Diplomarbeit

TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Deep Gestures

a human-machine drawing conversation

ausgeführt zum Zwecke der Erlangung des akademischen Grades
einer Diplom-Ingenieurin
unter der Leitung von

Christian Kern
Univ.Prof.Arch.Dipl.Ing.
E264/2 Institut für Kunst und Gestaltung
Dreidimensionales Gestalten und Modellbau
eingereicht an der Technischen Universität Wien
Fakultät für Architektur und Raumplanung
von

Michaela Nömayr
01325588

Wien, am 3. März 2022

Abstract

This master thesis explores the space of possibilities offered by the convergence of novel Machine Learning techniques and established Numerical Fabrication technology, by focusing on applications in the context of the early stages of the design process.

By experimentally engaging with the making and the programming of an interactive drawing machine in its hardware and software components, this research offers an alternative interpretation to the praxis of sketching and of intuitively representing ideas in the two-dimensional space.

Moving away from the concept of computers as passive receivers of pre-conceived creative impulses, the machine is here framed as an active instrument which is capable of perceiving and processing external inputs. As the machine co-participates in the unfolding of the creative process, the role of the designer/user is necessarily re-defined.

Currently, applications of Machine Learning in the creative field are mostly focused on the processing of pixel-based images or texts. As part of the thesis, the potential of a Hierarchical Generative Network for the creation of vector files in the SVG format (DeepSVG) is investigated. As opposed to image creation in raster format, vector representation opens the door to digital production, since its properties allow for a direct translation to machine code and CNC operations.

The core of the thesis consists in a set of experiments investigating user-machine interaction for a simple, feedback-based, design process. The physical drawing of a user is fed to a neural network by means of computer vision techniques (skeleton tracing).

The pre-trained network generates new content by interpolating the user input with a selected item from a large library of vector icons, the SVG-Icons8 dataset. The set of newly machine-generated frames is filtered by the user, who chooses a single solution to be printed back in the physical space by a custom-made pen plotter. By engaging in this open-ended process of interaction, the user can build up further drawn responses to feed to the machine, which in turn will keep on generating new and unexpected content until the process is stopped according to user defined criteria.

Rather than solving a specific design problem, an experimental Machine Learning application is designed and explored. The main objective of this work is not looking for a single optimal solution, but it is the creative exploration of the unfolding of a hybrid process of collaboration.

Abstrakt

Diese Masterarbeit erforscht die Möglichkeiten, die sich aus der Konvergenz neuartiger Techniken des maschinellen Lernens und etablierter Technologien der numerischen Fabrikation ergeben, indem sie sich auf Anwendungen im Kontext der frühen Phasen des Designprozesses konzentriert.

Durch die experimentelle Auseinandersetzung mit der Fertigung und Programmierung einer interaktiven Zeichenmaschine in ihren Hardware- und Softwarekomponenten bietet diese Forschung eine alternative Interpretation der Praxis des Skizzierens und der intuitiven Darstellung von Ideen im zweidimensionalen Raum.

In Abkehr vom Konzept des Computers als dem passiven Empfänger von vorgefassten kreativen Impulsen wird die Maschine hier als aktives Instrument verstanden, das in der Lage ist, externe Eingaben wahrzunehmen und zu verarbeiten. Da die Maschine an der Entfaltung des kreativen Prozesses teilnimmt, wird die Rolle des Designers/Nutzers zwangsläufig neu definiert.

Derzeit konzentrieren sich Anwendungen des maschinellen Lernens im kreativen Bereich meist auf die Verarbeitung von pixel-basierten Bildern oder Texten. Im Rahmen dieser Arbeit wird das Potenzial eines Hierarchischen Generativen Netzes für die Erstellung von Vektordateien im SVG-Format (DeepSVG) untersucht. Im Gegensatz zur Bilderstellung im Rasterformat öffnet die Vektordarstellung die Tür zur digitalen Produktion, da ihre Eigenschaften eine direkte Übersetzung in Maschinencode und CNC-Operationen ermöglichen.

Der Kern der Arbeit besteht aus einer Reihe von Experimenten zur Untersuchung der Benutzer-Maschine-Interaktion für einen einfachen, Feedback-basierten Designprozess. Die physische Zeichnung eines Benutzers wird mit Hilfe von Computer-Vision-Techniken (durch Abrufen des topologischen

Skeletts) in ein neuronales Netz eingespeist. Das trainierte Netzwerk generiert neue Inhalte durch Interpolation der Benutzereingabe mit einem ausgewählten Element aus einer großen Bibliothek von Vektorsymbolen, dem SVG-Icons8-Datensatz. Die maschinell erzeugten Bilder werden vom Benutzer gefiltert, der eine einzige Lösung auswählt, die dann mit einem speziell angefertigten Stiftplotter in den physischen Raum gedruckt wird. Durch diesen ergebnisoffenen Interaktionsprozess kann der Benutzer weitere gezeichnete Antworten erstellen, die ihrerseits immer wieder neue und unerwartete Inhalte generiert, bis der Prozess nach benutzerdefinierten Kriterien gestoppt wird.

Anstatt ein spezifisches Designproblem zu lösen, wird eine experimentelle Anwendung für maschinelles Lernen entworfen und erforscht. Das Hauptziel dieser Arbeit ist nicht die Suche nach einer einzigen optimalen Lösung, sondern die kreative Erforschung der Entfaltung eines hybriden Prozesses der Zusammenarbeit.

Acknowledgements

As I regard this thesis as a result of a collaboration with the people in my own network, I want to thank them for their support:

First I have to thank Christian Kern for extensive support, helpful suggestions and critical feedback in the creation of this body of work.

I would particularly like to thank Marco Palma and Florian Rist, without whom this work would not have been possible in this depth and scope. Thank you for all the support, tireless explanations, inspiring talks, your most valuable feedback and all the things you taught me.

I also have to thank the team of the model building workshop - Walter Fritz, Kornelia Fischer, Ronald Buchinger, Lena Grünbauer, Johanna Kübert, Katja Puschnik, Ruben Mahler and Alexander Ortner - for providing me with a first class workplace, technical support, equipment, coffee and nice words to keep up motivation and work pace.

A special thank you goes to my friends and family for support, endless pep talks, valuable feedback and guidance through a demanding year.

Index

1 Premise 11

1.1 Motivation 12
1.2 Summary 20

2 Theoretical Framework 23

2.1 Neural Networks 25
2.1.1 Artificial Neural Network 25
2.1.2 Deep Learning 31
2.1.3 Generative Adversarial Networks 41
2.1.4 Hierarchical Generative Networks 47

2.2 Digital Fabrication 49
2.2.1 Digital Fabrication 51
2.2.2 Computational Fabrication 55
2.2.3 Adaptive Computational Fabrication 59

2.3 Machine Learning, Digital Fabrication Feedback 63

3 Implementation 73

3.1 Concept 75
3.1.1 Methodology 75
3.1.2 Human-Machine Interaction 77
3.1.3 One iteration 79
3.1.4 Interpolation 81
3.1.5 Design Process 83
3.1.6 Experimental Setup 85

3.2 Hardware 87
3.2.1 Assembled Hardware - Mechanics 88
3.2.2 Assembled Hardware - Electronics 93
3.2.3 Drawing and Webcam Setup 109
3.2.4 Acoustic insulation 113

3.3 Software 115
3.3.1 Software Environment, Language, Platform 115
3.3.2 Data Pipeline 117
3.3.3 Firmware 122
3.3.4 Communication - Interoperability 124

4 Application 131

4.1 Abstract (simple) experiments 133
4.2 Figurative experiments 139
4.3 Conversation between human and machine 153
4.4 Path order 167
4.5 Composition/pattern creation 181
4.6 Superimposed drawings 187
4.7 Current limitations 203
4.7.1 Hardware limitations 203
4.7.2 Software limitations 203

5 Conclusions 205

5.1 Further implementations 210

6 Bibliography 213

1 Premise

This chapter provides the motivation for the thesis as well as an overview in the form of a short summary of the chapters to come.

1.1 Motivation

"The true challenge to artificial intelligence proved to be solving the tasks that are easy for people to perform but hard for people to describe formally—problems that we solve intuitively, that feel automatic, like recognizing spoken words or faces in images." [1]

1 Goodfellow, Bengio and Courville, 2016, Chapter 1 - Introduction, page 1

"Designers struggle to sketch with a technology as invisible and intangible as ML. (...) Carnegie Mellon human-computer interaction professor John Zimmerman explains, "When I teach a designer, I can give them cardboard and say, 'Make a stool; make a hat.' (...) But I can't send somebody to play with machine learning like that." (...) Since we can't cut it and fold it like cardboard, how do we engage with ML as a design material? One method is to focus not on how the tech works, but rather on its capabilities." [2]

2 Armstrong and Dixon, 2021, page 24f

The motivation for this thesis project was the development of a generative process - addressing human-machine interaction and involving Machine Learning and Numerical Production methods - rather than the development of an optimal design solution to a well described quantifiable problem.

This generative process shall enhance the designer's agency in the early stages of the design workflow, specifically by addressing the intuitive two-dimensional representation of concepts and ideas. Sketches are one of the tools of choice for architects and designers when it comes to convey and communicate ideas during the conceptual stage.

Following the evolution of numerically controlled production machines from passive tools to active collaborators, the role of the designer has also evolved. The relationship between machine and designer today is an entirely different one than it was 40 years ago. Researchers and artists in the field of computational design and fabrication recently started to address machines as their per-

ceiving and sensing counterparts, leveraging their interactive capabilities as well as their role as instruments that can extend the designer's creative reach in the exploration of novel design possibilities.

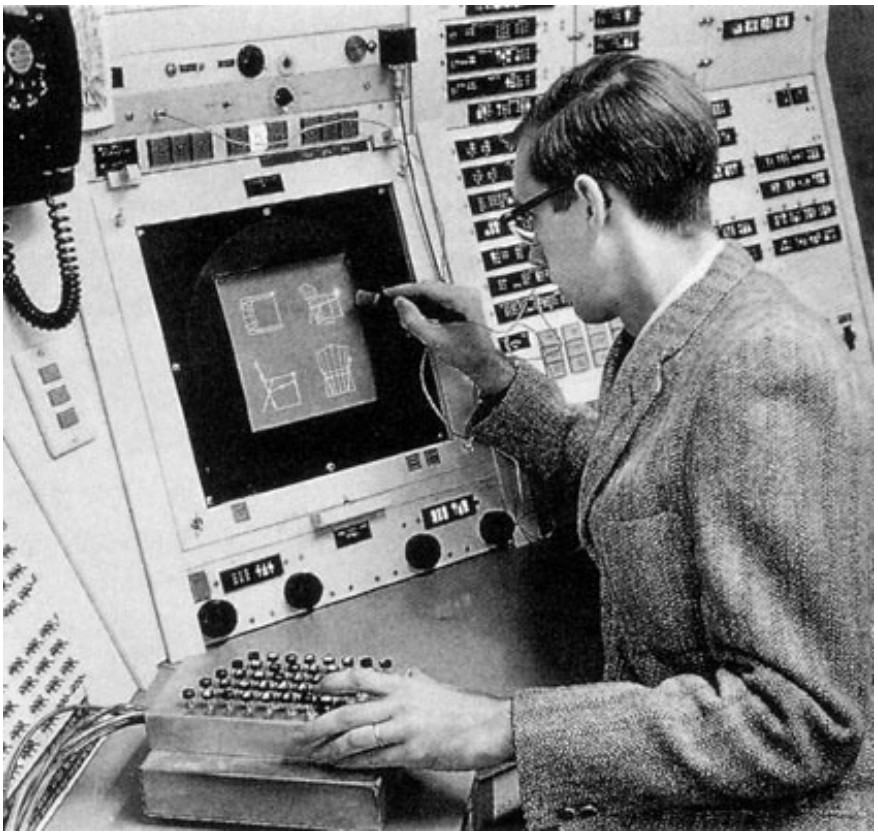


Figure 1.01:
Ivan Sutherland's
Sketchpad, 1962.

As part of his Phd. thesis, Ivan Sutherland developed Sketchpad, a precursor of modern CAD programs. Users were able to draw directly on a display using the Light Pen and a set of push buttons and turning knobs representing different geometrical operations. The results of the drawing process were plotted on paper using a connected pen plotter that acts as a passive receiver and produces the physical output of this feedback-based digital process.

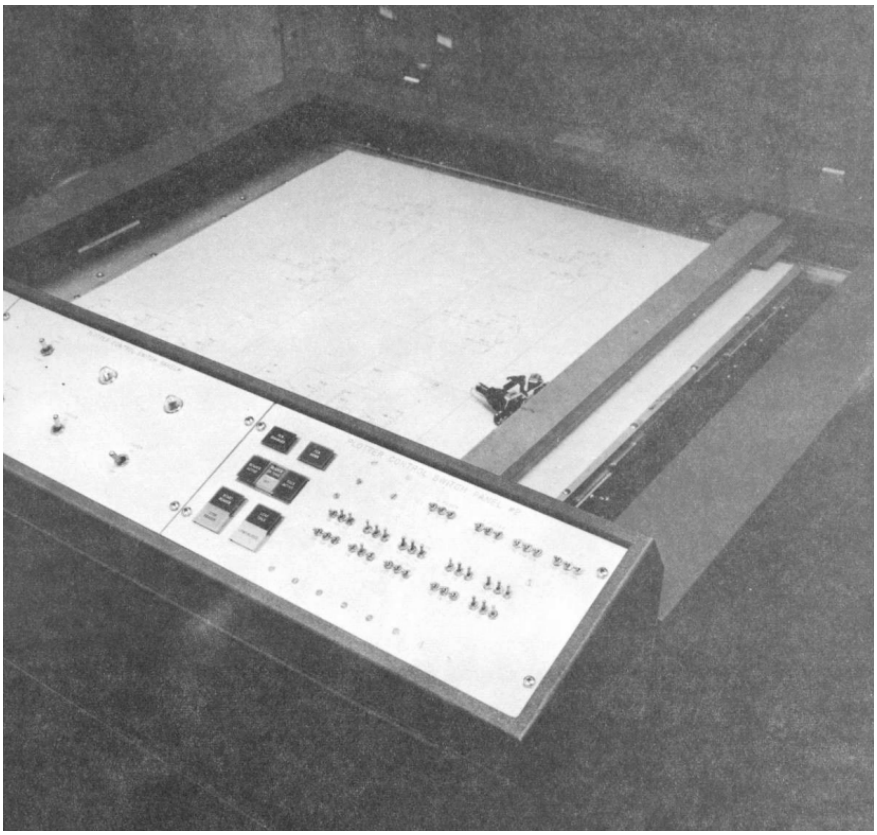


Figure 1.02:
Connected pen plotter
as passive receiver.

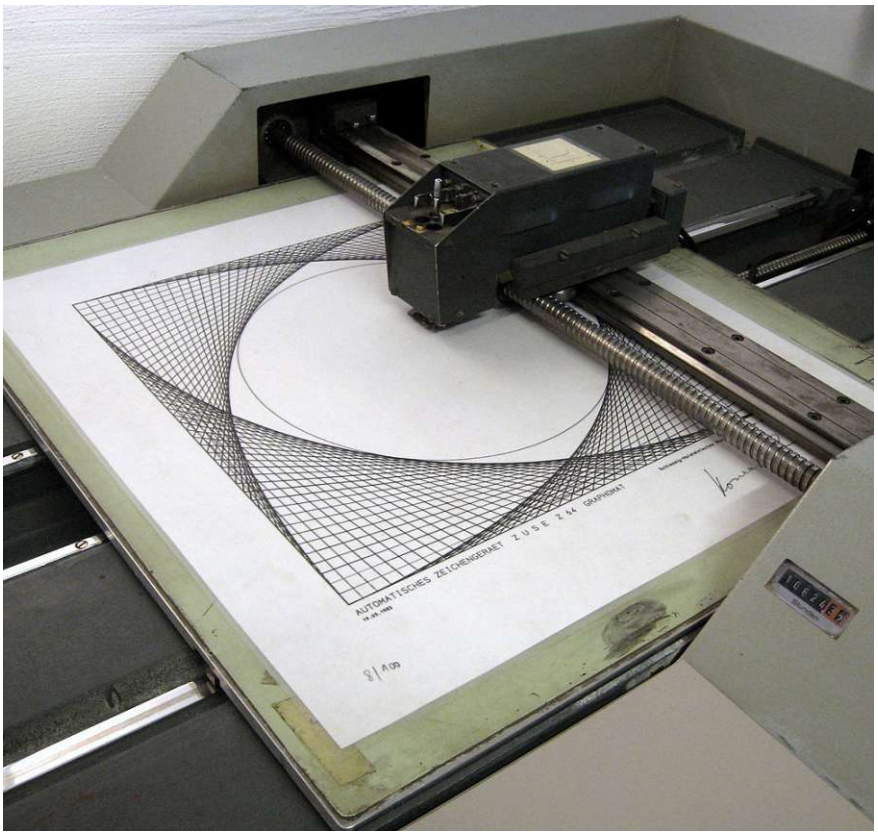
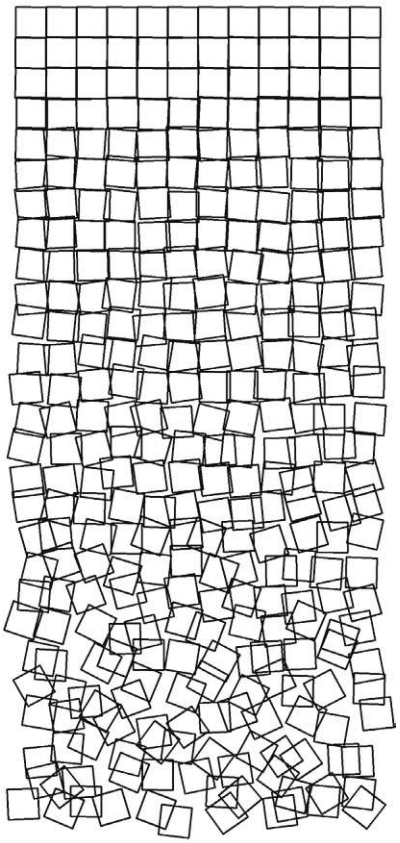


Figure 1.03:
Machine as Actuator.

The innovator of generative and computational art, Georg Nees, created his artworks in a fully mediated process. He used a programming interface which was connected to a pen plotter that still served as a passive receiver of coded instructions. Those were subsequently plotted directly onto a sheet of paper. The plots are the physical output of a linear digital process.



```
'BEGIN''COMMENT'SCHOTTER.,
'REAL'R,PIHALB,PI4T.,
'INTEGER'I.,
'PROCEDURE'QUAD., # generates the main figure, a square
'BEGIN'
'REALP1,Q1PSI.,'INTEGER'S.,
JE1.=5*I/264.,JA1.=-JE1., # random generator 1
JE2.=PI4T*(1+I/264)., # random generator 2
JA2.=PI4T*(1-I/264).,
P1.=P+5+J1.,Q1.=Q+5+J1.,# changing the coordinates
PSI.=J2.,
LEER(P1+R*COS(PSI),
Q1+R*SIN(PSI)).,
'FOR'S.=1'STEP'1'UNTIL'4'DO'
'BEGIN'PSI.=PSI+PIHALB.,
LINE(P1+R*COS(PSI),Q1+R*SIN(PSI))
'END'.,I.=I+1
'END'QUAD.,
R.=5*1.4142.,
PIHALB.=3.14159*.5.,PI4T.=PIHALB*.5.,
I.=0.,
SERIE(10.0,10.0,22,12,QUAD) # multiply the main figure
'END'SCHOTTER.,
```

Figure 1.04:
Schotter, 1968 (left).

Figure 1.05:
Code to Gravel written in the language ALGOL (right).

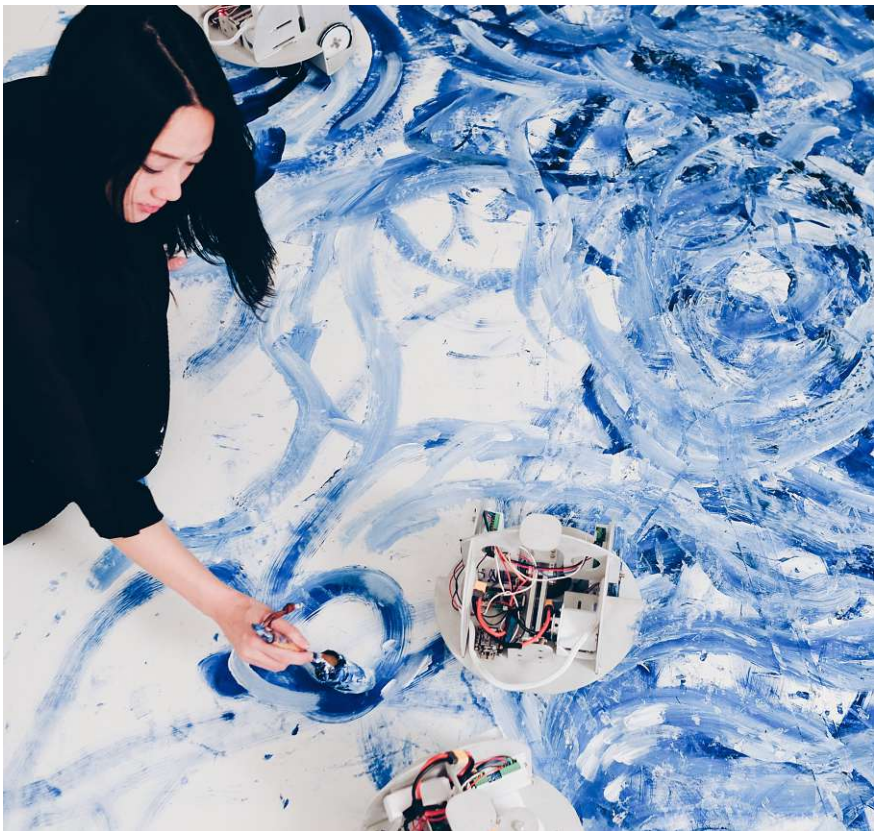


Figure 1.06:
Omnia per Omnia,
Sougwen Chung, 2018.

Sougwen Chung's series of collaborative art is based on AI systems that have been trained on data from her own work of art. The series culminated in a performance on a collective scale - a mutual drawing process of the artist and a set of twenty custom robots. The physical interface posing a room-sized horizontal, blank canvas is enhanced with sensing equipment. The drawing collective serves as sensors and actuators of a digital and physical feedback process. The robots receive digital input from camera feeds in New York City. While moving they are analyzing their position within the collective. While drawing they are miming a collective urban movement.

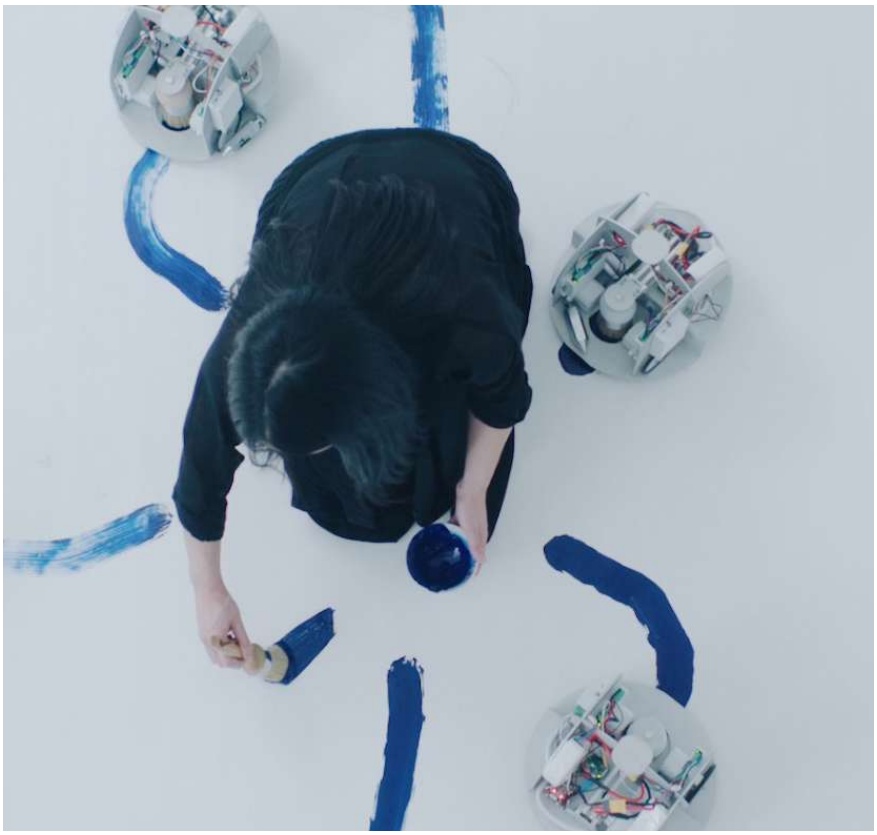


Figure 1.07:
Machines as Collaborators.

The tools or instruments that artists and designers use have a considerable impact on the user itself as well as on the process of creation. In recent years, numerically controlled machines have evolved from passive receivers to active, sensing collaborators. Simultaneously to that evolution, the relation between artist and their instrument of choice has changed as well as the output or work of art itself.

1.2 Summary

The following chapter **Theoretical Framework** provides a short introduction to the theory of Neural Networks and illustrates meaningful related applications in Digital Fabrication.

The chapter **Implementation** includes the concept for the experiments as well as a detailed description of the assembled hardware and implemented software.

In the chapter **Application**, visual and textual documentation of the experiments is presented and confronted with the current limitations of both code and electro-mechanical apparatus.

The final chapter presents the conclusions drawn from the experimental results and illustrates ideas for further developments and implementations.

2 Theoretical Framework

The following chapter provides the theoretical background for the following Implementation chapter. It gives an introduction to the theory of Artificial Neural Networks and illustrates meaningful related applications in Digital Fabrication.

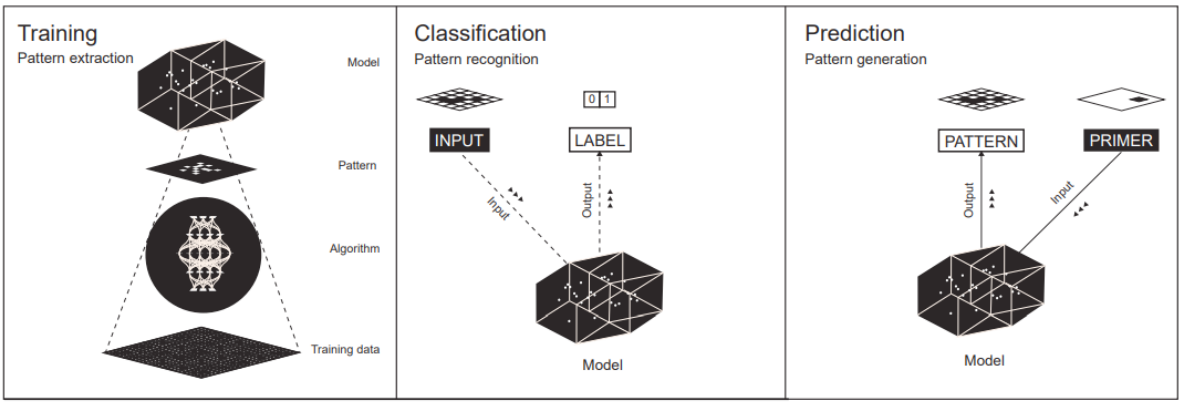


Figure 2.01:
Applications of Machine Learning
The Nooscope Manifest
Pasquinelli and Joler, 2020

2.1 Neural Networks

2.1.1 Artificial Neural Network

"As an instrument of knowledge, machine learning is composed of an object to be observed (training dataset), an instrument of observation (learning algorithm) and a final representation (statistical model)." [1]

1 Pasquinelli and Joler, 2020, page 3

ANNs are simplified models of the human brain consisting of a multitude of connected neural cells - the neurons. The neurons are linked between each other via their synapses. They pass electrical impulses from sending cells to receiving cells.

As in the natural model, the basic element of an artificial neural network is also called a neuron. The artificial neuron is defined by a mathematical function. On its activation the neuron does not receive electrical impulses but a numerical, initial value from the connected, sending neuron. The solution of the neuron's inherent mathematical function is passed onto the next artificial, receiving neuron.

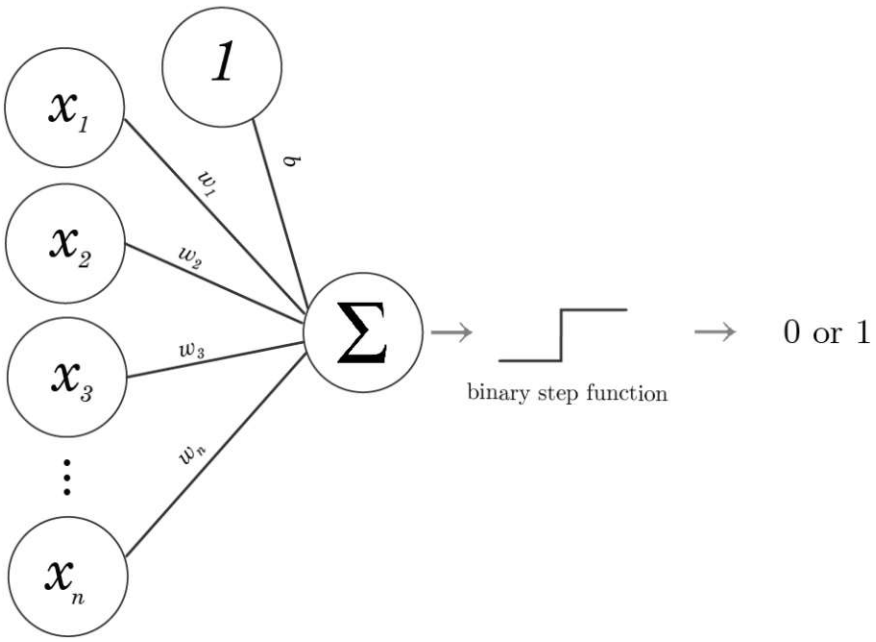


Figure 2.02: Scheme of a Perceptron.

