
Speculative Hybrids

Investigating the generation of conceptual architectural forms
through the use of Machine Learning.

Master Thesis Report
Panagiota Pouliou

Aalborg University
Architecture, Design, and Media Technology

Copyright © Aalborg University 2022

The software used for typesetting the document, and creating figures was overleaf, google forms, Photoshop, and Illustrator.



AALBORG UNIVERSITY

STUDENT REPORT

Electronics and IT

Aalborg University
<http://www.aau.dk>

Title:

Speculative Hybrids: Investigating the generation of conceptual architectural forms through the use of Machine Learning.

Theme:

Point cloud generation of architectural hybrids.

Project Period:

Spring Semester 2022

Project Group:

Panagiota Pouliou

Participant(s):

Panagiota Pouliou

Supervisor(s):

George Palamas

Anca-Simona Hovarth

Copies: 1**Page Numbers:** 80**Date of Completion:**

June 3, 2022

Abstract:

Design Thinking aims at solving problems that have no definite formulation, no explicit basis for terminating the problem-solving activity, and where differing outcomes may occur from differing problem formulations. Most of the design processes have already come under the spell of the digital age. In order to explore the realm and capabilities of contemporary architectural design, an analysis of its enhancement through digital tools and the suggestion of a method for problem-space exploration will be demonstrated. In *Speculative Hybrids* we employ machine learning algorithms in order to generate conceptual architectural forms with respect to site-specific regulations. More specifically, Generative Adversarial Networks are implemented to generate annotated point clouds. The iterative power of such a tool facilitates the design process by suggesting a plausible array of satisfactory outcomes in space-problem structuring, and can assist in the formulation of more efficient problem-solving processes.

The content of this report is freely available, but publication (with reference) may only be pursued due to

agreement with the author.

Contents

Preface	vii
1 Introduction	1
1.1 Reconsidering Otherness	2
2 Background and literature review	5
2.1 Digital Age in Architecture	5
2.1.1 Machine Learning applications in Architecture	6
2.2 Growing Neural Gas Algorithm	7
2.3 Generative Adversarial Networks	7
2.3.1 3D GANs	9
2.3.2 CPCGAN	10
2.4 Point clouds	11
2.4.1 Point cloud datasets	11
2.5 Dataset creation and augmentation	12
3 Design and Implementation	13
3.1 Dataset Creation	13
3.1.1 Module Geometries	14
3.1.2 Mesh Conversion	14
3.1.3 Point cloud Generation	14
3.1.4 Data Pre-processing	18
3.2 Building a virtual environment for training CPCGAN	19
3.2.1 Implementation of CLAAUDIA	20
3.3 Training CPCGAN and generating Samples	20
3.3.1 Pre-process the data	20
3.3.2 Training CPCGAN	21
3.3.3 Generate a sample	21
4 Evaluation	23
4.1 Participants	23
4.2 Creativity Support Index	23

4.3	Measurements	24
5	Results	27
5.1	Model Generations	27
5.2	Questionnaire Results	28
5.2.1	CSI Metrics	29
5.2.2	Open-ended Questions	32
6	Discussion	35
6.1	Design	36
6.2	Evaluation Process	37
6.3	User feedback	37
6.3.1	Quantitative Results	37
6.3.2	Qualitative Results	38
7	Conclusion	41
7.0.1	Future perspectives	42
	Bibliography	45
A	Scripts	53
A.1	csvtoptsseg.py	53
A.2	Dockerfile	55
A.3	devcontainer.json	56
A.4	samplepoints.py	56
A.5	samplestructurepoints.py	58
A.6	neuralgas.py	64
A.7	ptstocsv.py	68
A.8	gen_from_spc_and_z.py	68
B	CSI Metrics	73
B.1	Creative Support Index Questionnaire	73
B.2	Comparison Factors Questionnaire	76

I am overwhelmed in all humbleness and gratefulness to acknowledge my deep appreciation to all those who have helped me to fulfill *Speculative Hybrids*. Firstly, I would like to thank both my supervisors, George Palamas and Anca-Simona Horvarth. They both guided me in such inspirational and supportive manners that helped my project rise well above the level of simplicity and into something concrete. Additionally, I would really like to express my gratitude to all my Copenhagen family, specially Andrea-Roberto Insinga, Angeliki Alexiou - my flatmates-, and Zoe Tzika, who provided in this uphill process plenty of emotional support, and infinite time of peer review. Finally, I would like to thank both my parents and family without whom I would not have succeeded in any of this.

Aalborg University, June 3, 2022



Panagiota Pouliou
<ppouli20@student.aau.dk>

Chapter 1

Introduction

The process of design serves as a base for the formulation of theoretical analysis and a deeper understanding of the design itself. According to Oxman [60], there are four certain major classes describing the process of design: problem formulation, synthesis/generation, representation, and evaluation. By the mid-80s, Schön [71] shifted the focus of design toward the designer and initiated the study of Design Thinking. More specifically, Schön emphasizes the interaction of the designer with a given problem and creates a topological mind map of the process. Therefore, he suggests that design thinking is constituted of the following: a process of *reception* (perception), *reflection* (interpretation), and *reaction* (transformation).

Rowe et al. [68] summarize the understanding of design thinking in architecture and suggest its categorization between the properties of *incompleteness* - what Schön describes as design thinking (reception, reflection, and reaction) and *precision* - the evaluation and functionality of the results of design thinking. He, therefore, redefines design thinking as the process that lingers between the two, aiming to balance for continued problem-space structuring.

Already by the 1990s, architectural projects, conferences, competitions, and exhibitions start to formulate a theoretical discourse regarding digital design [60]. By the 2000s, most of the design processes had come under the spell of the digital age. This digitalization of processes enhances architectural design thinking by serving as a mediator between incompleteness and precision [68]. The digital age has made considerable contributions to design thinking in at least four areas:

- The *exploration of conceptual and technical options* regarding new representational methods (e.g. renders) and tools that provide higher precision (e.g. CAD drawings).
- The *iterative power of generate-and-test procedures* results in the generation of a plausible array of satisfactory outcomes. Therefore, it may assist in better problem-space structuring.

- ***Broader access to information.*** The evaluation and assessment of outcomes may be executed with higher degrees of accuracy, scope, and technical sophistication.
- ***Better research capabilities and simulation techniques.***

The prominent link between all the processes of design is the role of the designer. Even though this role is also evolving along with contemporary technologies, it still retains a vital position in the design process. The contemporary designer interacts with generative and performative mechanisms, while data has become a "new material" for them. The role of the designer adapts to the contemporary world, as practitioners, nowadays, adapt to the new technological tools, and sometimes even create their own apparatuses and digital equipment.

What the digital age has succeeded in is the improvement of capacities for acquisition, manipulation, and assessment of information. That said, further investigation of contemporary tools and methods should be applied in order to advance the workflow of architectural design. In *Speculative Hybrids*, we aim to explore the potentials of Machine Learning for architectural design, and therefore investigate how its use enhances design processes in the early design phases. More specifically, we suggest the implementation of a Generative Adversarial Network (GAN), trained with a dataset of building geometries in the form of annotated point clouds.

Speculative Hybrids are newly generated building geometries that are characterized by the features of the dataset that was used to train the model. Our hypothesis lays upon those hybrids and their ability to inspire architectural practitioners in the ideation phase through exploration, within the context of site-specific rules. Their purpose is to bridge the gap between incompleteness and precision. We believe that the iterative power of such a tool and the design solutions it can provide can inspire practitioners of the field and assist them in structuring fundamental problem-space.

1.1 Reconsidering Otherness

Speculative Hybrids is building on the research project *Reconsidering Otherness*, led by Anca-Simona Hovarth at the Research Laboratory for Art and Technology at Aalborg University. Reconsidering Otherness investigates the use of Machine Learning in designing conceptual architecture. The project proposes a method to assist practitioners in the field during the ideation phase. The project consists of three steps:

- The collection of two datasets: one dataset of texts describing conceptual architecture proposals, and one dataset of the representational images of those proposals.

- Training Machine Learning algorithms to generate new *architectures*: 3 Machine Learning models generating text from text, images from images (selected results can be seen in fig. 1.1), and images from text.
- The submission of a proposal to the eVolo architectural competition with the generated results.

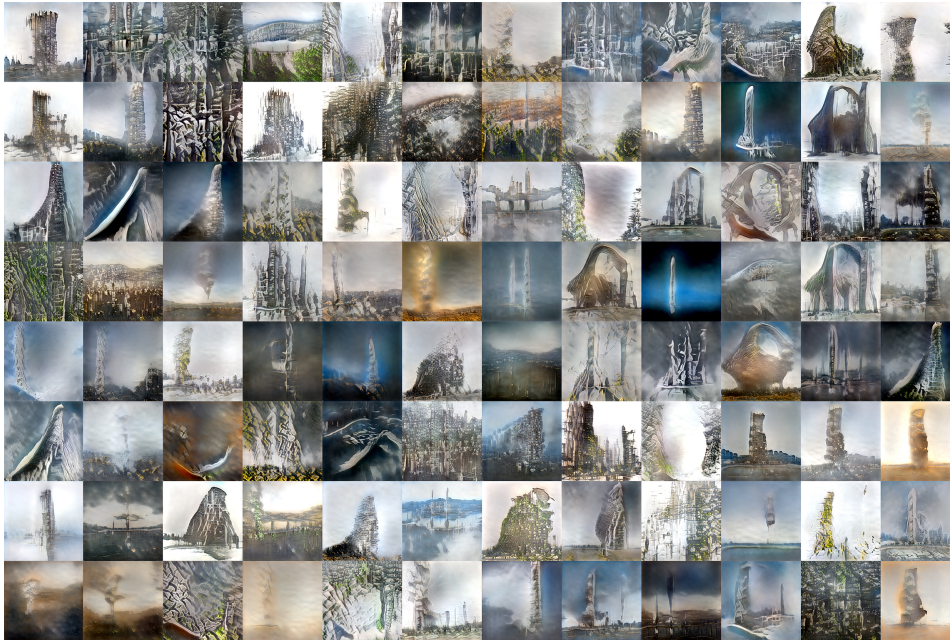


Figure 1.1: Images generated from a model trained with images that describe skyscrapers. The image is a part of Reconsidering Otherness, which aims to explore new 2D conceptual architecture mediums.

This initial work proved to be valuable in understanding and establishing design processes that involve Machine Learning in the design of conceptual architecture. However, while images and texts can be a source of inspiration during the ideation phase, the actuality of the design process must necessarily involve 3-dimensional (3D) shapes. Speculative Hybrids will shift from text and 2-dimensional (2D) images to 3D point clouds. The Speculative Hybrids project aims to:

- Establish a theoretical background regarding the use of Machine Learning in the formation of conceptual architecture.
- Investigate existing datasets of annotated point clouds resulting from 3D scanning technologies that have been processed and classified through deep learning algorithms.

- Create a dataset consisting of annotated point clouds that describe simple building geometries.
- Train a state-of-the-art 3D GAN model to generate new architecture volumes based on the created dataset.
- Explore if and how such a method can shape the contemporary design thinking.

Exploring text, 2D, and 3D Machine Learning algorithms for the creation of conceptual architecture, Speculative Hybrids, and Reconsidering Otherness together will create a multi-faceted perspective on design processes and the possible benefits and pitfalls of designing with Machine Learning.

Chapter 2

Background and literature review

The notion of digital design, Generative Adversarial Networks, and point clouds in the context of architectural design are explained and evaluated.

2.1 Digital Age in Architecture

There is no doubt that architecture has been significantly influenced by digital media and information technology. Digital technology, which can be described as the combination of the two, is assisting the field of architecture in areas such as design, fabrication, construction, maintenance, as well as in the way architecture is taught. However, contemporary technology has revolutionized not merely the way humans act upon designing but also the way they think [4]. Hovarth [32] analysed texts written between 2005 and 2020 that form a part of critical discourse in computational architecture in order to gain a better understanding of the evolution brought upon architecture due to the digitalization processes. There is a radical change in the perception of design because of the use of digital media, which are not only used as representative tools but also for design development, and manufacturing.

There is an extended number of state-of-the-art projects that describe the multi-dimensional approaches within which digital technology has influenced architecture. For instance, the application of digital technologies to architectural design has been the subject of research in a number of areas including the use of computer-aided design (CAD) in the design process [29], the use of integrated systems in architectural practice [22], the role of virtual environments in design [51, 86], the impact of digital technologies on the built environment [44], and the way architecture is taught and learned. [4] All these tools have provided the ability to generate construction information directly from design information, and evaluate the built environment by controlling and improving building components or systems. In conclusion, researchers in the field investigate how digital technology has changed

the architectural workflows, the thinking processes of the practitioners, and the communication of the ideation phase [48, 87].

2.1.1 Machine Learning applications in Architecture

The increasing implementation of Machine Learning algorithms and models is severely affecting the development of architecture. Tamke et. al [76] suggest that new architectural design practices might be based on Machine Learning approaches to better leverage data-rich environments and workflows. In their research, they propose five emergent practices for Machine Learning implementation:

- *Analysis of design space* - emergent parameterization [10, 91].
- *Designing* - short-circuiting simulation [3, 40].
- *Analysis* - defining descriptors for learning and classification [33, 50].
- *Operation and life data* - adapting ongoing behavior [35, 64].
- *Making* - adapting fabrication [9].

Caetano et al. [13] proposed a taxonomy for parametric, generative, and algorithmic design, all key terms of Computational Design in Architecture. Regarding the first emerging practice, a great interest has been gathered around generative design and its implementation in the design processes. More specifically, As et al. [7] are implementing Deep Neural Networks in order to generate conceptual designs. Additionally, Huang and Zheng proposed, in 2018 [33] a twofold project in which they recognized architectural drawings and then generated new ones. Algeciras-Rodriguez [6] utilized Self-organizing Maps [54] in order to produce hybrid forms that acquire characteristics from several input references. Dimensionality reduction tools are used for design data visualizations [50] and generative design exploration [31]. Finally, Campo et al. [21] question whether machines can dream. In order to come to an answer, they suggest a two dimensional approach to designing with the aid of Machine Learning, which serves as a proof of the concept. Steinfeld [75] summarizes the impact of Machine Learning processes in design within three categories: Machine Learning as actor¹, as material², and as provocateur³.

¹Machine Learning models that co-designs along with the designer.

²Generative models that provide new form of design “material”, usually curated by a designer.

³Machine Learning models that form new imagery.

2.2 Growing Neural Gas Algorithm

Feature space optimization and clustering methods are important factors regarding Machine Learning algorithms. Several algorithms attempt to succeed in the aforementioned techniques such as Self Organizing Maps [54], k-means [39], Principal Component Analysis [1], T-SNE [46], and Growing Neural Gas (GNG) [24]. The last one is an unsupervised topology learning algorithm that constructs a data space by interconnected units that focus on the most dense areas of that space [24, 82].

Growing Neural Gas is able to learn the important topological relations in a given set of input vectors using Hebbian Learning Rule [56]. The Hebbian Learning Rule is a learning rule that specifies how much the weight of the connection between two units should be increased or decreased in proportion to the product of their activation. The GNG dynamically adds or removes nodes and approximates the input space more accurately than a network with a predefined structure, such as the Kohonen self-organizing feature map [61].

More specifically, GNG algorithm initiates its process by creating two randomly placed neurons connected with each other by an edge. Then, a random data point is chosen from the given data distribution. The algorithm finds the neuron closer to that point, also known as the best-performing unit (BPU), and moves it closer to it, along with all the other neurons directly connected to it. The age of the edge between BPU and the second closest neuron to the selected point (SBPU) is set to zero, if it does not exist yet, it is at this point that it gets created. If an edge is older than a predefined maximum age, then that edge is deleted. New neurons spawn every once in a while in the in-between distance of the worst-performing neuron⁴ so far and its worst-performing neighbour, and the edge between them gets deleted. The process is usually terminated by a boundary condition, such as when the number of spawned neurons or the maximum number of iterations is reached. Fig. 2.1 shows the implementation of the GNG algorithm on an image. Additionally, the algorithm can be utilized for spatial data. Orts-Escalano et al. [22] are proposing a method for 3D colour object reconstruction based on Growing Neural Gas.

2.3 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were introduced in mid-2014 by Ian Goodfellow [28]. Generative modeling [27] is an unsupervised Machine Learning task that involves distinguishing and learning complex data distributions of input data, i.e. the training dataset, [59] in order to generate new samples that

⁴Worst-performing Units (WPU) are the neurons with the highest cumulative error (sum of distance from each data point over each iteration)



Figure 2.1: An example of a Growing Neural Gas algorithm performing. The parameters for GNG are: $\text{max_Nodes} = 2400$, $\text{n_iter_before_neuron_added} = 100$, $\text{after_split_error_decay_rate} = 0.5$, $\text{error_decay_rate} = 0.995$, $\text{min_distance_for_update} = 0.01$, $\text{step} = 0.2$, $\text{n_start_nodes} = 2$, $\text{max_edge_age} = 50$, and $\text{neighbour_step} = 0.005$

could have been part of the initial training dataset, meaning they have similar data distributions.

