3D Generative Design for Non-Experts: Multiview Perceptual Similarity with Agent-Based Reinforcement Learning

Robert Stuart-Smith[1], Patrick Danahy[2]

[1] University of Pennsylvania, Philadelphia, USA
University College London, London, UK
rssmith@design.upenn.edu
[2] University of Pennsylvania, Philadelphia, USA
pdanahy@design.upenn.edu

Abstract. Advances in additive manufacturing allow architectural elements to be fabricated with increasingly complex geometrical designs, however, corresponding 3D design software requires substantial knowledge and skill to operate, limiting adoption by non-experts or people with disabilities. Established non-expert approaches typically constrain geometry, topology, or character to a pre-established configuration, rather than aligning to figural and aesthetic characteristics defined by a user. A methodology is proposed that enables a user to develop multi-manifold designs from sketches or images in several 3d camera projections. An agent-based design approach responds to computer vision analysis (CVA) and Deep Reinforcement Learning (RL) to design outcomes with perceptual similarity to user input images evaluated by Structural Similarity Indexing (SSIM). Several CVA and RL ratios were explored in training models and tested on untrained images to evaluate their effectiveness. Results demonstrate a combination of CVA and RL motion behavior can produce meshes with perceptual similarity to image content.

Keywords: Generative Design, Machine Learning, Agent-Based Systems, Non-Expert Design, Deep Reinforcement Learning

1 Introduction

Recent advances in additive manufacturing (AM) allow architectural elements to be fabricated with increasingly complex geometrical designs and enhanced material and structural efficiencies. 3D modelling software has kept pace, providing approaches to the design of complex, multi-manifold surfaces. While AM supports user-customized design through one-off manufacturing on-demand, design software requires substantial knowledge in 3D modelling or software programming that is challenging for non-experts or people with disabilities to operate.

Researchers at MIT's Architecture Machine Group historically explored ways for non-expert participation in design by incorporating Machine Learning

(ML) in 3D software (Negroponte, 1972). More recently, Google Creative Lab™'s Autodraw™ cognifies an inexperienced artist's scribbles to suggest related content, allowing high-quality drawings to be developed in partnership with software (Motzenbecker & Phillips, 2017). Architectural research has incorporated ML in approaches to design (Bolojan & Vermisso, 2020), addressing performative aspects of user-created designs (Akizuki et al., 2020; Yousif & Bolojan, 2021), and assisting with assembly configurations of predefined design elements (Hosmer et al., 2020). 3D ML approaches have also been developed that leverage 3D Generative Adversarial Networks (3D-GAN) (Wu et al., 2016), StyleTransfer and GANs that map 2D image pixel content to a fixed 3D mesh topology's color and displacement maps, or interpolate several output images into a 3D voxel array (del Campo et al., 2021; Rehm, 2019).

Established design methods, however, do not allow a non-expert user to intuitively specify a 3D intention from multiple sourced, sketched or diagrammed 2D image-based viewpoints to generate a multi-manifold topology 3D model. Existing methods primarily operate on 2D raster images, 3D voxel fields or employ predefined 2D or 3D component parts. Such approaches to architectural design do not attempt to embody an aesthetic character from a user's sketched input by evaluating the perceptual equivalence of 3D design outcomes relative to user-specified 2D image content to inform ML model training.

While extensive research into computer vision (Ham et al., 2019) and ML-based approaches to 3D reconstruction (Soltani et al., 2017) from several sourced images exist within the field of computer science, these focus on image or voxel-based outcomes that are not integrated with other design methods or building performance considerations that would support the production of a polyvalent architectural design outcome. A flexible, non-indexical algorithmic design methodology for multi-manifold topologies is required that can incorporate other design-engineering workflows to safeguard the manufacturing and end-use viability of a non-expert user's design. Agent-based design methods are systemically open to the incorporation of several performance criteria, and have been developed to address AM-specific structural and geometric performance criteria (Stuart-Smith et al., 2020), alongside a narrow set of aesthetic criteria obtained by computer-vision-based visual character analysis methods (Stuart-Smith & Danahy, 2022).