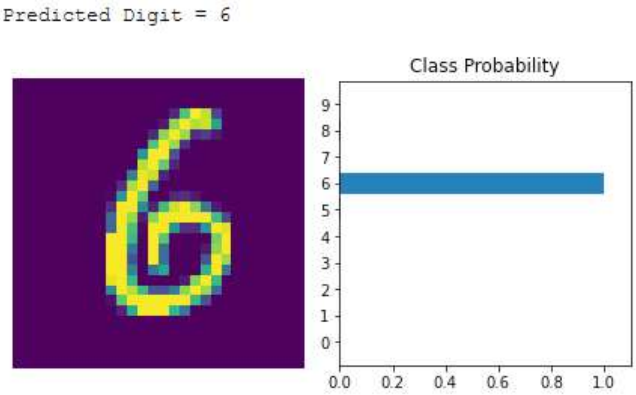
The Perceptron is a simplified artificial neural network consisting of a single neuron, adjustable weights and a threshold value. Introduced by Frank Rosenblatt in the late 1950s, it was the first type of artificial neuron and could be implemented for linear classification problems.

ANNs are usually constructed in layers. There are initial start layers that receive information. As briefly described above, information is thus passed from those start layers over a series of hidden layers to the output layers in one forward pass.

The training process of ANNs can be compared to any other natural learning process. Based on the principle of trial and error, different connections between neurons are strengthened or weakened. What stories and real-life situations represent for a human brain are the training and test dataset for an artificial neural network. One of the basic tasks for an ANN is solving a classification problem. Its application ranges from the detection of spam emails to cancer diagnosis. The network can, for instance, classify a tumor as benign or malicious. What happens will be made visible through the example of handwritten digit recognition in this context.

Both training and test dataset contain data that is paired with a list of the solution for this data. A popular example of datasets consists of photos of handwritten digits in Bitmap format, the MNIST dataset. It consists of 60,000 training images from 0 to 9 and 10,000 testing images. Each image is itemized as a matrix of 28x28 cells, each containing a grayscale pixel value.[2]

2 LeCun, Cortes and Burges, 2022



Successful classification of a handwritten digit.

The information of a single image - RGB values pixel per pixel - is passed through the network. After confronting the network with the right solution, all the weights are adjusted in a backward pass. The weights represent the connections between the neurons. The network's weights are fixed during the training process. The network is subsequently tested to accuracy using the test dataset.

Weights and the topology of a trained model can be saved and transferred. One talks about a learning transfer in this context.

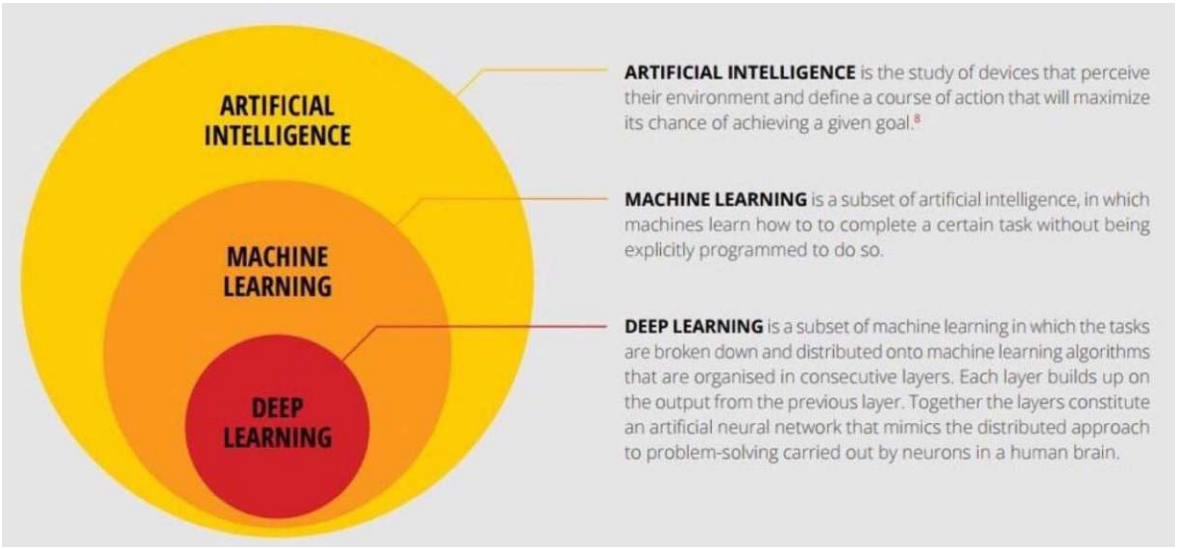


Figure 2.03: Classification of Deep Learning as a sub-category of Machine Learning.

2.1.2 Deep Learning

"(...) machine learning algorithms are the most powerful algorithms for information compression." [3]

"This solution is to allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined through its relation to simpler concepts. (...) The hierarchy of concepts enables the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning." [4]

Deep Learning poses one discipline of Machine Learning (see figure 2.03) and applies artificial neural networks as an essential component. While machine learning algorithms were constructed following mathematical logic, the neural networks that are applied in deep learning have been constructed after the natural model of the brain.

3 Pasquinelli and Joler, 2020, page 2

4 Goodfellow, Bengio and Courville, 2016, Chapter 1, Introduction, page 1f

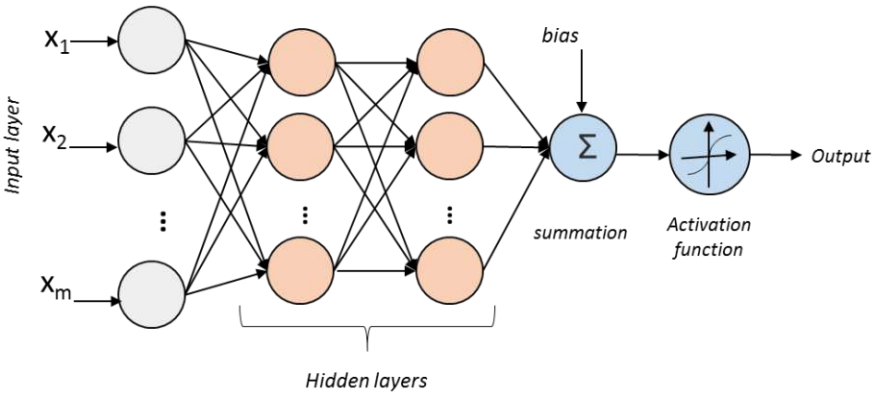


Figure 2.04:
Scheme of a Multilayer Perceptron.

The Multilayer Perceptron or Feedforward Neural Network represents a simplified artificial network for Deep Learning. It is a derivative of the Perceptron that was introduced with respect to Machine Learning. Its multiple layers removed the computational restrictions that the Perceptron was subject to. Another difference to the Perceptron is its non-linear activation. The activation function is determining if the output of one neuron equals 0 or 1. The MLP forms the basis of many important applications like Convolutional Neural Networks.[5]

5 Goodfellow et al., 2016, p. 164f

The model structure for deep learning algorithms is defined by multiple hidden layers between the input and output layer. The advantage of using artificial neural networks lies in the in-depth abstraction of correlations between input data and between the abstracted neural values and the output data.

Convolutional Neural Networks are specialized in image classification. Their structure is based on those of conventional neural networks. They dispose over an additional convolutional and pooling layer and can handle feature engineering more efficiently.

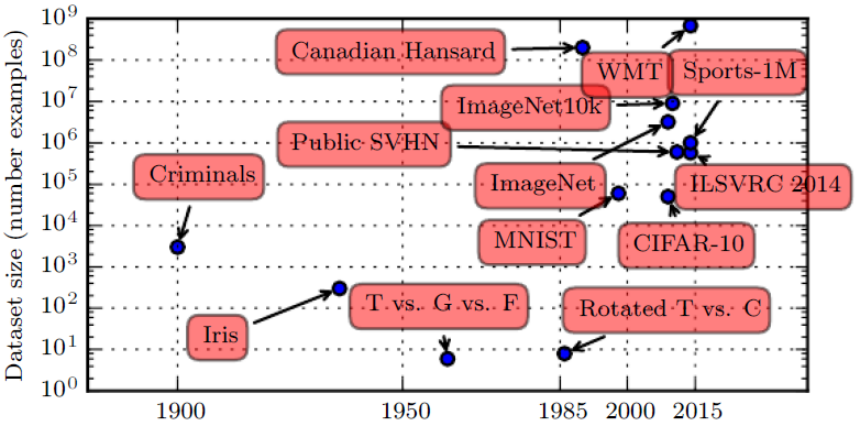


Figure 2.05:
Data size benchmark.

The concept of Deep Learning is not new: What is today referred to as Deep Learning (starting in 2006), has been the object of research as Connectionism in the 1980s and 1990s and as Cybernetics in the 1940s to 1960s.[6] One of the reasons why deep learning has become popular in recent years are advances in digital electronics resulting in increased computational performance of computers. Because computers can process a lot of information faster, it is possible to train deep neural networks in a decent amount of time.

6 Goodfellow et al., 2016, p. 12f

Computational speed is not the only resource needed: The Age of "Big Data" that is a consequence of the increasing "digitization of society" [7], that eventuates in large amounts of data that can be curated as datasets for DL applications. Figure 2.05 shows the benchmark of sizes of datasets over time.

7 Goodfellow et al., 2016, p. 19f

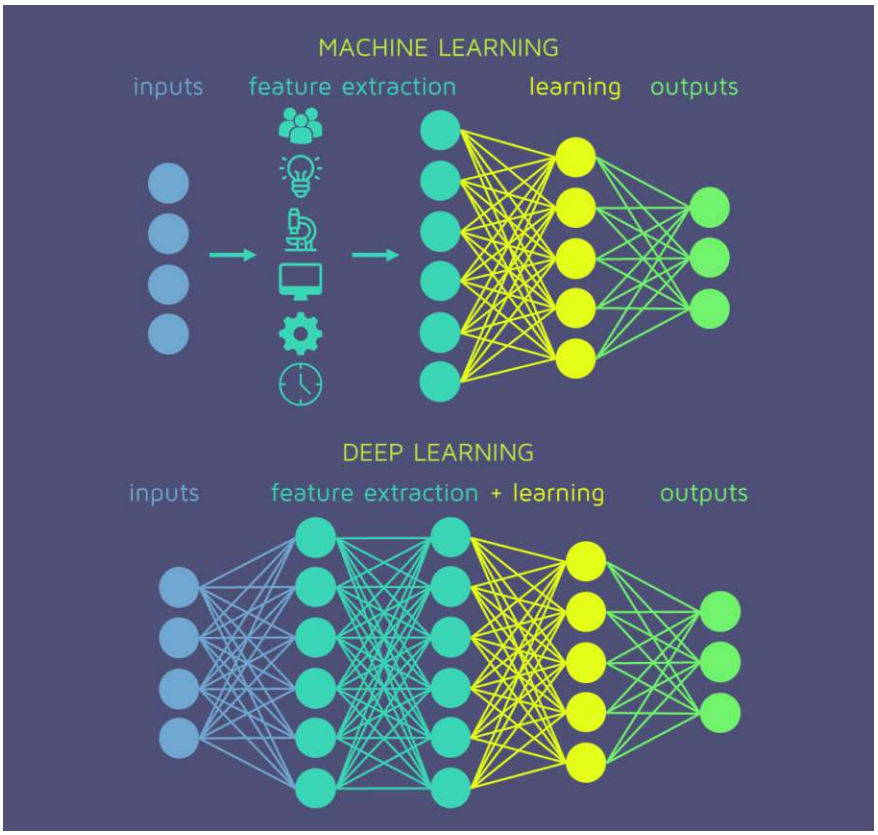


Figure 2.06: Different feature engineering processes for ML and DL.

Deep Learning models perform well with large amounts of data and are applied, when conventional machine learning models cannot improve after reaching a saturation point. They are also preferably used, when separate Feature Extraction might be waived. *"(...) a feature is any measurable input that can be used in a predictive model."*[8] While conventional ML models depend on manual feature engineering, ANNs can compress a multitude of input dimensions to their features because of their inherent structure. Feature extraction is essential for classification problems.

8 Patel, 2021

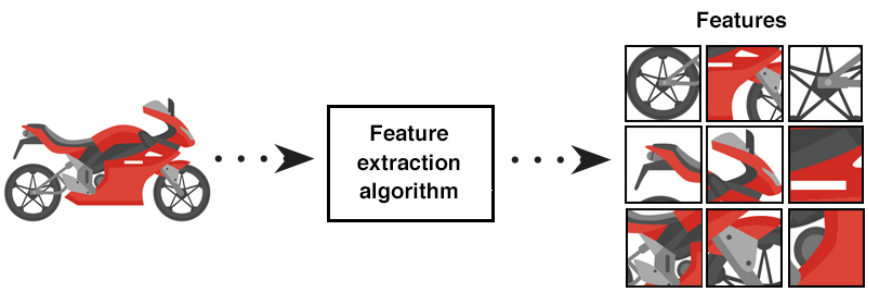


Figure 2.07: Schematic diagram of a feature extraction process.

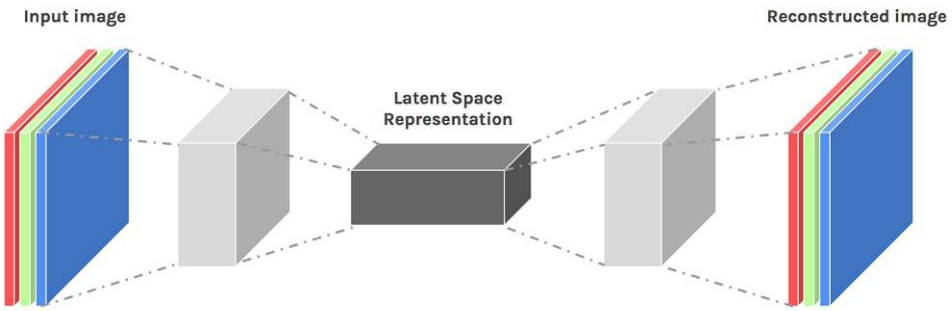


Figure 2.08:
Scheme of a Deep
Autoencoder.

At the moment, most artificial neural networks are applied for supervised learning, meaning regression or classification problems. Supervised learning requires labeled datasets and has a clearly predefined goal for the application of the algorithm.[9] Deep Autoencoders are applied for unsupervised learning problems. They are neural nets that reduce a large amount of data in input dimensions to a small number of dimensions. The reduction or encoding process is done in several steps, the reduced dimensions become feature vectors. During the decoding process, the dimensions are enlarged in several steps resulting in an abstract and reconstructed model of the original input (image). Abstract similarity models are created in this way.

9 Armstrong, 2021, page 154

2.1.3 Generative Adversarial Networks

"We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. (...) In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. (...) Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples." [10]

10 Goodfellow et al., 2014

"Neurography is short for "neural photography". It is the way of working with neurally generated images, since it is as much a process of search and discovery as it is one of creation. Like a photographer who goes into the world, discovers an interesting motif and then looks for the best ways to frame and capture it, I go into the multidimensional latent spaces of my GANs on the hunt for interestingness. The difference to classical photography is that by training my own GANs I can create and explore new and different worlds every time and am not limited to the single one we are all living in." [11]

11 Klingemann, 2019

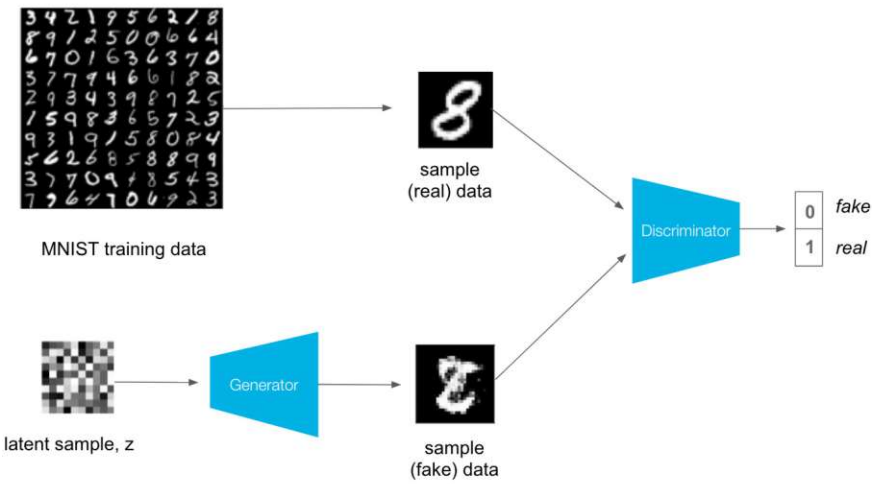


Figure 2.09:
The GAN Pipeline.

GANs were first developed in 2014 by Ian Goodfellow.[12] Two networks are working against each other: The Generator takes the latent sample z as input and learns to create fake images. The Discriminator is nothing but a traditional classifier. It uses a set of training data and compares it to the output of the Generator. The first few times, the Generator will not be able to fool the Discriminator into telling that his image is the training image. It will detect the fake image. After training, the Generator will get better and it will be more and more difficult for the Discriminator to tell which is the fake and which is the training image. In figure 2.09 the generation of new data based on the MNIST dataset using a GAN can be seen. There are GANs for several applications:

12 Goodfellow et al., 2014

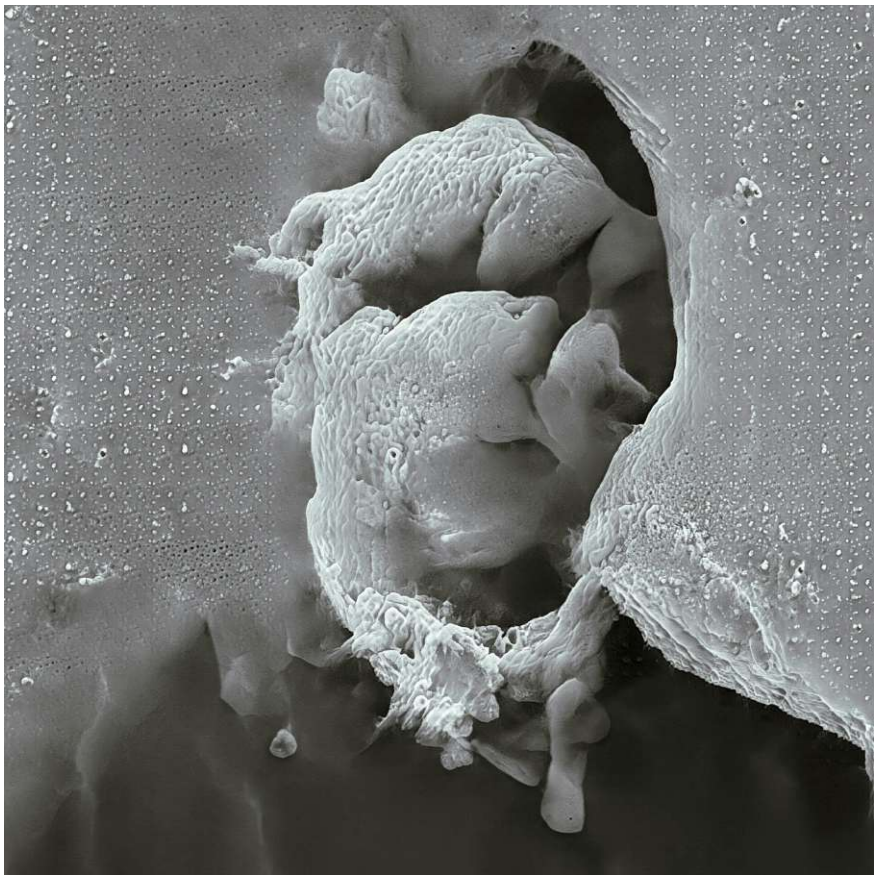


Figure 2.10:
"Nice scarf! Knitted it yourself?", 2017

Generative Art by Mario Klingemann using Generative Adversarial Networks.

The Pix2Pix network transforms Sketches to photorealistic images. CycleGAN performs style transfer and has the advantage that it does not require paired training data needed (unsupervised learning).

Artists like Mario Klingemann are interested in applying GANs for their works for their generative aspect.

"At the moment I am completely focused on making a new installation called "Circuit Training" for the show at the Barbican. It is definitely my biggest project to date. What I am trying to create is a machine that makes the art people want (or deserve, depending on how you read it). Being confined to an exhibition space the only material it can work with are the visitors themselves. It will work by continuously collecting images of people and using them to train data for models from which it will try to generate "art". This art will then be judged by the audience and from the feedback it gets it will try to learn which aesthetics, motifs, styles or compositions people prefer to see and adjust its training and classification processes accordingly. So, you could say that the whole system is a big GAN with humans being the data and the discriminators." [13]

13 Interview with Mario Klingemann, March 2019

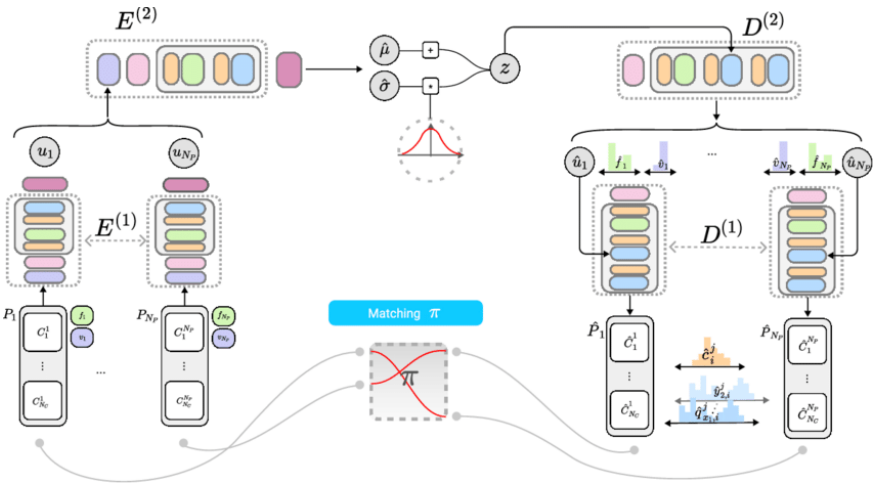


Figure 2.11: Schematic diagram of a Hierarchical Generative Network.

2.1.4 Hierarchical Generative Networks

In the following, a Hierarchical Generative Network architecture is explained using an example:

In figure 2.11, there is a schematic representation of a Hierarchical Generative Network. The network, that is by definition a variational auto-encoder, is constructed out of two main parts: an encoder network on the left side E and a decoder network on the right side D. The hierarchical representation of an SVG file is the base of both networks, meaning that each SVG file consists of multiple paths which are a description of commands written in XML.[14]

14 Carlier et al., 2020, page 4f

Every path of an SVG file is encoded separately using a path encoder $E^{(1)}$. $D^{(2)}$ contains the input of the latent space z and distributes it into blocks of information. A Multi-Layer Perceptron predicts SVG properties such as paths, attributes and visibility. The output of the MLP is decoded, retaining the symmetry of the network.

Figure 2.12: The 3D-printed stainless steel bridge was designed by Joris Laarman and built by MX3D using six-axis robotic arms equipped with welding gear. (Amsterdam, 2021)

2.2 Digital Fabrication

The evolution of fabrication technologies started with the implementation of CAD and CAM software for Digital Fabrication. The next step in this development was the use of integrated CAD-to-CAM processes for Computational Fabrication. At the moment this technology culminates at Adaptive Computational Fabrication which is using CAD-to-CAM technology incorporating feedback processes.

"Computer-Assisted Conception and Fabrication (CFAO) systems have two types of use in industry. [...]. CFAO systems have certainly increased the productivity of the idea, but fundamentally they offer no advances over the work done by hand. Now, we can envisage a second-generation of systems in which objects are no longer designed but calculated. The use of parametric functions opens two great possibilities for us. [...], these second-generation systems lay the foundation for a non-standard mode of production. In fact, the modification of calculation parameters allows the manufacture of a different shape for each object in the same series. Thus, unique objects are produced industrially." [15]

15 Cache, 1995

2.2.1 Digital Fabrication

"The MIT system combines digital and analogue processes under feedback control to govern a milling machine whose cutting tool moves in three planes relative to the work piece." [16]

16 Pease, 1952, page 109

The first documented example of digital fabrication is An Automatic Machine Tool.

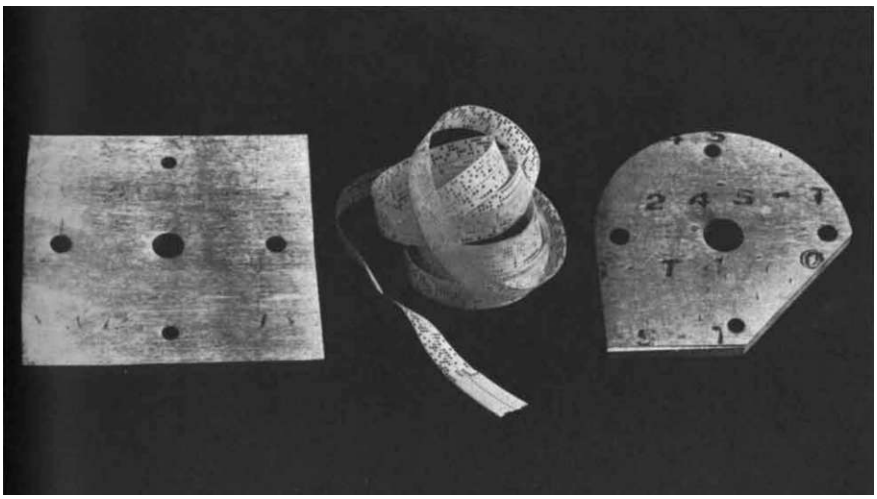


Figure 2.13:
Machine instructions were encoded on punch tape

The machine and its control system was developed by the MIT Servomechanisms Laboratory under the guidance of William Pease. In figure 2.15 the finished workpiece can be seen (on the right side). The machine instructions to move and control the cutting tool as well as the machine tabele were encoded on punch tape. The functioning system of a *"continuous-path numerically-controlled milling machine"*[17] was first presented in 1952.

17 Institute Archives, MIT Libraries, 2004

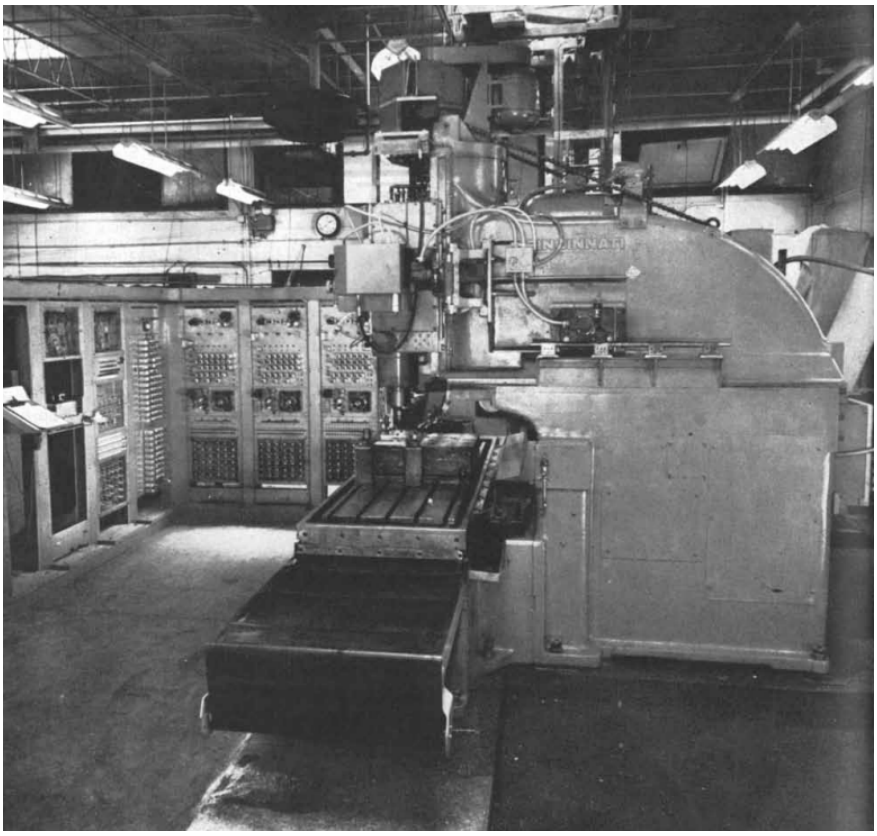


Figure 2.14:
An Automatic Machine Tool, Pease, 1952.

In this example, the representation was partially mediated by CAD software. The translation of the drawing to machine instructions was partially mediated by CAM software. The fabrication process was fully mediated by the CNC machine.