Generative Adversarial Networks consist of two sub-models: the *generator* model, which is trained to generate new samples, and the *discriminator* model, which consequently aims to classify these samples as either fake or real [28]. Fig. 2.2 demonstrates the basic architecture of a Generative Adversarial Network. Typically, the generative network learns to map data points from a latent space to a data distribution of a given training set, while the discriminative network distinguishes whether a sample is from the generative model distribution or the training dataset distribution. The two sub-models are trained together in a zero-sum game until the discriminator model cannot distinguish whether the sample that is evaluating is generated or pooled from the initial training set [18].

GANs have been successful in many applications including image synthesis, semantic image editing, style transfer, image super-resolution, and classification [18]. The implementation of Generative models has proven fruitful using a variety of data types, including text [8], images [36], and videos [5]. However, the increasing attention toward 3D (spatial) data has led GANs to succeed in several other research areas.

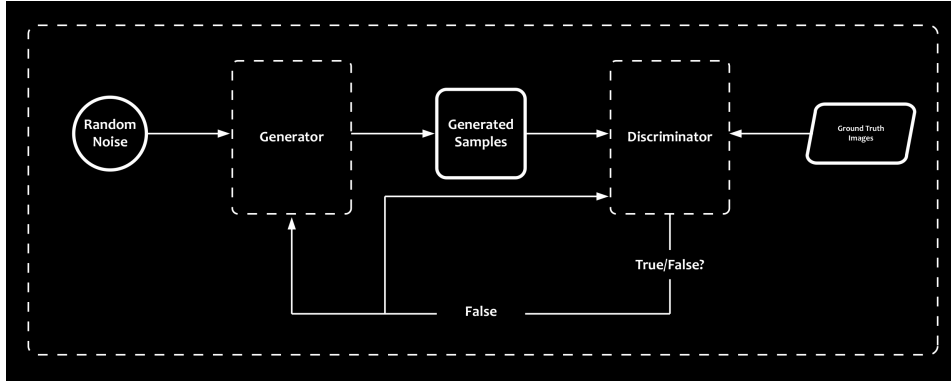


Figure 2.2: The overview of a Generative Adversarial Network’s structure.

2.3.1 3D GANs

The implementation of 3D GANs focuses on a variety of topics mostly using point clouds as input or output data. Applications of generative models have shown results in image-to-point-cloud transformation [42], text-to-voxel [70], point-cloud-to-point-cloud completion [94], and point cloud upsampling [43]. These methods have achieved impressive results in enhancing Computer Vision applications.

For the purposes of Speculative Hybrids, we investigate the capabilities of Generative Adversarial Networks in generating 3D point clouds from random latent codes. Even though Generative Adversarial Networks computing with 3D point clouds, are still under-explored [2], there are already several algorithms attempting to succeed in the task.

Achlioptas et al. [2] were the first ones in the field to suggest and implement a method for point cloud generation. Raw point cloud GAN (r-GAN) was the initial model for generating point clouds from raw data points, while latent-space GAN (l-GAN) was a simplified version of r-GAN incorporating pre-trained autoencoders for pre-processing the data [2]. Another GAN proposed by Valsesia et al. [80] is using a dynamic graph convolutional network instead of a typical generator. Tree-GAN was proposed in 2019 by [73] which shapes a hierarchical structure in feature space by using tree-structured graph convolutions. Finally, the most recent attempt at generating point clouds is introduced by Yang et al. in 2021 [90].

Controllable Point Cloud Generative Adversarial Network (CPCGAN) not only succeeds in higher performance rates regarding results and computational effectiveness than previous algorithms, but it also allows the manipulation of the generated output towards specific directions of preference. Additionally, CPCGAN provides semantic segmentation of the generated point clouds. In fig 2.3 a quantitative comparison can be seen between the performance metrics of the models as proposed by Achlioptas et al. [2], and performed by Yang et al. [90].

Class	Model	FPD↓	JSD↓	MMD-CD↓	MMD-EMD↓	COV-CD↑	COV-EMD↑
Chair	r-GAN*	1.860	0.238	0.0029	0.136	33	13
	Valsesia et al.*	-	0.100	0.0029	0.097	30	26
	tree-GAN	1.114	0.0725	0.00191	0.0900	60.21	33.92
	CPCGAN(ours)	0.877	0.0433	0.00186	0.0753	62.33	50.41
Airplane	r-GAN*	1.016	0.182	0.0009	0.094	31	9
	Valsesia et al.*	-	0.083	0.0008	0.071	31	14
	tree-GAN	0.549	0.0854	0.00039	0.0584	58.40	23.13
	CPCGAN(ours)	0.522	0.0296	0.00038	0.0417	59.63	45.36

Figure 2.3: The red and the blue values highlight the best and the second-best results respectively. The * indicates that the results were reported in [73] and [80], while [90] is cited for the implementation of the comparison.

2.3.2 CPCGAN

CPCGAN [90] succeeds in generating point clouds from random latent codes by implementing a two-stage GAN framework. The first network of CPCGAN is called Structure GAN while the second network is called Final GAN. Structure GAN is learning the distribution of 32-point structure point clouds, and outputs newly generated structure point clouds, along with their semantic labels. Subsequently, the output of the Structure GAN serves as an input for the Final GAN that learns the distribution of complete point clouds and therefore is able to populate the structure point clouds. A detailed representation of the algorithm can be seen in fig.2.4. The GANs are implementing typical generator models and PointNet-based [66] discriminators. Yang et al. [90] have used the ShapeNet-Partseg [92] in order to showcase the effectiveness of CPCGAN.

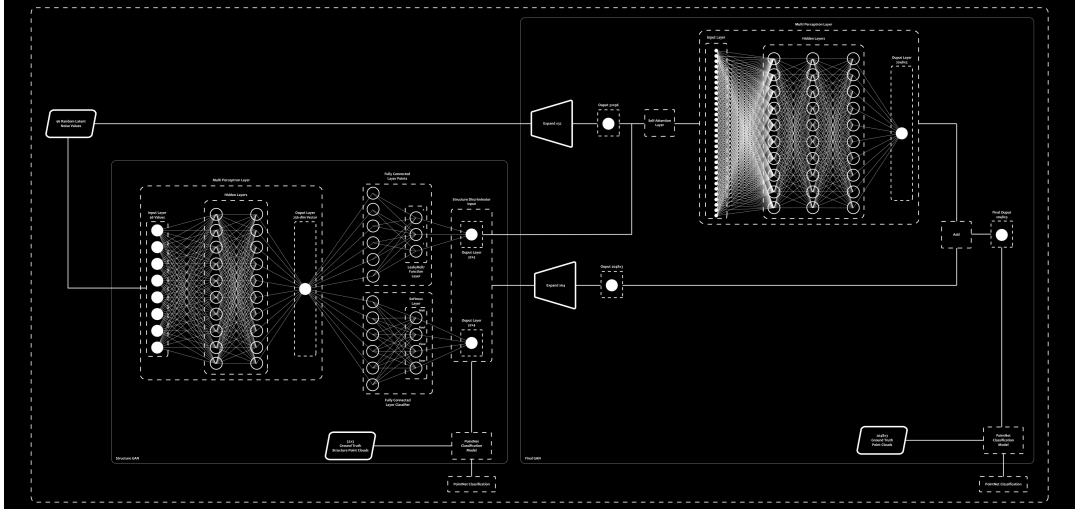


Figure 2.4: A diagram of the structure of the network of Controllable Point Cloud Generative Adversarial Network.

2.4 Point clouds

Even though there are many ways of storing 3D (spatial) information, most 3D Generative Adversarial Networks are employing point clouds. A point cloud is a set of data points in a three-dimensional coordinate system, defined by x, y , and z coordinates [30]. Besides the coordinate values, the dataset may also contain other features and attributes depending on the creation processing pipeline, such as reflection intensities and RGB color values.

The reason why point clouds are popular lies in the simplicity of their components. Using single points, with no attributes of scale, rotation, etc. can be handled and computed much easier in a large amount. This makes point clouds quite easy to edit, display, and filter. Additionally, the variety and accessibility of sensors (LiDar, Kinect, etc.) that construct point clouds have been raising more attention to this specific data type. Fig. 2.5 is depicting a point cloud captured with a LiDar Camera [85], consisting of x, y, z coordinates and R, G, B color values.

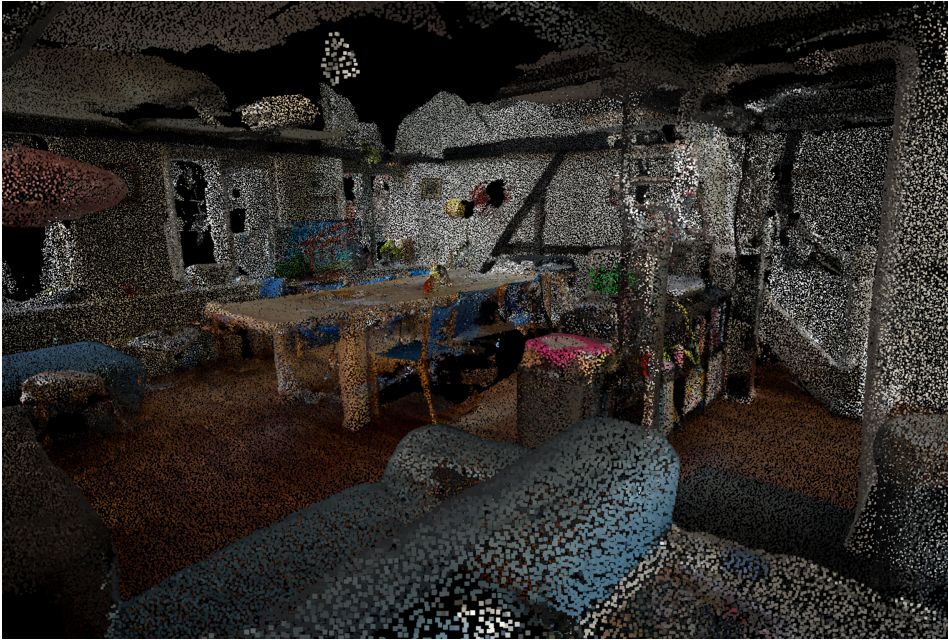


Figure 2.5: The 3D scan of an interior space exported and represented as a point cloud.

2.4.1 Point cloud datasets

Several research projects in the past have developed new approaches towards collecting or generating segmented point clouds. Most of these studies have been focusing on advancing the field of Computer Vision in different scopes. The KITTY

Vision Benchmark [25] was realized in order to assist autonomous driving by implementing urban scale spatial information such as cars, trees, roads, pedestrian streets, and building blocks. The ModelNet dataset [88] has a variety of indoor space data, mostly focusing on objects and furniture detection. The ShapeNet-Partseg [92] consists of 16 object classes, each of them segmented according to its parts (e.g. Airplane: tail, body, wheels, wings). The ArCH dataset [47] consists of large-scale heritage annotated point clouds. Finally, Croce et al. [19] present a semi-automated way of labeling heritage buildings and provide a dataset of 16 annotated point clouds of heritage buildings.

For the purposes of Speculative Hybrids, a dataset with segmented point clouds describing building geometries is needed. A dataset contains related data values that are usually collected or measured as part of a cohort study. However, in the context of Speculative Hybrids, no research has been done providing a suitable dataset. Therefore, the creation of a dataset consisting of annotated point clouds that describe building geometries is realized.

2.5 Dataset creation and augmentation

The output results of a Machine Learning model are determined by the dataset used to train it. The quality and size of the dataset is thus crucial for the learning process of the algorithm [15]. If a dataset is inadequate, the model can return inaccurate results (by learning things wrong) [84], or possibly not be able to learn anything at all.

Several aspects need to be taken into consideration when creating a new dataset [63]. Firstly, a dataset should reflect the world as realistically as possible, and be characterized by high fidelity regarding the information it is providing. Additionally, the biases that are possibly incorporated into a dataset should be eliminated or at least minimized. As far as privacy is concerned for copyrights and ownership, the dataset should be anonymized if necessary.

As mentioned above, the size of the dataset used to train a Machine Learning model will have a great impact on the quality of the resulting output. Deep learning algorithms typically require datasets of considerable size. Therefore, several data augmentation techniques can be used to expand the size of a dataset. Some data augmentation techniques are flip, rotate, scale, crop, translation, and adding noise [81].

Chapter 3

Design and Implementation

Several processes were tested in order to acquire the suitable data for the Speculative Hybrids project. In this report, we are demonstrating the methodology used to create and transform data, as well as the implementation of CPCGAN for the generation of mass building conceptual architecture.

3.1 Dataset Creation

As mentioned in the Subsec. 2.4.1, there is a lack of point cloud datasets of segmented building geometries. For the purpose of Speculative Hybrids a simple dataset is created manually in order to serve as an input for the CPCGAN algorithm. The created dataset is inspired by modern single-family houses around Denmark, some of which can be seen in fig. 3.1.



Figure 3.1: A variation of summer houses in Denmark.

The creation of the dataset is accomplished within four stages:

- **Module Geometries:** Creation of module geometries that represent wall components in Rhinoceros® [49].
- **Mesh Conversion:** Transformation of module geometries into mesh.
- **Point cloud Generation:** Conversion of the geometries into point clouds with the Cockroach plug-in [83].
- **Data Pre-processing:** Export of point clouds and file formatting in order to fit the purposes of CPCGAN.

3.1.1 Module Geometries

In order to be able to generate an adequate amount of building solutions, the implementation of module geometries was decided. 25 modules, describing wall variations, were created within the Rhinoceros® environment for an automated generation of a building series. Data augmentation methods were also applied in order to maximize the size of the dataset. More specifically, the modules were mapped into 3 different scales, and rotated 90 degrees for a wider feature map. A number of plan variations according to the wall modules can be seen in fig. 3.2. Additionally, 35 different roof modules were created and combined with each aforementioned wall module respectively, those variations are shown in fig. 3.3. Additional to the wall and roof modules, the generation of floor modules as simple surfaces was realized. The single components (surfaces) of each module were placed in specific layers, which in the end served as the segmentation and classification labels of the point cloud. The layers are: `WALL_Module`, `ROOF_Module`, and `FLOOR_Module`.

3.1.2 Mesh Conversion

An amount of generated building geometries are demonstrated in fig. 3.4. After the completion of the building generation, all surfaces included in one layer are joined and then converted into a mesh, which is saved in a new corresponding layer. The same process is repeated for all the three pre-existing modules. The new layers that host the meshes are named: `WALL_Mesh`, `ROOF_Mesh`, and `FLOOR_Mesh`. These meshes serve as an input for the generation of point clouds. This process is described in the following subsection.

3.1.3 Point cloud Generation

The Cockroach plug-in [83] was used in order to convert the meshes into point clouds. More specifically, Cockroach is a plugin that allows the implementation of various commands for point cloud post-processing and meshing into the

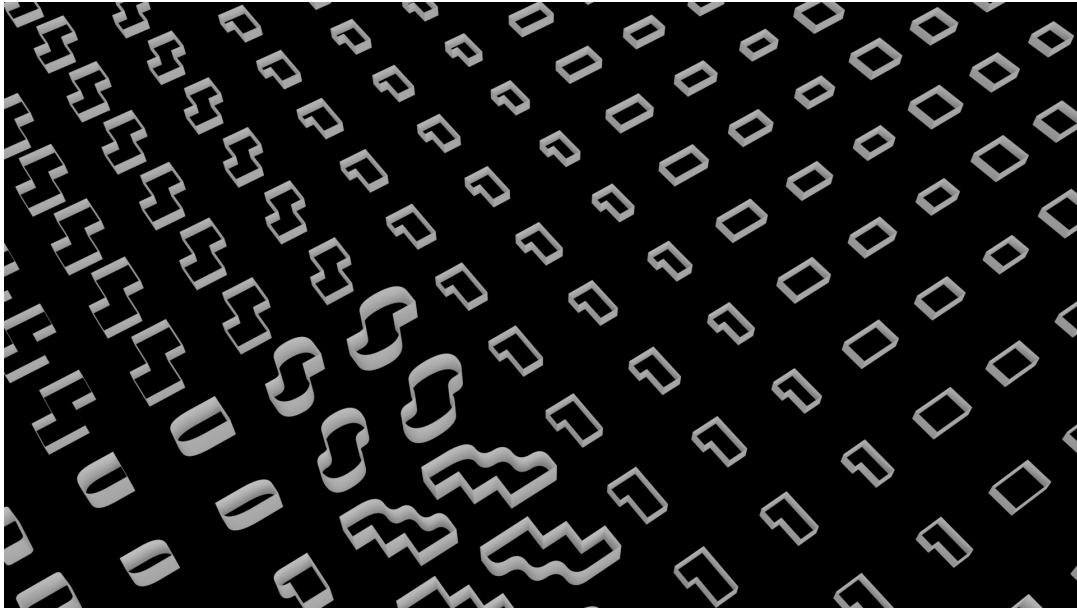


Figure 3.2: An array of the wall modules that shape the building geometries.