The ability of agent-based systems to weight several often-conflicting design criteria within a computed solution makes the approach well-suited to applications involving machine learning (ML). Agent-based models also enable designs with geometric continuity in three dimensions through vector-based image translation without indexical constraints, yet to date, only agent motion constrained to voxel-space orthogonal coordinates and evaluation methods has been demonstrated for architectural design ML applications (Akizuki et al., 2020). Such methods overly constrain a design's visual character and do not incorporate methods that support agent adaptation to image data which would facilitate such ML methods obtaining outcomes related to a user's specific inputs.

A novel 3D algorithmic design methodology for multi-manifold geometries is presented that employs reinforcement learning (RL) to train agent motion behavior to translate user-specified image inputs from multiple viewpoints into a 3D design solution generated by agent-based motion (Figure 1). The approach attempts to partially embody visual design character from the user's input images by adapting to image data obtained by computer vision analysis methods, and partly through an ML training approach that seeks to improve the perceptual similarity of design outcomes relative to user input images by Structural Similarity Indexing (SSIM) (OpenCV, 2021). The research does not address AM or specific architectural design or performance parameters as the approach is able to be integrated with prior research addressing these concerns (Stuart-Smith et al., 2020; Stuart-Smith & Danahy, 2022).

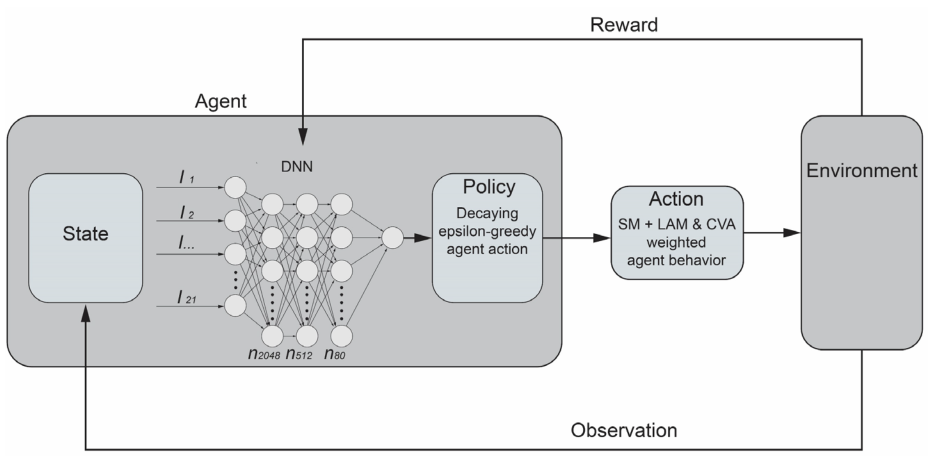
Reward
Agent
DNN
State
I_1
I_2
$I_{...}$
I_{21}
n_{2048} n_{512} n_{80}
Policy
Decaying epsilon-greedy agent action
Action
SM + LAM & CVA weighted agent behavior
Environment
Observation

Figure 1. Generative design methodology incorporating two agent-based motion methods, reinforcement learning and computer vision analysis. Source: Stuart-Smith & Danahy, 2022.

2 Methodology

A custom design methodology was developed that involved the creation of a user-orientated 3D setup environment that incorporates user-specified images, and two agent-based motion methods, one capable of adapting to image data using computer vision analysis, and the second, combined with a reinforcement learning (RL) method to train agent behavior to produce results that correspond to the input images. The weighting of computer vision-based adaptation relative to RL-trained behaviour was explored through a comparative analysis of several training runs, with the most successful model's results evaluated. The methodology involves the following components:

2.1 3D Environment Setup

A 3D environment was defined in Rhino3D (Figure 2a) with a specified 3D closed-volume brep geometry defining its extent. For the purposes of this initial study, a rectangular prism of 5 units length, 5 units width and 10 units height, is used as the environment although the environment could also be of an irregularly shaped geometry. Three perspective views are defined, each in separate viewports (the setup can be extended to any number of perspectival or parallel views). An image plane is defined in Grasshopper at the near and far frustum planes of each view with a proportion and location that corresponds to the environment bounding box. For the purposes of this study of column-like geometries, images are 3330 x 1574 pixels in resolution and can be positioned to be greater or smaller in coverage relative to the environment's bounding dimensions. In Grasshopper, a user is able to select these three input images from their computer. The input images could be photographs of hand-sketches, tablet sketched or drawn in Illustrator or CAD, or sourced reference images.