2.2.2 Computational Fabrication

"As the robot could be driven directly by the design data, without our having to produce additional implementation drawings, we were able to work on the design of the facade up to the very last minute before starting production." [18]

18 Gramazio & Kohler, 2009, page 90

An example of digital fabrication is the facade of the Winery Gantenbein in Switzerland.

Figure 2.15:
Interior view of the finished brick facade of the Winery Gantenbein by Gramazio & Kohler, 2006.

The project was designed by Bearth & Deplazes Architects and realized at the research facilities of the ETH Zürich under the guidance of Prof. Fabio Gramazio and Prof. Matthias Kohler. 72 individual facade elements were prefabricated in an automated process. The application of the two-component bonding agent that had to be applied to each brick in a different position and area as each brick had a different rotation posed one additional difficulty.

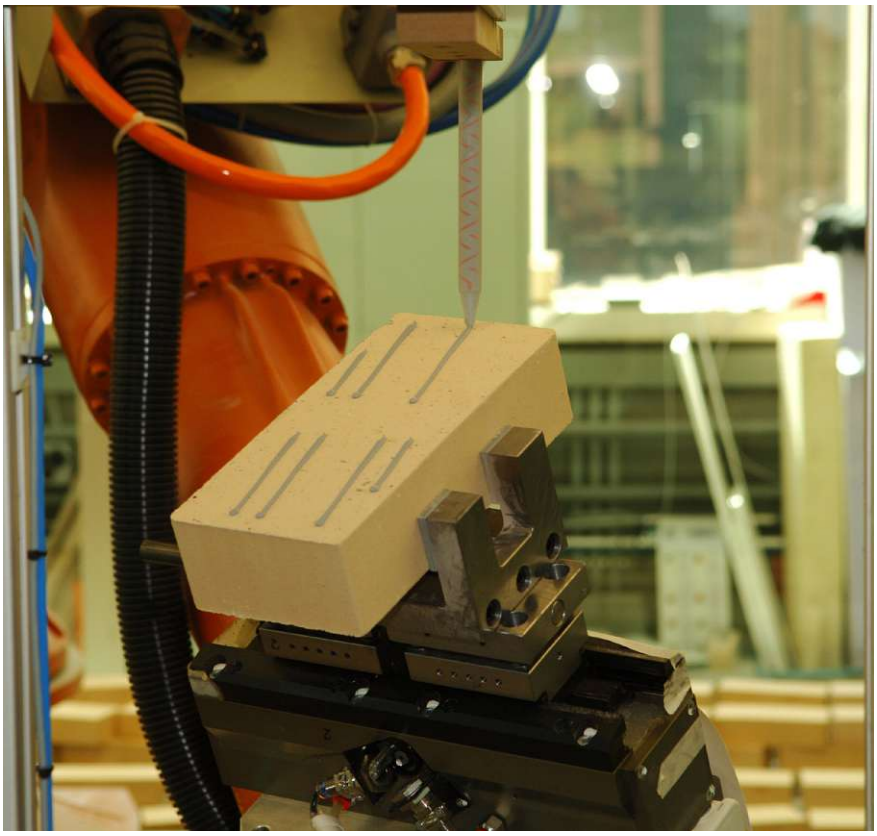


Figure 2.16: Automated robotic fabrication.

In this project, the representation was fully mediated by programming the CAD software. The translation of the drawing to machine instructions was fully mediated by programming the CAM software. The fabrication process was fully mediated by instructing the CNC machine.

2.2.4 Adaptive Computational Fabrication

"In order for a robot to build structures on the construction site with high accuracy, it needs to be able to track the position of the tool it is using with respect to some fixed reference frame. (...) These developments are broken up into three main functional parts: robot localisation within the construction site, alignment between the sensing reference frame and the CAD model, and feedback of the building accuracy during construction." [19]

19 Giftthaler, Sandy, Dörfler et al., 2017, page 4

The In-Situ Fabricator is an example of digital fabrication incorporating sensing equipment.

Figure 2.17:
In-Situ Fabricator,
Giftthaler, Sandy, Dörfler
et al., 2015.

The construction of a head-high, wave-like brick wall was realized using the In-Situ Fabricator, a mobile and autonomous robot. During the autonomous process of stacking discrete building elements, the Fabricator had to reposition itself which also required self-location and environment scanning. Visual sensing poses a prerequisite to these new advances of on-site digital fabrication. Before the start of the fabrication process, a 3D ambient scan was used to adjust the parametric model to the real dimensions of the building site.

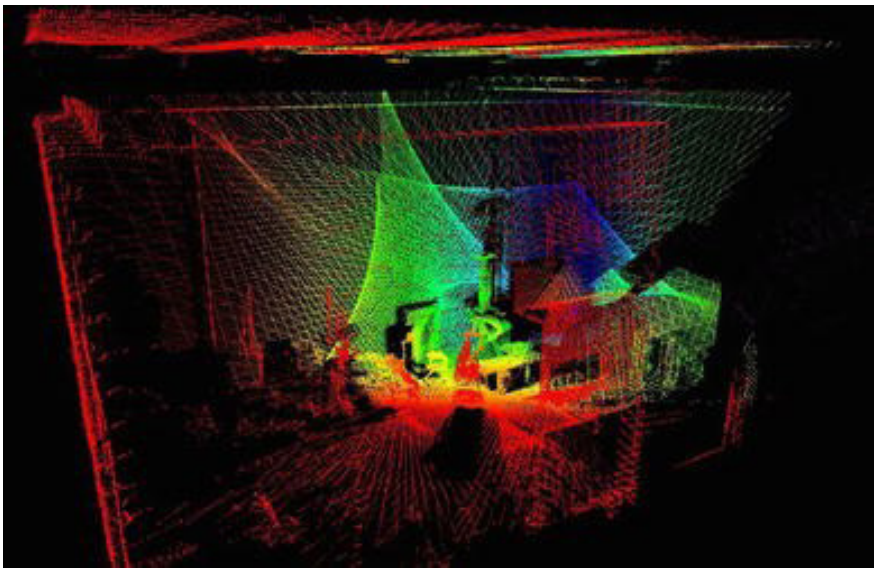


Figure 2.18:
Ambient point cloud.

In this example, sensing was fully mediated by programming a sensing device. The translation of the drawing to machine instructions was fully mediated by reprogramming the CAD software according to sensed information. Preparation was fully mediated by programming the CAM software. The fabrication process was fully mediated by instructing the CNC machine.

Figure 2.19:
SEEK, Negroponte and
The Architecture Ma-
chine Group, 1970.

2.3 Machine Learning, Digital Fabrication, Feedback

"Today, we are witnessing the transition from automated to autonomous systems of production. Increased access to advanced sensing, machine learning (ML), and artificial intelligence (AI) techniques are augmenting existing machines with new levels of understanding for their surroundings. Now, instead of being restricted to short, repetitive, pre-programmed tasks, fabrication machines are gaining the ability to dynamically see and respond to their changing environment without the need for human involvement." [20]

20 Gannon, 2018, page 14

The use of Machine Learning or Deep Learning algorithms in connection with Numerically Controlled machines in a creative context has the most illustrious precedents in the research produced by the Architecture Machine Group ast MIT under the guidance of Nicholas Negroponte between 1967 and 1985.

In most recent years, Machine Learning and Digital Fabrication have been widely used in significant artistic and research-based applications: three projects addressing automated feedback processes are presented in the following.

Figure 2.20: Mimus' installation at the Design Museum in London, Gannon, 2017.

In her work on Mimus, Madeline Gannon explores the use of an industrial robotic arm behaving like a living, breathing, mechanical creature. Mimus was shown as an installation at the Design Museum in London in 2017. It was able to engage and communicate with its visitors in body language and gestures showing danger, curiosity or empathy on a frequency only human beings could perceive. A belt of motion-tracking cameras provided visual feedback about

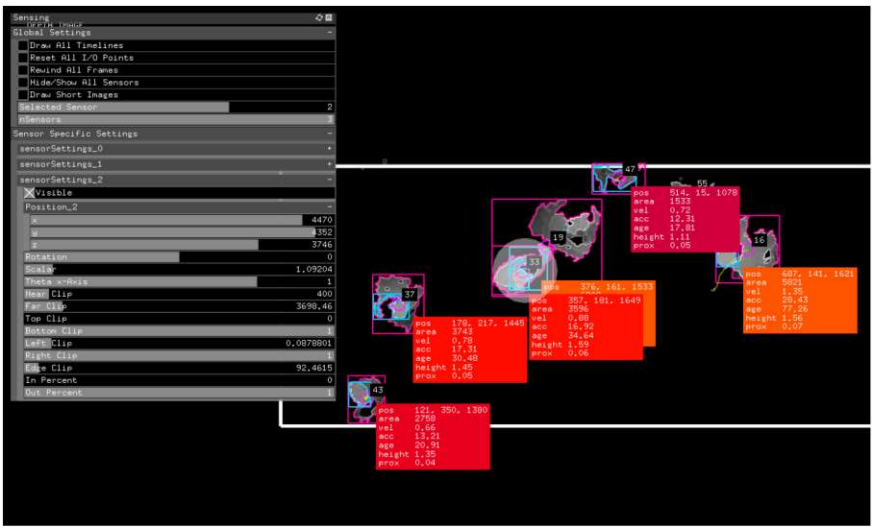


Figure 2.21:
Visual feedback of the visitors.

the visitors who could approach the attentive machine as far as a security glass fencing allowed it. Gannon proposed this kind of natural and human-centered language interface for autonomous fabrication machines to promote a more natural and spontaneous way of communicating.[21]

21 Gannon, 2018, page 150f

Figure 2.22:
Stochastic assembly,
Wu and Kilian, 2018.

In their paper, Wu and Kilian introduce a method for assembling naturally grown wooden blocks autonomously incorporating an industrial robotic arm and deep learning. Past applications are based on predefined geometric models as design goals for the finished structure. Using their approach, it is not necessary to define the geometry of the material before the beginning of the fabrication. An implemented and trained network based on CNNs is used to make suggestions for positioning the elements iteratively as well as it provides information regarding topological changes from potential connections as well as local assembly constraints.[22]

22 Wu and Kilian, 2018

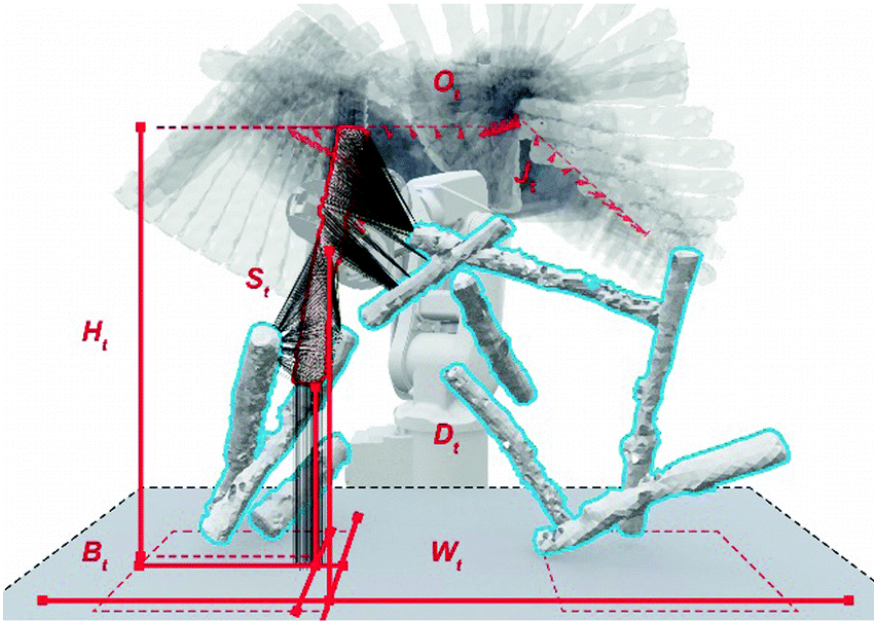


Figure 2.23:
Positioning of wooden logs using an industrial robotic arm.

Figure 2.24: NORAA - an interactive drawing installation was created by Jessica In in 2018.

NORAA [Machinic Doodles] is an interactive drawing installation created by Jessica In and her team that enables collaboration between human and machine. Being a customized pen plotter, NORAA is trying to understand the visitors' drawings and learning how to draw. NORAA is based on Magenta's Sketch-RNN (Magenta, SketchRNN, 2018), a network that was trained on Google's Quick Draw Dataset incorporating labeled sketches in 345 catego-

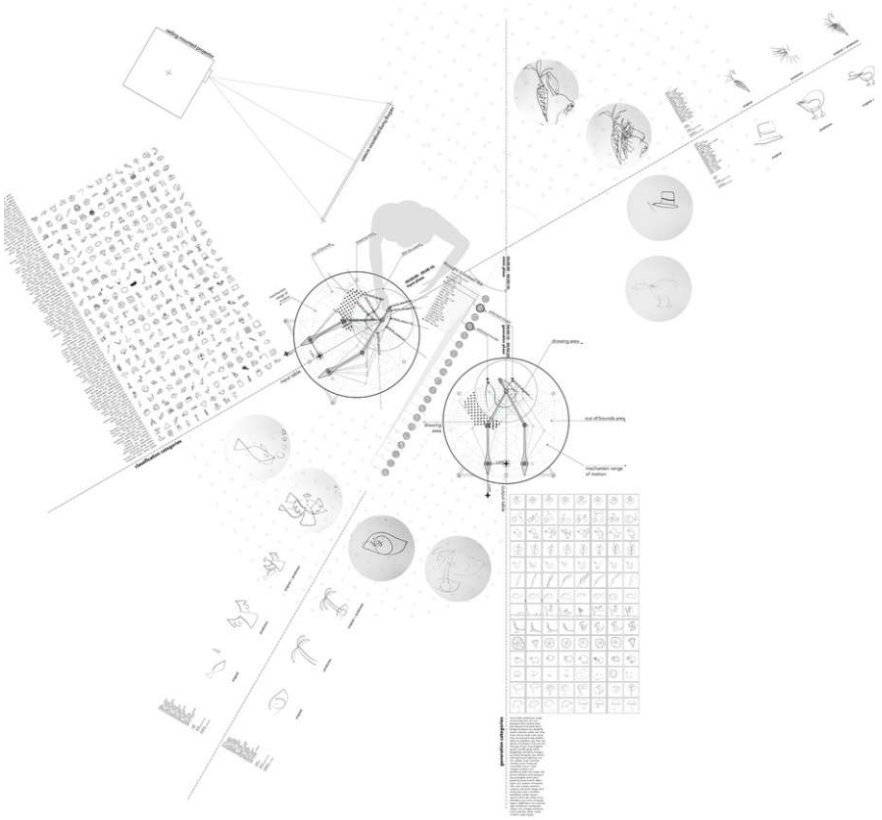


Figure 2.25:
Schematic sketch of installation concept.

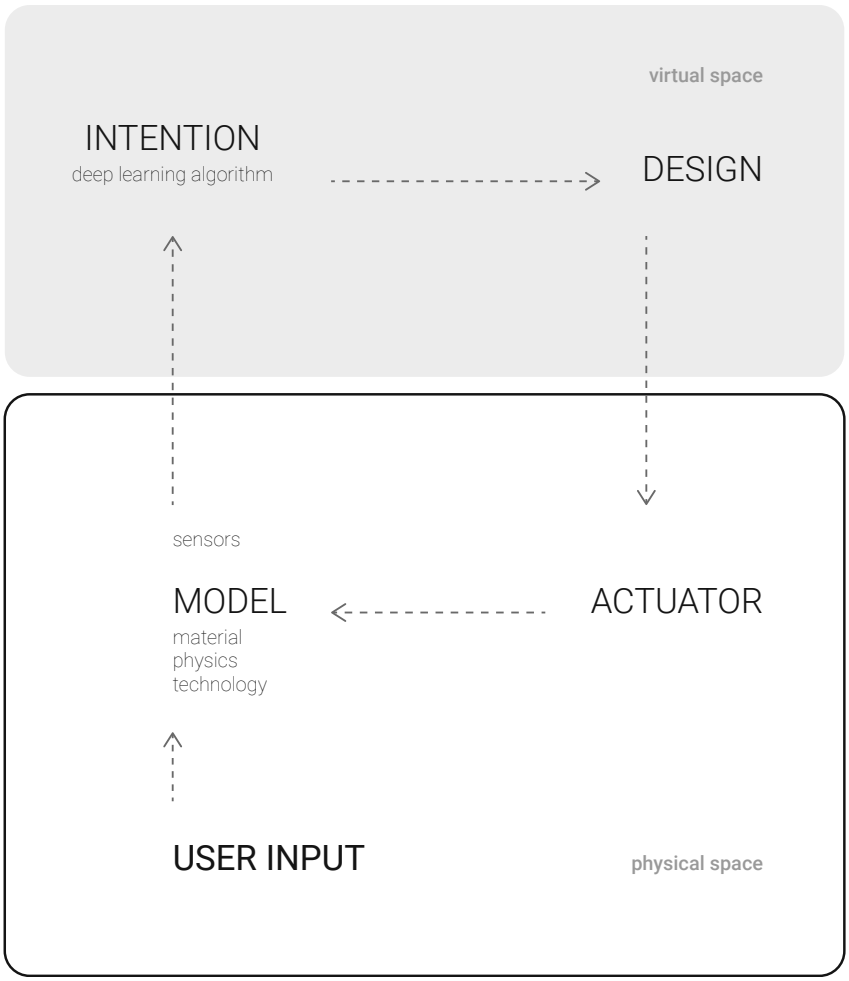
ries as time-stamped vectors (GoogleCreativeLab, Quick Draw Dataset, 2017). If NORAA recognizes the doodle it is confronted with, it will complete the drawing, if not, it hallucinates - meaning that it draws an answer in the form of a sketch that is closest or similar to what the visitor has drawn.[23]

23 In, 2018

Sketching and any comparable form of human intuitive activity poses a challenge for machine learning scientists. In the following, user-machine interaction incorporating sensing, Numerically Controlled Machines and Deep Learning will be investigated.

3 Implementation

The following chapter includes a conceptual and technical description of the project. Starting with concept and methodology, it covers the technical background to both assembled hardware and software and ends with a depiction of current limitations of code and electro-mechanical apparatus, respectively.



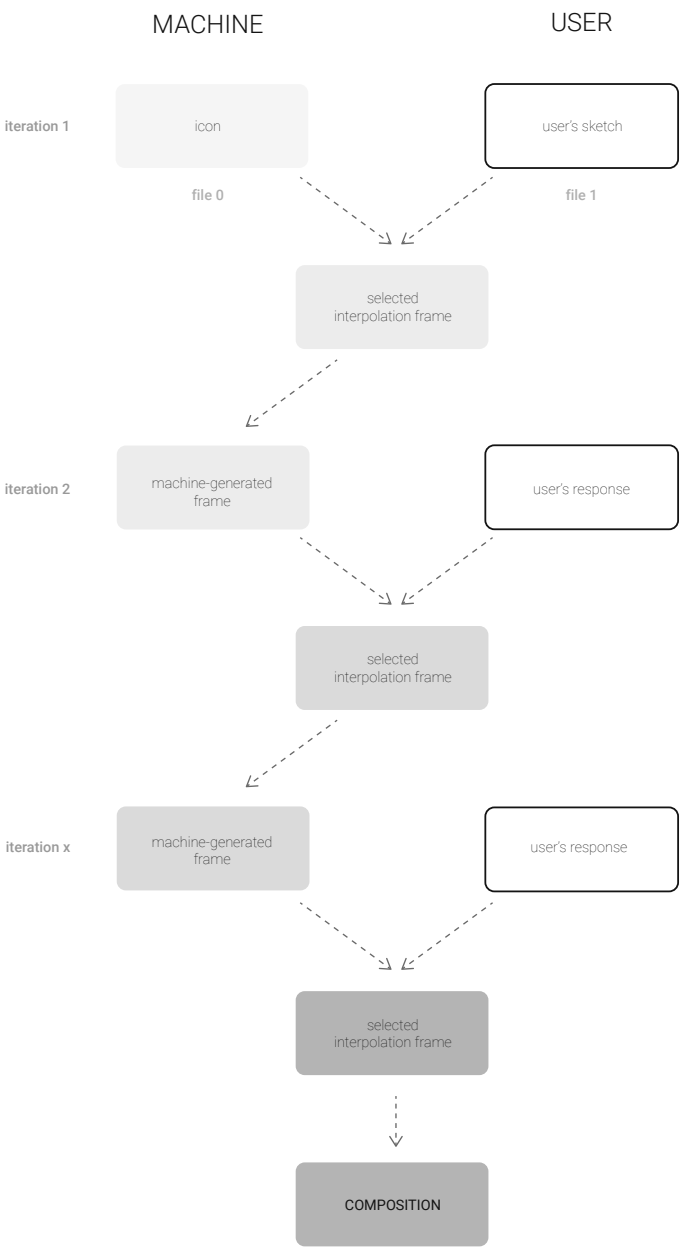
3.1 Concept

3.1.1 Methodology

As part of the thesis, the potential of a Hierarchical Generative Network for the creation of vector files in the SVG format (DeepSVG) is investigated. The core of the thesis is composed of a set of experiments investigating user-machine interaction for a simple, feedback-based, design process. The physical drawing of a user is fed to a neural network by means of computer vision techniques (by retrieving the topological skeleton). The pre-trained network generates new content by interpolating the user input with a selected item from a large library of vector icons, the SVG-Icons8 dataset. The set of newly machine-generated frames is filtered by the user, who chooses a single solution to be printed back in the physical space by a custom-made pen plotter. By engaging in this open-ended process of interaction, the user can build up further drawn responses to feed to the machine, which in turn will keep on generating new and unexpected content until the process is stopped according to user defined criteria.

The model in the diagram on the right depicts a design process that takes place in virtual and physical space. An actuator receives instructions that were designed in the virtual space, but is influenced by forces that are inherent to the physical such as the design intention.

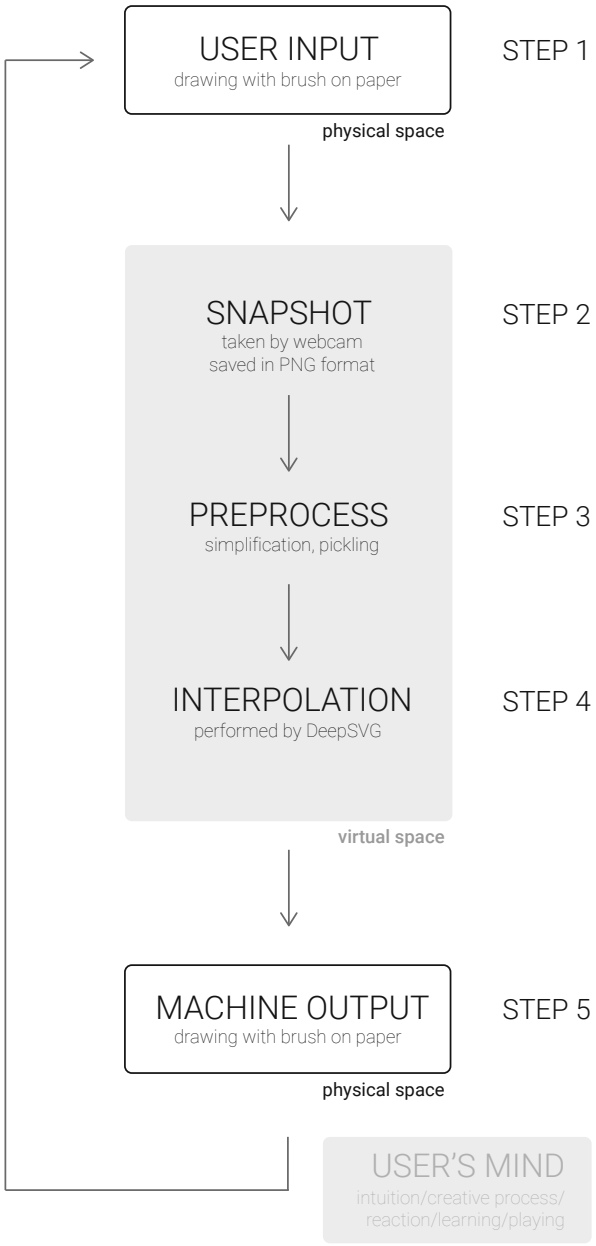
Opposite page:
Design process taking place both in virtual and physical space.



3.1.2 Human-Machine Interaction

Machine and user communicate in an iterative process. The conversation is started in Iteration I with an icon selected by the machine as File 0 and is replied with a sketch hand-drawn by the user as File 1. The machine generates a set of new frames which interpolate between File 0 and File 1. The user chooses one of the results of the interpolation as the new input for Iteration II on the machine side. The process is repeated in this manner until the user is satisfied with the output and decides to conclude the conversation at Iteration x. The last frame serves as input for a small program that generates a composition.

Opposite page:
Schematics of User-Machine Conversation.



3.1.3 One iteration

Step 1
The user draws a sketch consisting of up to eight strokes on the right half of the canvas.

Step 2
On pressing the Take snapshot button on the user interface, a webcam installed above the drawing space, is taking a photo. As the canvas is made of rewritable cloth used for calligraphy, the user's sketch, drawn with a brush filled with distilled water, will disappear within moments after the snapshot is taken.

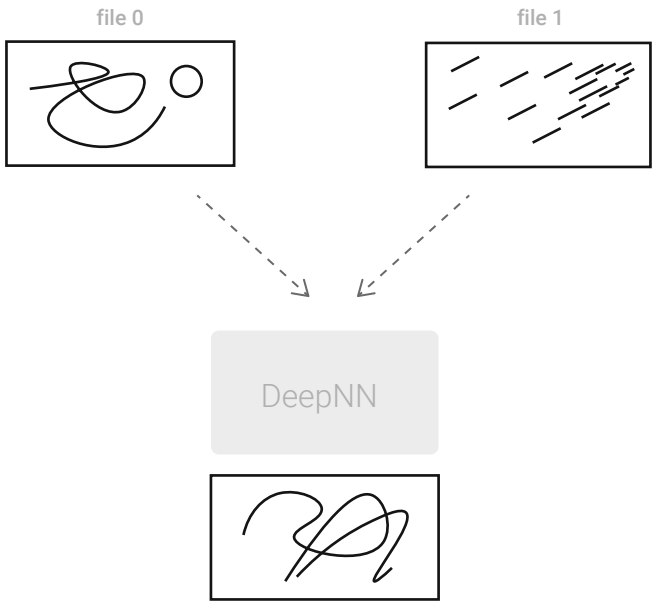
Step 3
The paths on the snapshot in Bitmap format are translated into vectors and saved in the SVG format. After preprocessing, the file is output in PKL format, which serves as input for the interpolation performed by the network.

Step 4
The interpolation is performed between the taken snapshot of the user's sketch and the output of the previous iteration. In iteration I, an icon of the dataset is used as the second value for the interpolation.

Step 5
The user is able to browse through 25 interpolation steps displayed on the computer screen by moving a slider on the user interface. In order to select one of the results, the Draw selection button has to be pressed on the interface. The SVG file gets translated into machine instructions and the machine draws the chosen step of the interpolation onto the left side sheet of magic paper. For a moment both, the machine's and the user's drawing will be present next to one another until they fade.

Step 6 / Step 1
The user starts the process again by drawing a response to the machine drawing and taking a snapshot of it.

Opposite page:
Schematics of information flow occurring in a single iteratio
of the user-machine conversation.



Interpolation performed by a pretrained Deep Neural Network.

3.1.4 Interpolation

Input
Two files in SVG format, as Variant 0 and 1, serve as the input for the interpolation. In order to be able to be processed by the network, they have to be preprocessed in two steps. The output of the second preprocessing step is a file in PKL format.

Interpolation
The interpolation is performed by a pretrained Deep Neural Network, the DeepSVG. The network was trained based on the SVG-Icons8 dataset.

Output
The interpolation process outputs 25 images in SVG format that lie between the two input values.