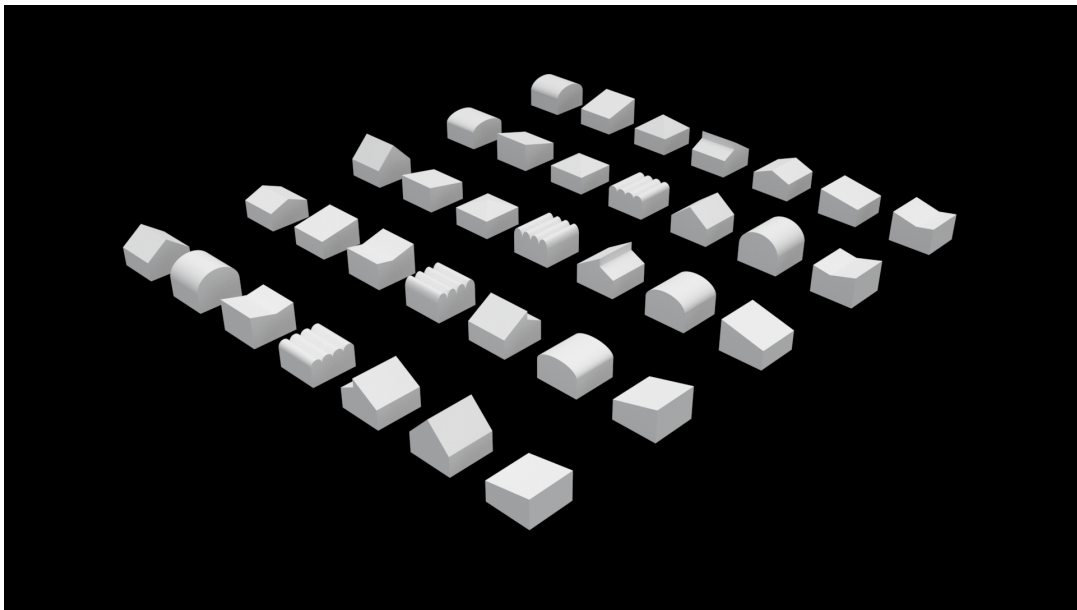


Figure 3.3: The 35 chosen roof versions.

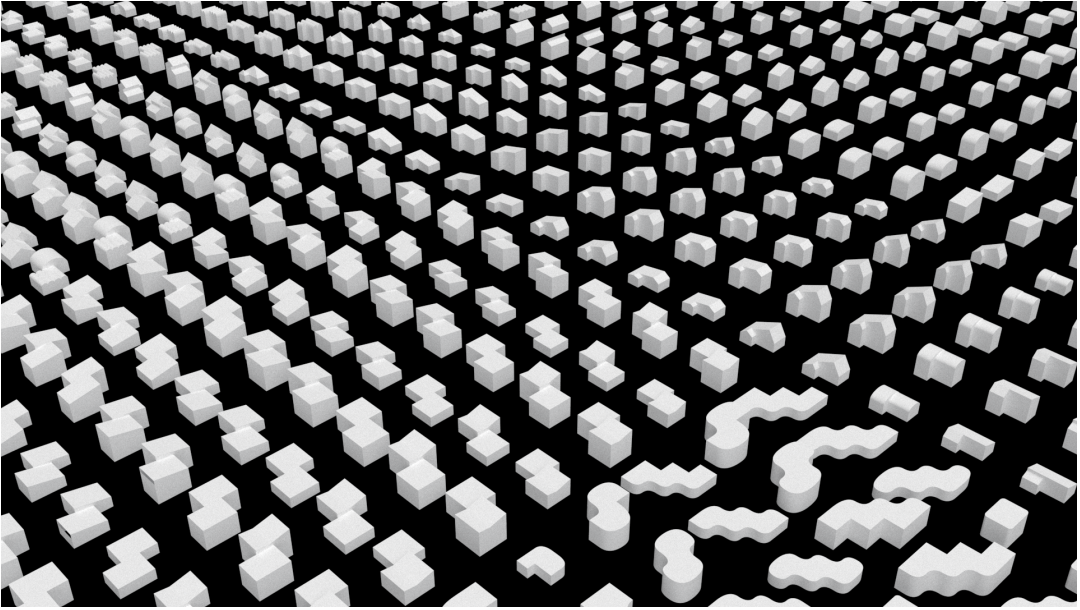


Figure 3.4: The complete array consists of 2,904 simple building geometries that were transformed into point clouds for the training of CPCGAN. In the picture more diverse building generations are showcased.

Mesh	Number of Points
Wall Meshes	1200
Roof Meshes	1000
Floor Meshes	400

Table 3.1: The input parameter for the `PopulateMesh` function of the Cockroach plug-in.

Rhinoceros® environment. The plugin is based on reference functions already existing in the open-source libraries Open3D [95], CGAL [11], Cilantro [93], and PCL [69].

For the generation of points, the function `PopulateMesh` is used. The particular function uses selected meshes, number of points for each mesh and the type of sampling as input. The `PopulateMesh` function is used three times, one for each building component, and its implementation can be seen on fig. 3.5. Poisson Disk Sampling [12] was preferred over random sampling, in order to provide a more uniform distribution of sample points along the converted meshes. Table 3.1 lists the set of inputs for each of the three times the function was used. The output of the function is a point cloud for each building component, which is then merged into a unified point cloud.

Further to the point cloud generation, further procedures apply for an auto-

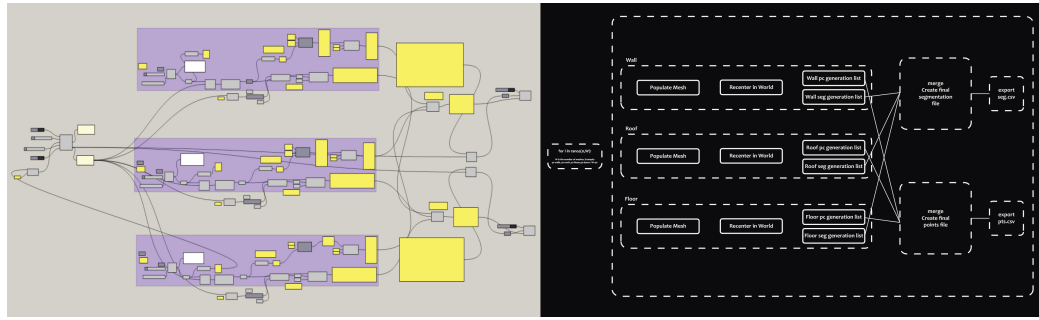


Figure 3.8: A. A screenshot taken from Grasshopper. B. A diagram describing the generation process of the dataset, the one that is demonstrated in the screenshot of image A..

3.1.4 Data Pre-processing

Fig. 3.9 represents the result of a completed point cloud. Consequently, both segmentation and point .csv files are edited through a python [78] script. The script, called `csv_to_pts_seg.py`, as seen in the Appendix Sec. A.1, reads the .csv files, removes the commas and generates two new files. A .pnt file stores information about the points and a .seg file storing information about the labels of each point cloud. The two different files have the same name and are stored in two separate folders. The .pnt files are stored in a folder called `train_data`, while the .seg files are saved in a folder called `train_label`. The procedure is repeated for all the coupled .csv files (point.csv and seg.csv) that are describing the point cloud of a building. The format of the two files can be seen in fig. 3.10

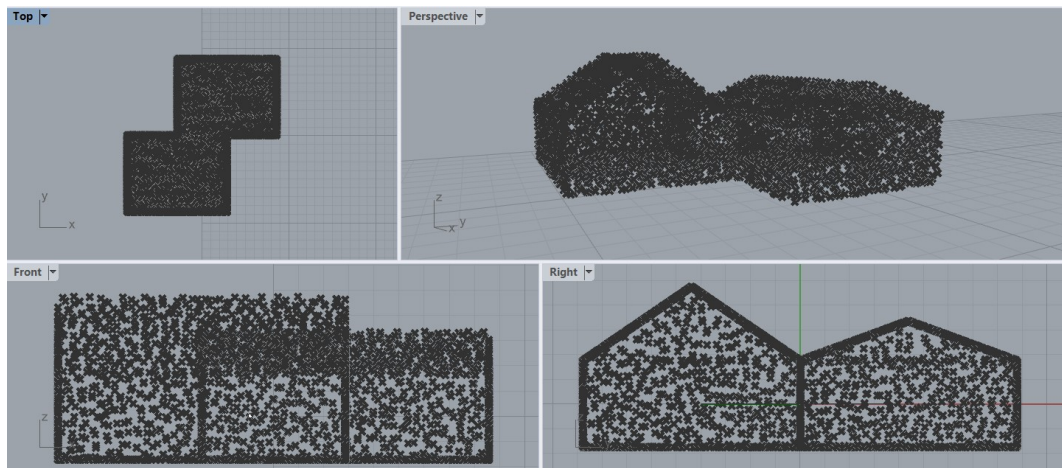


Figure 3.9: An example of a dense point cloud that was generated from the manual building solutions through the Cockroach plug-in.

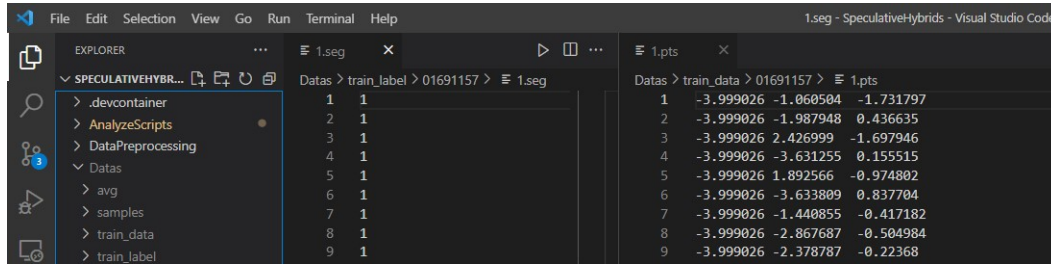


Figure 3.10: The .seg file describing the labels of each point and the .pnt file with the x,y, and z coordinates of each point.

3.2 Building a virtual environment for training CPCGAN

For developing and running the application, Visual Studio Code (VSCode) [52] and Docker [67] were used. Docker is an open platform that allows the separation of an application from the computer’s infrastructure, and is therefore able to deliver software quickly. Additionally, Docker provides the ability to package and run an application in a loosely isolated environment called a container. The creation of a container is accomplished through Docker Images (Dockerfile), which are read-only template instruction files.

The steps for creating a development container in VSCode are:

- Configuring the development container, e.g. installing new software, through the use of a Dockerfile, see Appendix Sec. A.2.
- Creating a devcontainer.json file, see Appendix Sec. A.3, which describes how VSCode should start the initialization of the container.

For the purpose of Speculative Hybrids, the implementation of the CPCGAN required the creation of an Ubuntu 18.04 [45] environment with PyTorch 1.7.0 [65] and CUDA 11.0 [58]. More specifically, a Dockerfile was created hosting information about the environment where the algorithm was trained. The instructions of the Docker Image are creating an environment with the following characteristics: Ubuntu 18.04, PyTorch 1.7.0, CUDA 11.0, Python 3.8.3, Git [26], H5py [17], Joblib [23], Pandas [62], Numpy [57], SciPy [72], TensorBoard [77], Sklearn [41], and DLNest [89]. In order to run such a container in a Windows11 [53] environment, the Windows Subsystem for Linux (WSL2) [55] was used. The information included in the devcontainer.json file are: naming the container, initializing it, running the Dockerfile, and installing some extensions to VSCode, namely the PyTorch Code Snippets, and Python for VSCode.

3.2.1 Implementation of CLAAUDIA

For the use of higher levels of computational power CLAAUDIA [79], the AI cloud of Aalborg University, was used. CLAAUDIA is working with Singularity [34] instead of Docker, therefore the Docker Image of Speculative Hybrids needed to be pulled and transformed into a Singularity Image. All the tasks that require high computational power run on the cloud.

3.3 Training CPCGAN and generating Samples

Foreword to the creation of a dataset and the set up of the environment is the training of CPCGAN and the generation of new Samples. The algorithm is divided into three steps:

- Preprocess the data
- Train CPCGAN
- Generate a sample / Control the generation

3.3.1 Pre-process the data

The original training of CPCGAN was realized with the implementation of the ShapeNet-PartSeg dataset. The particular dataset contains more than 13,000 segmented point-clouds of 16 different classes. In order to succeed in the generation of conceptual massing models the ShapeNet-PartSeg dataset was replaced. In more detail, a Building Class containing 2,904 segmented point clouds, within which each point is categorized by 1 out of 3 segmentation labels was used. The process of the creation of the Building Class was demonstrated in the Sec. 3.1.

Consequently, the data are passing through two scripts, the `sample_points.py` and the `sample_structure_points.py` in order to fit the purposes of CPCGAN. Both scripts can be seen in Appendix Sec. A.4 and A.5 respectively. The first script takes the `.pnt` and the `.seg` files and randomly samples 2048 points for the implementation of the training. Consequently, the same script is creating a samples folder where it saves all these chosen points in a `.sam` file for each point cloud. If one of the point clouds has less than 2048 points the script is printing a message communicating the problem.

In addition, the `sample_structure_points.py` script processes the sampled data and creates a structure point cloud for each point cloud in the dataset. The structure point cloud consists of 32 points and is created using the Growing Neural Gas algorithm. The script is shown in Appendix Sec. A.6. The maximum number of neurons, `max_neurons`, for each semantic label is determined by the percentage of the points with that label in the complete point cloud. The maximum amount of

iterations, `max_iter`, is 100, while the maximum age, `max_age`, is 10. The rest of the parameters were chosen as suggested by the creator of the algorithm [15]. Finally, the labels of the structure points are inherited through the GNG procedure.

In the original repository of CPCGAN, K-means was used instead. This choice was made in order to achieve higher performance rate in the creation of the structure point clouds, aiming for better topology preservation. K-means clustering may be faster, but GNG provides better results [20].

3.3.2 Training CPCGAN

The train of CPCGAN is achieved using DLNest [89]. DLNest is an experimental framework for training deep neural networks. The specific framework allows the training and automatic loading of Machine Learning models for experiments. For the purposes of Speculative Hybrids 3 models with different characteristics were created: a model of 10 epochs with applied FPD metrics, a model of 500 epochs with applied FPD metrics, and finally a model of 2600 epochs with no FPD metrics. The duration of the training for each model was approximately 20 minutes, 30 hours, and 96 hours. The results of the last training are the ones demonstrated in the following chapters.

3.3.3 Generate a sample

The generation of the new point clouds is realized after the training of the two GANs. This process is also achieved using DLNest, and through a script called `gen_a_sample.py`. This file consists of a function within which the model is called and generates a 32-point structure point cloud and a 2048-point fully populated point cloud.

Control the generation

In order to control the generation, a script was created in Python. The script is called `gen_from_spc_and_z.py` and is mapping the building generation towards a site and its specific building regulations. This is a mediating step between the Structure and the Final GAN, the two sub-models of CPCGAN. More specifically, all the points of the newly generated structure point cloud are set to have a maximum distance limit, both regarding width and length and according to the building percentage of the site. Regarding the wall points, a maximum height limit is set in order for the generation to respect the maximum facade height allowed. Additionally, the points of the roof have also a maximum height limit (usually different than the facade height) in order to follow the rule for the maximum total height of the building. The script can be seen in Appendix Sec. A.8.

Visualize results

After the generation of fully populated point clouds, the saved .pts files are processed through the `pts_pto_csv.py` file. The script can be seen in Appendix Sec. A.7. This file reads the .pts files and saves them as .csv files. For visualization purposes the .csv files are then imported into the Rhinoceros environment, where the point normals are calculated, and converted into mesh, through the `CloudMesh` function of the Cockroach plugin. The implementation of the function can be seen in fig. 3.11.

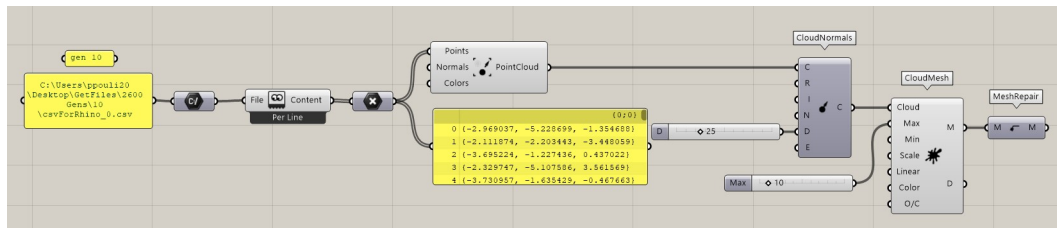


Figure 3.11: The `CloudMesh` function from the Cockroach plugin, that was used to transform the newly generated point clouds back to mesh.

Chapter 4

Evaluation

4.1 Participants

In total 23 subjects, 4.3% under 25, 87% 25-35, 8.7% 36-45 years old, participated in the method evaluation. The participants come from different educational backgrounds ¹: AUTH², Polytechnic University of Milan, SciArc³, IAAC⁴, KADK ⁵, UCL ⁶, Harvard GSD ⁷, and field specialisations: junior architect/designer, junior computational designer/researcher, landscape architect, master student and Phd Candidate in Architecture. The study aims to reach practitioners in architecture of different backgrounds and ages, therefore the selection of the participants was not based in any other particular criteria.

