Higher Order

Material Feedback in Robotic Extrusion

Masterarbeit

Higher Order

Material Feedback in Robotic Extrusion

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Diplom-Ingenieurs / Diplom-Ingenieurin
unter der Leitung von

Christian Kern
Univ.Prof.Arch.Dipl.Ing.
E264/2 Institut für Kunst und Gestaltung

eingereicht an der Technischen Universität Wien
Fakultät für Architektur und Raumplanung

von

Lidia Atanasova
00827335
Alliiertenstraße 16/23
1020 Wien

Wien, am

Unterschrift

Abstract

This work investigates possibilities of integrating physical environment into digital by allowing for deviations originating from varying material properties to gradually inform a custom fabrication strategy.

Current material practices in architecture invest immense efforts in exploring new materials and determining their behaviour. Nevertheless the way of manipulating them remains unchanged, imposing materials a given shape rather than triggering its spontaneous behaviour. This results primarily from the division between design and fabrication processes. Decisions concerning the manufacturing are taken in advance and the fabrication is a mere execution of machine instructions, requiring that materials perform in a predictable fashion. However processes could be responsive and adapt to different conditions and react to changes.

The presented work examines the potentiality of robotic fabrication based on additive manufacturing for iterative material aggregation informed by physical material behaviour. In a series of case studies material feedback proved crucial not only for correcting tool-paths and thus compensating for deviations but it enables a more general approach to materiality and formation processes.

A plastic extrusion head deposits material in 3D space guided by a 6-axis industrial robot that is been remotely controlled. This setup has been extended by additional devices inputting information from the physical environment to the external control system. Due to the unpredictable material behaviour feedback — position of structure parts in space — from the deposited layer has been introduced to the process defining the subsequent tool-paths for the robotic end effector and hence the areas of material accumulation. This facilitates the customisation of a robotically controlled additive manufacturing where the design is not predefined and a result of automated fabrication, but the material is affecting the robotic tool-paths and thus participating in the genesis of form.

Contents

Introduction

“Steel, especially mild steel, might euphemistically be described as a material that facilitates the dilution of skills… Manufacturing processes can be broken down into many separate stages, each requiring a minimum of skill or intelligence… At a higher mental level, the design process becomes a good deal easier and more foolproof by the use of a ductile, isotropic, and practically uniform material with which there is already a great deal of accumulated experience.”[1]

[1] James Edward Gordon. The Science of Structures and Materials. (Scientific American Library, 1988). p. 18

Materiality in Fabrication

Current material practices in architecture invest immense efforts in exploring new materials and determining their behaviour. Industrial metals for instance have experienced an intense process of uniformity and homogenization. This was partly based on questions of reliability and quality control, but it had also a social component:

> **“both human workers and the materials they used needed to be disciplined and their behaviour made predictable. Only then the full efficiencies and economies of scale of mass production techniques could be realized.”[2]**

Moreover this requires a division of design and fabrication processes. Decisions concerning the manufacturing are taken in advance and the fabrication is a mere execution of (machine) instructions using materials that perform in a predictable fashion.

This implies that material practices reduce processes to simple routines imposing materials a given shape rather than triggering its spontaneous behaviour. However fabrication processes could be flexible and responsive — adapt to different conditions and react to changes — allowing intrinsic physical and chemical material properties to actively participate in the design process and hence bridge the gap between design, fabrication and construction.

Material Feedback

Since fabrication deals with physical manipulation of materials, its properties are crucial for the design process; its response to manipulation determines the applied fabrication technique and if for instances material is added or subtracted. Therefore it is the material feedback and the technique of manipulation that define the physical outcome. Material suggests processes and ways of being formed. A fabrication process using just raw material and its inherent properties can be realized using contemporary tools for sensing and extracting data from the material in real time.

Digital tools and technology offer possibilities of bridging the gap between design and fabrication by informing a computational model with data extracted from the physical environment. A way of dealing with material and physical fabrication processes is by introducing material properties on an early design stage allowing them to influence the process and subsequently the final design.