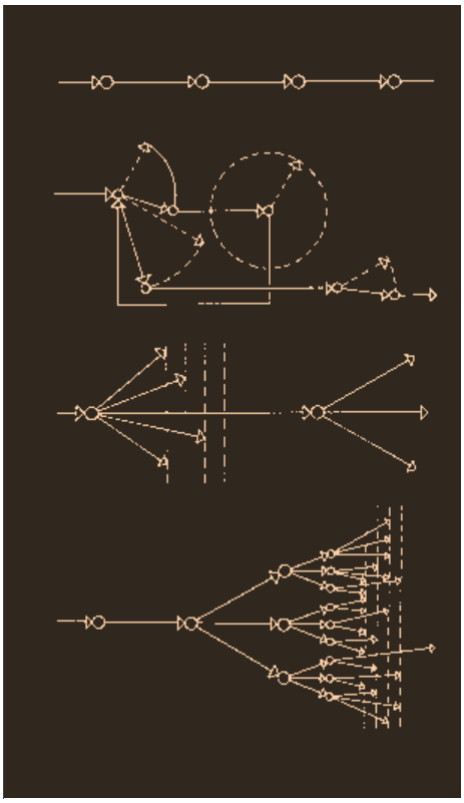


Figure 3.01:
Horst Rittel's design processes of generating variety and reducing variety:

a) linear sequence
b) testing or scanning
c) systematic production of several alternative approaches
d) forming alternatives in a multi-step process

3.1.5 Design process

"Design processes can be described in terms of sequences of different forms depending on the level of observation. Horst Rittel's diagrams show four basic possibilities for structuring the design process. He calls designing an "iterative process of generating variety and reducing variety". (Rittel 1992, pp. 75 ff.) The individual iterative steps could be seen as a circular, constantly recurring sequence of the work stages described above, which take the form of a spiral curve.
(...)
If further work shows that this approach is not going to produce the desired result, then the designer goes back to the beginning and tries a different route to the solution. Rittel calls the formation of alternatives the systematic production of several alternative approaches to a solution and the selection of the best one using an evaluation filter that covers all relevant aspects. It can also be done by forming alternatives in a multi-step process. In order to eliminate as many nonsensical alternatives as possible from the outset, Rittel recommends working with constraints (self-imposed limitations) for this process, which help to cut down the variety of possible design solutions to a sensible and manageable number."[1]

1 Gänshirt, 2021, page 80

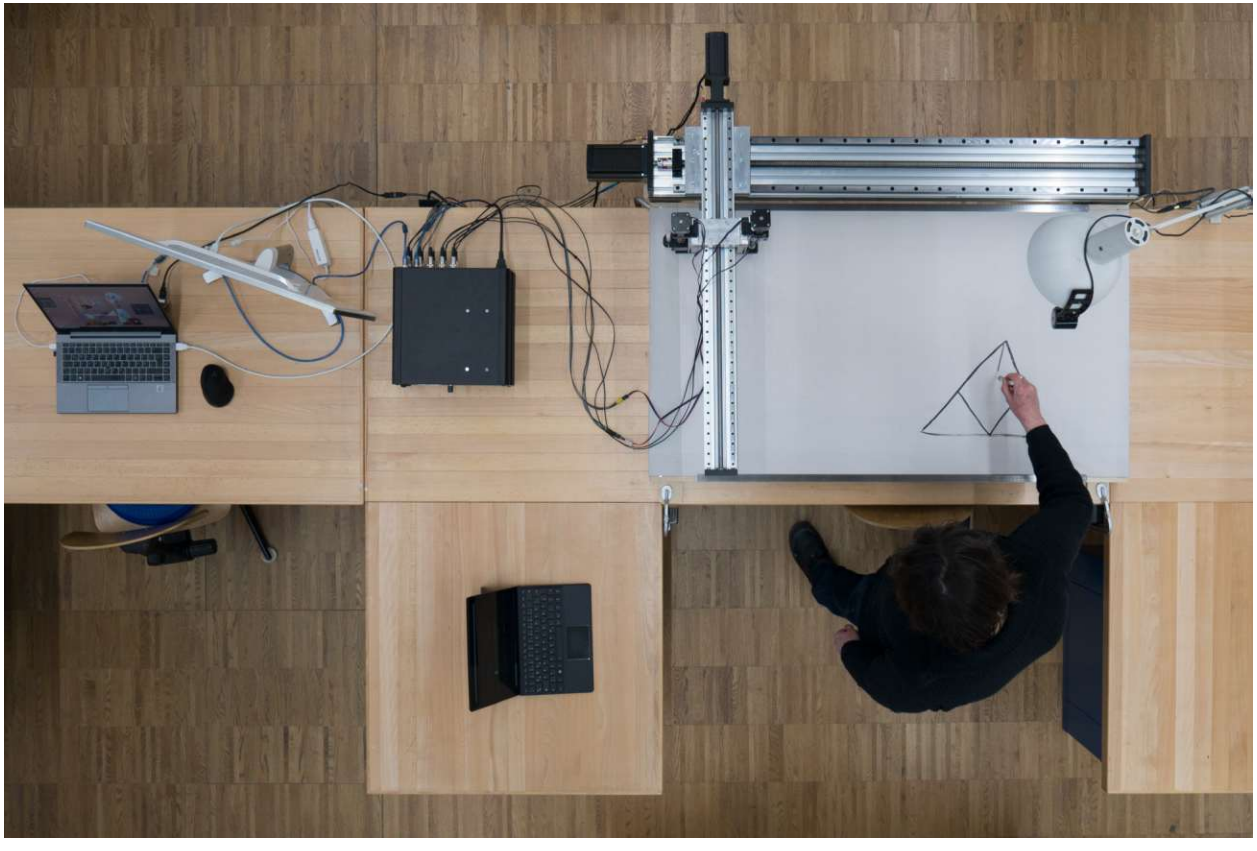
Horst Rittel describes the design process as an iterative process which coincides with the ideas mentioned above and reinforces the ideas of this project.

Different forms of this iterative process are visually displayed in figure 3.01: *"the linear sequence, testing or scanning, a systematic production of several alternative approaches and the forming of alternatives in a multi-step process"*. The third form describes this work best, it generates alternatives at each iteration.

3.1.6 Experimental Setup

The experimental setup consists of a machine equipped with drawing and photography setup, a computer that sends information to the machine's controlling unit, a Deep Neural Network that performs an interpolation based on the information that is fed in and a graphical user interface that enables user interaction. The machine, as the essence of the setup, is custom-built and therefore documented in detail in this chapter.

To ensure communication between all components, custom programs have been mostly written in the Python language and are supported by the Visual Studio Code platform.



3.2 Hardware

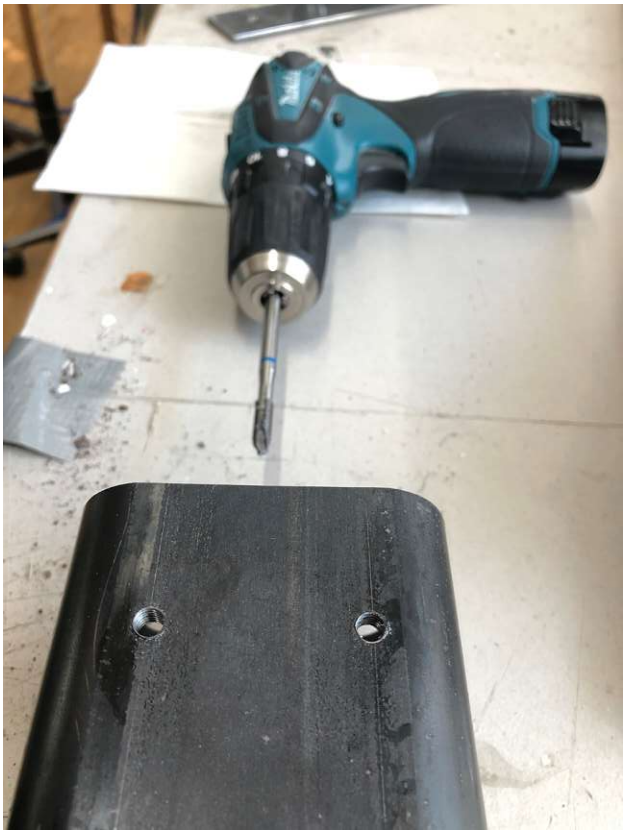
The following chapter describes the hardware that was used in the building process of the pen plotter that is the basis for the set of experiments that was carried out and is described in the next chapter.

Oppisite page:
Top view of the machine setup.

Drilling holes into the connecting pieces.

3.2.1 Assembled Hardware - Mechanics

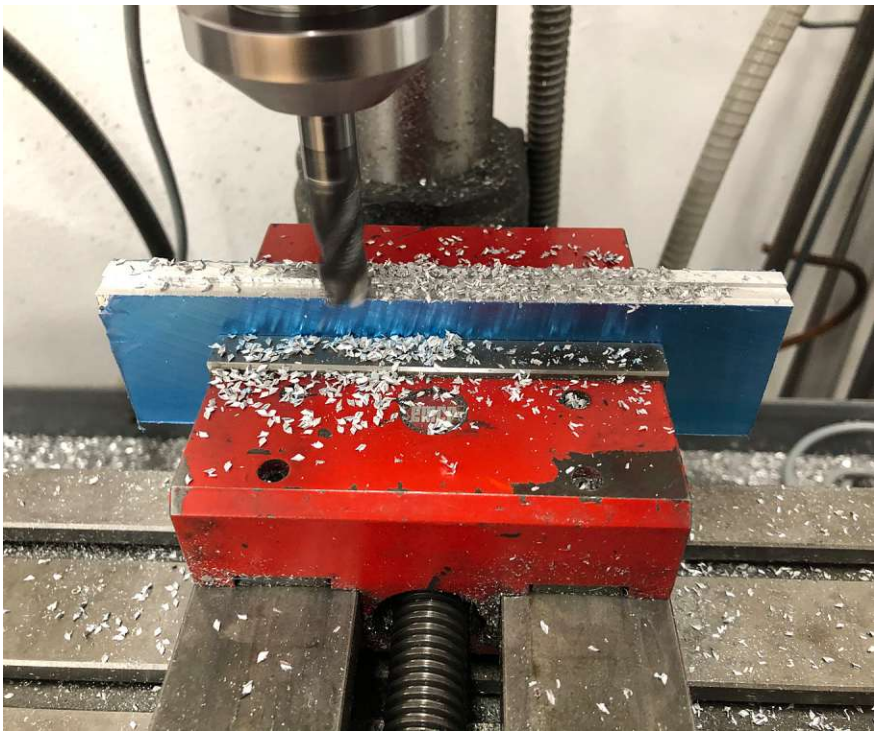
The mechanics of the assembled hardware can be itself divided into two parts: the base that is built out of connected steel tubes and the four linear axes that are assembled with aluminum connectors.



Tapping threads using a thread-cutting tap.

Base

The customized pen plotter is mounted on a wooden desk, but it is designed in a way that it can be mounted on a different sturdy base. Steel tubes of rectangular and square profile form the basis of the machine which are connected using screws. The photos on the left show the production of connecting pieces, drilling holes and the tapping of threads.

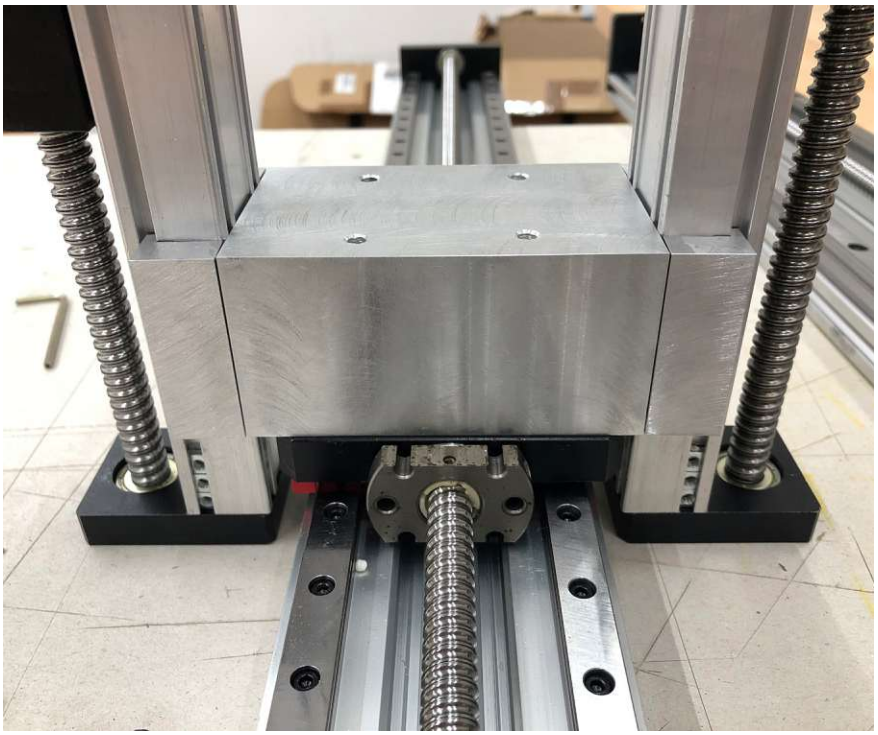


Milling connectors using the milling machine.

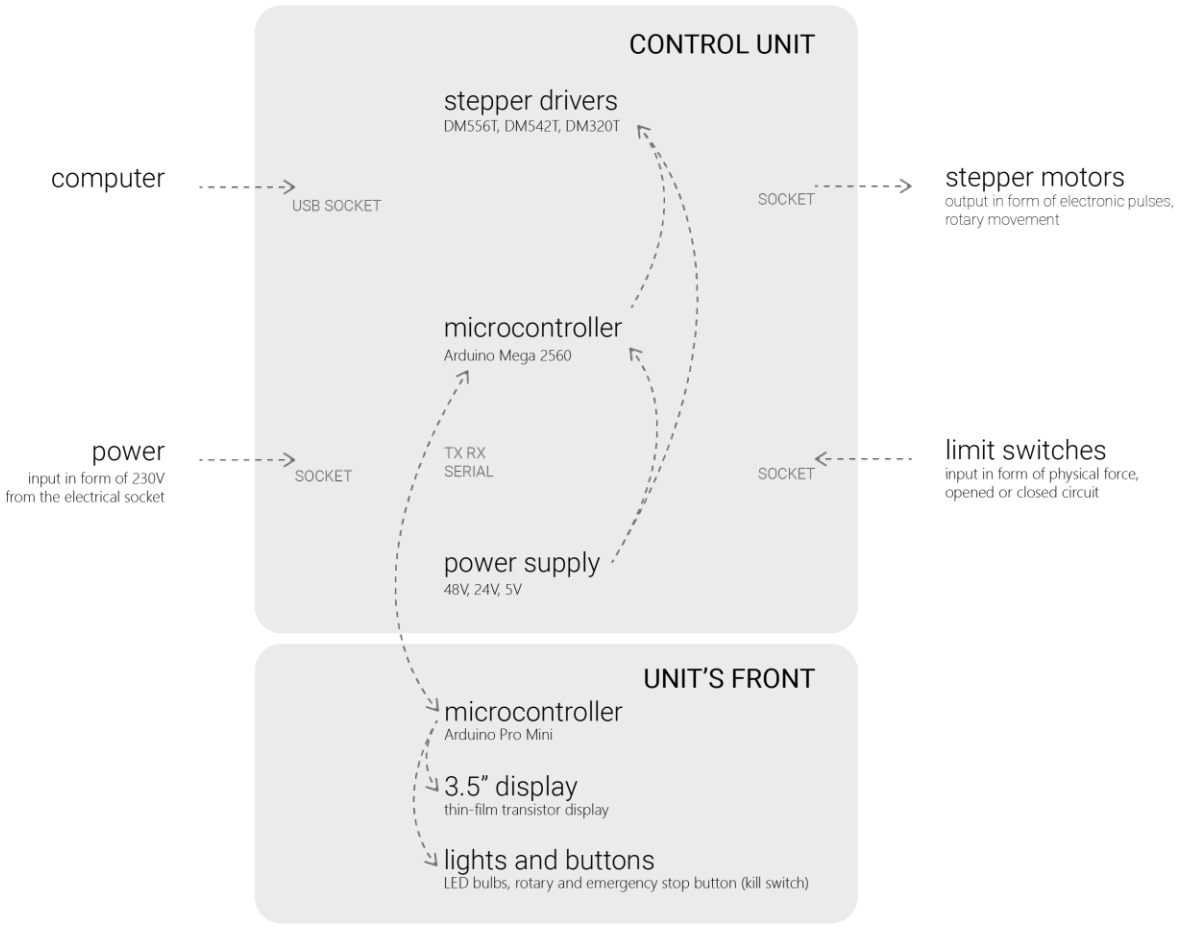
Linear axes

The x-axis with a length of 1260 mm is mounted directly onto the basis. Connectors were milled out of aluminum, holding the four axes in a position and forming a right-handed xyz-coordinate system with an additional z-axis.

All axes work with a spindle or ball screw drive which ensures precision of repetition and positioning. The x-axis and y-axis (with a length of 1000 mm) form a drawing space of approximately 1000 x 600 mm. Both z-axes have a travel length of 220 mm.



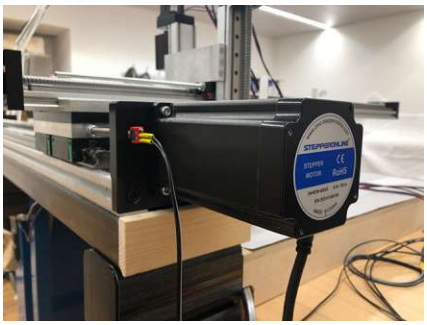
Mounting of z-axes assembling aluminum connectors.



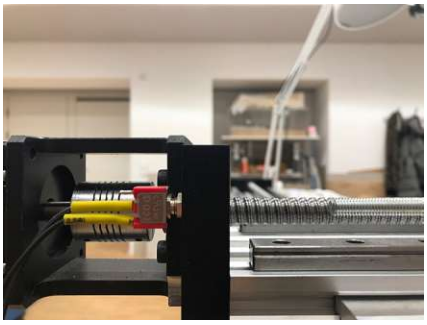
Simplified diagram of the electronic setup. All devices are connected to ground.

3.2.1 Assembled Hardware - Electronics

Essential for the correct movement of each of the pen plotter's axes is a control unit that receives information from the computer and communicates with the machine. There are two sockets for each axis: The first one sends signals to the respective stepper motor in the form of electric pulses. The second one receives information from the limit switches.



Stepper motor per axis.




Pressure switch on y-axis. (left)

Figure 3.16: Roller lever microswitch on z-axis. (right)

Stepper motors

All axes use bipolar stepper motors with a step angle of 1.8° and 200 steps per revolution. The other specifics from weight to shaft length differ from motor to motor as the axes require different power per axis. For instance, the holding torque is 12.0 Nm, (x-axis) 300 Ncm (y-axis), 52 Ncm (z-axis), respectively.

Limit switches

Limit switches are electromechanical devices that are operated by applying a physical force. There is one limit switch mounted at the origin of each axis to identify the maximum travel length. When the linear carriage triggers the switch, an electric circuit is closed or opened, depending on the type and configuration of the switch. Signals from the limit switch are essential for the homing of the machine. Two types of switches were installed: Threaded rods trigger a pressure switch on the x- and y-axis. Roller lever microswitches are triggered by the linear carriages on the z-axes.

Control Unit

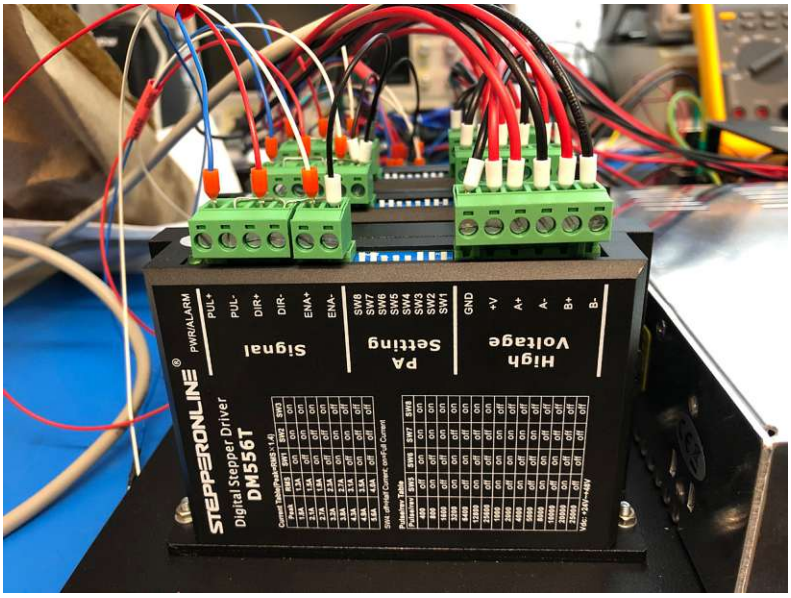
In order to be able to control the pen plotter's movements, an aluminum case was equipped with a power supply, stepper motor drivers, an Arduino board and connections to the pen plotter and the computer.

Power supply

Power enters the case at the 230V power socket and is distributed to the three power supply units. The power supply was installed for 48V, 24V - the stepper drivers require different voltages - and 5V each. The Arduino board is powered with 5V.

Opposite page:
The open aluminum case, still without wiring between its components:

1) 230V power socket
2) sockets to connect the incoming wiring from the axes
3) two USB sockets connecting the Arduino with the computer
4) 5V power supply
5) 24V power supply
6) 48V power supply
7) stepper drivers
8) 3,5" shield display
9) 2 LED bulbs and the emergency stop button

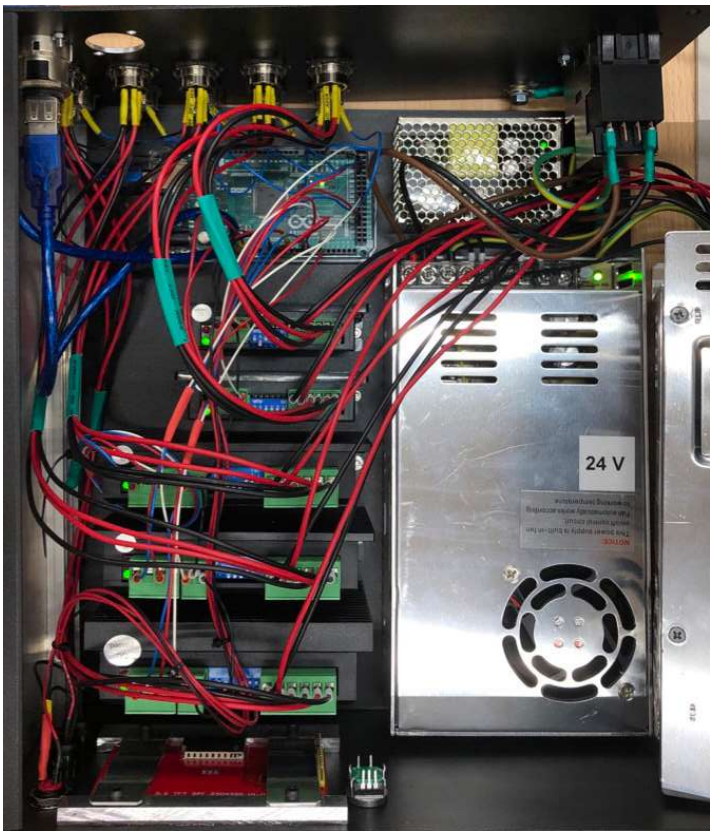


Stepper driver DM556T (x-axis).

Stepper drivers

Five stepper drivers of different types (DM556T (x-axis), DM542T (y-axis and tool) and DM320T(z-axes)) are used to receive signals from the microcontroller and translate them into electric pulses as input for the stepper motors. Each of the five stepper drivers is connected to the microcontroller, to ground, to the power supply and to the output socket in the case that is connected to the wiring to the respective stepper motor (wiring left to right, see photo on the left side).

One can specify the current and microstep resolution by setting an 8-bit DIP switch. In figure 3.15, the first three switches (SW1-SW3) are setting the dynamic current to its maximum of 5.6A for the peak current, and 4.0A for the RMS current. Switch SW4 is set to full current and the last four switches (SW5-SW8) set 8000 microsteps per revolution which is 40 microsteps per 1.8°.



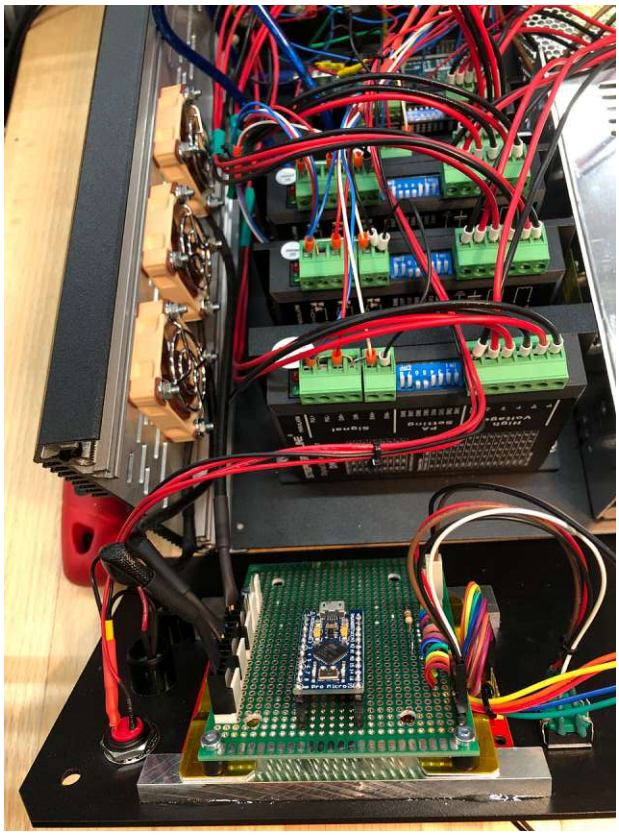
The open aluminum case with wiring.

Arduino Mega 2560

An Arduino Mega 2560 is used to mount GRBL and enable the communication between computer and pen plotter. The G-Code files are sent to the Arduino via USB, received by GRBL and are executed by the machine.

Front

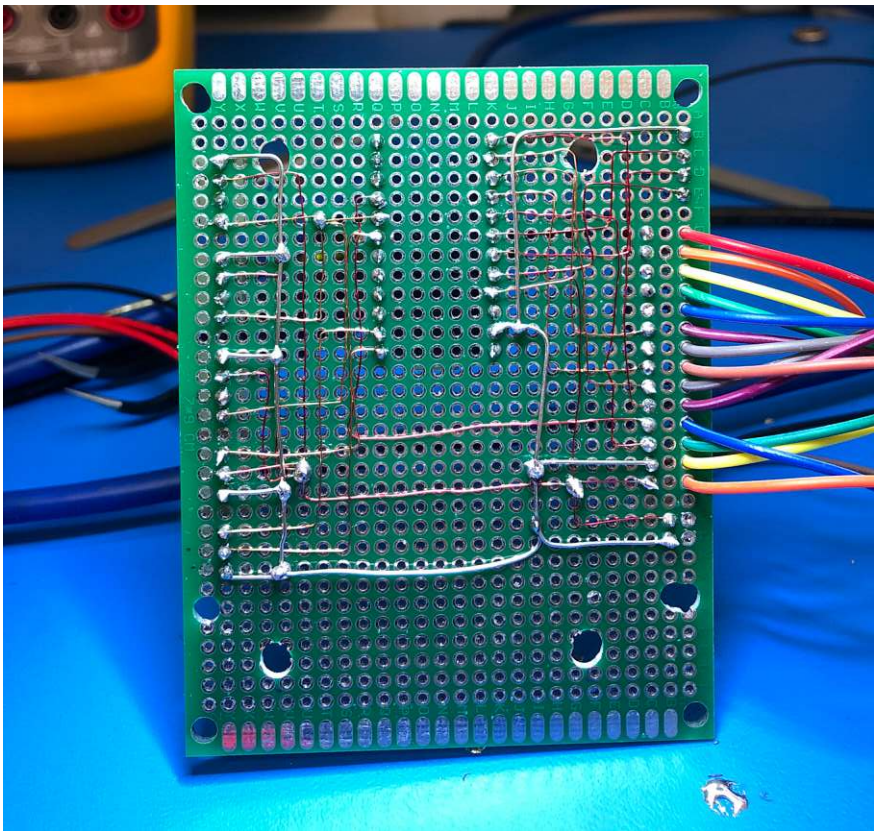
The front of the controlling unit was designed and assembled in a second step. It accommodates the display, a rotary button to control the display, LED lights



Mounted fans and stripboard with components.

which show the current state of the machine and a kill switch. All the openings on the front side were milled out of the aluminum plate to ensure precision.

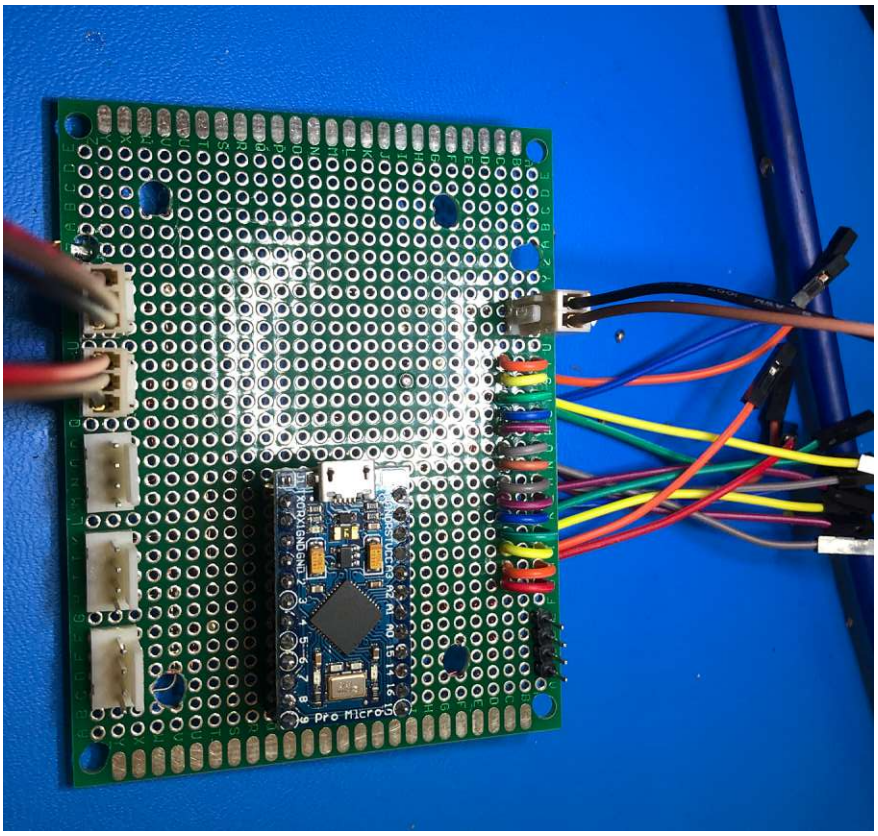
A stripboard or double sided prototyping board was used to mount the microcontroller and other prefabricated modules that are used to connect the other devices (LED lights etc.). All connections were manually wired.