4.2 Creativity Support Index

The Creativity Support Index (CSI) is a psychometric method designed to evaluate how a creativity support tool assists a user engaged in creative work (Carroll and Latulipe 2009). Based on current research, CSI can be effectively utilized to prove the usefulness of computational creativity tools (Cherry and Latulipe 2014). The CSI measures creativity support within the context of six categories: *Exploration*, *Collaboration*, *Engagement*, *Effort/Reward Trade-off*, *Tool Transparency*, and *Expressiveness*.

In addition, the CSI highlights the aspects of the creativity support tool that

¹ All schools are from the global north, and most focus on Technology and Architecture.

² Aristotle University of Thessaloniki

³ Southern California Institute of Architecture

⁴ Institute for Advanced Architecture in Catalonia

⁵ The Royal Danish Academy of fine Arts in Copenhagen

⁶ The Bartlett School of Architecture in London

⁷ Harvard University Graduate School Of Design in Massachusetts

might need further attention. The six factors are evaluated based on one statement each and a final CSI score of 100 is generated for the tool being used. The formula for calculating the CSI score requires the mean values of the answers to the questions for each of the six categories in a scale of 20. Additionally, comparison factors, regarding the importance weight of each category, are implemented in order to yield higher accuracy evaluation results. The mean values are multiplied by each comparison factor, then summed all together and divided by 3.

The higher the score is, the better it indicates creativity support. A score above 90 is an “A,” and indicates excellent support for creative work. On the other hand, a score below 50 is an “F”, showing that the tool does not support creative work that well.

4.3 Measurements

Self-reporting measurements⁸ were used in order to evaluate Speculative Hybrids. The participants were introduced to the method and provided with a case study of a specific architectural site and its building regulations. Ten building options were generated and served as a starting point for their design process. The participants were asked to answer questions regarding how the results of the method described would support their creativity in the early design stages. Additionally, we mentioned that this is a showcase of the tool, that the method can implement more complicated building geometries from a specific architectural type (e.g. brutalism), or a specific architect (e.g. Gaudi), and can help practitioners in the field to explore larger solution space.

The items of the questionnaire were divided into three sections:

- Demographic Information
- Introduction to task and CSI metrics
- Open-ended questions regarding computational design: tools, limitations, and ethical considerations

The questionnaire items are shown in fig. B.4 and fig. B.5, while the complete questionnaire can be seen in Appendix Sec. B.1.

Open-ended questions were also added to the questionnaire in order to acquire qualitative feedback for the method. The questions were the following:

- Which software or tools do you use in the conceptual design phase?
- How do you currently use technology in the ideation phase of the design process?

⁸The questionnaire was created and distributed through Google forms.

- What are the limitations of computational design tools?
- What are the ethical implications of using generative design tools?
- How the increasing use of computational design tools will help the future of design?
- Further feedback

An additional section was created and distributed to the participants in order to estimate the comparison factors for each one of the six categories. In this last section the subjects were asked to show their preference between dual combinations of the categories (e.g. Exploration or Collaboration). The section can be seen in Appendix Sec. B.2.

Chapter 5

Results

In this chapter, the generated outputs of CPCGAN trained with buildings is demonstrated. Additionally, we are showcasing the results of the CSI evaluation of the proposed method.

5.1 Model Generations

The initial output of the algorithm is a 32-point structure point cloud. Forward to that, a 2048-point point cloud is created from the population of the initially generated structure point cloud. An example of the two generations can be seen in fig. 5.1.

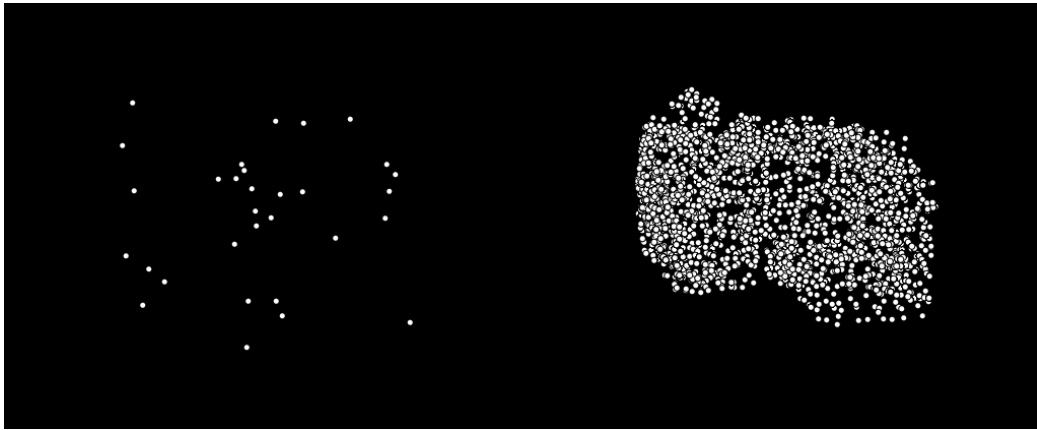


Figure 5.1: On the left side of the image a newly generated structure point cloud is shown, while on the right side of the image we see a fully populated point cloud of 2048 points.

For a better understanding of the results, as mentioned in Subsec. 3.3.3, the fully populated point clouds are converted into a mesh. To demonstrate the functionality of the tool, 25 generations are converted into mesh and depicted in fig.

5.2 - perspective view of the generations, and in fig. 5.3 - top view of the buildings.

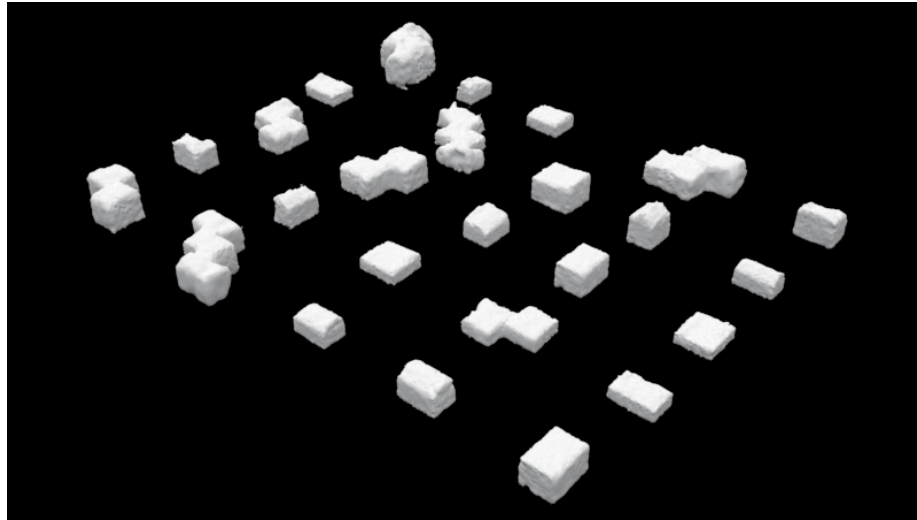


Figure 5.2: The generated Speculative Hybrids from a perspective view.

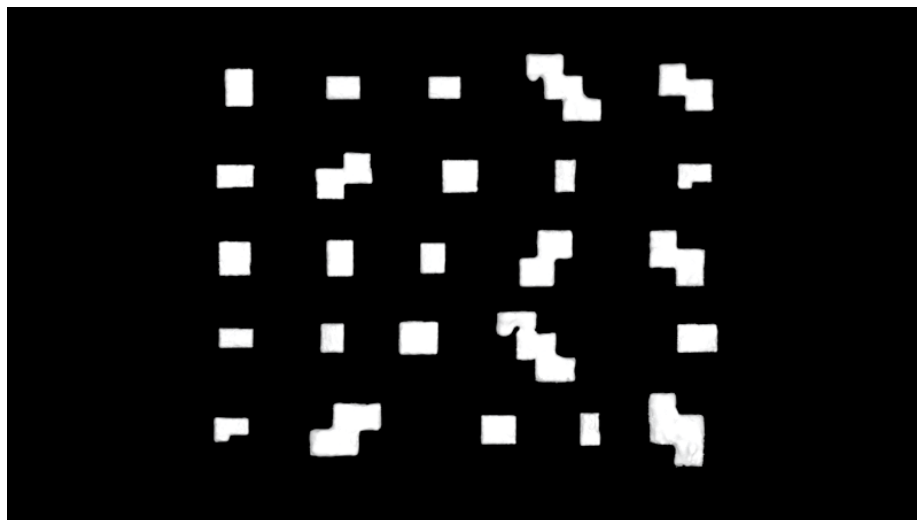


Figure 5.3: The top view of the generated hybrids.

5.2 Questionnaire Results

In this chapter both the quantitative and qualitative results gathered from the questionnaire are demonstrated.

5.2.1 CSI Metrics

An overview of the results regarding the 6 categories for measuring creative support can be seen in the figure 5.4. The overall scores gathered from the questionnaire for each category are: *Exploration* = 77%, *Collaboration* = 78%, *Engagement* = 75%, *Effort/Reward Trade-off* = 76%, *Tool Transparency* = 72%, and *Expressiveness* = 73%. Figures 5.5, 5.6, 5.7, 5.8, 5.9, 5.10 show the distribution of answers for various CSI related dimensions. In general, the tool gathered an average score of 75 out of 100, which means it ranks as a "C" in the aforementioned scale.

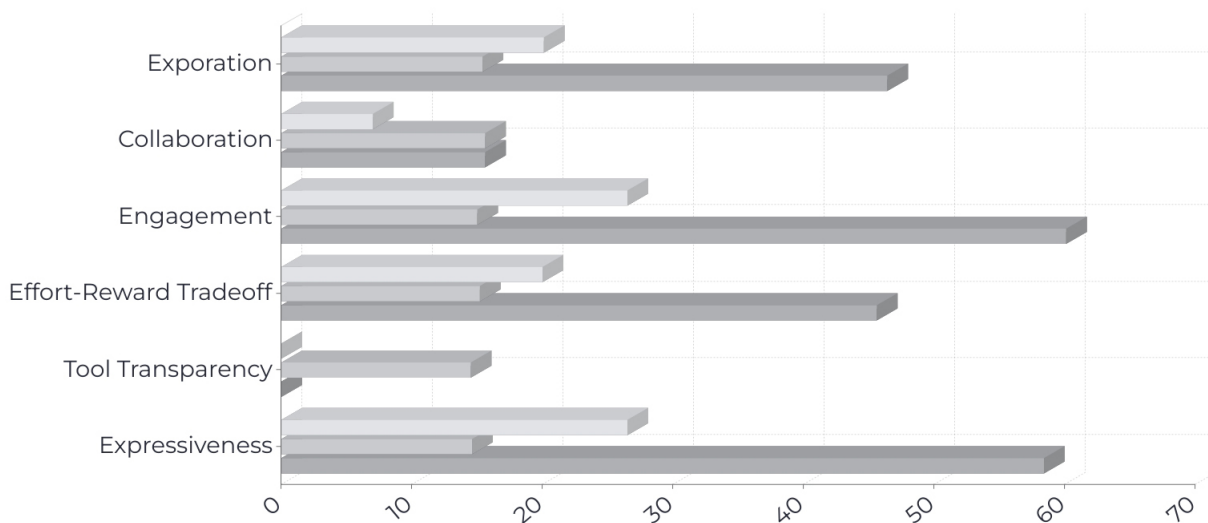


Figure 5.4: Overall CSI score for each category.

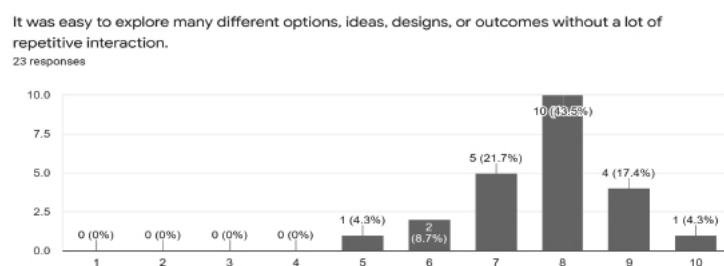


Figure 5.5: The question and the answers regarding *Exploration*, on a scale from 1-10, where 1 = Strongly disagree and 10 = Strongly agree.

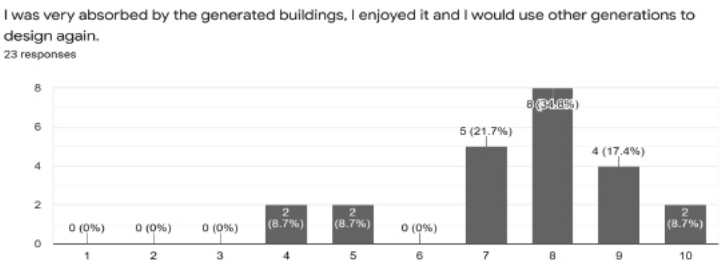


Figure 5.6: The question and the answers regarding *Collaboration*, on a scale from 1-10, where 1 = Strongly disagree and 10 = Strongly agree.

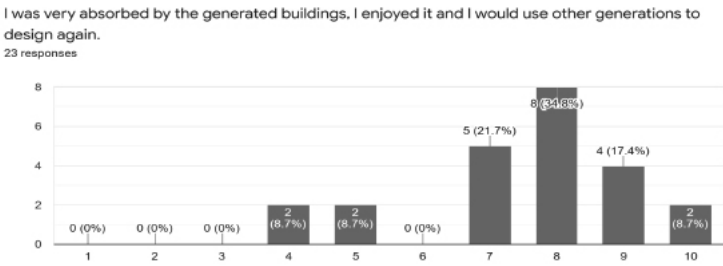


Figure 5.7: The question and the answers regarding *Engagement*, on a scale from 1-10, where 1 = Strongly disagree and 10 = Strongly agree.

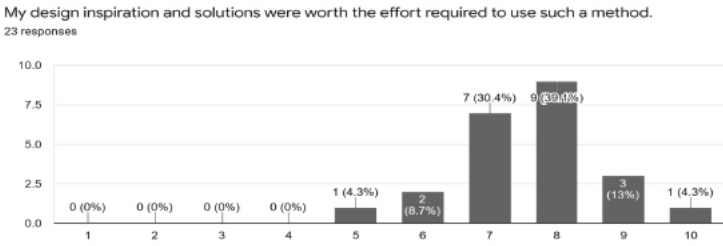


Figure 5.8: The question and the answers *Effort/Reward Trade-off*, on a scale from 1-10, where 1 = Strongly disagree and 10 = Strongly agree.

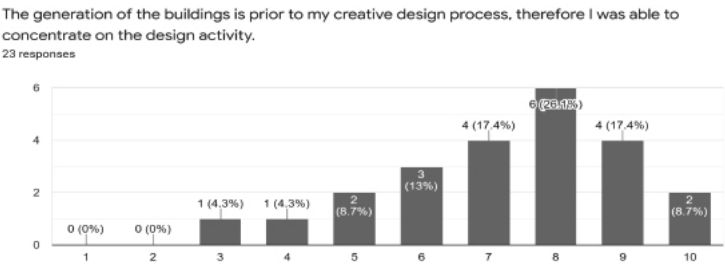


Figure 5.9: The question and the answers regarding *Tool Transparency*, on a scale from 1-10, where 1 = Strongly disagree and 10 = Strongly agree.

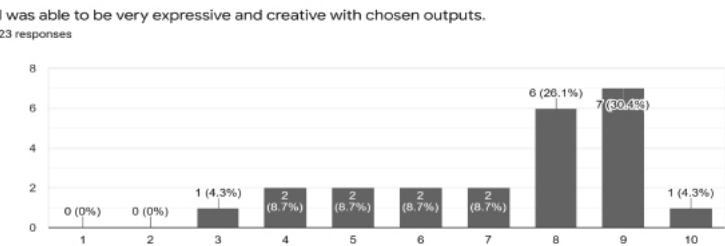


Figure 5.10: The question and the answers regarding *Expressiveness*, on a scale from 1-10, where 1 = Strongly disagree and 10 = Strongly agree.

5.2.2 Open-ended Questions

due to the open-ended questions, practitioners in the field stated that they use both digital and analogue tools for their ideation phase. More specifically, they mentioned that they use hand drawings and sketches, digital drawings (Adobe Creative Suite, Miro, Autocad), physical modeling (blue foam, cardboard models), and digital 3D sculpting (Rhinoceros, SketchUp, Archicad, Grasshopper). Most of the subjects noted that they initiate their design process with conceptual sketches on paper and then they facilitate digital software in order to test the accuracy of the sketch (scale, proportions, etc). Some participants mentioned that they use generative algorithms in order to produce different solutions, while some highlighted the importance of performance calculated by technology (scripts), and therefore indicate that technological tools steer the design outcomes.