[2] Manuel DeLanda, “Uniformity and Variability: An Essay in the Philosophy of Matter”, 2005

Methodology

Custom Robotic Fabrication

The present work examines the potentiality of robotic fabrication based on additive manufacturing for iterative material aggregation informed by physical material behaviour.

A plastic extrusion head deposits material in 3D space guided by a 6-axis industrial robot that is been remotely controlled. This setup has been extended by additional devices inputting information from the physical environment to the external control system. Due to the unpredictable material behaviour feedback — position of structure parts in space — from the deposited layer has been introduced to the process defining the subsequent tool-paths for the robotic end effector and hence the areas of material accumulation. This facilitates the customisation of a robotically controlled additive manufacturing where the design is not predefined and a result of automated fabrication, but the material is affecting the robotic tool-paths and thus participating in the genesis of form.

Experiment

By means of physical experiments and through variation of predefined parametric values a self-organising behaviour has been deliberately triggered. To achieve “higher order” however a material feedback needs to be introduced. Why? Establishing a link between digital design and physical object so that a real time control over the fabrication and necessary changes resulting from material behaviour is incorporated in the design during fabrication.

In the series of case studies material feedback proved crucial not only for correcting tool-paths and thus compensating for deviations but it enables a more general approach to materiality and formation processes. The developed design and fabrication system allows for material feedback to flow into the design process and participate in the genesis of form equally as design idea and procedure.

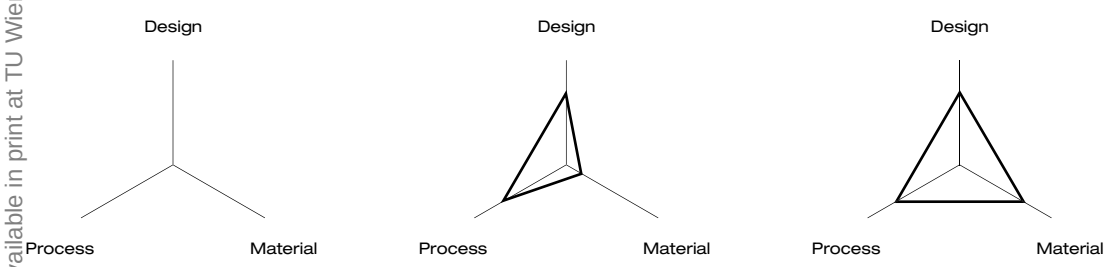
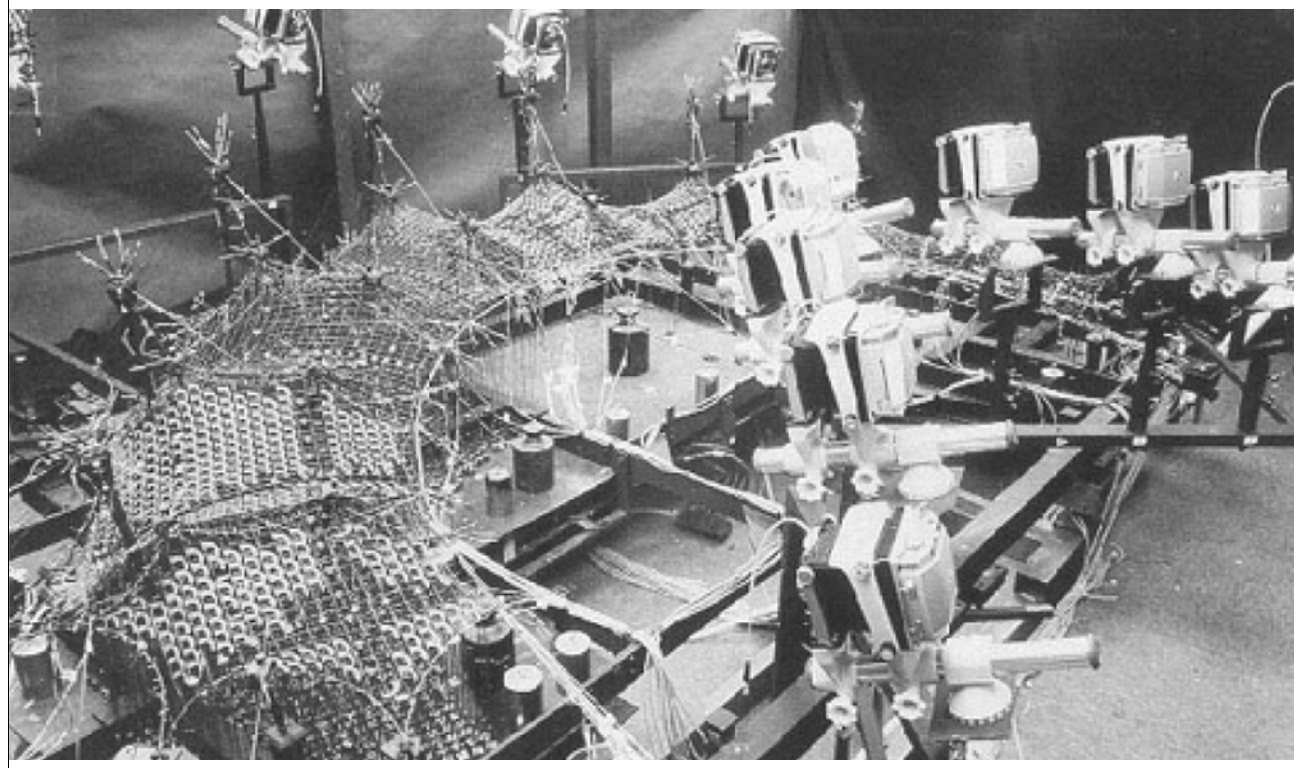