The backside of the stripboard.

Display

A thin-film transistor (TFT) display of 3.2 inch and a resolution of 240x320 pixels was mounted to show additional information about the machine status. There are various libraries that provide example code: the open-source hardware distributor Adafruit , the TFT Adafruit Bus IO Library and Arduino's TFT Library .



All components and connectors assembled.

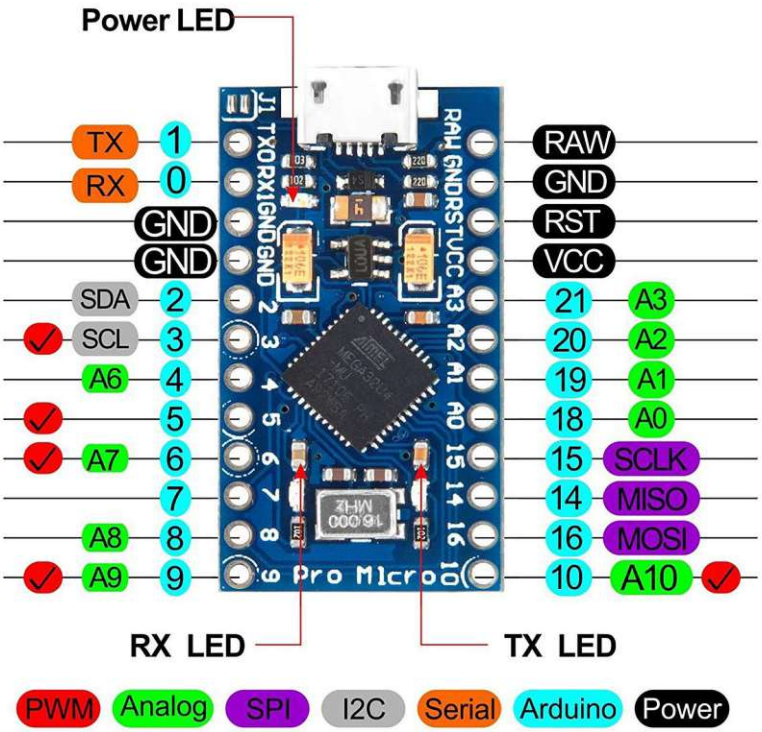
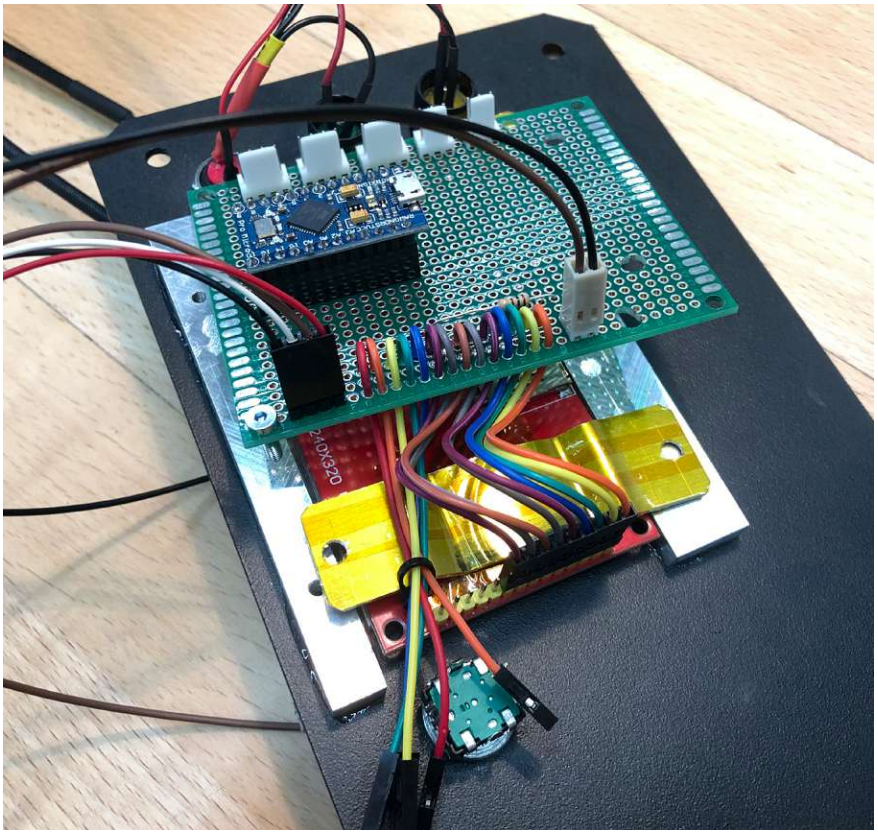


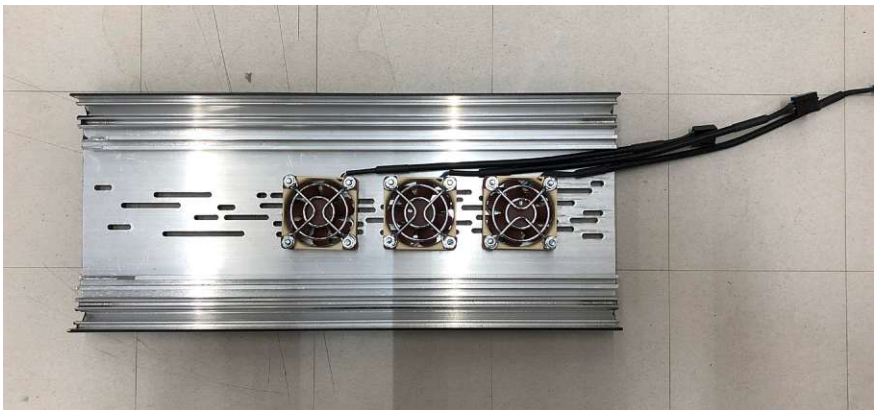
Figure 3.02: Arduino Pro Mini Pinout.

Arduino Pro Mini

In order to be able to control the display as well as the buttons and LED lights, an Arduino Pro Mini was added. It is positioned on the back of the display (see figure on the right). On the Pinout in figure 3.02 all the inputs and outputs of the Arduino board can be seen. The Arduino board uses an ATmega32U4 microcontroller. The two Arduino boards communicate with each other via TX RX serial communication.



Mounting of the strip-board on top of the TFT display.



Mounted fans.

Temperature sensor

The DS18B20 is a programmable temperature sensor that is used to monitor the operating temperature of the power supplies.

The side of the case with the mounted fans.

Fans

The Arduino Pro Micro is controlling the fans using pulse-width modulation. PWM reduces the power of an electrical signal. Ventilations slits were milled into both lateral parts of the aluminum case to enable the circulation of air.

Philips webcam. (left)

Drawing setup. (right)

3.2.3 Drawing and Webcam Setup

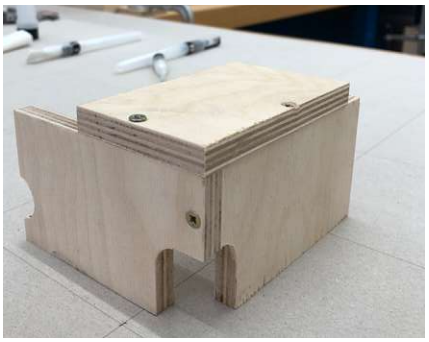
Photo setup
A snapshot of the user's drawing is made using a Philips webcam. It is currently positioned directly above the area on the paper where the user is drawing.

Drawing setup
The output from the network is drawn using a brush filled with distilled water onto a rewritable cloth that is used to learn calligraphy. The refillable brush is mounted onto one z-axis using a customized 3d printed pen holder.

Other part of the drawing setup in form of a wooden writing pad and rewritable cloth.

Canvas

A wooden writing pad was built to ensure the correct distance from the brush that has been mounted onto the z-axis to the canvas. Magnets were inserted into both longitudinal sides of the wooden pad. Two steel rulers are fixated on the magnets and hold the textile canvas into place. The rewritable cloth is usually used for practicing calligraphy.

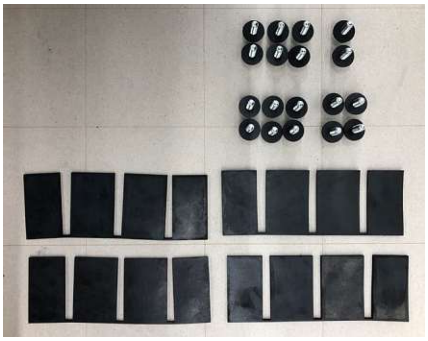


Above: Distilled water is used to draw on the textile canvas.

Below: Several prototypes were built.

M6 screws that usually hold the base together, covered by rubber rings.

The comb-like layers of rubber were inserted in between the steel tubes before inserting the above mentioned screws.



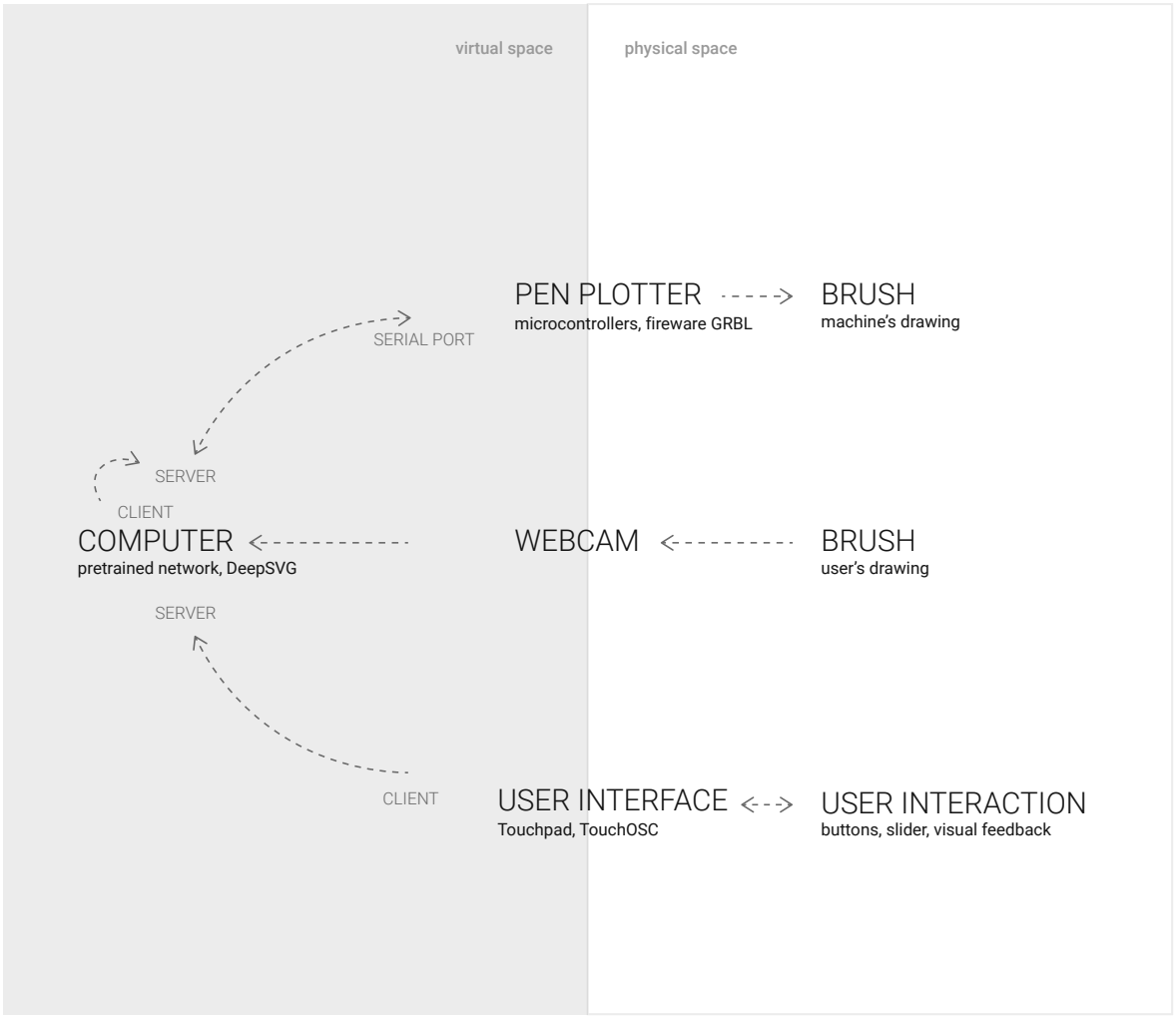
3.2.4 Acoustic insulation

Rubber damper

In an additional step, rubber was inserted in between the steel tubes that form the base of the machine.

TPU lids

The steel tubes themselves were filled with plastic granulate mats and closed with customized 3d printed lids in TPU.



3.3 Software

The following chapter describes the software and framework that was used to ensure interoperability and which is the basis for the set of experiments that was carried out and is described in the next chapter.

3.3.1 Software environment, language, platform

Python
Interoperability was ensured using the Python programming language. As the DeepSVG environment works with this version, Python 3.7 was used for the entire project.

Anaconda
Anaconda is a distribution of the Python programming languages for scientific computing that aims to simplify package management and deployment.[2]

2 Anaconda (Python distribution)

Visual Studio Code
Visual Studio Code is a source-code editor that is provided by Microsoft and is available cross-platform for Windows, macOS and Linux. In contrast to Visual Studio, VS Code does not work using a project system, but users are able to open one or more directories. The editor is popular because of its environment tool.

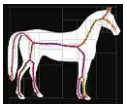
The first deep learning exercises were all done using Jupyter Notebooks. This web-based platform was abandoned because of its dependence on a stable internet connection.

Opposite page:
Higher order diagram of the software implementation

USER INPUT
take snapshot

takeSnapshot

applySkeletonTracing



preprocessing

performInterpolation

DeepSVG

USER INPUT
browse through/display interpolation output

displayPNGs

USER INPUT
select a drawing and send the created G-Code to the machine

selectSVG

saveSVGtoGCode

3.3.2 Data Pipeline

The data pipeline is described in its basic structure in both diagrams (on the left and the next page) and describes the sequence of steps that have to be computed in one iteration (that was conceptually and sequentially described in chapter 3.1.3). A more in-depths description of what happens on the code side is provided in the following:

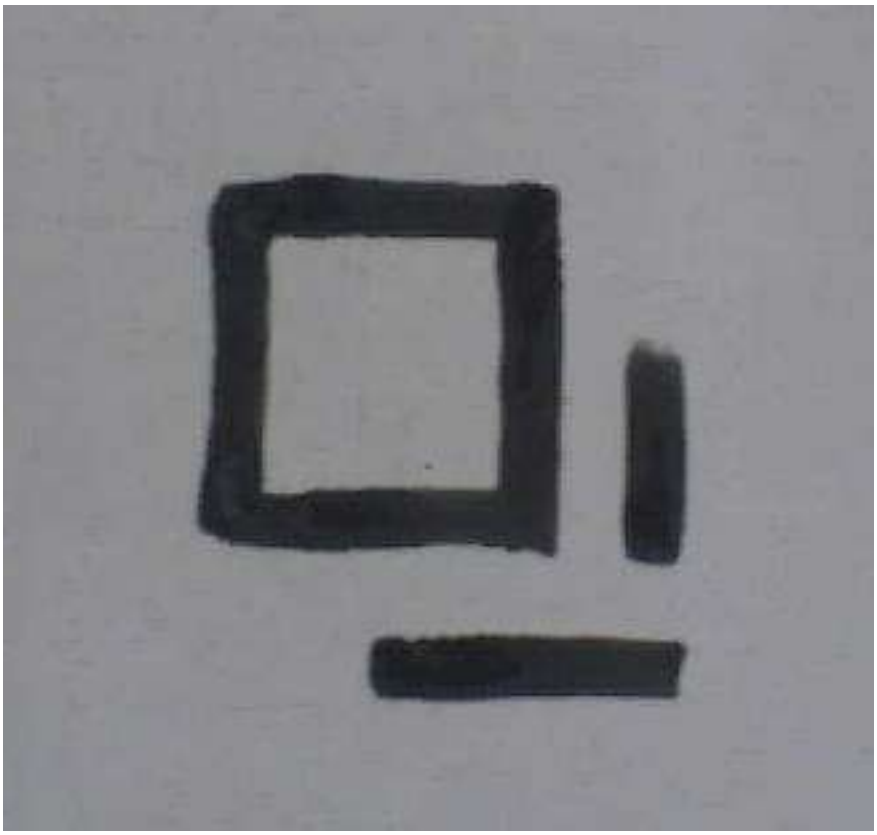
The iteration starts with a snapshot of the user's drawing which works as the first input of data in the form of a Bitmap image. A PNG is saved to the first folder of the directory. The brush strokes are extracted and saved in SVG format.

A first attempt to do this was implementing RhinoInside, Grasshopper and its Firefly plugin. The Traveling Salesman Algorithm was used to connect extracted points from the Bitmap photo in the order of the shortest distance to one another. This method was abandoned as it was quite computation-intensive and did not produce satisfying results. The parameters could not be set in a way that a broad range of different user drawings could be translated into polylines that would replicate them.

Skeletonization is used in the actual version of the pipeline. A more detailed description of this process is provided in the following.

Preprocessing of the resulting SVG images is a prerequisite for the interpolation. The interpolation is performed using a Deep Hierarchical Network, the pretrained model published as DeepSVG.

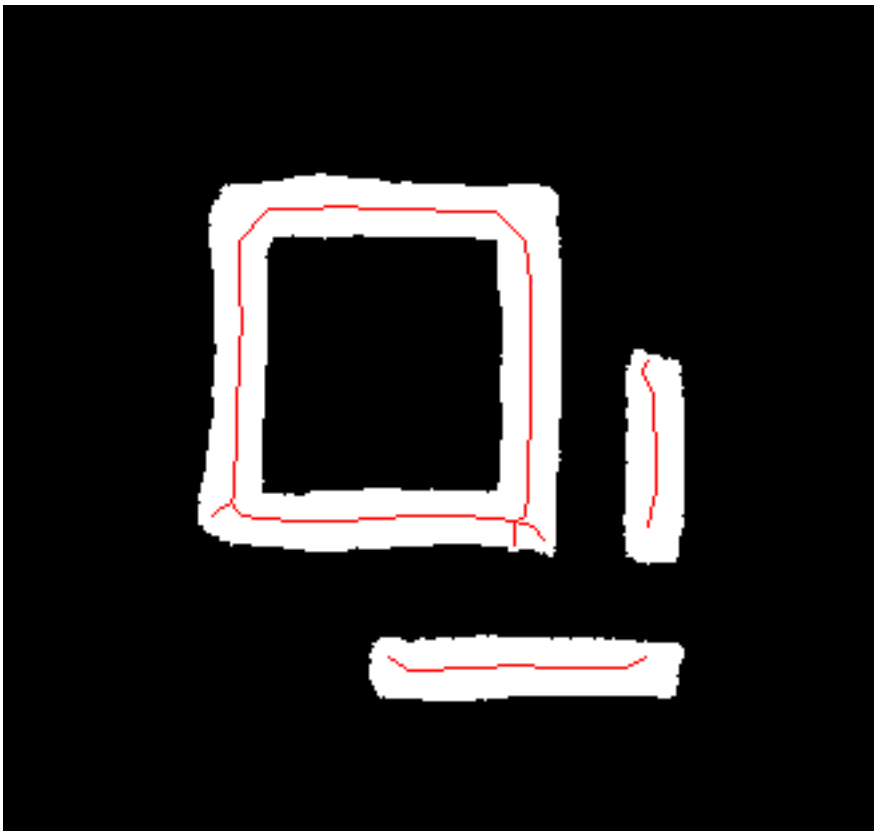
Opposite page:
Pseudocode application software



Snapshot user drawing.

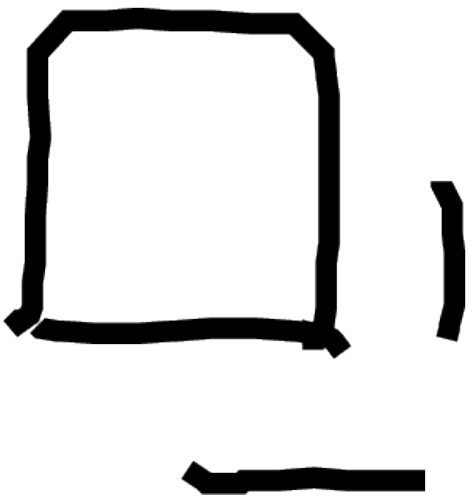
The user inputs data here for the second time within the process. Browsing through the output of the interpolation is possible and a selection of one of the frames results in the SVG being translated into machine instructions and sent to the pen plotter.

As introduced before, the code was written in Python, in the Anaconda environment and using Visual Studio Code editor. All the user input is handled via TouchOSC.



Skeletonization.

The project comprises around 750 lines of code that were written in the scope of this project. The program performing Skeleton Tracing includes around 300 lines of code. The code from DeepSVG that was used in this project is not included in this summation.



Result Skeletonization in SVG format.

The user inputs data here for the second time within the process. Browsing through the output of the interpolation is possible and a selection of one of the frames results in the SVG being translated into machine instructions and sent to the pen plotter.

As introduced before, the code was written in Python, in the Anaconda environment and using Visual Studio Code editor. All the user input is handled via TouchOSC.

DeepSVG

As mentioned above, the code and pretrained network that was published in 2020 accompanying the paper "'DeepSVG: A Hierarchical Generative Network for Vector Graphics Animation" was used for the interpolation process.

RhinoInside

RhinoInside made it possible to embed Grasshopper files in the data pipeline.

Grasshopper

Grasshopper was used for geometry manipulation.

SkeletonTracing

The SkeletonTracing App was developed at the Frank-Ratchye Studio for Creative Inquiry at Carnegie Mellon University and is available in all common languages on Github. In this morphological operation, a binary image is reduced to its topological skeleton. Another term for the process of skeletonization is thinning.[3]

3 Huang, L., Skeleton Tracing

Github

It would go beyond the scope of this book to publish all written code. The code is open-source and available on this link: https://github.com/mnoemayr/experiment-1

3.3.3 Firmware

G-Code

G-Code is widely used for computer numerical control. The following commands were mostly used to communicate with the pen plotter:

```
G90
(Set to absolute positioning;
coordinates are absolute to the origin of the machine)

G1 X0 Y225.5 Z0
(Linear move;
Xnnn (position to move to on X axis)
Ynnn (position to move to on Y axis)
Znnn (position to move to on Z axis)
Fnnn (feed rate per minute / speed of travel from point to
point))

G4 P0.1
(Dwell; pauses the machine for a defined period of time:
Pnnn (time to dwell in milliseconds)
Snnn (time to dwell in seconds))
```

GRBL

Grbl is a free firmware used to control CNC milling machines. It runs on all Arduino boards with the Atmega 328 microchip. Grbl supports G-Code, but also comes with its own commands.

```
$ commands are used for the configuration of the machine, give
user feedback or execute specific machine jobs:

$H (Run homing cycle)

? (Status Report Query)
```

3.3.4 Communication between IDE and Firmware and Application Software - Interoperability

The smooth operation of the written programs and communication between the devices is essential for the information exchange between virtual and physical environments.

There are two parts of code that make this information exchange possible: The first one exchanges information between the pen plotter and the computer and is responsible for the execution of sent machine instructions. The second one assures communication between the user interface and computer, embeds the code of the Deep Neural Network and receives data created by the user in the form of webcam snapshots.

UDP Protocol

Figure 3.06 shows the communication process of UDP server and client in theory.

There is one server-client couple on the machine side: In order to perform the homing cycle once and be able to send updating G-Code commands to the machine, a UDP-based server and client are set up.

The other server-client couple is handling all information regarding drawings in vector and raster format, the implementation of the pretrained Deep Neural Network and user input.

UDP Server

```
#Create socket
UDPServerSocket    =    socket.socket(family=socket.AF_INET,
type=socket.SOCK_DGRAM)

#Bind to Port and IP
localIP     = "127.0.0.1"
localPort   = 20001
UDPServerSocket.bind((localIP, localPort))

#Listen for incoming messages
while(True):
      bytesAddressPair = UDPServerSocket.recvfrom(buffSize)
      message = bytesAddressPair[0]
      address = bytesAddressPair[1]
      clientMsg = "Message from Client:{}".format(message)
      clientIP  = "Client IP Address:{}".format(address)

      # Sending a reply to client
      UDPServerSocket.sendto(bytesToSend, address)
```

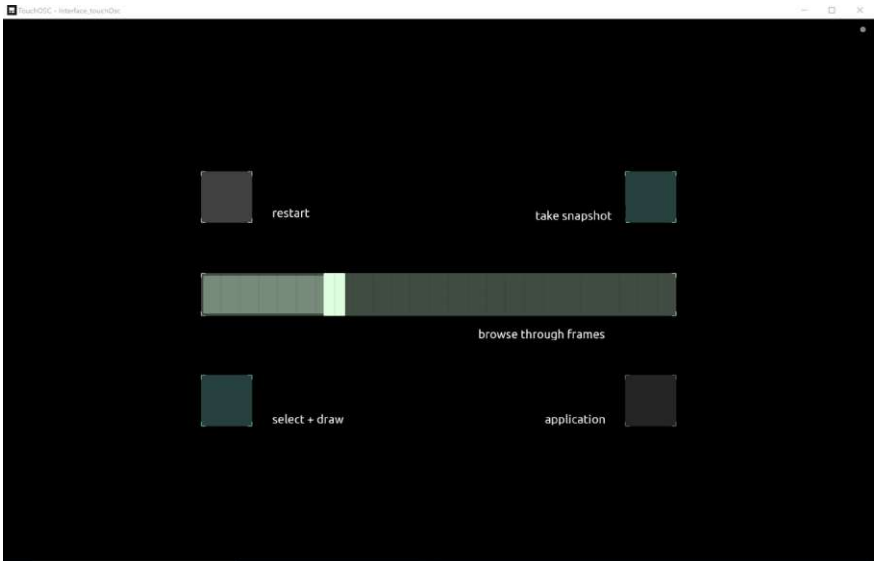
UDP Client

```
#Create socket
UDPClientSocket    =    socket.socket(family=socket.AF_INET,
type=socket.SOCK_DGRAM)

#Send message to server
UDPClientSocket.sendto(bytesToSend, serverAddressPort)

#Receive message from server
msgFromServer = UDPClientSocket.recvfrom(bufferSize)
msg = "Message from Server {}".format(msgFromServer[0])
```

2 UDP - Client And Server Example Programs In Python, 2022



Graphical user interface.

TouchOSC

TouchOSC calls itself a modular control surface. It supports all platforms and sends and receives MIDI and OSC messages. OSC is a protocol that is used to control music instruments and stage lighting control systems. TouchOSC enables easy creation of interfaces and provides an option for sending and receiving signals via UDP.

Touchpad and User Interface.

The TouchOSC uses a so-called bridge connection to communicate from the touchpad to the computer. It functions as a UDP client in this case. The server on the computer listens to data from this client.

4 Application

This chapter includes a visual and textual documentation of the set of experiments that have been conducted in the scope of this thesis.

The following targets were set for the experiments: The behavior of the network and setup were tested under predefined conditions that are mentioned at the beginning of each block of tests. For the first two blocks, the focus has been set on abstract and figurative drawings. Expected results were compared with what actually happens in a simple experimental setup with one iteration maximum. The third block of experiments documents a conversation conducted between user and machine.