Additionally, in this section of the questionnaire participants note probable limitations of computational design tools. Most of the subjects highlight a disconnection regarding the creative process of the designer (mind-output). One of the participants, specifically mentioned that " (such tools)... force you to think like a technologist rather than to think like a creative". They also added that the reasoning of the design investigations in some tools are too strictly framed or without any socio-spatial considerations, and does not provide big or high quality solution space. High fidelity -especially in the field of performance and simulation studies- often lead to computationally intensive workflows, and therefore result in higher costs and processing time.

Regarding ethical consideration of the use of generative design tools, practitioners in the field seem to linger between two opposite positions. Some of them find no ethical implication in using such tools, especially if they are not used as a monopoly but as a tool for negotiation and collaborative design. However, some stated that potential biases might be implanted in the tool from their creators. On the other hand, some architects are doubting whether these tools will eliminate the functionality and role of the designer, and oversimplify the design process. They added that generative design processes might oversee human-scale, social responsiveness or contextual consideration, all features that are conventionally regarded as being central to design. Finally, the fact of social unfairness is also commented, with regard to the people who actually have access to such tools.

When the participants were asked how the use of computational design tools will help the future of design several different answers were given. They stated that architectural design practice is facing higher demand in terms of predictability and performance of its outcomes, therefore such tools will assist the field to evolve into more functional solutions. More specifically, they expect material use optimization, efficient management and enhanced properties of built environment, and sustainable design options. Moreover, the subjects would hope to see technology successfully incorporating in the future human psychology parameters, and to

be accessible to everyone.

Finally, the subjects stated that computational design gives the opportunity to generate and handle massive amounts of information, numerous editions of an idea, and complex geometries. However, aspects like: connection with the surroundings, view and orientation, aesthetics and ergonomics, are leading to decisions that need to be taken outside of an algorithm. They added that technology is undoubtedly the future in every field and suggest to find ways to make it work for our benefit. Additionally, one of the participants mentioned that they would like to have more information regarding the function of the tool, and another one questioned whether the tool is directed to AI-specialized practitioners only.

Chapter 6

Discussion

Speculative Hybrids is a mixed-method research consisting of computational research and deployment, design research, and quantitative and qualitative evaluation techniques. The project aims to make a contribution to architectural design processes and inform theories about how design thinking for architecture change with the use of digital and computational tools by suggesting a new generative method to it. This is achieved by the implementation of Generative Adversarial Networks, in order to ease and inspire architectural practitioners in the early design phase.

Initially, the documentation of a theoretical background for understanding design thinking is set, and followed by the set-up of a method capable of generating multiple variants of buildings that abide to site-specific regulations. These generations, so called *Speculative Hybrids*, may enhance design thinking while providing a tangible description of the solution space.

Fig. 6.1 shows a particularly instructive subset of the input and output of the CPCGAN algorithm. Evidently, some of the generated outputs presented features that were not included in the initial dataset (e.g. differentiated curves). This is precisely the intended outcome since it proves the usefulness of Machine Learning with respect to the creativity aspect (incompleteness) of the design process.

In addition, by inspecting the geometries of the generated buildings, we confirmed that these comply with site-specific regulations. On the contrary, the same does not apply for all the input samples used to train the network. This was achieved by controlling the output of the Structure GAN, before it is fed to the Final GAN. It is thus obvious the relevance of the method in relation to the functionality aspect (precision) of the design process.

The two previous observations demonstrate how the proposed approach is able to facilitate a connection between incompleteness and precision. This is achieved thanks to the architecture of the CPCGAN which, being composed of two sub-models, allows for generation control. However, there are several aspects of the

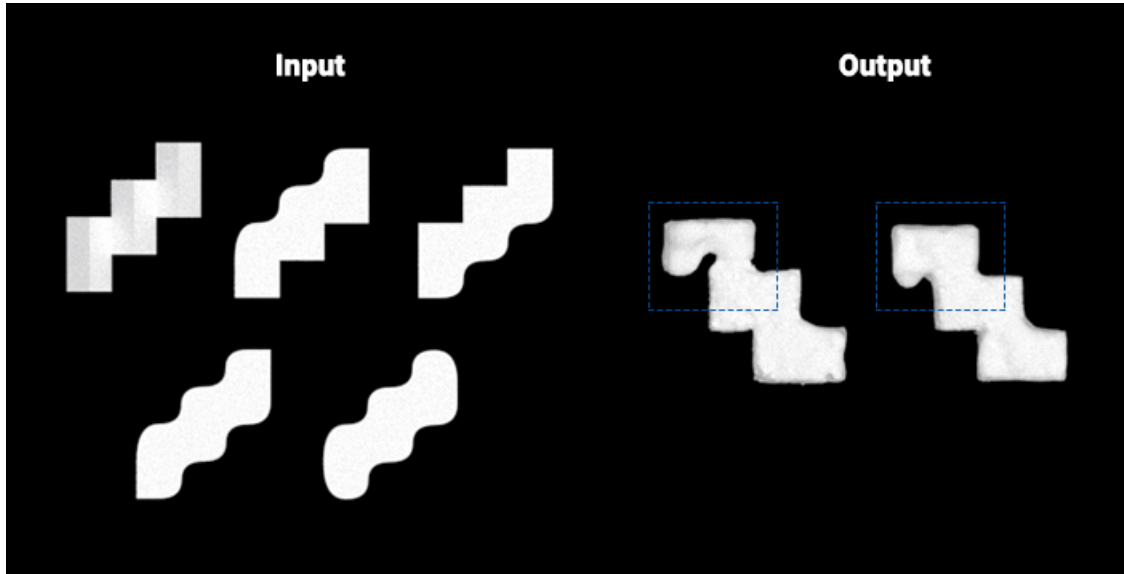


Figure 6.1: A closer look at the input and output data regarding the trained CPCGAN model.

project that require deeper understanding and further exploration.

6.1 Design

Speculative Hybrids serves as a proof of concept for the implementation of Generative Adversarial Networks for the creation of 3D conceptual architecture. Both the dataset creation and the training of the algorithm could be further explored and improved.

Despite exhibiting a significant variety, all the outputs belong to the same building typology as those used to train the model. Being limited to an artificially generated dataset, they are also less detailed than real-life buildings. A sufficiently large collection of segmented point clouds of realistic buildings would have allowed us to explore even further the potentiality of this tool.

As stated in Sec. 3.3.2, the last training of the algorithm consisted of 2600 epochs, which already provided satisfactory results. However, a larger number of iterations would have improved the quality even more. The number was decided due to the time-limitation of usage of the computational resources of CLAAUDIA.

As mentioned in the Sec. 3.3.1, the algorithm was slightly altered in order for his performance to be increased. Specifically, the GNG was implemented to create the structure point clouds preserving the same proportion between the number of points for each label as in the original point cloud. Our point clouds were created using Poisson Disc Sampling, therefore they have no density differentiation regarding each category. However, the use of the proposed method would achieve

significant results if applied to point clouds deriving from scanning technologies, since these would most likely present non homogeneous densities.

6.2 Evaluation Process

The Creativity Support Index evaluation has provided fruitful feedback regarding the proposed method. However, the low number of participants in the evaluation process is concerning with regard to the overall accuracy of the results. To counterbalance that, the quantitative metrics taken from the questionnaires were supplemented by corresponding open-ended questions which were analyzed qualitatively using grounded theory [16].

The sample of architectural practitioners that took part in the evaluation process was verbally informed about the method. As mentioned in the questionnaire Appendix Sec. B.1, the participants were asked to answer the questions depending on the description of the method and the generated results. The results could have been even more insightful if the participants would have had the possibility of using the method themselves.

The Comparison Factors questionnaire (Appendix Sec. B.2) was distributed to the subjects in a second phase. Carroll and Latulipe [14] mention in their research that the factor comparison can be tedious for a participant. Therefore, they suggest that the factors should be ranked independently of the actual conditions. It is thus justifiable to administer the factor comparison part of the survey just once, at the end of the study. That choice led to a smaller sample of answers regarding the comparison factors, but since it is an independent scale we believe there is no significant impact regarding the overall CSI score.

6.3 User feedback

The CSI evaluation process was very informative and provided an invaluable insight regarding how the method could be further improved. The overall CSI score for Speculative Hybrids is 75%, which translates as a "C" in an "A" to "F" scale. As mentioned in the Evaluation chapter, an "A" ranking indicates excellent support for creative work, while a ranking of "F" means that the tool does not provide creativity support that well. Therefore, Speculative Hybrids seem to provide creativity support at a satisfactory level.

6.3.1 Quantitative Results

Taking a closer look at the results, interesting observations can be made. The individual scores for each one of the 6 evaluated categories are between a range

of 72-78%. Therefore, the tool provides good performance in supporting the creative process. As revealed/highlighted by the comparison factor questionnaire, Engagement and Expressiveness were the two most relevant factors. These categories acquired a score of 75% and 73% percent respectively. The second most relevant factors were Exploration (77%), and Effort/Reward Trade-off (76%). The Collaboration factor, which had the highest ranking score (78%), gathered a small comparison factor. It was exceptionally interesting that the Tool Transparency factor was not relevant at all regarding the suggested method, even though it achieved a 72% score. This may be caused due to the fact that the participants were introduced to the method but did not implement it themselves. From our perspective, we expected that the most relevant categories would be: Exploration, Expressiveness, Effort/Reward Trade-off and Collaboration, since our method aims to inspire architectural practitioners in the early design phase. The overall score and the individual score of each one of them are promising, opening a road-map for further research on the method and its implementation.

6.3.2 Qualitative Results

Even more fruitful insights were gathered by the open-ended questions. Some practitioners of the field still favor analogue techniques for expressing themselves and for initiating their conceptual design phase. On the contrary, some others implement computational design tools, and base their designs on results accountable for sustainability and performance. This may be caused by the limitations of computational design tools, due to their inaccessibility, or due to the cost of their implementation. However, we foresee a future where such tools will be accessible by everyone and embedded in the educational systems of architecture.

Additionally, participants correctly noticed that the reasoning of the design investigations in most tools are too strictly framed or without any socio-spatial considerations. Therefore, we would like to do further research in order to suggest a process of choosing the best generated outputs. Several tools can be used for such an analysis, for example considering environmental impacts with computational tools such as Ladybug [40], and LearnCarbon [37].

Even though some of the participants find no ethical implications in using generative design tools, it is important to take them under consideration, especially while designing them. Potential biases might be included in a Machine Learning model, and therefore the designer of the tool needs to be cautious with the implemented dataset. Moreover the use of these tools does not aim to eliminate the functionality and role of the designer, but rather assist their workflow. It is inevitable that the role of the architect is changing as technology progresses, but the final outcome always needs to be evaluated and realized upon human factors.

Computational design gives the opportunity to generate and handle massive

amounts of information, numerous editions of an idea, and complex geometries. Speculative Hybrids succeeds in providing solutions and assist the field in formulating inspirational shape-making.

Chapter 7

Conclusion

By the 2000s, most of the design processes had started to incorporate digital tools. This digitalization of processes enhances architectural design thinking by serving as a mediator between incompleteness (creative exploration) and precision (functionality and evaluation) - both inherited properties of the design process. More specifically, the contemporary technologies are currently assistive in many architectural areas such as design, fabrication, construction, and maintenance. However, the incorporation of Machine Learning approaches may assist architectural design practices to better leverage data-rich environments and workflows. In our attempt to bridge the gap between incompleteness and precision, we suggest the implementation of CPCGAN, which is an unsupervised Machine Learning Generative Adversarial Network, that allows the control of its output generations.

The implementation of Generative models has proven fruitful using a variety of data types including text, images, and videos. However, architectural design practices often communicate their ideas through the use of 3D models, that include a plethora of information regarding shape, building components, materials etc. Point clouds are a method of representing spatial information, and they are raising in popularity because of the simplicity of their components. For the purposes of the project, a dataset of annotated point clouds describing building geometries is required. There is, to the best of our knowledge no such dataset, and therefore we suggest a semi-automated method for its realization.

The quality and size of the dataset is crucial for the learning process of an algorithm. If a dataset is inadequate, there is the possibility that the model will not manage to learn anything at all. Therefore, we decided to create a simple dataset inspired by a specific building typology: single-family summer houses in Denmark. The realization of the dataset is completed within four stages: the creation of wall, roof, and floor modules combined in different ways, their conversion into meshes, the point cloud generation through the `PopulateMesh` function of the Cockroach plug in, and finally, the point clouds export and file formatting. Throughout this

process a dataset of 2.904 segmented point clouds describing architectural forms was realized.

Forward to the dataset creation, the point clouds were downsized to 2048 points, and a structure point cloud of 32 points was created using the Growing Neural Gas algorithm, a topology learning, and preservation algorithm, for each one of them. The structure and complete point clouds serve as an input to the CPC-GAN's two adversarial sub-models: the Structure GAN and the Final GAN. CPC-GAN was trained with different numbers of iterations, however, the 2600 epochs training, lasting approximately 96 hours, was the most successful. Consequently, a script was created in order to map the the generation of new building geometries depending on the regulations of a specific site (building percentage, facade height, total height).

For the evaluation of the proposed method 25 buildings were generated, out of which 10 were introduced to 23 practicing architects, along with a description of the method. The Creativity Support Index was used to acquire quantitative results for the performance of the method regarding 6 categories: Exploration, Collaboration, Engagement, Effort/Reward Tradeoff, Tool Transparency and Expressiveness. Additionally, qualitative feedback was gathered regarding the use of technology in conceptual design phases, its ethical implications, limitations, and the future of it.

The method achieved an overall score of 75%, which is promising, opening a road-map for further research on the method and its implementation. We foresee a future where such tools will be accessible to use by everyone and embedded in the educational systems of architecture. In the present study we proposed a method for generating *Speculative Hybrids*. These hybrids are able, at a certain level, to inspire architectural practitioners in the ideation phase through exploration, within the context of site-specific regulations. Concluding, we believe that there is a great potential in exploring new methods to enhance creativity in the architectural design with respect to functionality and precision, in order to find new solutions for a better future.

7.0.1 Future perspectives

A number of investigation avenues emerge as a natural continuation of this project.

The dataset that was used to train CPCGAN is relatively simple, describing different scales of single-family houses. Despite its simplicity, generating such a dataset is a rather time-consuming process. Therefore, we are planning to make the dataset publicly accessible for the research community.

In order to automate and expand the process of the dataset creation, generative algorithms could be implemented. More specifically, the Wave Function Collapse algorithm (WFC) [38] which is a constraint-based algorithm that takes an input image, either 2D or 3D, and procedurally generates a larger space in the same style.

The implementation of the algorithm is realised within the Rhinoceros environment though the Monoceros [74] plugin.

Additionally, we would like to explore the possibility of using real-life data for the same process. This could be achieved with the use of 3D scanning technologies like LiDar cameras [85]. As mentioned before, CPCGAN requires segmented point clouds, therefore the use of 3D scanning outputs requires the development of a method for segmenting them. A PointNet classification model [66] would be the most suitable algorithm for this purpose. Implementing and testing this method would be a logical next step. Moreover, a collaboration with an architectural practise could be established in order to train the CPCGAN with point clouds generated from their own BIM models.

Regarding the performance of CPCGAN, we would like to employ the Growing Neural Gas algorithm for downsizing the point clouds to 2048 points, a process that happens randomly in CPCGAN in its current implementation.

Another logical development would be to implement the proposed method in a ready-to-use Rhinoceros plugin. In combination with a wider availability of suitable datasets, this would allow us to explore in even more depth the potential of this approach. It would thus be extremely valuable to address the evaluation to architectural practitioners by asking them to implement and utilize the plugin. It is thus clear that the present work opens a road-map for future investigations.

Bibliography

- [1] Hervé Abdi and Lynne J Williams. “Principal component analysis”. In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.
- [2] Panos Achlioptas et al. “Learning Representations and Generative Models for 3D Point Clouds”. In: *arXiv:1707.02392 [cs]* (July 2017). URL: <https://arxiv.org/abs/1707.02392> (visited on 05/13/2022).
- [3] Zeynep Aksöz and Clemens Preisinger. “An Interactive Structural Optimization of Space Frame Structures Using Machine Learning”. In: *Impact: Design With All Senses* (Aug. 2019), pp. 18–31. DOI: 10.1007/978-3-030-29829-6_2. (Visited on 05/25/2022).
- [4] Jamal Al-Qawasmi and Karim Hadjri. *Architecture in the digital age: The effect of digital media on the design, production and evaluation of the built environment*. June 2007. (Visited on 05/24/2022).
- [5] Nuha Aldausari et al. “Video Generative Adversarial Networks: A Review”. In: *ACM Computing Surveys* 55 (Mar. 2023), pp. 1–25. DOI: 10.1145/3487891.
- [6] Jose Algeciras-Rodriguez. *Stochastic Hybrids From references to design options through Self-Organizing Maps methodology*. 2018. (Visited on 05/25/2022).
- [7] Imdat As, Siddharth Pal, and Prithwish Basu. “Artificial intelligence in architecture: Generating conceptual design via deep learning”. In: *International Journal of Architectural Computing* 16 (Nov. 2018), pp. 306–327. DOI: 10.1177/1478077118800982.
- [8] Cyprien de Masson d’Autume et al. *Training Language GANs from Scratch*. Neural Information Processing Systems, 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/a6ea8471c120fe8cc35a2954c9b9c595-Abstract.html> (visited on 05/13/2022).
- [9] Bernd Bickel et al. “Design and fabrication of materials with desired deformation behavior”. In: *ACM Transactions on Graphics* 29 (July 2010), pp. 1–10. DOI: 10.1145/1778765.1778800. (Visited on 03/17/2021).
- [10] Plugin Biomorpher. *Biomorpher*. Food4Rhino, Aug. 2017. URL: <https://www.food4rhino.com/en/app/biomorpher> (visited on 05/25/2022).

