Fig. 11 creates rather noisy and vague models without any particularly distinct spatial properties. In the most successful ones to the right, the legend and the noise appears less arbitrary. Here the distribution of materials in the clouds show some clarity that starts imitating the characteristics of the original 3D models.

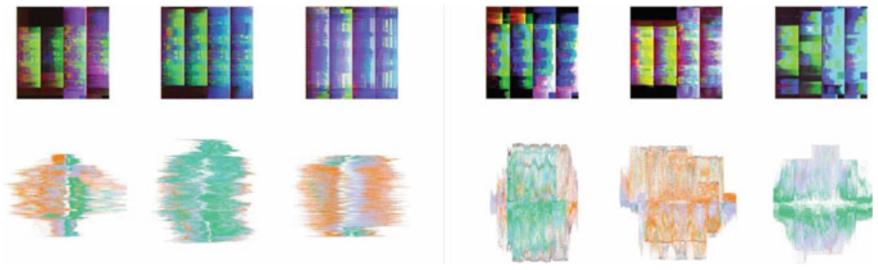

Fig. 11. Decoded results, full-size synthetic pixel plots. To the left are three common outcomes, while the right side displays three of the best, but also uncommon, outcomes.

4 Conclusion

In this paper, we presented a new method for architectural form-finding using existing concepts of houses. We showed that our method can be used to encode complex 3D models into a single image, decode a single image back to complex 3D models, and create architectural 3D models, using image editing techniques. Through the comparison of encoded images, one could recognize spatial patterns, representing the different material distributions in space. Thanks to the developed legend of the encodings, models could be merged, collaged and blended without losing the unity of decoded clouds. With our second method, we also showed how new, unseen 3D point clouds can be generated by a styleGAN, after being trained on the encoded images. While the "architectural" performance of the GAN generated clouds were not even close to the successful image edited concept-collages, -blends and -legend exchanges, in some cases we could still recognize the transferred spatial qualities of the original dataset. Since our current training data was rather small, the results are keeping us motivated.

However, during this research, we have encountered several limitations that need to be acknowledged. Our current encoding technique is working exclusively with solid models, making the preparation of the dataset time-consuming. Although the encoding and decoding of 3D models can be processed in the Rhino-Grasshopper environment, there is no direct connection yet between the image editing tools and the modelling software. This lack of immediacy makes it hard to properly explore and iterate within a design process.

Concerning the StyleGan our lack of training data generated results that often appeared spatially arbitrary. While the segmentation of pixel plots increased that dataset size drastically, successful training on entire pixel plots would also require more data. A larger training dataset would also allow us to train a StyleGan without utilising transfer learning, potentially improving the results. The supplementation of the authors' limited expertise relating to the customization of GAN setups would significantly push the outlined encoding implementation within artificial neural networks.

In this article, we have taken existing architecture as a basis for the discovery of new patterns. By using special encoding techniques and machine intelligence, we attempted to discover new ways of form-finding that transfer design knowledge to concept models at the early stages of architectural design. With this research, we introduced a new format for the representation of architectural concepts and aimed to highlight its potential as training data for artificial neural networks and also as new mediums for the creation of architectural concepts.

Acknowledgements. We would like to thank our students at the University of Innsbruck, Institute for Structure and Design, for their help in producing the dataset. The work was funded by the University of Innsbruck and the Austrian Science Fund (FWF) project F77 (SFB "Advanced Computational Design").

References

1. Achlioptas P, Diamanti O, Mitliagkas I, Guibas L (2018) Learning representations and generative models for 3D point clouds. arXiv:1707.02392 [cs]
2. Hellsten J, Karras T (2022) NVlabs/ffhq-dataset. NVIDIA Research Projects. https://github.com/NVlabs/ffhq-dataset. Accessed 16 Mar 2022
3. Hellsten J, Karras T (2022) NVlabs/stylegan2-ada-pytorch. NVIDIA Research Projects. https://github.com/NVlabs/stylegan2-ada-pytorch/blob/6f160b3d22b8b178ebe533a50d4d5e63aedba21d/README.md. Accessed 15 Mar 2022

4. Hirschberg U, Hovestadt L, Fritz O (eds) (2020) Atlas of digital architecture: terminology, concepts, methods, tools, examples, phenomena. Birkhauser, Boston
5. Karras T, Aittala M, Hellsten J, Laine S, Lehtinen J, Aila T (2020) Training generative adversarial networks with limited data. arXiv:2006.06676 [cs, stat]
6. Kench S, Cooper SJ (2021) Generating 3D structures from a 2D slice with GAN-based dimensionality expansion. arXiv:2102.07708 [cs]
7. Marinčić N (2019) Computational models in architecture: towards communication in CAAD. Spectral characterisation and modelling with conjugate symbolic domains. Birkhäuser
8. de Miguel J, Villafañe ME, Piškorec L, Sancho-Caparrini F (2019) Deep form finding using variational autoencoders for deep form-finding of structural typologies. In: Blucher design proceedings. Editora Blucher, Porto, Portugal, pp 71–80
9. Ren Y, Zheng H (2020) The spire of AI—voxel-based 3D neural style transfer. In: Anthropocene, design in the age of humans, vol 2. CAADRIA, Bangkok, Thailand, pp 619–628
10. Vasconcelos LO, Sato A (2020) Texture animation: applying morphing and vertex animation techniques. Wildlife Studios Tech Blog. https://medium.com/tech-at-wildlife-studios/texture-animation-techniques-1daecb316657. Accessed 15 Mar 2022
11. West J, Bergstrom J (2019) Which face is real? Which face is real? https://www.whichfaceisreal.com/methods.html. Accessed 15 Mar 2022
12. Zhang H, Huang Y (2021) Machine learning aided 2D–3D architectural form finding at high resolution, pp 159–168.https://doi.org/10.1007/978-981-33-4400-6_15