First drawing (left)
result image tracing
(right)

Second drawing (left)
result image tracing
(right)

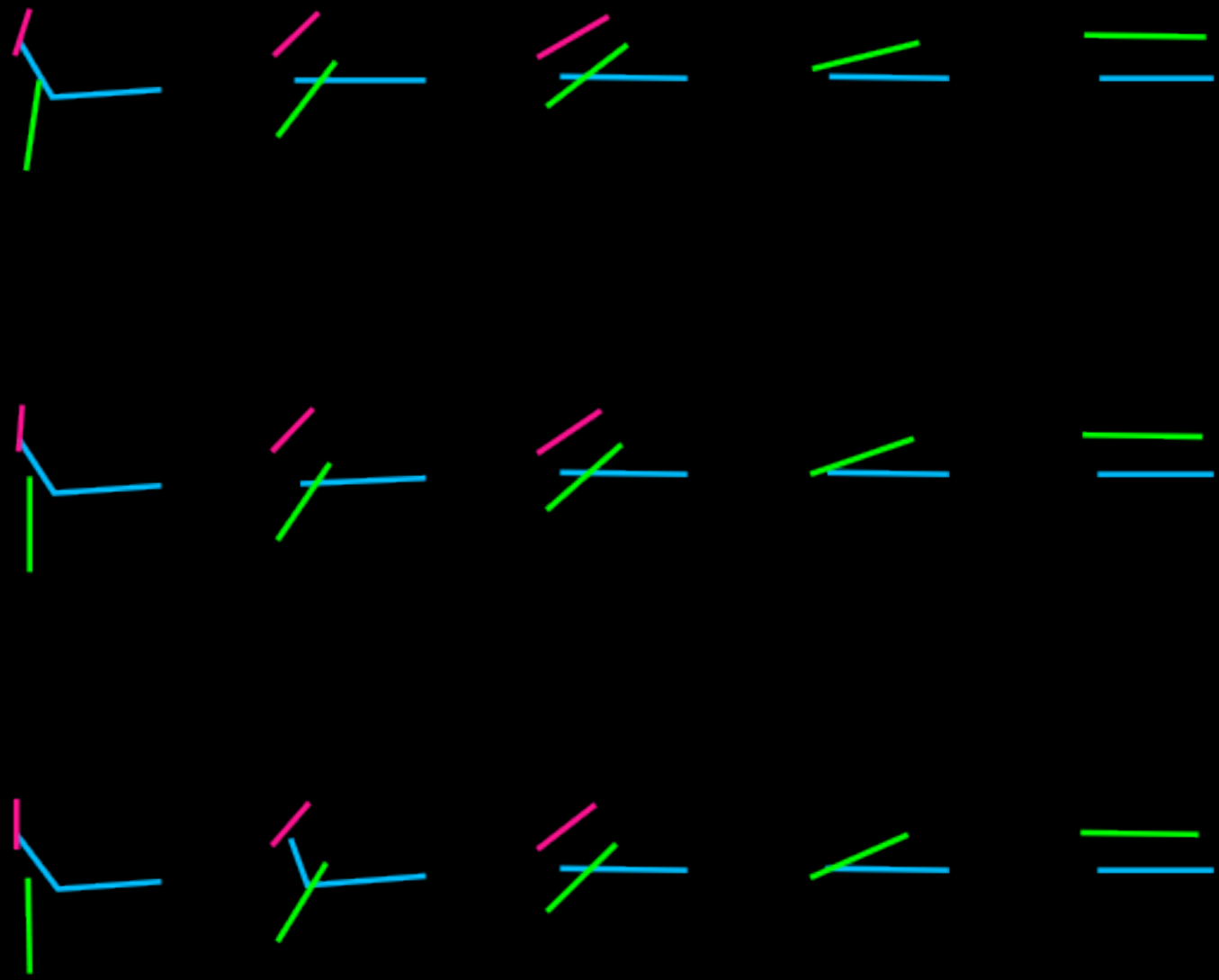
4.1 Abstract experiment

Prerequisites

Abstract, geometric interpolations were tested in this first block of experiments. User drawings of simple geometry, such as open, straight, parallel lines and closed, curved lines, were used. The output was evaluated according to the comprehensibility of the performed interpolation and the complexity of the geometry generated.

Results

The results of what was tested, has been visually and textually documented on the website https://deepgestures.carrd.co/ as a research portfolio in the first set of abstract experiments (1.1 to 1.10). In order to improve readability, the visual results of experiment 1.3 Abstract interpolation between two parallel lines and two orthogonal lines will be presented on the next two pages.



Interpolation frames of the abstract, geometric interpolation between two straight, parallel and two straight, orthogonal lines.

Conclusions

This set of experiments was conducted in order to gain a basic understanding of what happens during the interpolation in the deep neural network. Beginning with single straight or curved strokes, complexity was added incrementally.

In some cases, the process of image tracing adds new aspects to both initial geometries. If reflections occur on the PNG file, the image cannot be translated correctly by the Skeleton Tracer. It is possible that a single straight brush stroke is depicted as two segments. Thus, new serendipitous and unpredictable frames emerge during the interpolation process. However, a realistic picture of the interpolation between both figures is not drawn in that case.

The developers of the pre-trained deep neural network DeepSVG state that the network would be able to process inputs in SVG format with up to eight paths (Carlier, 2020). The results of the experiments show that in the case of manually drawn and preprocessed input (PNG - image tracer - preprocessing), up to four paths can be used. However, this circumstance does not undermine the concept.

Figurative Experiment A
Digital input

First drawing

Second drawing

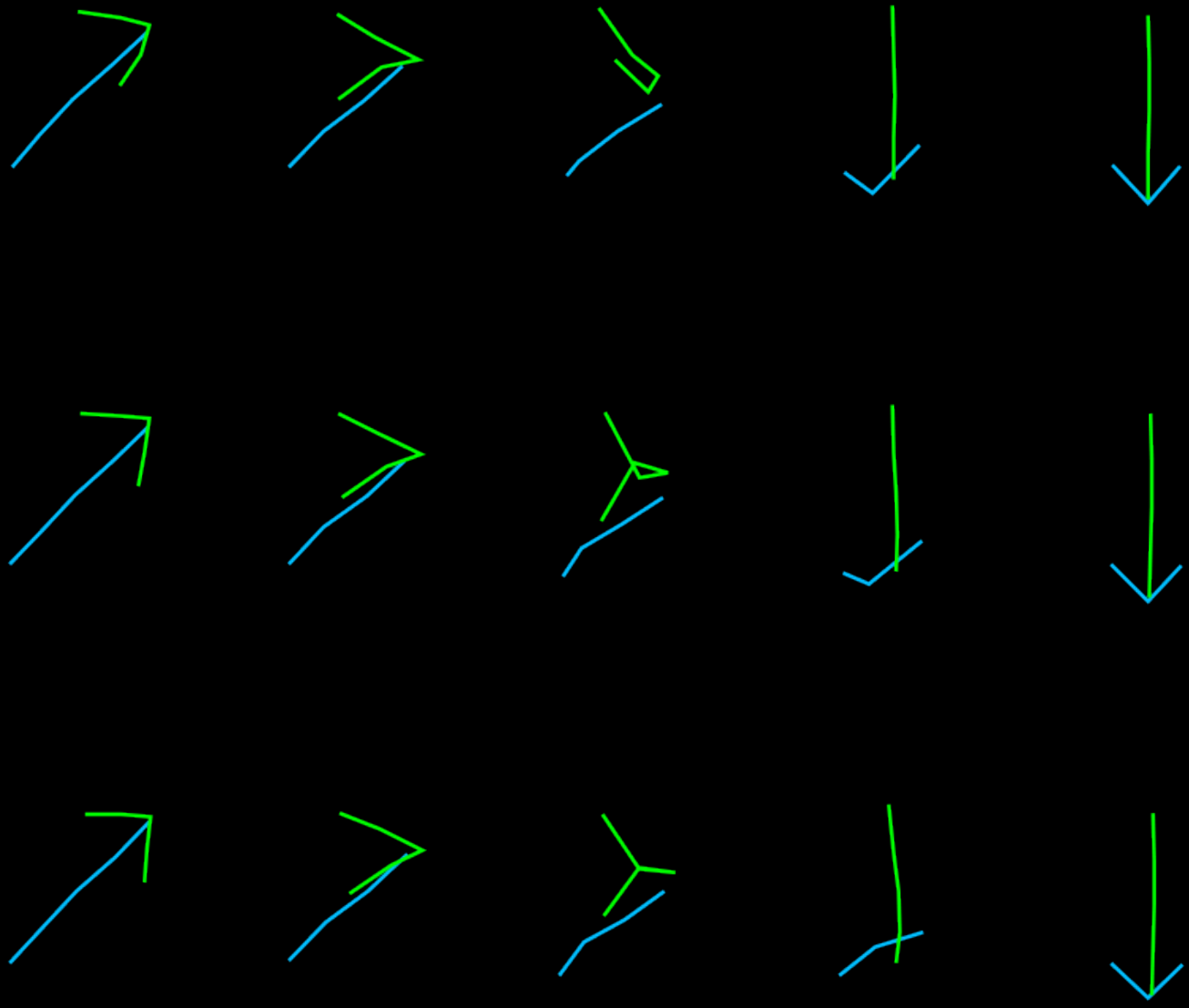
4.2 Figurative experiments

Prerequisites

Figurative interpolations were tested in this second block of experiments. Input was provided both in digital form (icons from the Icon8-dataset) as in the form of user drawings (made after the icons). The output of both was compared and evaluated according to the comprehensibility of the performed interpolation and the complexity of the geometry generated.

Results

The results of what was tested, has been visually and textually documented on the website https://deepgestures.carrd.co/ as a research portfolio in the set of figurative experiments (2.1 to 2.3). The visual results of these three individual experiments will be presented on the next pages.

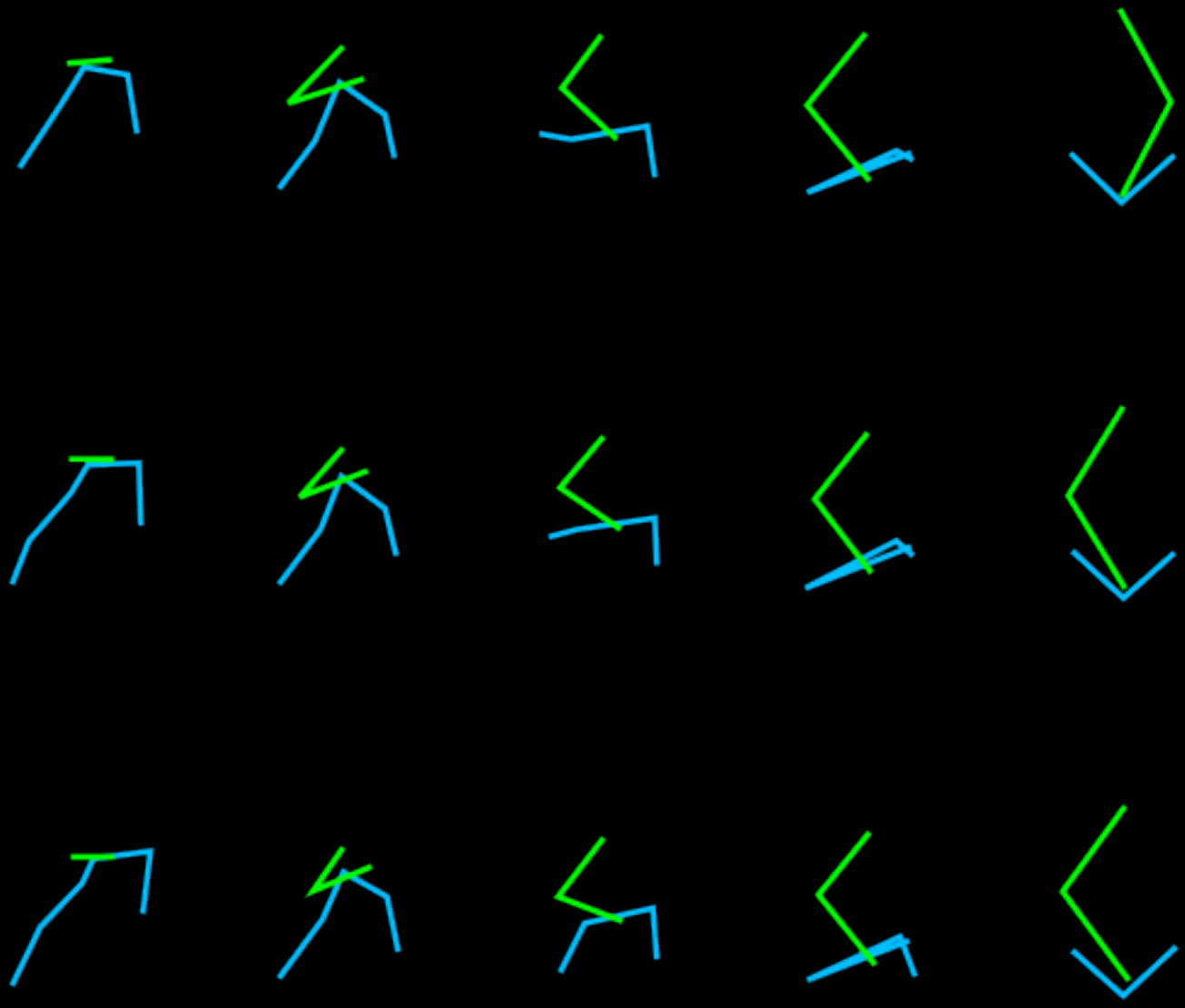


Interpolation frames of the figurative interpolation between two icons with two strokes each. The icons were chosen from the Icon8-dataset which the DeepSVG network was trained on.

Figurative Experiment B
Manually drawn input

First drawing (left)
result image tracing
(right)

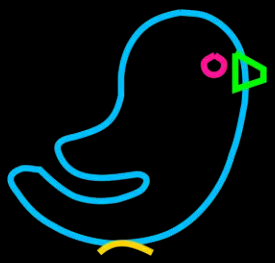
Second drawing (left)
result image tracing
(right)



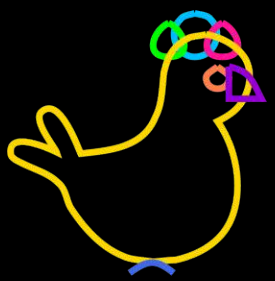
Interpolation frames of the figurative interpolation between two icons with 2 strokes each. The images were drawn after the icons from the last experiments.

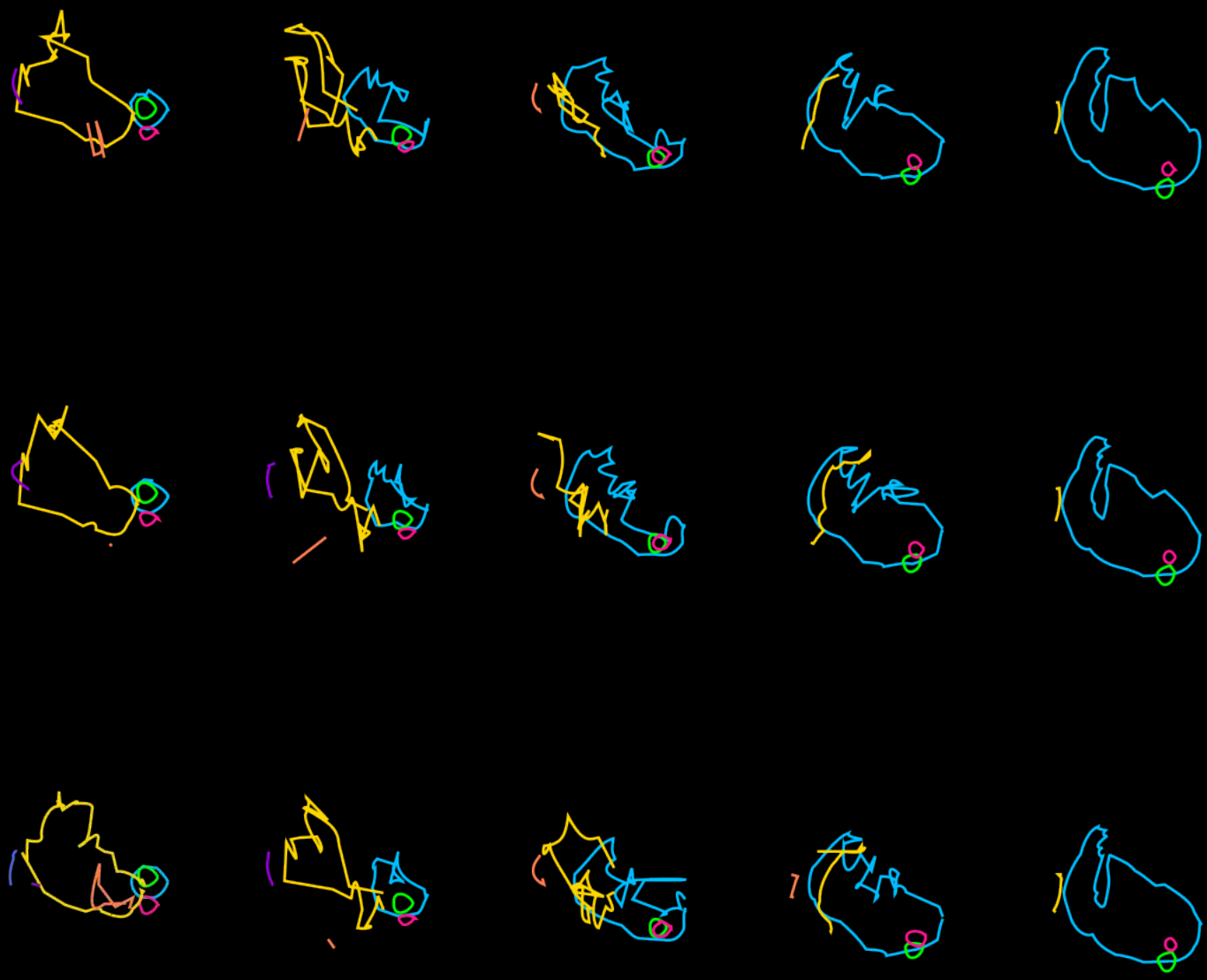
Figurative Experiment C
Digital input

First drawing



Second drawing





Interpolation frames of the figurative interpolation between four and seven paths. The icons were chosen from the Icon8-dataset which the DeepSVG network was trained on.

Conclusions

This set of experiments was conducted in order to gain a better understanding of what happens during the interpolation in the deep neural network. Some geometric and semantic relations could only be grasped using figurative input.

Each figure or icon is composed out of multiple strokes possessing a defined meaning. When the order of the strokes differ in each icon, regarding the semantic meaning of the stroke, a complex interpolation output is the result, that leads to interesting and unforeseen geometry, but is not comprehensible.

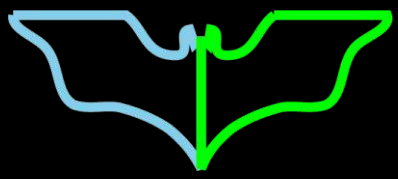
Without the sorting of the SVG paths before the interpolation, new unexpected behavior is introduced in the process, but at the same time it loses in comprehensibility. This is visible in both experiment using digital input:

In the first experiment, the arrow-head and -body are differently colored, the same color depicting the same position in the order of paths in the SVG code. From frame to frame, the arrow does not rotate from one direction to the other as intended, but the arrowhead is formed of the body of the other arrow and vice versa.

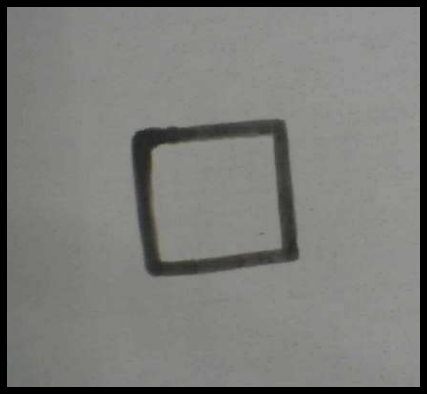
In the second experiment, this can be observed as well. The body of the chick is translated into the rooster's comb, the chick's claw into the rooster's body etc.

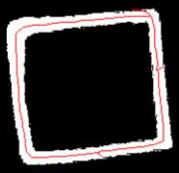
The limit of four strokes for physical input in the form of a user drawing is circumvented using digital input. The interpolation is still performed using seven strokes as input.

Iteration 1



First drawing





Second drawing (left)
result image tracing
(right)

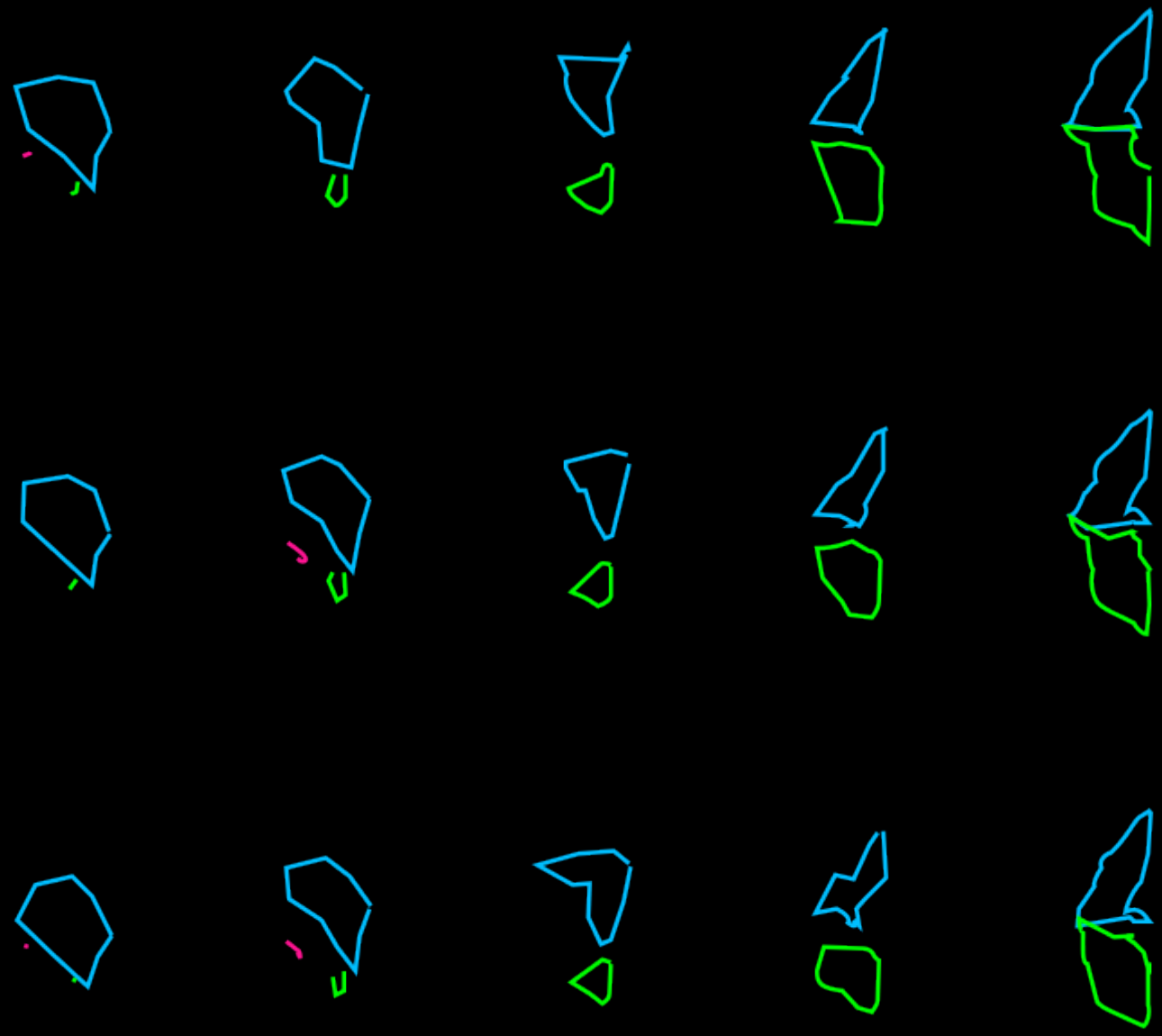
4.3 Conversation between human and machine

Prerequisites

A human-machine conversation was registered in this third block of experiments. The conversation starts with one icon from the Icon8-Dataset one the machine side, but the topic is changed by the user and is drawn in the direction of basic geometry.

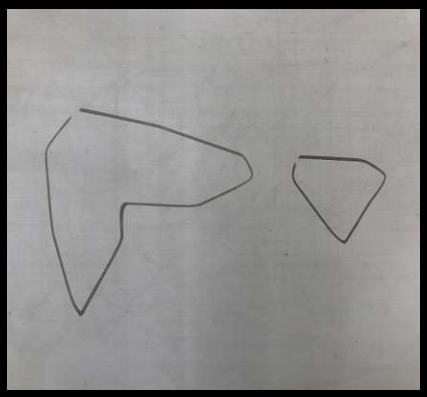
Results

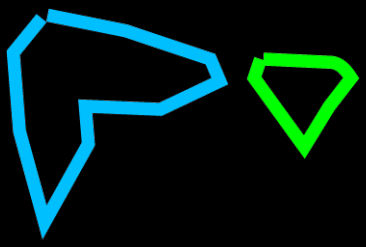
The results of what was tested, has been visually and textually documented on the website https://deepgestures.carrd.co/ as a research portfolio in section 3 on a conversation between user and machine. A visualization of this conversation will be presented on the next pages.



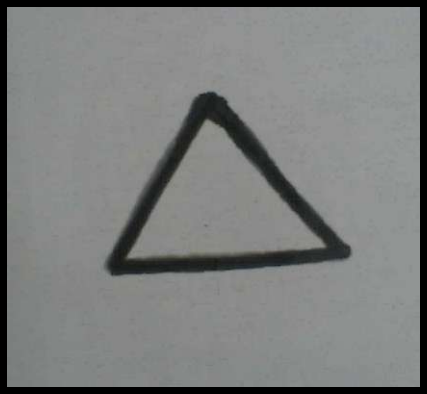
Interpolation frames of the first iteration of the conversation between user and machine. The conversation starts with one icon from the Icon8-Dataset one the machine side, but the topic is changed by the user and is drawn in the direction of basic geometry.

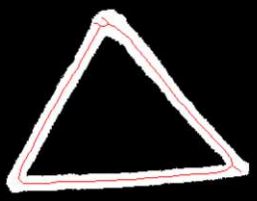
Iteration 2



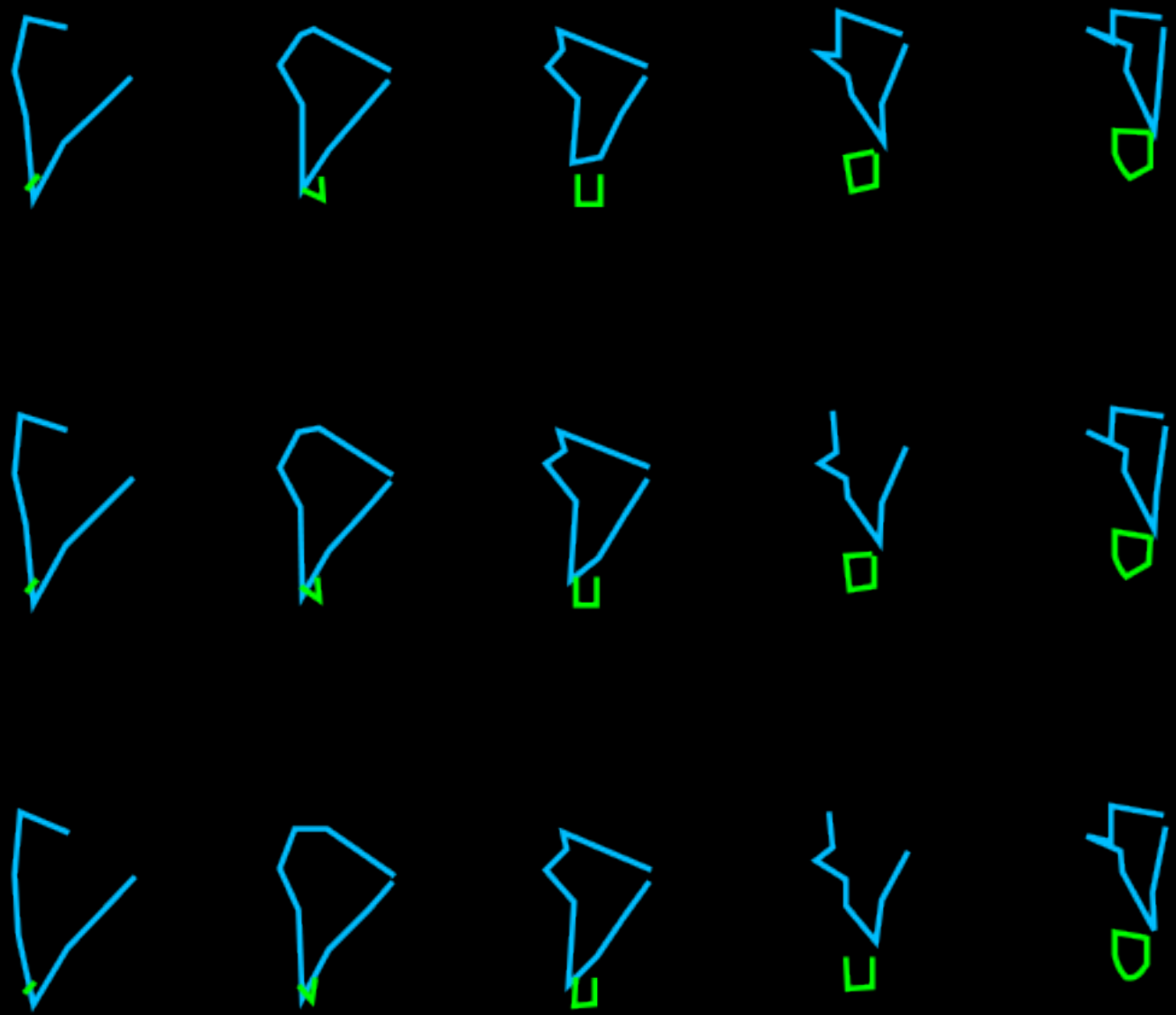


Machine drawing (left)
selected frame 12
(right)





Second drawing (left)
result image tracing
(right)

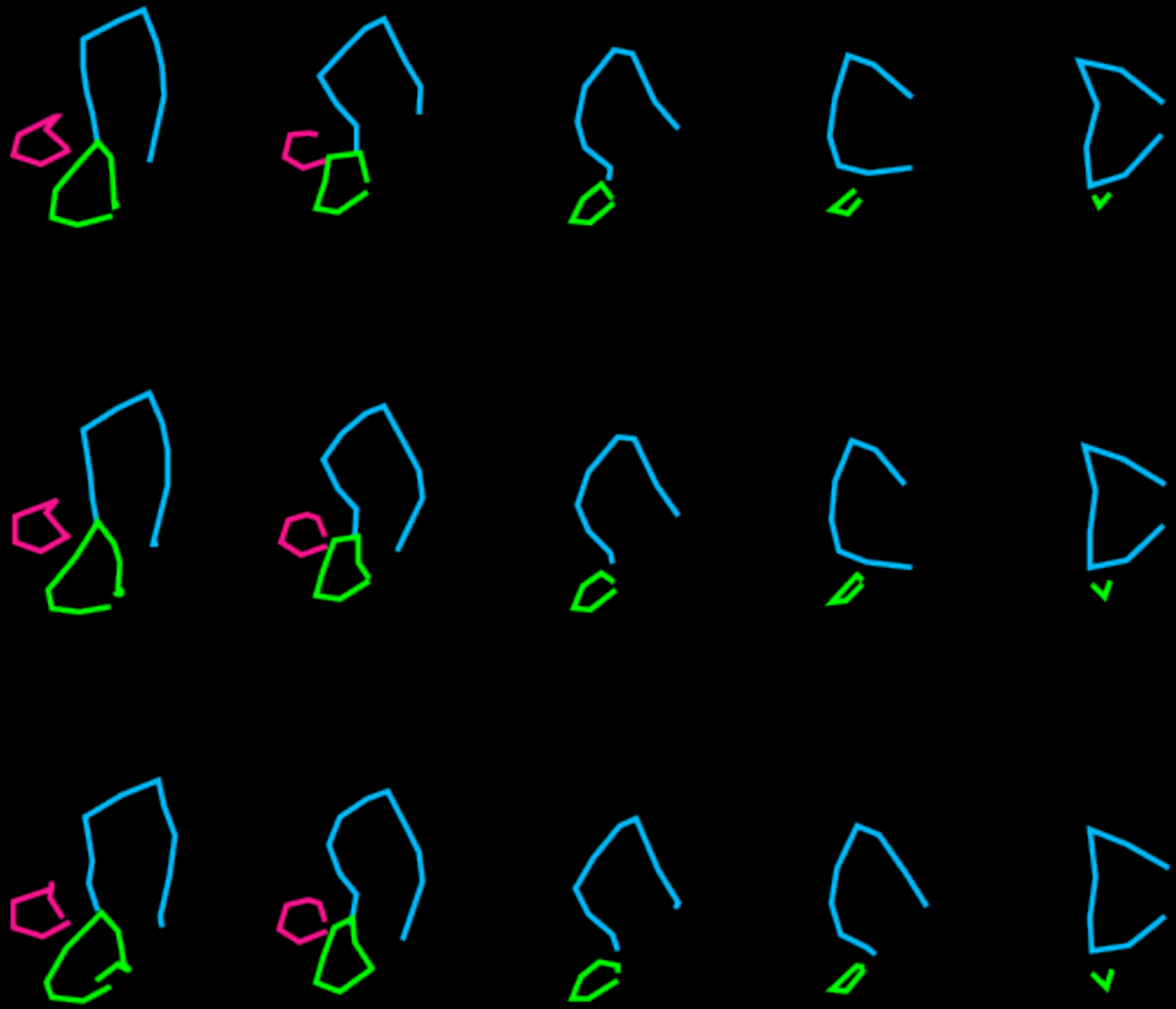


Interpolation frames of the second iteration of the conversation between user and machine.