2.2 Agent-Based Design Method

A custom-developed agent-based simulation builds on prior research into performance-based multi-manifold architectural design (Stuart-Smith et al. 2020), tasking an autonomous particle to make event-based decisions to descend through the 3D environment to define a motion trajectory over a simulated sequence of time-frames which is translated into a 3D Nurbs curve and isosurfaced to create a volumetric mesh. Agent motion is calculated in a series of time-steps where an agent's velocity is added to its current position to define its next position. Two methods were developed to determine an agent's velocity vector over simulated time.

Method 1, Sequential Motion (SM) method: moves an agent incrementally by selecting one of 9 possible vector moves each time frame in simulated time. These moves are derived from a set of four vectors equally distributed around the velocity axis along a conical surface, with two different conical surface angles allowing for sharp and shallow turning at 15 and 40 degrees away from the velocity vector, in addition to a ninth vector pointing straight ahead. These vectors are calculated relative to the agent's current velocity vector each timeframe (Figure 2b, 2c). Agents determine which of the nine moves to implement as part of an RL action (see 2.4).

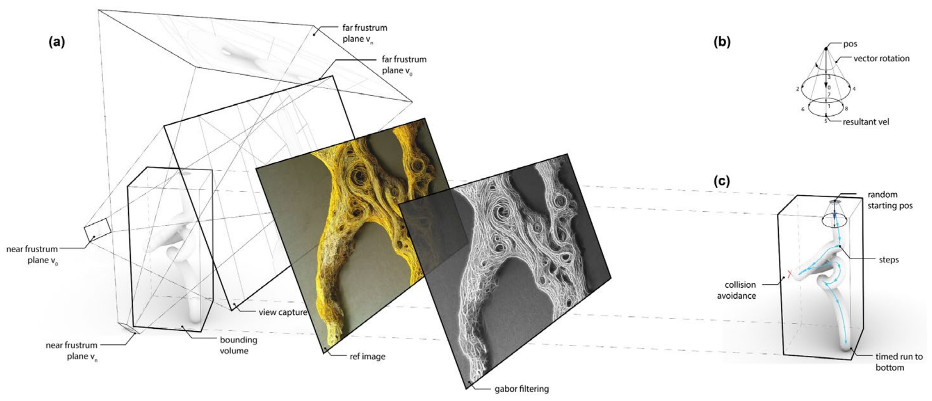

Figure 2. User-sketched 3D geometry generation using Deep Reinforcement Learning (RL), (a) 3D environment, user-input sketches/images and correlated camera view projections, (b) Agent SM: 9 x vector motion options for each timeframe relative to current velocity, (c) Agent SM sequence during RL training. Source: Stuart-Smith & Danahy, 2022.

Method 2, Locally Adaptive Motion (LAM) method: sums several vectors of influence including seeking, avoiding and aligning to several locally perceived environmental data sources. This method is used to adapt agent motion to user-specified image content (see 2.3). Inspired by Craig Reynolds steering behaviors, each vector influence is calculated as a steering vector relative to the current velocity vector, as a turning force (Reynolds, 1999). Each steering behavior is calculated as the average of all influences within a specified radius and field of view from the agent. Behaviors are weighted relative to one another to provide a combined overall agent motion behavior (Figure 3b).

2.3 Computer Vision Adaptation (CVA)

Computer vision methods are used to analyze input images and extract features such as the image silhouette and directional grain, and to remap these within a 3D field, which agents can locally perceive and respond to. Each of the three input images is processed using the computer vision methods ‘Canny Contouring’ and ‘Probabilistic Hough Line Transform’ (OpenCV, 2021). The Canny Contouring edge detection method is utilized to first identify continuous edges in the geometry due to its ability to produce an image that depicts continuous edges in white on a black background through a multistep process that leverages spatial-frequency filtering and hysteresis thresholding (Davies, 2017, p. 136). Using the Canny Edge image output as an input, OpenCV's Probabilistic Hough Lines Transform method is utilized to create a series of line segments that approximate continuous edges within each reference image (OpenCV, 2021).