- [11] CGAL Editorial Board. *The Computational Geometry Algorithms Library*. Cgal.org, 2019. URL: <https://www.cgal.org/>.
- [12] Robert Bridson. “Fast Poisson disk sampling in arbitrary dimensions.” In: *SIGGRAPH sketches* 10.1 (2007), p. 1.
- [13] Inês Caetano, Luís Santos, and António Leitão. “Computational design in architecture: Defining parametric, generative, and algorithmic design”. In: *Frontiers of Architectural Research* 9.2 (2020), pp. 287–300.
- [14] Erin A. Carroll and Celine Latulipe. *The Creativity Support Index*. 2009. (Visited on 05/30/2022).
- [15] Rendi Chevi. *GitHub - RendChevi/growing-neural-gas*. GitHub, July 2020. URL: <https://github.com/rendchevi/growing-neural-gas> (visited on 05/31/2022).
- [16] Ylona Chun Tie, Melanie Birks, and Karen Francis. “Grounded theory research: A design framework for novice researchers”. In: *SAGE open medicine* 7 (2019), p. 2050312118822927.
- [17] Andrew Collette. *HDF5 for Python*. www.h5py.org. URL: <https://www.h5py.org/> (visited on 05/31/2022).
- [18] Antonia Creswell et al. “Generative Adversarial Networks: An Overview”. In: *IEEE Signal Processing Magazine* 35 (Jan. 2018), pp. 53–65. DOI: 10.1109/msp.2017.2765202.
- [19] Valeria Croce et al. “From the Semantic Point Cloud to Heritage-Building Information Modeling: A Semiautomatic Approach Exploiting Machine Learning”. In: *Remote Sensing* 13 (Jan. 2021), p. 461. DOI: 10.3390/rs13030461. (Visited on 05/14/2022).
- [20] Michael Daszykowski, Beata Walczak, and Desire L Massart. “On the optimal partitioning of data with k-means, growing k-means, neural gas, and growing neural gas”. In: *Journal of chemical information and computer sciences* 42.6 (2002), pp. 1378–1389.
- [21] Matias Del Campo, Alexandra Carlson, and Sandra Manninger. “Towards Hallucinating Machines-Designing with Computational Vision”. In: *International Journal of Architectural Computing* 19.1 (2021), pp. 88–103.
- [22] Randy Deutsch. *BIM and integrated design : strategies for architectural practice*. Wiley, 2011.
- [23] Joblib developers. *Joblib: running Python functions as pipeline jobs — joblib 0.14.1.dev0 documentation*. joblib.readthedocs.io, 2008. URL: <https://joblib.readthedocs.io/en/latest/> (visited on 05/31/2022).
- [24] Bernd Fritzke. “A growing neural gas network learns topologies”. In: *Advances in neural information processing systems* 7 (1994).

- [25] Andreas Geiger, Philip Lenz, and Raquel Urtasun. *Are we ready for autonomous driving? The KITTI vision benchmark suite*. IEEE Xplore, June 2012. DOI: 10.1109/CVPR.2012.6248074. URL: <https://ieeexplore.ieee.org/abstract/document/6248074/>.
- [26] Git. *Git*. Git-scm.com, 2019. URL: <https://git-scm.com/>.
- [27] Ian Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks*. arXiv.org, 2016. URL: <https://arxiv.org/abs/1701.00160>.
- [28] Ian Goodfellow et al. *Generative Adversarial Nets*. 2014. (Visited on 04/17/2022).
- [29] Mikell P Groover and Emory W Zimmers. *CAD / computer-aided design and manufacturing*. Prentice-Hall, 1984.
- [30] Xian-Feng Hana et al. "A comprehensive review of 3D point cloud descriptors". In: *arXiv preprint arXiv:1802.02297* 2 (2018).
- [31] John Harding. "Advances in Architectural Geometry 2015 - Dimensionality Reduction for Parametric Design Exploration". In: (2016). DOI: 10.3218/3778-4_19. (Visited on 05/25/2022).
- [32] Anca-Simona Horvath. "How we talk (ed) about it: Ways of speaking about computational architecture". In: *International Journal of Architectural Computing* (2022), p. 14780771211070006.
- [33] Weixin Huang and Hao Zheng. *Architectural Drawings Recognition and Generation through Machine Learning 1 Apartment floor plan: recognition and generation through generative adversarial network*. Oct. 2018. (Visited on 05/25/2022).
- [34] Sylabs Inc. *Introduction to Singularity — Singularity container 3.5 documentation*. sylabs.io, 2017. URL: <https://sylabs.io/guides/3.5/user-guide/introduction.html> (visited on 05/31/2022).
- [35] Jens Jørgensen, Martin Tamke, and Kåre Poulsgaard. *Occupancy-informed: Introducing a method or flexible behavioural mapping in architecture using machine vision*. 2020. (Visited on 05/25/2022).
- [36] Salome Kazeminia et al. "GANs for medical image analysis". In: *Artificial Intelligence in Medicine* 109 (Sept. 2020), p. 101938. DOI: 10.1016/j.artmed.2020.101938. URL: <https://arxiv.org/pdf/1809.06222.pdf> (visited on 11/22/2020).
- [37] Kritika Kharbanda et al. "LearnCarbon: A tool for Machine Learning prediction of Global Warming Potential from abstract designs". In: *eCAADe* (Sept. 2022).
- [38] Hwanhee Kim et al. "Automatic generation of game content using a graph-based wave function collapse algorithm". In: *2019 IEEE Conference on Games (CoG)*. IEEE. 2019, pp. 1–4.

- [39] K Krishna and M Narasimha Murty. “Genetic K-means algorithm”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 29.3 (1999), pp. 433–439.
- [40] Plugin Ladybug. *Ladybug Tools*. Food4Rhino, Oct. 2016. URL: <https://www.food4rhino.com/en/app/ladybug-tools> (visited on 05/25/2022).
- [41] scikit learn. *scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation*. Scikit-learn.org, 2019. URL: <https://scikit-learn.org/stable/> (visited on 05/31/2022).
- [42] Chun-Liang Li et al. “Point Cloud GAN”. In: *arXiv:1810.05795 [cs, stat]* (Oct. 2018). URL: <https://arxiv.org/abs/1810.05795> (visited on 05/14/2022).
- [43] Ruihui Li et al. *PU-GAN: A Point Cloud Upsampling Adversarial Network*. openaccess.thecvf.com, 2019. URL: https://openaccess.thecvf.com/content_ICCV_2019/html/Li_PU-GAN_A_Point_Cloud_Upsampling_Adversarial_Network_ICCV_2019_paper.html (visited on 05/14/2022).
- [44] Eric C.W. Lou, Nguyen The Quan, and Kenneth Park. “Guest editorial: Disruptive and innovative technologies for sustainable development in the built environment”. In: *International Journal of Building Pathology and Adaptation* 40 (May 2022), pp. 301–304. DOI: 10.1108/ijbpa-06-2022-173. (Visited on 05/25/2022).
- [45] Canonical Ltd. *Ubuntu 18.04.5 LTS (Bionic Beaver)*. releases.ubuntu.com, 2018. URL: <https://releases.ubuntu.com/18.04/>.
- [46] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).
- [47] Eva Savina Malinverni et al. “Deep learning for semantic segmentation of 3D point cloud.” In: *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* (2019).
- [48] Scott Marble. *Digital Workflows in Architecture: Design–Assembly–Industry*. Walter de Gruyter, 2012.
- [49] Robert McNeel et al. “Rhinoceros 3D, Version 6.0”. In: *Robert McNeel & Associates, Seattle, WA* (2010).
- [50] Leo Meng, Jeremy Graham, and M Haeusler. *T-SNE: A DIMENSIONALITY REDUCTION TOOL FOR DESIGN DATA VISUALISATION*. 2020. (Visited on 05/25/2022).
- [51] Paul R. Messinger et al. “Virtual worlds — past, present, and future: New directions in social computing”. In: *Decision Support Systems* 47 (June 2009), pp. 204–228. DOI: 10.1016/j.dss.2009.02.014.
- [52] Microsoft. *Visual Studio Code*. Visualstudio.com, Apr. 2016. URL: <https://code.visualstudio.com/>.

- [53] Microsoft. *Windows 11 Specs and System Requirements* | Microsoft. Windows. URL: <https://www.microsoft.com/en-us/windows/windows-11-specifications> (visited on 05/31/2022).
- [54] Dubravko Miljković. “Brief review of self-organizing maps”. In: *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. 2017, pp. 1061–1066.
- [55] craigloewen msft. *Install WSL*. Microsoft.com, Feb. 2022. URL: <https://docs.microsoft.com/en-us/windows/wsl/install>.
- [56] Yuko Munakata and Jason Pfaffly. “Hebbian learning and development”. In: *Developmental science* 7.2 (2004), pp. 141–148.
- [57] Numpy. *NumPy — NumPy*. Numpy.org, 2009. URL: <https://numpy.org/>.
- [58] NVIDIA. *CUDA Toolkit 11.0 Download*. NVIDIA Developer, Aug. 2020. URL: <https://developer.nvidia.com/cuda-11.0-download-archive> (visited on 05/31/2022).
- [59] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. *Pixel Recurrent Neural Networks*. proceedings.mlr.press, June 2016. URL: <https://proceedings.mlr.press/v48/oord16.html> (visited on 05/13/2022).
- [60] Rivka Oxman. “Theory and design in the first digital age”. In: *Design Studies* 27 (May 2006), pp. 229–265. DOI: 10.1016/j.destud.2005.11.002. URL: <https://www.sciencedirect.com/science/article/pii/S0142694X05000840#fig5> (visited on 01/22/2020).
- [61] George Palamas and J Andrew Ware. “Sub-goal based robot visual navigation through sensorial space tessellation”. In: *International Journal of Advanced Research in Artificial Intelligence* 2.11 (2013).
- [62] Pandas. *Python Data Analysis Library — pandas: Python Data Analysis Library*. Pydata.org, 2018. URL: <https://pandas.pydata.org/>.
- [63] Amandalynne Paullada et al. “Data and its (dis) contents: A survey of dataset development and use in machine learning research”. In: *Patterns* 2.11 (2021), p. 100336.
- [64] Panagiota Pouliou et al. “Sensing Behavior Framework: Acquisition of occupancy behavior data.” In: *Symposium on Simulation for Architecture and Urban Design. SimAUD* (July 2022).
- [65] PyTorch. *PyTorch*. www.pytorch.org. URL: <https://pytorch.org/get-started/previous-versions/> (visited on 05/31/2022).
- [66] Charles R Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *arXiv preprint arXiv:1612.00593* (2016).

- [67] Babak Bashari Rad, Harrison John Bhatti, and Mohammad Ahmadi. "An introduction to docker and analysis of its performance". In: *International Journal of Computer Science and Network Security (IJCSNS)* 17.3 (2017), p. 228.
- [68] Peter G Rowe et al. *Design Thinking In The Digital Age*. StenbergPress, 2018.
- [69] Radu Bogdan Rusu and Steve Cousins. *Point Cloud Library (PCL): PCL API Documentation*. pointclouds.org, 2011. URL: <https://pointclouds.org/documentation/> (visited on 05/25/2022).
- [70] Aditya Sanghi et al. "CLIP-Forge: Towards Zero-Shot Text-to-Shape Generation". In: *arXiv:2110.02624 [cs]* (Apr. 2022). URL: <https://arxiv.org/abs/2110.02624> (visited on 05/14/2022).
- [71] Donald A. Schön. *The Reflective Practitioner*. Routledge, Mar. 2017. DOI: 10.4324/9781315237473. URL: <https://www.taylorfrancis.com/books/9781351883160>.
- [72] SciPy. *SciPy.org* — *SciPy.org*. Scipy.org, 2020. URL: <https://scipy.org/>.
- [73] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. *3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions*. openaccess.thecvf.com, 2019. URL: https://openaccess.thecvf.com/content_ICCV_2019/html/Shu_3D_Point_Cloud_Generative_Adversarial_Network_Based_on_Tree_Structured_ICCV_2019_paper.html (visited on 05/14/2022).
- [74] Subdigital s.r.o. *Monoceros*. monoceros.sub.digital, 2021. URL: <https://monoceros.sub.digital/> (visited on 05/31/2022).
- [75] Kyle Steinfeld. "Significant others: Machine learning as actor, material, and provocateur in art and design". In: *The Routledge Companion to Artificial Intelligence in Architecture*. Routledge, 2021, pp. 3–12.
- [76] Martin Tamke, Paul Nicholas, and Mateusz Zwierzycki. "Machine learning for architectural design: Practices and infrastructure". In: *International Journal of Architectural Computing* 16.2 (2018), pp. 123–143.
- [77] Get started with TensorBoard. *Get started with TensorBoard | TensorFlow*. TensorFlow, 2019. URL: https://www.tensorflow.org/tensorboard/get_started.
- [78] Python TM. *Python Release Python 3.8.3*. Python.org. URL: <https://www.python.org/downloads/release/python-383/> (visited on 05/31/2022).
- [79] Aalborg University. *CLAAUDIA - Research Data Services*. www.claaudia.aau.dk. URL: <https://www.claaudia.aau.dk/> (visited on 05/31/2022).
- [80] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. *Learning Localized Generative Models for 3D Point Clouds via Graph Convolution*. openreview.net, Sept. 2018. URL: https://openreview.net/forum?id=SJeXSo09FQ&source=post_page----- (visited on 05/13/2022).

- [81] David A Van Dyk and Xiao-Li Meng. "The art of data augmentation". In: *Journal of Computational and Graphical Statistics* 10.1 (2001), pp. 1–50.
- [82] Elio Ventocilla et al. "Scaling the Growing Neural Gas for Visual Cluster Analysis". In: *Big Data Research* 26 (2021), p. 100254.
- [83] Petras Vestartas and Andrea Settimi. *Cockroach*. Food4Rhino, Dec. 2020. URL: <https://www.food4rhino.com/en/app/cockroach> (visited on 05/25/2022).
- [84] Bertie Vidgen and Leon Derczynski. "Directions in abusive language training data, a systematic review: Garbage in, garbage out". In: *Plos one* 15.12 (2020), e0243300.
- [85] Ulla Wandinger. "Introduction to lidar". In: *Lidar*. Springer, 2005, pp. 1–18.
- [86] Jennifer Whyte and Jennifer Whyte. *Virtual Reality and the Built Environment*. Routledge, Aug. 2007. doi: 10.4324/9780080520667. (Visited on 05/25/2022).
- [87] Thomas Wortmann and Bige Tunçer. "Differentiating parametric design: Digital workflows in contemporary architecture and construction". In: *Design Studies* 52 (2017), pp. 173–197.
- [88] Zhirong Wu et al. "3d shapenets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920.
- [89] Ximing Yang. *DLNest*. GitHub, May 2022. URL: <https://github.com/SymenYang/DLNest> (visited on 05/31/2022).
- [90] Ximing Yang et al. *CPCGAN: A Controllable 3D Point Cloud Generative Adversarial Network with Semantic Label Generating*. May 2021. URL: <https://www.aaai.org/AAAI21Papers/AAAI-4341.YangX.pdf>.
- [91] Zhangsihao Yang, Haoliang Jiang, and Lan Zou. "3D Conceptual Design Using Deep Learning". In: *Advances in Intelligent Systems and Computing* (Apr. 2019), pp. 16–26. doi: 10.1007/978-3-030-17795-9_2. (Visited on 05/25/2022).
- [92] Li Yi et al. *A Scalable Active Framework for Region Annotation in 3D Shape Collections*. web.stanford.edu, Nov. 2016. URL: http://web.stanford.edu/~eric/yi/project_page/part_annotation/index.html (visited on 05/14/2022).
- [93] Konstantinos Zampogiannis, Cornelia Fermüller, and Yiannis Aloimonos. *kzampog/cilantro*. GitHub, 2018. URL: <https://github.com/kzampog/cilantro> (visited on 05/25/2022).
- [94] Junzhe Zhang et al. *Unsupervised 3D Shape Completion Through GAN Inversion*. openaccess.thecvf.com, 2021. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Zhang_Unsupervised_3D_Shape_Completion_Through_GAN_Inversion_CVPR_2021_paper.html (visited on 05/14/2022).
- [95] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. *Open3D – A Modern Library for 3D Data Processing*. Open3d, 2018. URL: <http://www.open3d.org/>.