Iteration 3

Machine drawing (left)
selected frame 8 (right)

Second drawing (left)
result image tracing
(right)

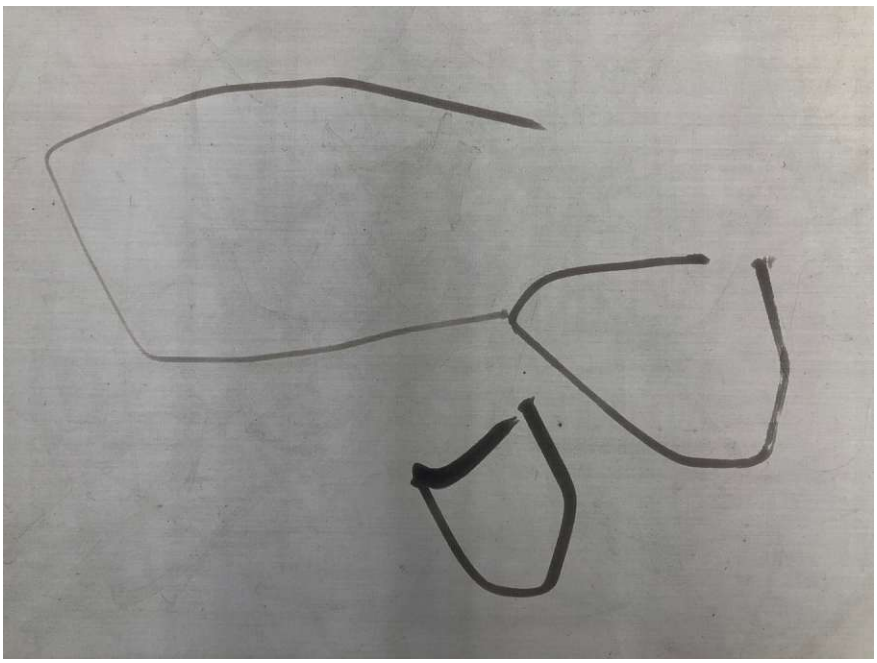


Interpolation frames of the third iteration of the user-machine conversation.

Conclusions

This set of experiments was conducted in order to gain a better understanding of what happens in the course of a conversation between a user and the machine. Some relations between user input, image visualization and machine output could only be grasped by visualizing them sequentially.

The user is leading the conversation as he or she is the one selecting the frame on which the next iteration is based. The dialogue can always be led into a direction that he or she prefers or be altered completely. It is an open-ended process of interaction that can at any iteration be concluded or picked up again, as all the frames are stored in the digital memory.



Conversation end:
Machine drawing of
selected frame 20

Experiment A

First drawing

Second drawing

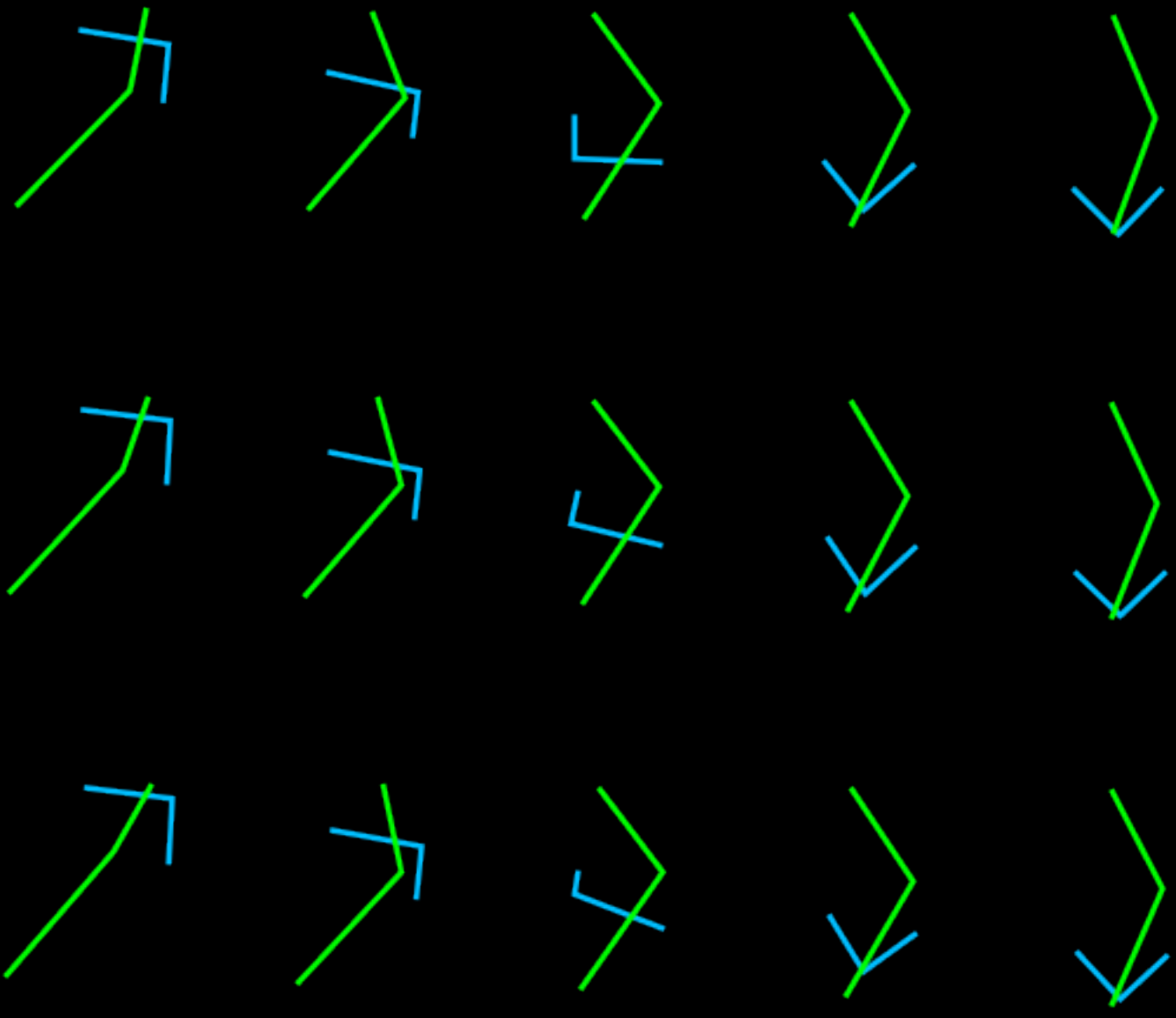
4.4 Path order

Prerequisites

In this block of experiments figurative interpolations were tested for a second time and improved with focus on the semantics of the images. Input was provided in digital form (icons from the Icon8-dataset)v. Before performing the interpolation, the input was revised regarding the order of the paths. The output of both was compared and evaluated according to the comprehensibility of the performed interpolation and the complexity of the geometry generated.

Results

The results of what was tested, has been visually and textually documented on the website https://deepgestures.carrd.co/ as a research portfolio in section 4 on revised figurative experiments (4.1 to 4.2).The results of these two experiments are visualized on the following pages.

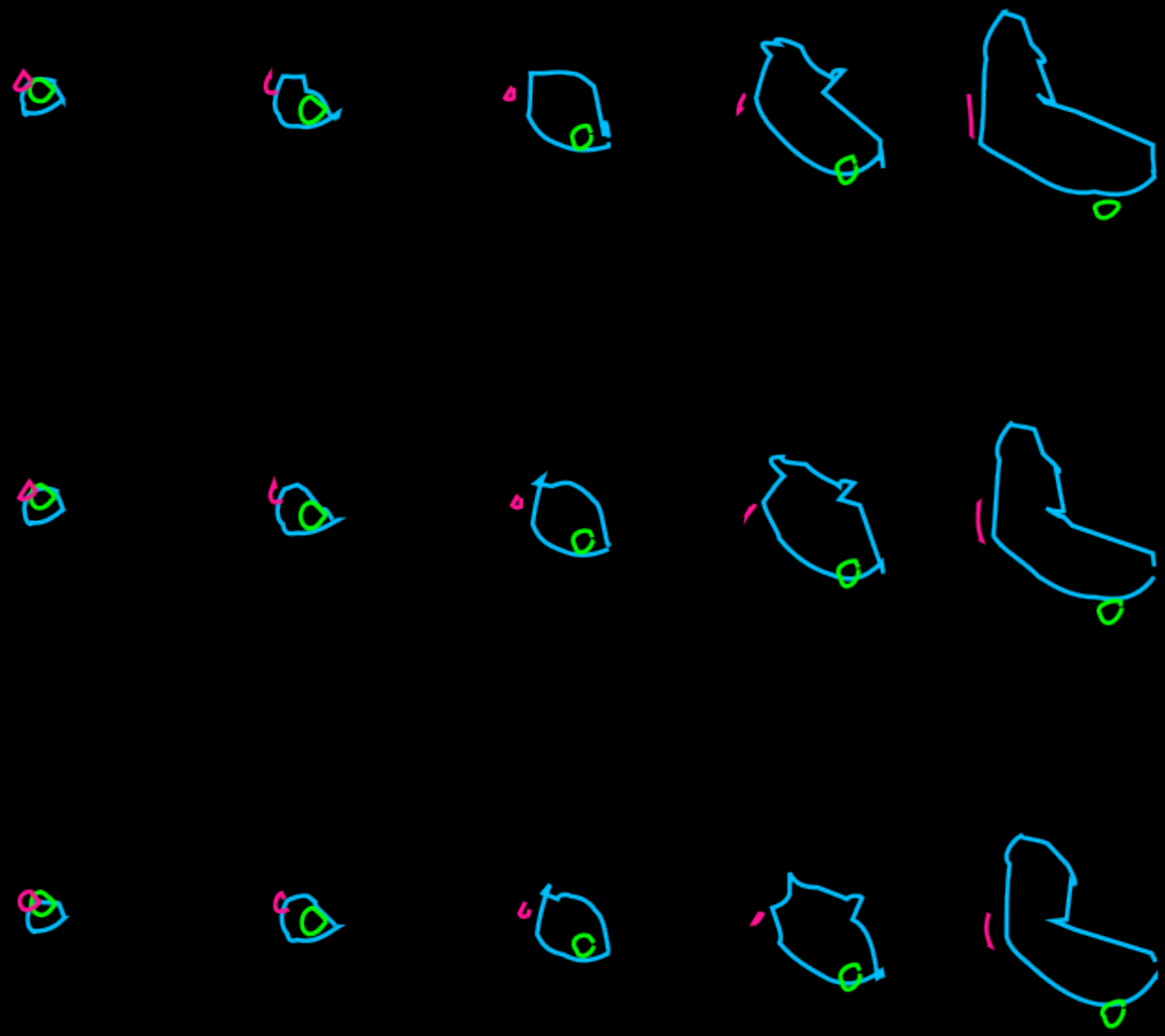


Interpolation frames of the revised figurative interpolation. The SVG paths were sorted before performing the interpolation.

Experiment B

First drawing

Second drawing



Interpolation frames of the revised figurative interpolation. The SVG paths were sorted before performing the interpolation.

Conclusions

Although sorting of the SVG paths was done manually in this step, it led to the desired results in the first experiment. An automated way to sort the SVG paths according to the semantic meaning is proposed. Some additional code would be necessary to automate this step and implement it in the preprocessing step of the data pipeline. Although SVG code is readable by humans, automated handling of SVG code reduces the risk of manual mistakes in this process.

In the second experiment, the interpolation process shows unexpected, unsatisfying results. As the interpolation proved to work using other icons as input, the preprocessing steps are reexamined.

First drawing

Second drawing

Simplified drawing (left)
Pickled and unpickled drawing (right)

Simplified drawing (left)
Pickled and unpickled drawing (right)

Conclusions

It is visible that in this step already - not only in the interpolation process - the geometry of the two drawings changes. Additionally, some small shape gets lost in the pickling process.

What exactly happens in this pickling step, in the process of object serialization (converting an object hierarchy into stream of bytes using the pickle module in Python), is not visible - as the output of this process is code that is not readable for humans. The results of this pickling process are only visible, when unpickling those .pkl files again.

Conclusion of the conversation

4.5 Composition/pattern creation

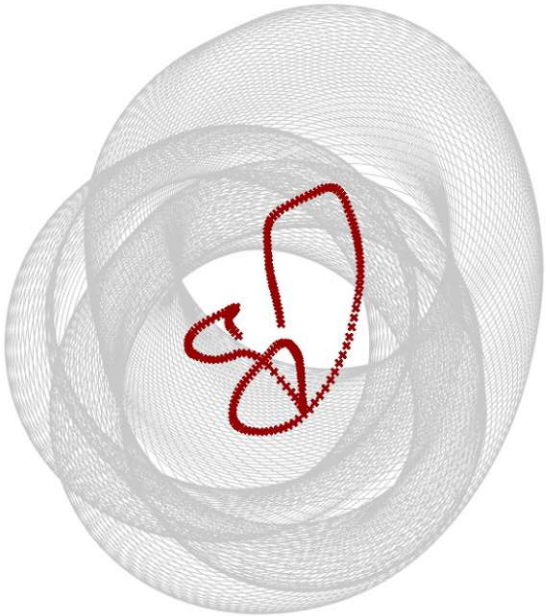
Prerequisites

Each user-machine dialogue concludes at a point and a certain geometry. In this example, the implementation of this newly generated content into an algorithm for pattern creation is examined.

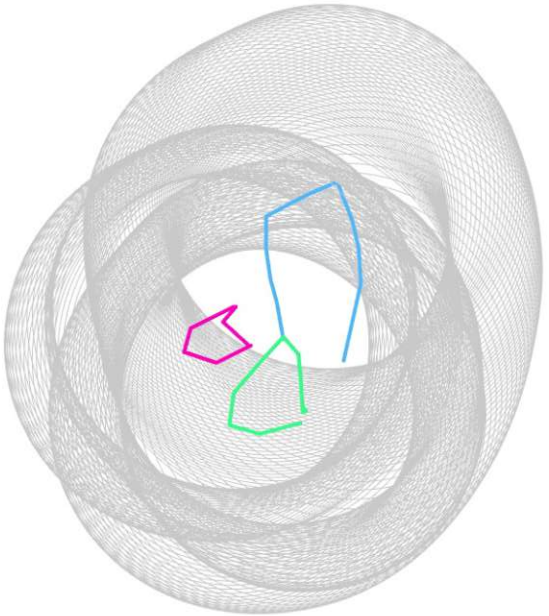
Results

The results of what was tested, has been visually and textually documented on the website https://deepgestures.carrd.co/ as a research portfolio in section 5 on composition/pattern creation. The result can be seen on the following pages.

Point subdivision



Comparison
input/output



Composition

Conclusions

The output of the composition quickly adds complexity to the relatively simple input curves (which are the result of the conversation of user and machine of chapter 4.3).

This creation canm be visualized on a screen during the user experimentation. Another option would be to make use of the second z-axis, use a brush filled with paint and a sheet of paper to paint it.

This composition uses some lines code in Grasshopper and C#. Some variables have to be predefined, such as the radii of the circles. This could also be made interactive, bearing in mind that some dependent processes, like the scaling of the drawing, have to be automated.

Iteration 1

First drawing

Second drawing (left)
result image tracing
(right)

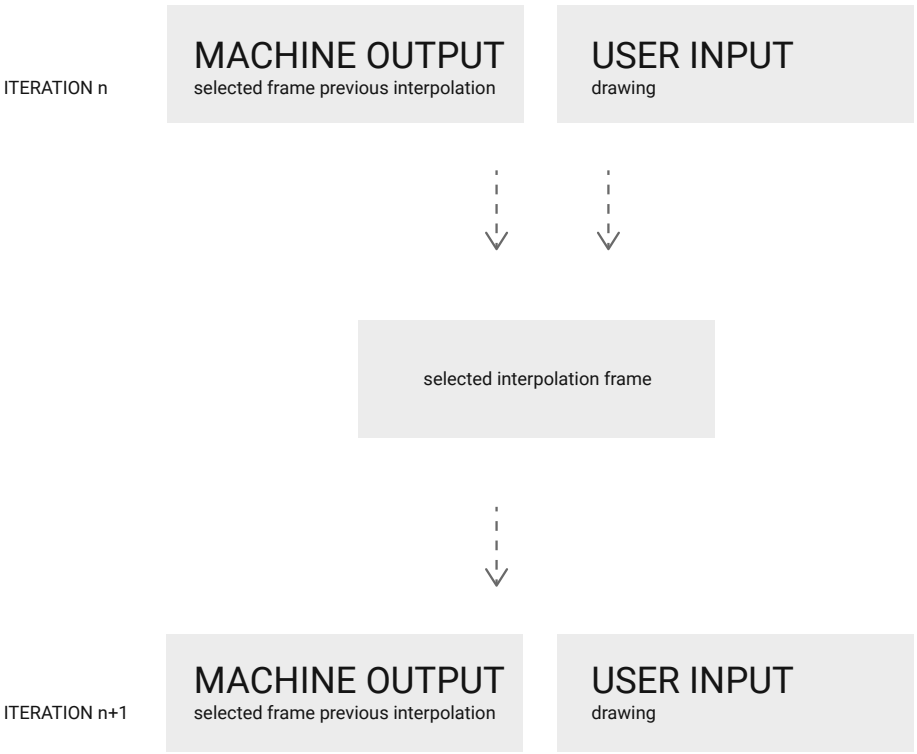
4.6 Superimposed drawings

Prerequisites

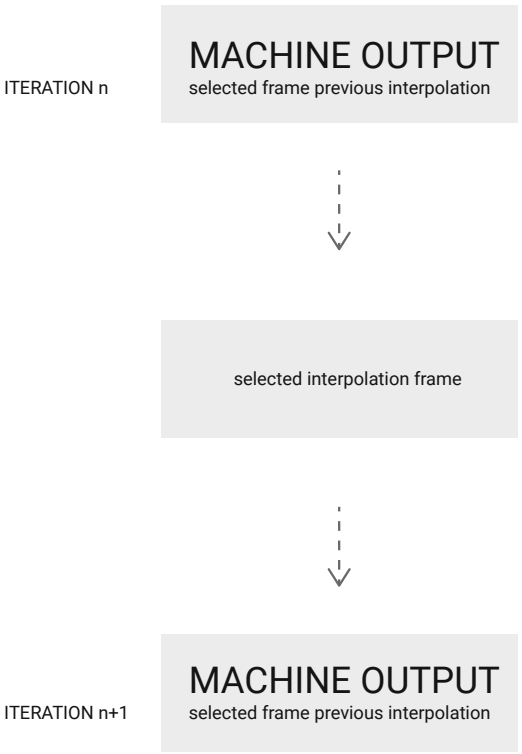
One conversation in this mode is registered. The conversation starts with one icon from the Icon8-Dataset one the machine side. The user draws a response and after that, it is tested, if the machine can take its own drawings as input for the next iteration (concept and distinction to the setup of the previous experiments, see diagram above).

Results

The results of what was tested, has been visually and textually documented on the website https://deepgestures.carrd.co/ as a research portfolio in section 6 on superimposed drawings. A visualization of this conversation will be presented on the next pages.



PREVIOUS EXPERIMENTS

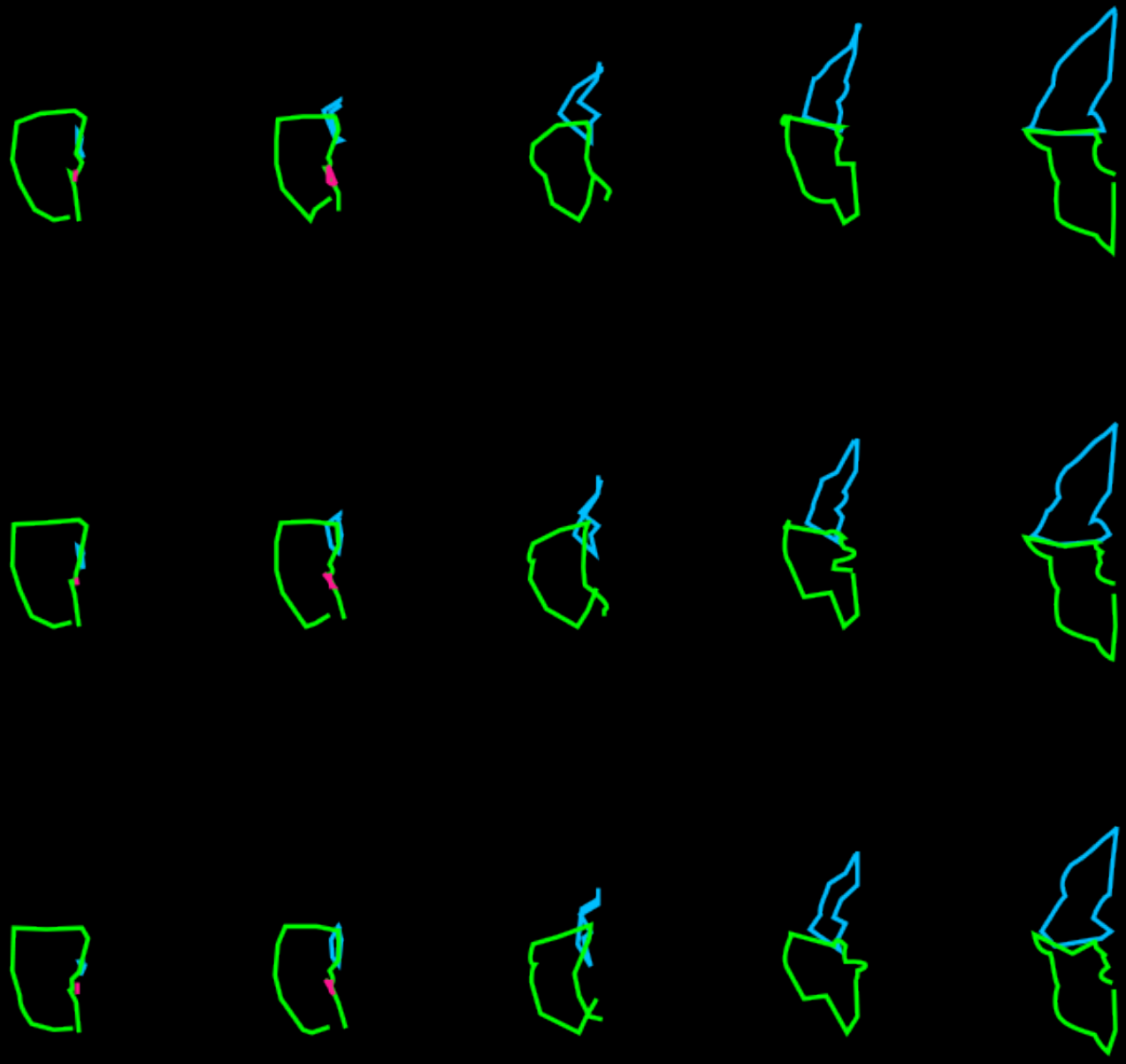


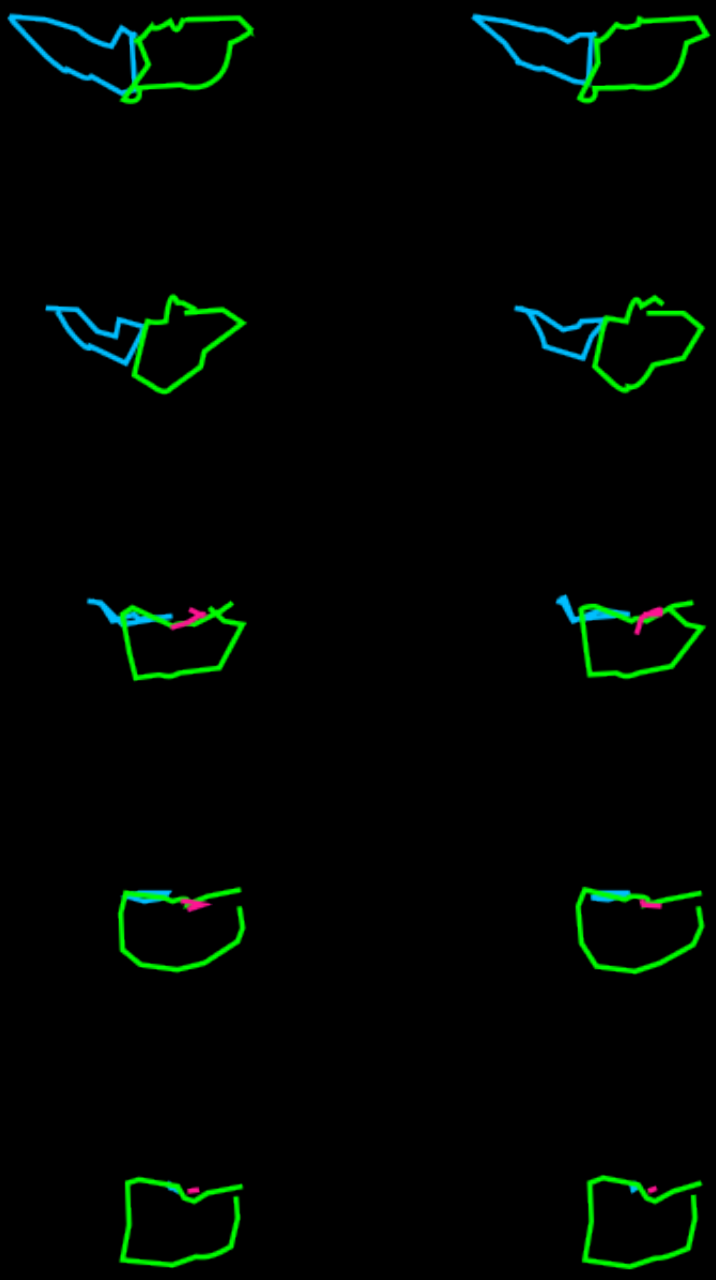
NEXT SET OF EXPERIMENTS

Opposite page and this page:

Superimposed drawings: same output as input.

The diagram shows the difference between the last and the next sets of experiments.