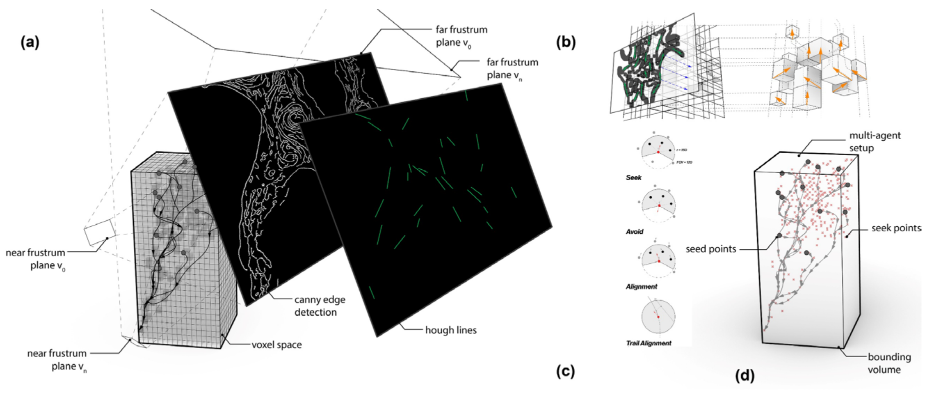

Figure 3. Agent-based Computer Vision Adaptation (CVA) motion. (a) edges from Canny Contour image data and lines from Hough Line Transform are projected into a voxel environment, (b) Canny data is summed as an integer value per voxel, with Hough Lines translated to a single summed vector at the center of each voxel, (c) LAM agent steering behaviors, (d) agent LAM response to CVA data. Source: Stuart-Smith & Danahy, 2022.

Each analysis yields a 2D output image (Figure 3). Hough lines create a series of short lines along features in the image that can be calculated as vectors. Canny contouring illustrates key features in white pixels whose pixel grid indexes can be identified. White pixels are assigned a value of ‘1’ with all other colors assigned ‘0’. The Canny contouring analysis is remapped to a 3D voxel space within the 3D environment comprised of voxels at a resolution equivalent to 100 x 100 pixels. The custom voxel class in Python operates similar to a Bin Lattice, and contains variables to store data from each of the methods. The voxel field used in this initial research comprised of 80x80x180 voxels. Each voxel sums values obtained from corresponding image pixels projected through the 3D voxel space. 2D Data from each image is remapped as projections through the voxel field using the near and far frustum planes to create a virtual trapezoidal volume. Voxels inside this volume are assigned a value of ‘1’. Where there is corresponding data from multiple image projections, this information is summed. For example, one voxel relates to 4 pixels from one image and 4 from a second image. The variable for Canny Contours, therefore, has a value ranging from 0 to 8. If a third view also overlapped in this area the value range would be 0 to 12. The Hough Line Transform outcomes are projected by the same method, but stored as vectors in the center of each voxel. Lines from multiple viewports are summed as one vector result per voxel (Figure 3).

This data-rich 3D environment locally influences LAM behavior of agents as they descend through the environment ensuring they adapt to data content from the user-provided images, seeking high values from Canny Contouring and Gabor Filter analysis, and aligning to vectors obtained from Hough Line analysis. The complexity of LAM behaviour however is immense, and the agent-based rule set creates a level of abstraction that might at times prevent the agents from fully matching outcomes to image content. As such, an RL method was developed to fine-tune agent behavior in relation to image content.