Figure 01. Frei Otto, Analogue Form-finding techniques for Olympia Stadium, Munich



Background Theories

“Architecture is one of the most urgent needs of man, for the house has always been the indispensable and first tool that he has forged for himself. Man’s stock of tools marks out the stages of civilization, the stone age, the bronze age, the iron age. Tools are the result of successive improvement; the effort of all generations is embodied in them. The tool is the direct and immediate expression of progress; it gives man essential assistance and essential freedom also.”[1]

This chapter gives a brief overview of the history of computational design and digital fabrication outlining concepts and approaches relevant for this work. Through examples it presents the impact of (technological) advancements in architectural tools for design and fabrication on the contemporary architectural practice and eventually suggests an integration of computational and manufacturing processes in a dynamic design system.

[1] Le Corbusier, Toward an Architecture, 1923

Computation

„The dominant mode of utilizing computers in architecture today is that of computerization; entities or processes that are already conceptualized in the designer's mind are entered, manipulated, or stored on a computer system."[1]

First Digital Tools

Developed as a part of his thesis „Sketchpad: A man-machine graphical communication system" at MIT in 1963, Ivan Sutherland's Sketchpad is the first graphical user interface. Although not conceived for architectural purposes the program is considered as the precursor of the computer aided design (CAD).[2]

Around the same time Nicholas Negroponte conceptualised first ideas about the application of computation for solving architectural problems. Convinced that architects cannot handle neither large scale problems because of their high level of complexity nor small problems since they are too specific and hence trivial, Negroponte identified computing services as a way to liberate designers and provide them with more time to deal with actual design problems.[3]

Furthermore he suggested:

> **"There are three possible ways of having machines assist the design process:**
>
> 1. **Current procedures can be automated, thus speeding up and reducing the cost of existing practices.**
> 2. **Existing methods can be altered to fit within the specifications and construction of a machine, where only those issues are considered that are supposedly machine compatible**
> 3. **The process considered as being evolutionary, can be introduced to a mechanism (also considered as evolutionary), and a mutual training, resilience and growth can be developed."[4]**

Inspired by these ideas, The Architecture Machine Group — led by Negroponte — provided first practical experiments of using a com-

Figure 02. Michael Hansmeyer Subdivided Columns 2010

[1] Kostas Terzidis, Algorithmic Architecture, 2006, p.xi
[2] Ivan Sutherland, Sketchpad: A Man-Machine Graphical Communication System, 1980
[3] Nicholas Negroponte, Toward a Theory of Architecture Machines, 1969
[4] Nicholas Negroponte, Towards a Humanism through Machines, 1969

puter as a design tool rather than a simple manipulator of an existing design concept.