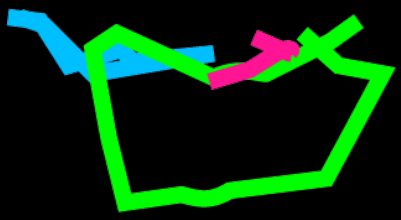




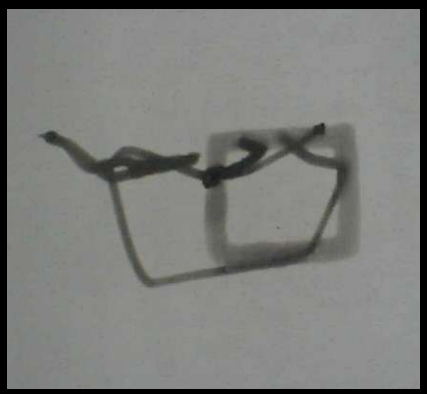
Interpolation frames of a human-machine dialogue with superimposed drawings. The conversation starts with one icon from the Icon8-Dataset one the machine side. The user draws a response and the machine takes its own drawing as input for the next iteration.

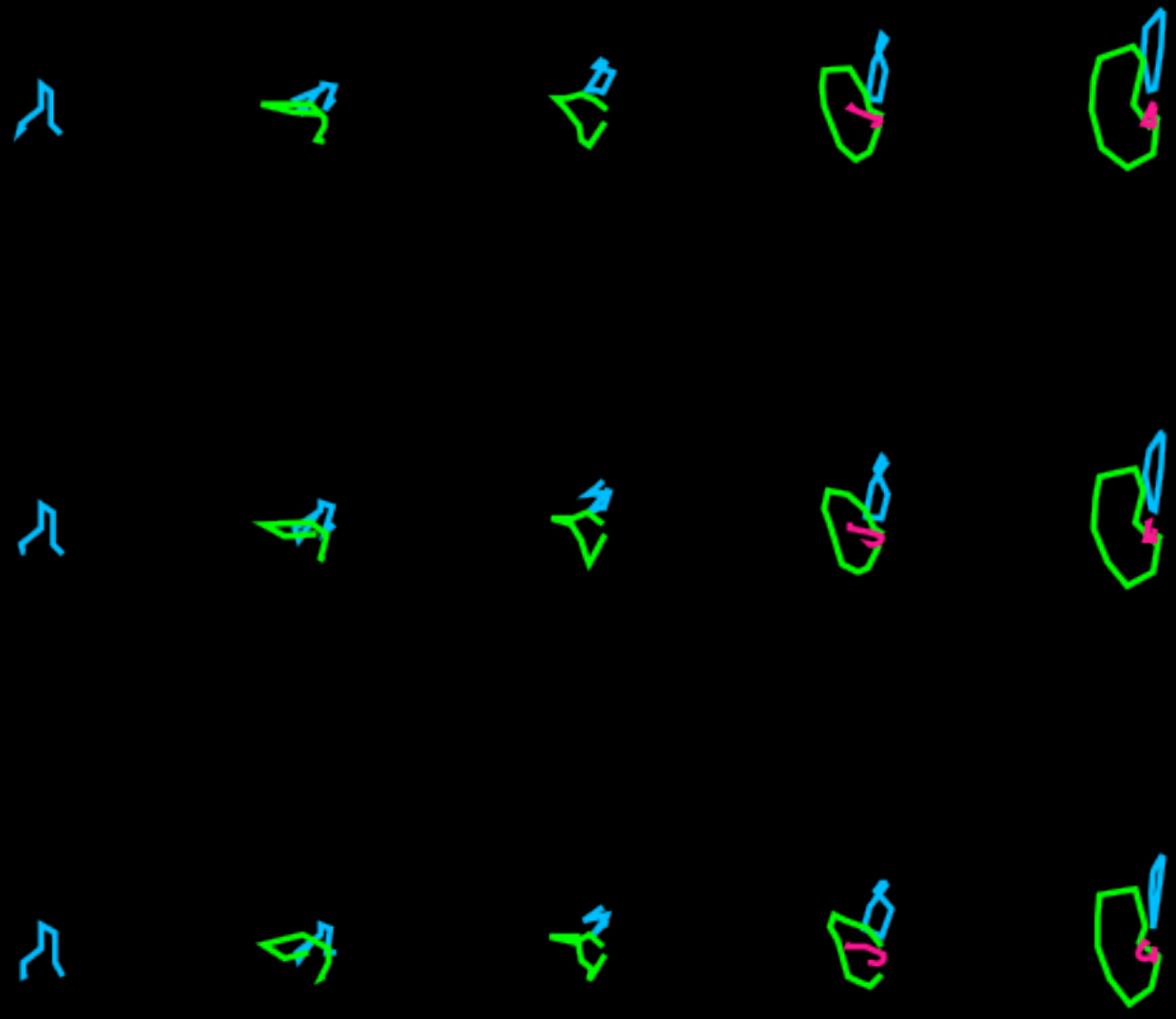
Iteration 2

Selected frame 13 as first drawing



Machine drawing (left) result image tracing (right) as second drawing

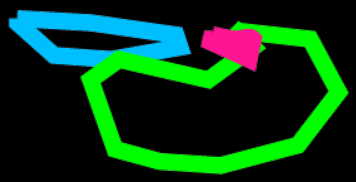




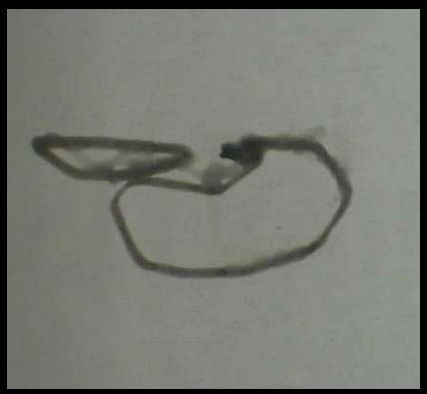
Interpolation frames of the second iteration of a human-machine dialogue with superimposed drawings.

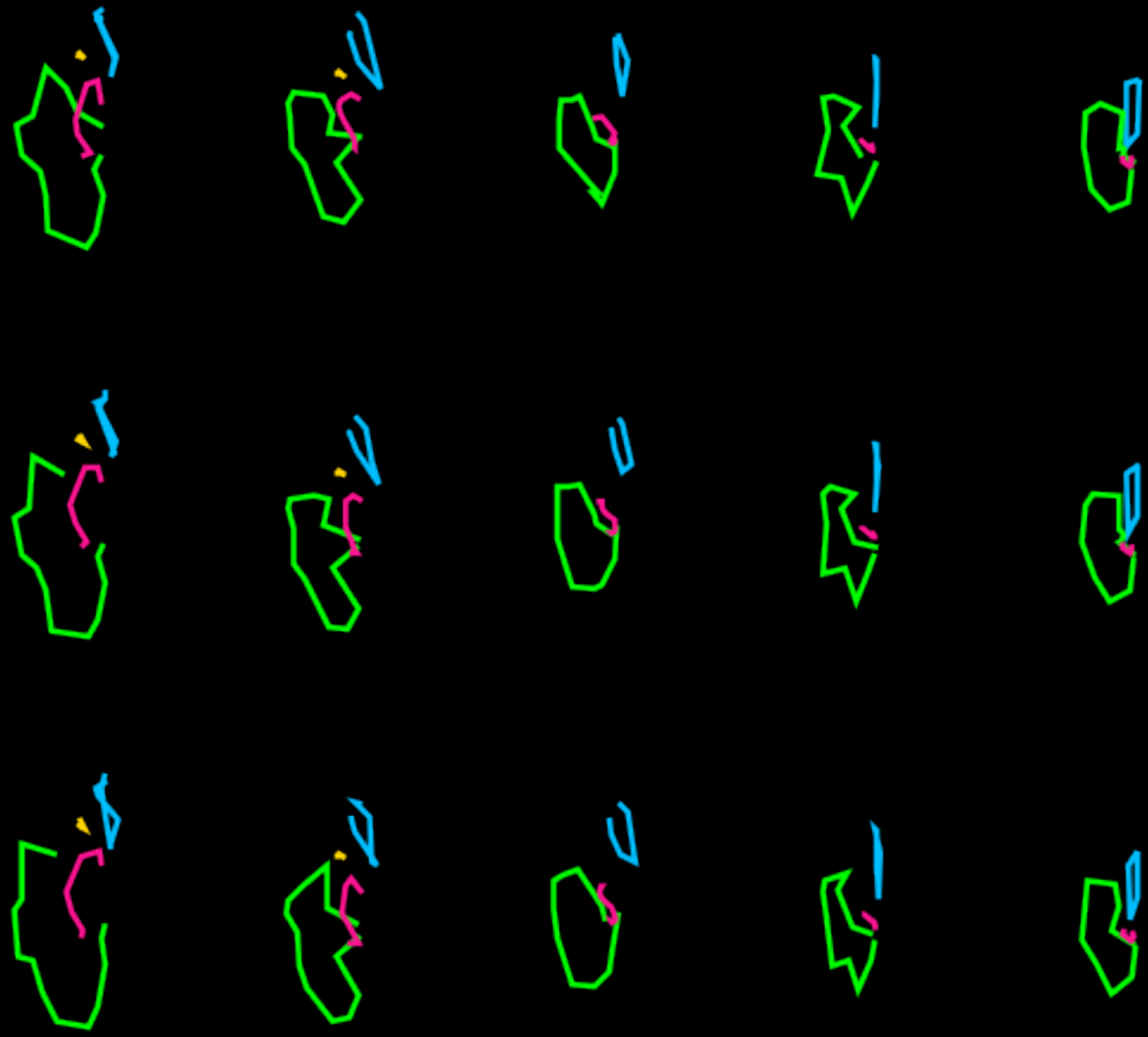
Iteration 2

Selected frame 0 as first drawing



Machine drawing (left) result image tracing (right) as second drawing





Interpolation frames of the third iteration of a human-machine dialogue with superimposed drawings.

Conclusions

This set of experiments was conducted in order to gain a better understanding of what happens in the course of a conversation between a user and the machine adding complexity through superimposed drawings.

The user is still leading the conversation by selecting the frame on which the next iteration is based. The dialogue can not be led into another preferred direction easily or be altered completely. It is an open-ended process of interaction that can at any iteration be concluded or picked up again, as all the frames are stored in the digital memory.

Conversation end:
Machine drawing of
selected frame 22

4.7 Current limitations

4.7.1 Hardware limitations

Because of the time it takes once the snapshot is taken of the user's drawing to process and preprocess data, feed it into the network and perform the interpolation, there is a slight downside to interactivity and user feedback.

Comparing user input and machine output side to side, it is visible that the brush sizes vary in thickness. On the user side, it is essential to draw using a thicker brush and more water in order to have enough time for taking the snapshot. In addition to that, the drawing sensation is a different one using a thicker brush that might allow some inaccuracy and corrections in the drawing process.

The thin brush is used with machinic precision, albeit drying very fast.

4.7.2 Software limitations

The DeepSVG network has been trained based on the SVG-Icons8 dataset. Interpolations with SVG files with more than eight vector paths are not possible. Using physical input, in the form of a user drawing, only four- strokes or vector paths are the limit for the interpolation. In the first case, using more than eight paths and digital input, the interpolation fails. In the second case, using more than four paths and physical input, only four paths are displayed as the result of the interpolation process.

As the order of the paths of the input files has not been regarded, the interpolation process of some figurative drawings possesses interesting complexity but lacks replicability. A solution to this is posed in paragraph 4.4.

5 Conclusions

This final chapter presents the conclusions drawn from the experimental results and illustrates ideas for further developments and implementations.

Conclusions

The presented research project investigated the technical and creative space of possibilities situated at the intersection of novel Machine Learning techniques and established Numerical fabrication technologies, with a focus on graphical applications in the 2-dimensional space.

The majority of Machine Learning applications in the creative field is currently focused on the processing of pixel-based images or texts. As part of this work, the potential of a Hierarchical Generative Network for the creation of vector files in the SVG format (DeepSVG) was investigated. As opposed to image creation in raster format, vector representation opens the door to digital production, enabling a direct translation of geometry to CNC machine operations.

The experimental focus of this work has been on the design and development of a custom-made interactive computational system in its hardware and software components. To validate such technical developments, the system was creatively explored and critically analyzed in its behaviors and responses to a variety of user inputs.

By critically looking at the process of interaction between user and machine and by rejecting the conception of computers as passive receivers of pre-conceived creative impulses, this work framed the machine as active instrument which is capable of perceiving and processing external inputs to generate new content, while at the same time it challenged and re-defined the user in its modern role of creative author.

Experiments

The set of experiments conducted in the scope of this work aimed at documenting and evaluating the process of interaction between user and machine, with the main objective of playfully exploring the system to unravel its inner

logics and generative potential and, concurrently, of testing its technical and performative response.

The results showed that the system was able to add further complexity even during simple and abstract interpolations. Some of those serendipitous and unpredictable frames emerge during the latent space operation in the deep neural network. In some cases, the process of image tracing adds new aspects to the initial geometries. The addition of complexity, however, comes at the cost of comprehensibility, as a realistic image of the interpolation is not drawn in that case.

Some geometric and semantic relations could only be grasped using figurative input. Interpolating two icons of comparable content showed that the order of the respective SVG paths was important and should correspond, as neglecting this aspect would alter the semantic content of the interpolation results completely. An automatic sorting algorithm during the preprocessing step is proposed.

Initial experiments culminated in a dialogue between user and machine. The user led the conversation by selecting the topic as well as by affecting the machine's response in each iteration. This open-ended process of interaction could be concluded or picked up again at any time, as all the machine's answers in the form of interpolation frames are stored in the digital memory.

On a more technical note, the results of the experiments showed that, in the case of manually drawn input, the DeepSVG network is able to process an SVG file of four paths maximum, whereas the developers of DeepSVG state that the network would be able to process input with up to eight paths (Carlier, 2020). Even if this circumstance does not undermine the concept, it certainly poses a limit to the user's creative possibilities.

Final Application

The last interpolation frame of the human-machine dialogue serves as input for the final experiment. Deriving from the chosen conversation topic, this application was the purpose of having this conversation in the first place.

Conceptualized in virtual space, the output of this application can be actualised in a multitude of forms depending on a variety of variables, such as its realization in 2- or 3-dimensional space, on its scale or materiality. This form provides an indication on who the user might be. In this context, one could imagine an architect, artist or designer exploring the generative aspect of the machine.

The role of the designer

In the process of employing (or developing) a deep neural network as a creative tool, the designer has to challenge the knowledge about her role as a creator and reflect on the design intention on a broader scale. Based on the direct experience of this research, I saw the role of the designer being affected in two ways:

Firstly, by expanding her level of engagement, the designer stops from being solely a focused task-solver looking for a specific solution to a well posed design problem. She also becomes the creator of a higher level system which enables a new type of cooperative design. This function can comprise a larger set of tasks - such as the curation of a dataset at the beginning of the process, the evaluation of a trained model or the fabrication at the end of the "outsourced" process.

Secondly, the role of the designer who is exposed to such systems, is bouncing back and forth from the one of a playful explorer to one of a focused exploiter. The appropriation of and the continuous interaction with a complex system not only fosters a deeper rational understanding on how these complex networks process data, but also provides a real learning environment for the designer who is exposed to a variety of initially unexpected inputs, which overtime she will eventually incorporate and transform in a novel design sensibility.

5.1 Further implementations

Although this project can be considered concluded in its prototypical state, research could be expanded in multiple directions.

Performance

As certain calculations performed by the network involve rather heavy and time-expensive computation, it would not be true to describe the drawing experience as real-time. In halting and non-fluent moments, trains of thought cannot be immediately processed, leading to moments of excessive reflection or irritation, significantly affecting the user's experience.

In order to minimize the delay and to bring the interaction closer to real-time, the computational performance would have to be reviewed. The processing of code could be improved by using more powerful processors, switching to another programming language, or, most importantly, by fine tuning and implementing new computational strategies.

Alternative networks

The application of SketchRNN - based on Google's Quick Draw Dataset - was considered in the scope of this project. One advantage would be a faster response to user input as well as the introduction of categories of sketches. As the network is based on TensorFlow, a new environment and new libraries would have to be introduced.

Customized dataset

The creators of DeepSVG chose a very large (and thus unspecified) dataset of icons as the basis for training of the network. One next consequent step would be the curation of an alternative, customized dataset in order to train and employ a new network with a more specific semantic, aesthetic or disciplinary target.

6 Bibliography

List of Figures

If the figure is not included in this list, it is provided by the author.

Premise

Figure 1.01	Ivan Sutherland's Sketchpad, 1962. https://tamarind.unm.edu/wp-content/uploads/Ivan_Sutherland1962.jpg accessed 13 January 2022
Figure 1.02	Connected pen plotter as passive receiver. Sutherland, 2003, Sketchpad, A man-machine graphical communication system, Technical report based on dissertation submitted in January 1963, Cambridge, page 21
Figure 1.03	Machine as Actuator https://upload.wikimedia.org/wikipedia/commons/thumb/e/e6/Automatisches_Zeichengeraet_ZUSE_Z64_ubt.JPG/1280px-Automatisches_Zeichengeraet_ZUSE_Z64_ubt.JPG accessed 4 January 2022
Figure 1.04	Gravel, 1968 (left). https://raw.githubusercontent.com/LindomarRodrigues/Georg-Nees-Schotter-Python/master/Lindomar%20Rodrigues%2C%20Schotter.png accessed 13 January 2022
Figure 1.05	Code to Gravel written in the language ALGOL (right). http://cmuems.com/2013/a/wp-content/uploads/sites/2/2013/09/weiss-fig4-493x480.jpg accessed 13 January 2022
Figure 1.06	Omnia per Omnia, Sougwen Chung, 2018. https://sougwen.com/wp-content/uploads/2018/05/sougwen-2018_omnia_03.jpg accessed 13 January 2022
Figure 1.07	Machines as Collaborators. https://sougwen.com/wp-content/uploads/2018/05/sougwen-2018_omnia_02.jpg accessed 17 January 2022

Theoretical Framework

Figure 2.01	Applications of Machine Learning, the Nooscope Manifest , Pasquinelli and Joler, 2020 Pasquinelli and Joler, 2020, The Nooscope Manifested: Artificial Intelligence as Instrument of Knowledge Extractivism, Essay, page 14
Figure 2.02	Scheme of a Perceptron. https://miro.medium.com/max/1068/1*Z1_IgFO1c6tq4Tz1iwJraw.png accessed 5 January 2022
Figure 2.03	Classification of Deep Learning as a subcategory of Machine Learning. https://miro.medium.com/max/1400/1*JVbomzzzOuV7rhU3ErGBrw.jpeg accessed 5 January 2022
Figure 2.04	Scheme of a Multilayer Perceptron. https://static.packt-cdn.com/products/9781787121089/graphics/B12043_02_04.png accessed 17 January 2022

Figure 2.05	Data size benchmark Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. Cambridge, MIT Press, Chapter 1, Introduction, page 19
Figure 2.06	Different feature engineering processes for ML and DL. https://quantdare.com/wp-content/uploads/2019/06/deep_learning.png accessed 13 January 2022
Figure 2.07	Schematic diagram of a feature extraction process. https://miro.medium.com/max/800/0*sQzmiOf8Yb_18HX1.png accessed 13 January 2022
Figure 2.08	Scheme of a Deep Autoencoder. https://hackernoon.com/hn-images/1*8ixTe1VHLsmKB3AquWdxpQ.png accessed 17 January 2022
Figure 2.09	The GAN Pipeline. https://github.com/udacity/deep-learning-v2-pytorch/raw/661c38e1c6a1f6734c26e-cd899b958533c41cf1f/gan-mnist/assets/gan_pipeline.png accessed 5 January 2022
Figure 2.10	*"Nice scarf! Knitted it yourself?"*, 2017 Generative Art by Mario Klingemann using Generative Adversarial Networks. https://i.pinimg.com/originals/d8/96/de/d896de6bfc31cddaca2ce17636260a91.png accessed 6 January 2022
Figure 2.11	Schematic diagram of a Hierarchical Generative Network. Carlier, A., Danelljan, M., Alahi, A., Timofte, R., 2020, DeepSVG: A Hierarchical Generative Network for Vector Graphics Animation, arXiv:2007.11301v3, page 5
Figure 2.12	The 3D-printed stainless steel bridge was designed by Joris Laarman and built by MX3D using six-axis robotic arms equipped with welding gear. (Amsterdam, 2021) https://starts-prize.aec.at/wp-content/uploads/2018/05/Amsterdams3DPrintedSteel-Bridge3.jpg accessed 21 March 2022
Figure 2.13	Machine instructions were encoded on punch tape. Pease, W., 1951, An Automatic Machine Tool, Scientific American Inc, page 101
Figure 2.14	An Automatic Machine Tool, Pease, 1952. Pease, W., 1951, An Automatic Machine Tool, Scientific American Inc, page 102
Figure 2.15	Interior view of the finished brick facade of the Winery Gantenbein by Gramazio & Kohler, 2006. https://gramaziokohler.arch.ethz.ch/data/ProjectImages/02_Web/M/036/060823_036_Dokumentation_Ralphfeiner_006_WM.jpg accessed 13 January 2022 Copyright 2016, Gramazio Kohler Research, ETH Zürich, Switzerland
Figure 2.16	Automated robotic fabrication. https://www.aic-iac.org/wp-content/uploads/Element-production.jpg accessed 13 January 2022 Copyright 2016, Gramazio Kohler Research, ETH Zürich, Switzerland
Figure 2.17	In-Situ Fabricator, Giftthaler, Sandy, Dörfler et al., 2015. https://punkt4.info/fileadmin/user_upload/bauroboter_e.jpg accessed 13 January 2022
Figure 2.18	Ambient point cloud. https://media.springernature.com/lw685/springer-static/image/art%3A10.1007%2Fs41693-017-0003-5/MediaObjects/41693_2017_3_Fig2_HTML.jpg accessed 13 January 2022
Figure 2.19	SEEK, Negroponte and The Architecture Machine Group, 1970. http://cyberneticzoo.com/wp-content/uploads/Seek_p3-x640.jpg accessed 13 January 2022

Figure 2.20 Mimus' installation at the Design Museum in London, Gannon, 2017.
https://www.cmu.edu/news/stories/archives/2017/february/images/mimus_853x480-min.jpg.jpeg
accessed 6 January 2022

Figure 2.21 Visual feedback of the visitors.
https://images.squarespace-cdn.com/content/v1/5758289d27d4bdf581e1c031/1479333536100-LXSVWM0MZKS9JS5A0XPP/bridge_early_screengrab.jpg?format=750w
accessed 13 January 2022

Figure 2.22 Stochastic assembly, Wu and Kilian, 2018.
https://media.springernature.com/original/springer-static/image/chp%3A10.1007%2F978-3-319-92294-2_2/MediaObjects/450164_1_En_2_Figa_HTML.png
accessed 17 January 2022

Figure 2.23 Positioning of wooden logs using an industrial robotic arm.
https://media.springernature.com/original/springer-static/image/chp%3A10.1007%2F978-3-319-92294-2_2/MediaObjects/450164_1_En_2_Fig5_HTML.png
accessed 17 January 2022

Figure 2.24 NORAA - an interactive drawing installation was created by Jessica In in 2018.
https://images.squarespace-cdn.com/content/v1/58b2e5605016e199ea87f8d3/1540675105287-4KR87YCA4NDNGRKQ5212/31036081008_345929067c_o_sm.jpg
accessed 13 January 2022

Figure 2.25 Schematic sketch of installation concept.
https://images.squarespace-cdn.com/content/v1/58b2e5605016e199ea87f8d3/1551096706381-UU0JURBOC8HNKEKSFXD2/190216_2D_Choreography2-01_medRes.png?format=1000w
accessed 13 January 2022

Implementation

Figure 3.01 Horst Rittel's design processes of generating variety and reducing variety.
Gänshirt, C., 2021, Tools for Ideas, Birkhäuser, Basel, page 80

Figure 3.02 Arduino Pro Mini Pinout.
https://www.eitkw.com/wp-content/uploads/2019/11/promicropinout.jpg
accessed 28 January 2022

Figure 3.03 Python logo.
https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3/Python-logo-notext.svg/1024px-Python-logo-notext.svg.png
accessed 31 January 2022

Figure 3.04 Anaconda logo.
https://www.generalcatalyst.com/wp-content/uploads/2017/10/logo-anaconda-372x372-1.png
accessed 31 January 2022

Figure 3.05 Visual Studio Code logo.
https://upload.wikimedia.org/wikipedia/commons/thumb/9/9a/Visual_Studio_Code_1.35_icon.svg/2048px-Visual_Studio_Code_1.35_icon.svg.png
accessed 31 January 2022

Figure 3.06 UDP Server and Client communication in theory.
https://media.geeksforgeeks.org/wp-content/uploads/UDP.png
accessed 31 January 2022

Figure 3.07	TouchOSC logo. https://hexler.net/site/images/app_icons/png_white/TouchOSC-icon.png accessed 31 January 2022
Figure 3.08	DeepSVG logo. https://repository-images.githubusercontent.com/281817032/68d82800-cc83-11ea-846c-1fe78e3ef266 accessed 31 January 2022
Figure 3.09	Rhino3D logo. https://www.filou.de/wp-content/uploads/2020/12/Rhino7-Logo-2_200.jpg accessed 31 January 2022
Figure 3.10	Grasshopper logo. http://3.bp.blogspot.com/_ZKKPQHRaR0I/S_VP9YeSoqI/AAAAAAAAApw/J4pdVe9_eHk/s400/Grasshopper_logo.png accessed 31 January 2022
Figure 3.11	Skeleton Tracing logo. https://user-images.githubusercontent.com/7929704/79626790-c39c3980-8100-11ea-82c8-3da4380c1128.png accessed 31 January 2022
Figure 3.12	Github logo. https://logosmarken.com/wp-content/uploads/2020/12/GitHub-Logo.png accessed 31 January 2022

References

Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. Cambridge (EE. UU.): MIT Press.

Armstrong, H., 2021, Big Data, Big Design, Why Designers Should Care About Artificial Intelligence, Princeton

Matteo Pasquinelli and Vladan Joler, "The Nooscope Manifested: Artificial Intelligence as Instrument of Knowledge Extractivism", KIM research group (Karlsruhe University of Arts and Design) and Share Lab (Novi Sad), 1 May 2020 (preprint forthcoming for AI and Society). https://nooscope.ai

LeCun, Y., Cortes, C., Burges, C., MNIST handwritten digit database, [online] Available at: <http://yann.lecun.com/exdb/mnist/> [Accessed 8 January 2022]

Patel, H., 2021, What is Feature Engineering — Importance, Tools and Techniques for Machine Learning, [online] Available at: <https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10> [Accessed 17 January 2022]

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014, Generative Adversarial Nets, Technical paper, arXiv:1406.2661v1

Klingemann, M., 2019, Interview with Mario Klingemann, [online] Available at: <https://www.artmarket.guru/le-journal/interviews/mario-klingemann/> [Accessed 13 January 2022]

Cache, B., 1995, Earth Moves: the furnishing of territories, MIT Press, Cambridge, Mass.

Carlier, A., Danelljan, M., Alahi, A., Timofte, R., 2020, DeepSVG: A Hierarchical Generative Network for Vector Graphics Animation, Technical paper, arXiv:2007.11301v3

Pease, W., 1951, An Automatic Machine Tool, Scientific American Inc

Heads of the MIT Servomechanisms Laboratory, Prepared by the Institute Archives, MIT Libraries
September 2004, [online] Available at: <https://libraries.mit.edu/mithistory/research/labs/mit-servomechanisms-laboratory/> [Accessed 13 January 2022]

Gramazio & Kohler (2009), Facade Gantenbein Winery, Non-Standardised Brick Facade, Vol. 8

Giftthaler, M., Sandy, T., Dörfler, K., Brooks, I., Buckingham, M., Rey, G., Kohler, M., Gramazio, F., Buchli, J., 2017, Mobile Robotic Fabrication at 1:1 scale: the In situ Fabricator, System Experiences and Current Developments, Technical paper, arXiv:1701.03573v1

Gannon, M., 2018, Human-Centered Interfaces for Autonomous Fabrication Machines, Doctoral Thesis, Carnegie Mellon University

Wu, K., Kilian, A., 2019, Designing Natural Wood Log Structures with Stochastic Assembly and Deep Learning, Copyright Springer Nature Switzerland AG 2019, J. Willmann et al. (Eds.): ROBARCH 2018, Robotic Fabrication in Architecture, Art and Design 2018, pp. 16-30, 2019.

Magenta, Sketch RNN, 2018 [online] Available at: <https://magenta.tensorflow.org/assets/sketch_rnn_demo/index.html> [Accessed 13 January 2022]

GoogleCreativeLab, Quick Draw Dataset , 2017 [online] Available at: <https://github.com/googlecreativelab/quickdraw-dataset> [Accessed 13 January 2022]

In, J., NORAA , 2018 [online] Available at: <https://www.jessicain.net/pagesnoraa> [Accessed 13 January 2022]

Huang, L., Skeleton Tracing, Frank-Ratchye STUDIO for Creative Inquiry at Carnegie Mellon University, [online] Available at: <https://skeleton-tracing.netlify.app/> [Accessed 31January 2022]

UDP - Client And Server Example Programs In Python, [online] Available at: <https://pythontic.com/modules/socket/udp-client-server-example> [Accessed 31 January 2022]

Gänshirt, C., 2021, Tools for Ideas, Birkhäuser Verlag GmbH, Basel

Anaconda (Python distribution), [online] Available at: <https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)> [Accessed 22 March 2022]

Carlier, A., 2020, [online] Available at: <https://github.com/alexandre01/deepsvg/issues/15> [Accessed 8 February 2022]