2.4 Reinforcement Learning (RL)

To provide better design adaptation to user-specified images, a method was developed to support SM agent behavior derived from deep reinforcement learning (RL). An RL method was developed within Rhino3D by connecting to the Windows Powershell via a socket connection. SM agent behavior is adjusted using Keras™'s RL API for TensorFlow 2 (Chollet & others, 2015). A Functional Model comprised of several ReLU dense layers trains non-image-based agent vector steering over 2000 iterations. The RL training model utilises a reward function that is evaluated at every motion step within an SM agent simulation. An agent that attempts to leave the 3D environment is penalized with a negative value of -10, while agents descending all the way to the bottom are rewarded with a value of 20. To encourage multi-manifold topological design solutions the genus of each agent-generated mesh is determined and also provided as a positive integer reward value.

To evaluate the similarity of the results to the image inputs, screen capture is performed from the same viewport as each corresponding image input plane and compared to the input image using OpenCV's Structural Similarity Indexing (SSIM) method which samples different parts of each image, comparing their luminance, contrast and structure to provide an averaged metric of their perceptual similarity (Wang et al., 2004). This evaluation is performed every second time frame. A high SSIM equates to a high reward.

To reward proximity to key features in reference images, Gabor Filtering is performed to highlight local features and edges by evaluating the frequency and orientation of image content (Anuj shah, 2018). Agent proximity to the reference images' Gabor Filter features is rewarded by performing a Gabor Filtering of each reference image and averaging their results. In greyscale Gabor images, features are highlighted by white pixels. The reward function samples the closest pixel projection to the agent and its eight adjacent neighboring pixels, assigning each pixel a value between zero and one (0 = black, 1 = white), summing these values. This is repeated using each input reference image and averaged to provide an overall reward value for image feature content. Following training over 2000 iterations, the trained model can be provided with different input images of similar content. The combined, weighted RL reward function can be described as:

$$f(x) = \frac{(w_1 * \mathrm{g}) \; + (W_2 * \mathrm{s})}{2} + \; n \; + t - B + A$$

Where, g = $\left\lceil \frac{v-e+f-2}{-2} \right\rceil$

with the following variables:
f = number of mesh faces
e = number of mesh edges
v = number of mesh vertices
g = topological genus
n = neighborhood pixel average from Canny contour analysis remapped to 0-1 summed across mesh vertexes in range (V0+V1+Vn.../n)
t = time alive count (reward for surviving for a long time)
s = average SSIM between each view and its user-input reference image
A = reward for arriving at the bottom of the bounding box environment
B = punishment for hitting boundary, ends episode

W_1, W_2, B and A are all constants that can be waited by a user undertaking RL training.

2.5 Behavioral Weighting Evaluation

CVA (see 2.3) ensured LAM agent behavior directly responded to locally perceived input image data yet had no ability to evaluate overall image perception similarity. Conversely, SM agent behavior used in the RL method (see 2.4) trained an agent model to achieve high SSIM values yet had no direct relation to easily obtained local image content. Although combining these approaches was likely to produce more successful (3D designs with more similarity to the input images), the exact ratio these methods should be weighted relative to one another had to be evaluated. To determine this, a series of separate RL training runs were performed which weighted the ratio of CVA relative to the RL method at 25% increments. 2000 iterations of training were performed on separate models with ratios of; 0%/100%, 25%/75%, 50%/50%, 75%/25%, 100%/0%. In each model, incremental motion is performed by weighting and summing a vector calculated by LAM based CVA and a vector calculated using the SM RL method. These are summed and averaged to provide a single motion vector to govern incremental agent motion with a reward evaluated and fed back into the RL training session. Each model was trained on the same input images. The model with the most successful loss and reward metrics was selected for evaluation.