Appendix A

Scripts

A.1 csvtoptsseg.py

```
from re import S
import pandas as pd
import numpy as np
#from sqlalchemy import column, false
import os

from requests import head

#-----Set file paths-----

dirPoints= 'rawData\points'
dirSeg= 'rawData\segments'
i = 1
a = 1

#-----Creating .pts file-----

for filename in os.listdir(dirPoints):
    f = os.path.join(dirPoints,filename)
    if os.path.isfile(f):

        #Read .csv file
        df = pd.read_csv(str(f), usecols=[0,1,2], names=['x', 'y', 'z'])

        first_column = df.iloc[:, 0]
        sec_column = df.iloc[:, 1]
        third_column = df.iloc[:, 2]
        #third_column = third_column.replace("}", "")
```

```

f_list = first_column.values.tolist()
f_list = [w.replace('{', '') for w in f_list]

f_df = pd.DataFrame(f_list)

th_list = third_column.values.tolist()
th_list = [w.replace('}', '') for w in th_list]

th_df = pd.DataFrame(th_list)

f_df['y'] = sec_column
f_df['z'] = th_df

new_header = f_df.iloc[0] #grab the first row for the header
f_df = f_df[1:] #take the data less the header row
f_df.columns = new_header #set the header row as the df header

#print (f_df)

#Save as .csv file
f_df.to_csv("Dats/csvs/" + str(i) + ".csv", index=False,
            header=False)

text = open("Dats/csvs/" + str(i) + ".csv", "r")

#Remove "," and "'"
text = ''.join([i for i in text]) \
        .replace(",", " ")
x = open("A_oldProcess/Dump/output_pnt.csv", "w")
x.writelines(text)
x.close()

read = pd.read_csv("A_oldProcess/Dump/output_pnt.csv")

read.to_csv("Dats/train_data/" + str(i) + ".pts", index=False,
            header=False)

i = i + 1

#-----Creating .seg file-----

for filename in os.listdir(dirSeg):
    b = os.path.join(dirSeg, filename)
    if os.path.isfile(b):

        #Read .csv file
        s_df = pd.read_csv(str(b))

```

```

#Save as .csv file
s_df.to_csv("Datas/segs/" + str(i) + ".csv", index=False,
            header=False)

text = open("Datas/segs/" + str(i) + ".csv", "r")

x = open("A_oldProcess/Dump/output_seg.csv", "w")
x.writelines(text)
x.close()

read = pd.read_csv("A_oldProcess/Dump/output_seg.csv")

#Save as .seg file
read.to_csv("Datas/train_label/" + str(a) + ".seg", index=False,
            header=False)

a = a + 1

```

A.2 Dockerfile

```

FROM pytorch/pytorch:1.7.0-cuda11.0-cudnn8-devel

# [Optional] Uncomment this section to install additional OS packages.
RUN apt-get update -y && export DEBIAN_FRONTEND=noninteractive

#Import CV2
RUN apt-get update && apt-get install git -y

RUN conda install -y ipython h5py nltk joblib pandas scipy \
    # Install DLNest
    && pip install git+https://github.com/SymenYang/DLNest.git \
    # Install other libs
    && pip install tensorboard

RUN pip install sklearn

RUN pip install numpy

RUN pip install igraph

RUN pip install plyfile

```

A.3 devcontainer.json

```
// For format details, see https://aka.ms/devcontainer.json. For config
// options, see the README at:
//
// https://github.com/microsoft/vscode-dev-containers/tree/v0.217.4/containers/docker-existing-dockerfile
{
    "name": "Pytorch 1.7.0 Ubuntu 18.04",
    "runArgs": ["--init", "--network=host", "--gpus=all"],

    // Sets the run context to one level up instead of the .devcontainer
    // folder.
    "context": "..",

    // Update the 'dockerFile' property if you aren't using the standard
    // 'Dockerfile' filename.
    "dockerFile": "../Dockerfile",

    // Set *default* container specific settings.json values on container
    // create.
    "settings": {},

    // Add the IDs of extensions you want installed when the container is
    // created.
    "extensions": ["ms-python.python", "sbsnippets.pytorch-snippets"],

    "features": {
        "git": "latest",
        "github-cli": "latest"
    }
}
```

A.4 samplepoints.py

```
import pathlib
from pathlib import Path
import json
import random

root = Path(__file__).absolute().parent.parent
config_file = root / "dataset_config.json"
```

```

def sample_k_points(config,k=2048):
    cat = {}

    cat_path = root / config["category_information_path"]
    with cat_path.open("r") as f:
        for line in f:
            ls = line.strip().split()
            cat[ls[0]] = ls[1]

    meta = {}
    for item in cat:
        meta[item] = []
        dir_point = root / Path(config["full_pointcloud_path"] ) / cat[item]
        dir_seg = root / Path(config["semantic_label_path"] ) / cat[item]
        dir_sampling = root / Path(config["pointcloud_sample_path"] ) /
            cat[item]

        fns = dir_point.iterdir()

        for fn in fns:
            token = fn.stem
            meta[item].append((
                dir_point / (token + '.pts'),
                dir_seg / (token + '.seg'),
                dir_sampling / (token + '.sam'),
                dir_sampling
            ))

    for cls_key in meta:
        print(cls_key)
        cls_list = meta[cls_key]
        for item in cls_list:
            ifp = item[0].open('r')
            if not item[3].exists():
                item[3].mkdir(parents=True,exist_ok=True)
            ofp = item[2].open('w')

            lines = ifp.readlines()
            lst = [str(i) + '\n' for i in range(len(lines))]
            if len(lst) < k:
                ifp.close()
                ofp.close()
                item[0].unlink()
                item[1].unlink()
                item[2].unlink()
                print("points not enough ",item[0])

```

```

        continue
    slines = random.sample(lst,k)
    ofp.writelines(slines)
    ifp.close()
    ofp.close()

if __name__ == "__main__":
    with config_file.open("r") as f:
        config = json.load(f)["dataset_config"]
        print("Sampling ground truth points")
        sample_k_points(config)

```

A.5 samplestructurepoints.py

```

import pathlib
from pathlib import Path
import json
from pyexpat import model
import numpy as np
from sklearn.cluster import KMeans

root = Path(__file__).absolute().parent.parent
config_file = root / "dataset_config.json"

def read_pointclouds(points_path,semantic_path,sample_path):
    """
    args:
        points_path: pathlib.PATH for pointclouds
        semantic_path: pathlib.PATH for semantic labels
        sample_path: pathlib.PATH for pointcloud sample

    return:
        points: [n,3] sampled points
        sem_labels: [n] sampled labels
    """
    points = np.loadtxt(points_path).astype(np.float32)
    sample = np.loadtxt(sample_path).astype(np.int64)
    sem_label = np.loadtxt(semantic_path).astype(np.int64)

    points = points[sample]
    sem_label = sem_label[sample]

    return points,sem_label

```

```

def get_sem_counts(sem_label):
    """
    args:
        sem_label: [n] semantic label

    return:
        classes: INT semantic label classes count
        counts: [k] points num for every of k semantic label
    """
    counts = []
    for item in sem_label:
        while item > len(counts):
            counts.append(0)
        counts[item - 1] += 1
    return len(counts), counts

def get_sem_split(sem_counts, points_num, k):
    """
    args:
        sem_counts: [k] points num for every of k semantic label
        points_num: INT number of points in pointcloud
        k: INT number of sampled points

    output:
        dvi: [k] sampled points num for every semantic label
    """
    ratio = [sem_counts[i] / points_num * k for i in range(len(sem_counts))]
    ratio_int_part = [int(ratio[i]) for i in range(len(ratio))]
    ratio_float_part = [(ratio[i] - ratio_int_part[i], i) for i in
                        range(len(ratio))]
    dvi = ratio_int_part
    cnt = 0
    for item in ratio_int_part:
        cnt += item
    rest = k - cnt

    ratio_float_part.sort(key=lambda x : -x[0])
    for i in range(rest):
        dvi[ratio_float_part[i][1]] += 1

    return dvi

def k_means(points, k, id):
    """
    args:
        points: [n] point cloud of same semantic label

```

```

        k: INT the k in k_means
        id: INT id of the semantic label

    return:
        sampled_points: [k,4] center of k clusters
    """
    estimator = KMeans(n_clusters=k)
    estimator.fit(points)
    centroids = estimator.cluster_centers_
    ids = [[id] for i in range(k)]
    print (centroids)
    return np.concatenate([centroids, ids],axis=1)

from GNG_implementation.neuralgas import GrowingNeuralGas

def gng(points,k,id):

    arr = np.array(points)
    print (np.shape(arr))

    """
    args:
        points: [n] point cloud of same semantic label
        k: INT the k in k_means -> n_inputs
        id: INT id of the semantic label

    return:
        sampled_points: [k,4] center of k clusters
    """

    gng = GrowingNeuralGas(
        max_neurons = k,
        max_iter = 100,
        max_age= 10,
        eb = 0.1,
        en = 0.006,
        alpha = 0.5,
        beta = 0.995,
        l =1,
        dataset = arr
    )

    centroids, model = gng.learn()
    centroids = np.array(centroids)
    centroids = centroids[:-1]

    print (k)

```



```

print(np.shape(centroids))

ids = [[id] for i in range(k)]
ids = np.array(ids)
print(np.shape(ids))
print(type(ids))

return np.concatenate([centroids, ids],axis=1)

def get_avg_sem_samples(points,sem_label,k):
    """
    args:
        points: [n,3] pointclouds
        sem_label: [n] semantic label
        k: INT number of sampled points we want to get

    return:
        sem_classes: INT number of semantic classes
        sampled_points: [k * 4] k points from sampled results concat with
            its semantic label
    """
    sem_classes,counts = get_sem_counts(sem_label)
    points_num = len(points) #2048?

    sub_points = [[] for i in range(sem_classes)] #In our case
    sub_points=[[],[],[],[],[]]
    sem_split = get_sem_split(counts,points_num,k) #This is a list with the
    number of points ??that all have to sum at 32?? on each label ex. 1
    -> 17pnts 2->5 3->5 3-> 100 4-> 4 5-> 1

    for i in range(points_num):
        sem = sem_label[i] # =1 =wall
        sub_points[sem - 1].append(points[i]) # appends all the points of
        the same label in an array. all arrays from all labels are in
        one array called sub_points

    sampled_points = None
    for i in range(sem_classes):
        if sem_split[i] == 0:
            continue

        if sampled_points is None:
            sampled_points = gng(sub_points[i],sem_split[i],i + 1) #chance
            with gng
        else:
            sampled_points = np.concatenate(

```

```

        [sampled_points,
         gng(sub_points[i],sem_split[i],i + 1)], #chance with gng
         axis = 0
    )
    return sem_classes,sampled_points

def sample_semantic_points(config,k = 32):
    """
    args:
        config: {} config dict
        k: INT number of sampled points
        type: "avg" or "equal" or "nosem"

    return:
        None
    """
    cat = {}

    cat_path = root / config["category_information_path"]
    with cat_path.open("r") as f:
        for line in f:
            ls = line.strip().split()
            cat[ls[0]] = ls[1]

    meta = {}
    for item in cat:
        meta[item] = []
        dir_point = root / Path(config["full_pointcloud_path"] ) / cat[item]
        dir_seg = root / Path(config["semantic_label_path"] ) / cat[item]
        dir_sampled = root / Path(config["pointcloud_sample_path"] ) /
            cat[item]
        dir_sem_sampling = None
        dir_sem_sampling = root /
            Path(config["semantic_avg_sampled_pointcloud_path"]) / cat[item]

        fns = dir_point.iterdir()

        for fn in fns:
            token = fn.stem
            meta[item].append((
                dir_point / (token + ".pts"),
                dir_seg / (token + ".seg"),
                dir_sampled / (token + ".sam"),
                dir_sem_sampling / (token + ".pts"),
                dir_sem_sampling
            ))

```

```

meta_info = {}
for cls_key in meta:
    print(cls_key)
    cls_list = meta[cls_key]
    max_sem_classes = 0
    cnt = 0

    for item in cls_list:
        if cnt % 100 == 0:
            print(cnt)
            cnt += 1
            if not item[4].exists():
                item[4].mkdir(parents=True, exist_ok=True)
            out_fp = item[3].open('w')

            points, sems = read_pointclouds(item[0], item[1], item[2])
            #points_path, semantic_path, sample_path

            sem_classes, out_pointclouds = get_avg_sem_samples(points, sems, k)

            max_sem_classes = max(sem_classes, max_sem_classes)
            np.savetxt(
                out_fp,
                out_pointclouds
            )
            out_fp.close()

    meta_info[cls_key] = {
        "max_sem_classes": max_sem_classes,
        "item_num": len(cls_list)
    }
    print(meta_info[cls_key])

path_meta = root / Path(config["semantic_avg_sampled_pointcloud_path"])
    / "meta.json"
meta_fp = path_meta.open('w')
json.dump(
    meta_info,
    meta_fp
)
meta_fp.close()

if __name__ == "__main__":
    with config_file.open("r") as f:
        config = json.load(f)["dataset_config"]
        print("Sampling structure points")

```

```
sample_semantic_points(config)
```

A.6 neuralgas.py

```
from tqdm import tqdm
from sys import stdout

import igraph as ig

import numpy as np
from numpy.linalg import norm

class GrowingNeuralGas:

    def __init__(self, max_neurons, max_iter, max_age, eb, en, alpha, beta,
                  l, dataset):
        '''
        -----
        Growing Neural Gas' Parameter Declarations
        -----
        1. max_neurons ; Maximum # of neurons generated by the network
        2. max_iter ; Maximum # of iterations the
        3. max_age ; Maximum # of age
        4. eb ; fraction of distance between nearest neuron and input signal
        5. en ; fraction of distance between neighboring neurons and input
           signal
        6. alpha ; multiplying scalar for local error
        7. beta ; multiplying scalar for global error
        8. l
        9. dataset
        '''
        # Parameters declared by user
        self.max_neurons = max_neurons
        self.max_iter = max_iter
        self.max_age = max_age
        self.eb = eb
        self.en = en
        self.alpha = alpha
        self.beta = beta
        self.l = l
        self.dataset_original = dataset.copy()
        self.dataset = dataset.copy()
```

```

    # Variable for tracking learning evolution
    self.verts_evolve = []
    self.edges_evolve = []

def initialize_gng(self):
    '''
    Initialize Growing Neural Gas
    '''
    # Get random datapoints from target dataset
    t0 = np.random.randint(0, int(self.dataset.shape[0] / 2))
    #print (t0)
    t1 = np.random.randint(int(self.dataset.shape[0]/2),
                           self.dataset.shape[0])
    #print (t1)

    # Initialize Growing Neural Gas
    self.gng = ig.Graph()
    s = self.gng.add_vertex(weight = self.dataset[t0,:], error = 0)
    self.gng.add_vertex(weight = self.dataset[t1,:], error = 0)
    self.gng.add_edge(0, 1, age = 0)
    print (s['weight'])

def learning_position(self):
    for _ in range(0, self.l):
        # Step 1. Get a random datapoint from target dataset
        t = np.random.randint(0, self.dataset.shape[0])
        random_input = self.dataset[t, :]
        #print (t)

        # Step 2. Find 2 nearest neuron from random_input
        nearest_index = np.array([norm(weight - random_input)**2 for
                                weight in self.gng.vs['weight']]).argsort()
        neuron_s1 = self.gng.vs[nearest_index[0]]
        #print (neuron_s1)
        neuron_s2 = self.gng.vs[nearest_index[1]]
        #print (neuron_s2)

        # Step 3. Increase the age of all neighboring edges from nearest
        neuron (neuron_s1)
        for edge_id in self.gng.incident(neuron_s1.index):
            self.gng.es[edge_id]['age'] += 1

        # Step 4. Add error to the nearest neuron
        self.gng.vs[neuron_s1.index]['error'] +=
            norm(neuron_s1['weight'] - random_input)
        # Step 5.1. Update position of nearest neuron

```

```

neuron_s1['weight'] += (self.eb * (random_input -
    neuron_s1['weight']))
# Step 5.2. Update position of nearest neuron's neighbors
for neuron in self.gng.vs[self.gng.neighbors(neuron_s1.index)]:
    neuron['weight'] += (self.en * (random_input -
        neuron_s2['weight']))

# Step 6. Update edge of nearest neurons
EDGE_FLAG = self.gng.get_eid(neuron_s1.index, neuron_s2.index,
    directed = False, error = False)
if EDGE_FLAG == -1: # FLAG for no edge detected
    self.gng.add_edge(neuron_s1.index, neuron_s2.index, age = 0)
else:
    self.gng.es[EDGE_FLAG]['age'] = 0

# Step 7.1. Delete aging edge
for edge in self.gng.es:
    src = edge.source
    tgt = edge.target
    if edge['age'] > self.max_age:
        self.gng.delete_edges(edge.index)

# Step 7.2. Delete isolated neuron
for neuron in self.gng.vs:
    if len(self.gng.incident(neuron)) == 0:
        self.gng.delete_vertices(neuron)

# Step 8. Reduce global error
for neuron in self.gng.vs:
    neuron['error'] *= self.beta

# Step 9.1. Remove generated random input from target dataset
self.dataset = np.delete(self.dataset, t, axis = 0)
# Step 9.2. Reset dataset if datapoints are depleted
if self.dataset.shape[0] == 1:
    self.dataset = self.dataset_original.copy()
    #print (self.dataset)

def update_neuron(self):
    # Adding new neuron from previous learning
    if len(self.gng.vs) <= self.max_neurons:
        # Get neuron q and f
        error_index = np.array([error for error in
            self.gng.vs['error']]).argsort()
        neuron_q = self.gng.vs[error_index[-1]]