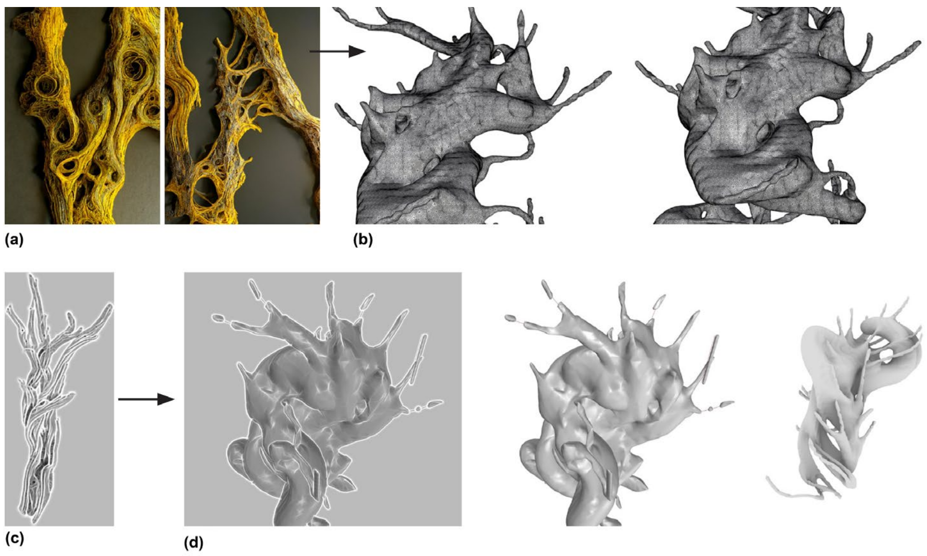

Figure 4. Outcomes from the selected trained model with ratio of LAM CVA agent behavior to SM agent motion of 1:3. Two different sets of image inputs are shown (a & c), and corresponding mesh results (b & d). Source: Stuart-Smith & Danahy, 2022.

3 Results

The training model with a LAM CVA to SM agent behavior ratio of 1:3 exhibited the best loss and reward metrics. 3D results from this model exhibit an abstract approximation of input images, demonstrating multi-manifold design outcomes that embodied some degree of visual characteristics of user-specified input images (Figure 4). Although outcomes only partially related to their respective input images, data from the training model (Figure 5) suggests that the correlation is sufficient to demonstrate the merits of the method. Loss trends downwards indicating model training is optimizing correctly. Reward values also trend upwards, although more training iterations would yield a clearer logarithmic curve for the loss function and a continuous trend upwards in the reward function. This data demonstrates that design outcomes were able to improve SSIM perceptual similarity metrics throughout training.

4 Discussion & Conclusion

The presented method aims to allow a designer or non-expert user to sketch a 3D design from multiple camera view 2D projection planes to generate a 3D multi-manifold mesh design that takes into consideration the perceptual similarity of the design outcome relative to the input images, while

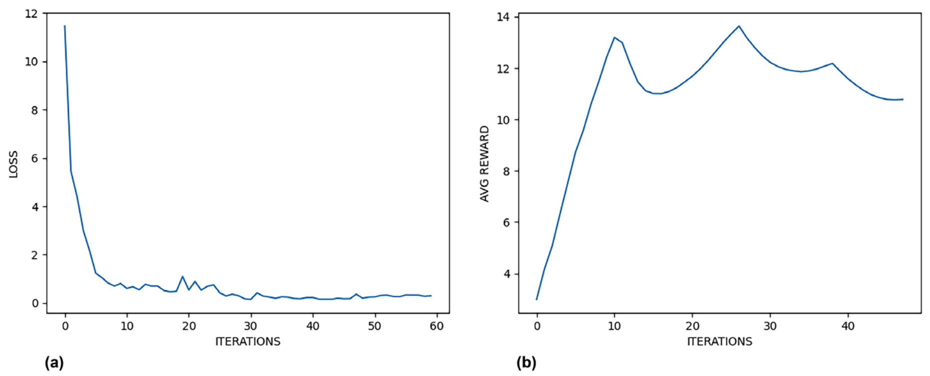

Figure 5. Data from most successful RL model: (a) loss function, (b) reward values.

also aligning to features in the input images. The agent-based approach leveraged CVA to adapt to input image content and RL to train agent motion to improve perceptual similarity and thus benefited from both local behavioral adaptation in addition to goal-orientated deep RL. While the agent RL training was sequential, future work will explore training being distributed and operating within a multi-agent system. The methodology was developed to be compatible with the authors' prior research, allowing future research to incorporate building performance criteria such as structural, geometric, and visual character optimization methods (blind), ensuring the approach can be developed for architectural design applications in future research.

The research demonstrates an intuitive approach to complex geometric design that offers great potential for applications such as user-customized design or rapid response designs for manufacturing on-demand. Questions remain around user engagement with model training. Ideally, a user is able to train a model that aligns with their specific input image types. This would require more explorations into a user-friendly GUI and hardware. Although this proof of concept was undertaken in Rhino3D, the approach could be ported to operate on physical devices such as smartphones, tablets and augmented reality (AR)/virtual reality (VR) headsets to enable greater accessibility and interface enhancements.

It is hoped the research spurs more activity in non-expert design customization, providing end-users greater accessibility to design both alone and in dialogue with professional designers, illustrating a means by which they can co-design the built environment via more user-friendly, adaptive AI-based design software. Recent and emerging metaverse and AR software and hardware applications suggests the research offers a glimpse into greater user-engagement within the design world of tomorrow.

References

Akizuki, Y., Bernhard, M., Kakooee, R., Kladeftira, M., & Dillenburger, B. (2020). Generative modelling with design constraints: Reinforcement learning for object generation. RE: Anthropocene, Design in the Age of Humans - Proceedings of the 25th International Conference on Computer-Aided Architectural Design Research in Asia, CAADRIA 2020, 1.

Anuj shah. (2018, June 17). Through The Eyes of Gabor Filter. Medium. https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97

Bolojan, D., & Vermisso, E. (2020). Deep Learning as heuristic approach for architectural concept generation Creativity and Artificial Intelligence. Proc. ICCC 2019.

Chollet, F., & others. (2015). Keras. https://keras.io

Davies, E. R. (2017). Computer Vision: Principles, Algorithms, Applications, Learning. Elsevier Science.

del Campo, M., Carlson, A., & Manninger, S. (2021). Towards Hallucinating Machines - Designing with Computational Vision. International Journal of Architectural Computing, 19(1). https://doi.org/10.1177/1478077120963366

Ham, H., Wesley, J., & Hendra. (2019). Computer vision based 3D reconstruction : A review. International Journal of Electrical and Computer Engineering, 9(4). https://doi.org/10.11591/ijece.v9i4.pp2394-2402

Motzenbecker, D., & Phillips, K. (2017, May 6). AutoDraw by Google Creative Lab - Experiments with Google. Google Creative Lab. https://experiments.withgoogle.com/autodraw

Negroponte, N. (1972). The Architecture Machine: Toward a More Human Environment. M.I.T. Press.

OpenCV. (2021). About - OpenCV. Open CV Team. https://opencv.org/about/

Rehm, M. C. (2019). Complicit: The creation of and collaboration with intelligent machines. Architectural Design, 89(2). https://doi.org/10.1002/ad.2417

Reynolds, C. W. (1999). Steering behaviors for autonomous characters. Game Developers Conference, 763–782.https://doi.org/10.1016/S0140-6736(07)61755-3

Soltani, A. A., Huang, H., Wu, J., Kulkarni, T. D., & Tenenbaum, J. B. (2017). Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January. https://doi.org/10.1109/CVPR.2017.269

Stuart-Smith, R., & Danahy, P. (2022). Visual Character Analysis Within Algorithmic Design, Quantifying Aesthetics Relative To Structural And Geometric Design Criteria. In J. van Ameijde, N. Gardner, K. H. Hyun, D. Luo, & U. Sheth (Eds.), CAADRIA 2022, POST-CARBON - Proceedings of the 27th CAADRIA Conference - vol. 1, Sydney, 9-15 April 2022 (pp. 131–140).

Stuart-Smith, R., Danahy, P., & Rotta, N. R. la. (2020). Topological and material formation. Proceedings of the 40th Annual Conference of the Association for Computer Aided Design in Architecture: Distributed Proximities, ACADIA 2020, 1.

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4). https://doi.org/10.1109/TIP.2003.819861

Wu, J., Zhang, C., Xue, T., Freeman, W. T., & Tenenbaum, J. B. (2016). Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. Advances in Neural Information Processing Systems.

Yousif, S., & Bolojan, D. (2021). Deep-performance: Incorporating deep learning for automating building performance simulation in generative systems. Projections - Proceedings of the 26th International Conference of the Association for Computer-Aided Architectural Design Research in Asia, CAADRIA 2021, 1.