```

```

        error = np.array([(neuron['error'], neuron.index) for neuron in
                           self.gng.vs[self.gng.neighbors(neuron_q.index)]])
        error = np.sort(error, axis = 0)
        neuron_f = self.gng.vs[int(error[-1, 1])]

        # Add neuron between neuron q and f
        self.gng.add_vertex(weight = (neuron_q['weight'] +
                                       neuron_f['weight']) / 2, error = 0)
        neuron_r = self.gng.vs[len(self.gng.vs) - 1]

        # Delete edge between neuron q and f
        self.gng.delete_edges(self.gng.get_eid(neuron_q.index,
                                                neuron_f.index))

        # Create edge between q-r and r-f
        self.gng.add_edge(neuron_q.index, neuron_r.index, age = 0)
        self.gng.add_edge(neuron_r.index, neuron_f.index, age = 0)

        # Update neuron error
        neuron_q['error'] *= self.alpha
        neuron_f['error'] *= self.alpha
        neuron_r['error'] = neuron_q['error']

    pnts=[]
    for neuron in self.gng.vs:
        pnts.append(neuron['weight'])

    return pnts

def learn(self):
    # Initialize GNG
    self.initialize_gng()
    # GNG learning iteration
    for iter, _ in zip(range(0, self.max_iter),
                      tqdm(range(self.max_iter))):
        # Track evolution
        self.verts_evolve.append(np.array([neuron['weight'] for neuron
                                           in self.gng.vs]))

        self.edges_evolve.append(np.array([(neuron.source + 1,
                                             neuron.target + 1) for neuron in self.gng.es]))
        # Learn new posititon
        self.learning_position()
        pnts = self.update_neuron()

    return pnts, self.gng

```

A.7 ptstocsv.py

```
import pandas as pd
import numpy as np
from sqlalchemy import column, false
import os
import csv

def read_pts(filename):
    return np.loadtxt(filename, comments=("version:", "n_points:", "{",
                                          "}"))

dir= 'Output'
i=0
for filename in os.listdir(dir):
    f = os.path.join(dir,filename)

    n_array = read_pts(f)

    df = pd.DataFrame(n_array, columns = ['Column_A','Column_B','Column_C'])

    test=df.to_csv("Output\Transformed\csvForRhino_" + str(i) + ".csv",
                   index=False, header=False)

    i=i+1
```

A.8 gen_from_spc_and_z.py

```
from pathlib import Path
import numpy as np
import torch
import pandas as pd
from sqlalchemy import column, false
import os
import csv

def modify_pc_0(pc_0_path):
    def read_pts(filename):
        return np.loadtxt(filename, comments=("version:", "n_points:", "{",
                                              "}"))
```



```
def distance(x, y):
    if x >= y:
        result = x - y
    else:
        result = y - x
    return result

dir = pc_0_path
i=0

for filename in os.listdir(dir):
    f = os.path.join(dir,filename)

    n_array = read_pts(f)

    df = pd.DataFrame(n_array, columns =
        ['Column_A','Column_B','Column_C', 'Column_D', 'Column_E',
        'Column_F', 'Column_X'])
    #print(df)
    #print(len(df))

    labels = []
    for i in range(0,len(df)):
        a = df['Column_D'].values[i]
        b = df['Column_E'].values[i]
        c = df['Column_F'].values[i]

        largest = 0

        if a > b and a > c:
            largest = 1
        if b > a and b > c:
            largest = 2
        if c > a and c > b:
            largest = 3

        labels.append(largest)

    df['Labels'] = labels

    for i in range(0,len(df)):
        label = df['Labels'].values[i]
        x = df['Column_A'].values[i]
        y = df['Column_B'].values[i]
        z = df['Column_C'].values[i]
```

```

##PARAMETERS AND RULES FOR MAPPING regarding building width -
length

for s in range(0,len(df)):
    xs = df['Column_A'].values[s]
    distanceX = distance(x,xs)
    limitX = 25 #The limit is a variable chosen by the user
                (depending on the field) maximum x distance

    if distanceX > limitX:
        w = distance/2
        if x>=xs:
            if x>=0:
                cent = x - w
                df.replace(x, cent+(limitX/2))
                df.replace(xs, cent-(limitX/2))
            else:
                if xs>=0:
                    cent = xs - w
                    df.replace(x, cent-(limitX/2))
                    df.replace(xs, cent+(limitX/2))

        ys = df['Column_B'].values[s]
        distanceY = distance(y,ys)
        limitY = 20 #The limit is a variable chosen by the user
                    (depending on the field) maximum y distance

    if distanceY > limitY:
        w = distance/2
        if y>=ys:
            if y>=0:
                cent = y - w
            else:
                cent = w + y
            df.replace(y, cent+(limitY/2))
            df.replace(ys, cent-(limitY/2))
        else:
            if y>=0:
                cent = ys - w
            else:
                cent = w + ys

        df.replace(y, cent-(limitY/2))
        df.replace(ys, cent+(limitY/2))

```

```

    ##PARAMETERS AND RULES FOR MAPPING regarding Facade height
    if label == 1:
        limitZ = 5.5 #The limit is a variable chosen by the user
                      (depending on the field) maximum z distance from wall
                      points
        if z>limitZ:
            df.replace(z, limitZ)

    ##PARAMETERS AND RULES FOR MAPPING regarding Maximum B height
    height
    if label == 2:
        limitZmin = 5.5 #The limit is a variable chosen by the user
                        (depending on the field) maximum z distance from wall
                        points (limitZ)
        limitZmax = 9.5 #The limit is a variable chosen by the user
                        (depending on the field) maximum z distance (total
                        building height)
        if z<limitZmin:
            df.replace(z, limitZmin)
        if z>limitZmax:
            df.replace(z, limitZmax)

    df = df.drop(['Labels'], axis=1)

    print(df)

    #Save as .pts
    df.to_csv(pc_0_path, index=False, header=False)
    i=i+1

    with pc_0_path.open("r") as f:
        pc_0 = np.loadtxt(f)
    return pc_0

def experience(self):
    model = self.model
    # load z and structure point cloud
    z_path = Path("./z.npy")
    pc_0_path = Path("./pc_0.pts")
    z,pc_0 = None,None
    with z_path.open("rb") as f:
        z = torch.load(f).view(1,-1)

    # Modify the structure point cloud
    pc_0 = modify_pc_0(pc_0_path)
    # To GPU
    z = z.cuda()

```

```
pc_0 = torch.from_numpy(pc_0).float().view(1,-1,7).cuda()

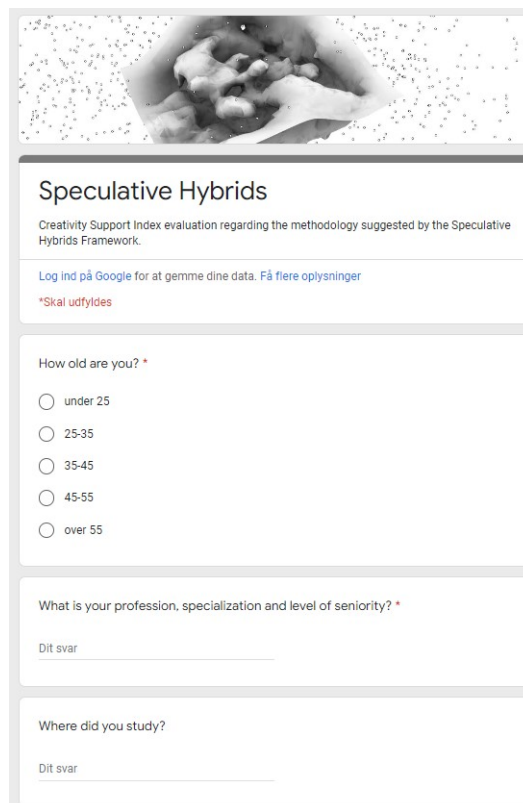
f_pc_1 = model.gen_from_given_z_and_pc0(z,pc_0)

# Save generated point cloud
pc_1_path = Path("./pc_1.pts")
with pc_1_path.open("w") as f:
    np.savetxt(f,f_pc_1[0].cpu().numpy())
```

Appendix B

CSI Metrics

B.1 Creative Support Index Questionnaire



The image shows a screenshot of a web-based questionnaire titled "Speculative Hybrids". At the top, there is a header image showing two hands holding a small, glowing, crystalline object. Below the header, the title "Speculative Hybrids" is displayed, followed by a subtitle: "Creativity Support Index evaluation regarding the methodology suggested by the Speculative Hybrids Framework." A link "Log ind på Google for at gemme dine data. Få flere oplysninger" is provided. A red asterisk indicates that the survey is anonymous. The first question is "How old are you? *", with radio button options: "under 25", "25-35", "35-45", "45-55", and "over 55". The second question is "What is your profession, specialization and level of seniority? *", with a text input field and a "Dit svar" label. The third question is "Where did you study?", also with a text input field and a "Dit svar" label.

Speculative Hybrids

Creativity Support Index evaluation regarding the methodology suggested by the Speculative Hybrids Framework.

[Log ind på Google for at gemme dine data. Få flere oplysninger](#)

*Skal udfyldes

How old are you? *

☐ under 25

☐ 25-35

☐ 35-45

☐ 45-55

☐ over 55

What is your profession, specialization and level of seniority? *

Dit svar

Where did you study?

Dit svar

Figure B.1: The first section of the questionnaire regarding demographic information.

CSI Evaluation

In the present research, we are introducing the framework Speculative Hybrids, which consists of a theoretical background regarding AI applications in Architecture, and the implementation of CPC-GAN in order to generate conceptual architecture massing models. CPC-GAN is an Adversarial Neural Network that can succeed in the controllable generation of point clouds from random latent noise. In this demonstration of the method, CPC-GAN was trained with a dataset of 2.904 simple building geometries in the form of point clouds and was used in order to generate 10 new building shapes. The newly generated buildings were controlled in order to map the building regulations of a specific site. Forward to the generation, the point clouds are imported in Rhinoceros 3d and converted into mesh with the Cockroach plugin.

Site Regulations

Plot area: 2.460 m2
Building height: 9.5m
Facade Height: 5.5m
Building percentage: 10%

Image of the building site




Figure B.2: The second section of the questionnaire, describing the method proposed by Speculative Hybrids, and the the regulations of the building site.

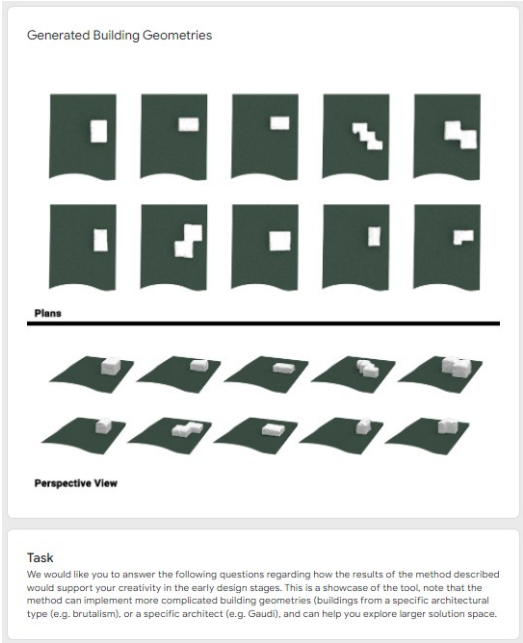


Figure B.3: The second section of the questionnaire, depicting the generations of CPCGAN on the chosen building site, and introducing the participants to their task.

It was easy to explore many different options, ideas, designs, or outcomes without a lot of repetitive interaction. *

1 2 3 4 5 6 7 8 9 10

Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree

The results of the method help in order to work better/easier while designing with others. *

1 2 3 4 5 6 7 8 9 10

Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree

I was very absorbed by the generated buildings. I enjoyed it and I would use other generations to design again. *

1 2 3 4 5 6 7 8 9 10

Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree

Figure B.4: The second section of the questionnaire, the questions regarding Exploration, Collaboration, and Engagement.

My design inspiration and solutions were worth the effort required to use such a method. *

1 2 3 4 5 6 7 8 9 10

Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree

The generation of the buildings is prior to my creative design process, therefore I was able to concentrate on the design activity. *

1 2 3 4 5 6 7 8 9 10

Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree

I was able to be very expressive and creative with chosen outputs. *

1 2 3 4 5 6 7 8 9 10

Strongly disagree ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Strongly agree

Figure B.5: The second section of the questionnaire, the questions regarding Effort/Reward Tradeoff, Tool Transparency, and Expressiveness.

B.2 Comparison Factors Questionnaire

Open-ended questions

Elaborate feedback on the proposed method

Which software or tools do you use in the conceptual design phase? *

Dit svar

How do you currently use technology in the ideation phase of the design process? *

Dit svar

What are the limitations of computational design tools? *

Dit svar

What are the ethical implications of using generative design tools? *

Dit svar

How the increasing use of computational design tools will help the future of design? *

Dit svar

Further feedback *

Dit svar

Figure B.6: The third section of the questionnaire, asking participants some open-ended questions.

Comparison between questions

Each one of the prior questions are describing creativity support within 6 categories: Exploration, Collaboration, Engagement, Effort/Reward Tradeoff, Tool Transparency, and Expressiveness. We would like you to compare each other towards the one that are more important to you for the method.

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Exploration

☐ Collaboration

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Exploration

☐ Engagement

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Exploration

☐ Effort/Reward Tradeoff

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Exploration

☐ Tool Transparency

Figure B.7: The questionnaire regarding the Comparison Factors.

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Exploration

☐ Expressiveness

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Collaboration

☐ Engagement

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Collaboration

☐ Effort/Reward Tradeoff

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Collaboration

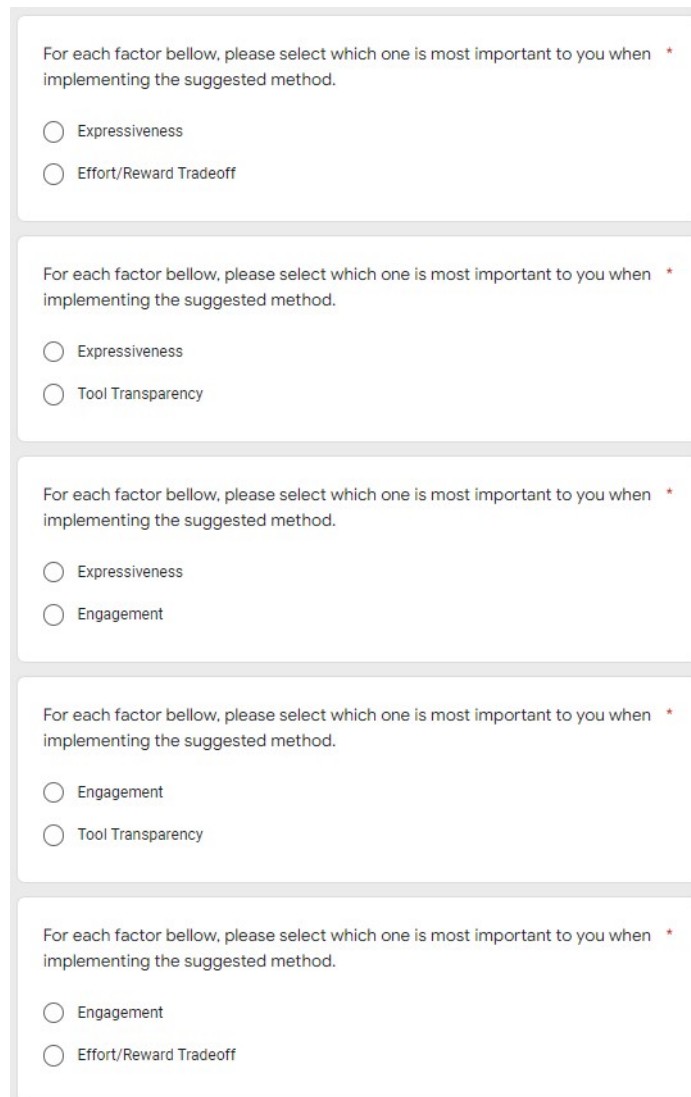
☐ Tool Transparency

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Collaboration

☐ Expressiveness

Figure B.8: The questionnaire regarding the Comparison Factors.



For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Expressiveness

☐ Effort/Reward Tradeoff

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Expressiveness

☐ Tool Transparency

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Expressiveness

☐ Engagement

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Engagement

☐ Tool Transparency

For each factor bellow, please select which one is most important to you when implementing the suggested method. *

☐ Engagement

☐ Effort/Reward Tradeoff

Figure B.9: The questionnaire regarding the Comparison Factors.