Computerization vs. Computation

These early examples illustrate the current situation and differentiated approaches in the contemporary architectural practice. The dominant use of computers encourages the execution of repetitive tasks and introducing certain level of automation in order to accelerate processes. Nevertheless due to their specific ability to deal with large amount of information and therefore apply individual solutions to specific problems “intelligent” digital tools which could not only assist but participate the design process are been greatly explored. This is facilitated by the growing amount of technological advancements and computational tools in particular made accessible for designers and architects.

Figure 03. Ivan Sutherland’s Sketchpad: Using a „light pen“, a Sketchpad user sketches directly on the screen and can both position parts of the drawing on the display or point to them to change them.

Parametric design

It has mostly been used as a form-finding tool and a way of manipulating form through parameters and their dependencies. Parametric design gives freedom to explore an infinite number of design possibilities and generate fabrication data, apply structural analysis for optimization purposes and facilitate the fabrication and construction of geometrically complex structural components and buildings. Moreover “parameters can provide for a powerful conception of architectural form by describing a range of possibilities, replacing in the process stable with variable, singularity with multiplicity.”[5]

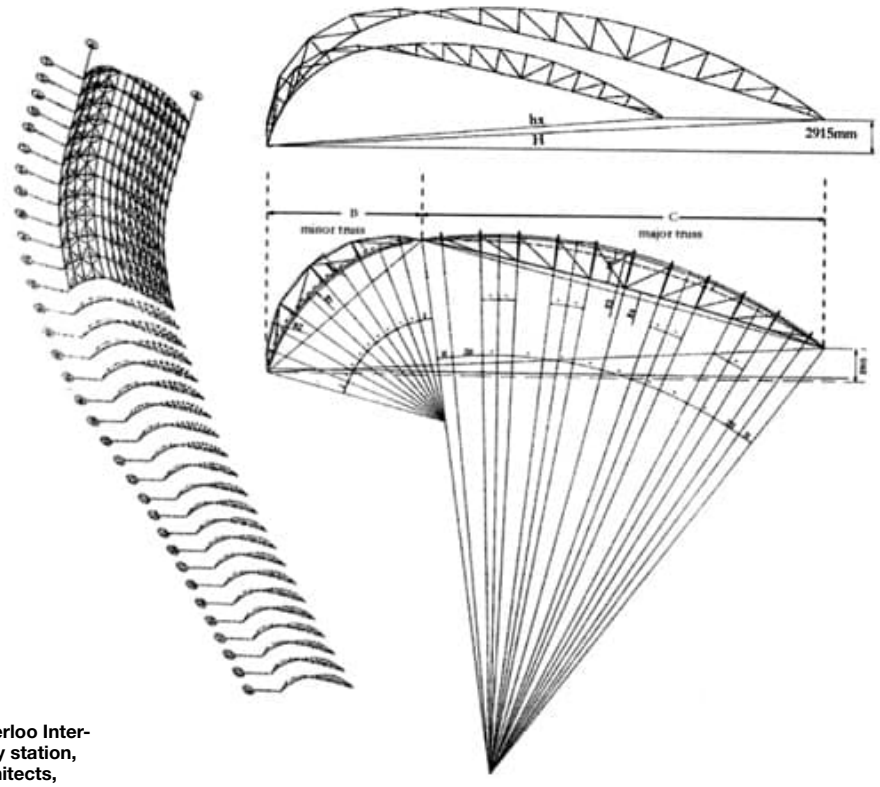


Figure 04. Waterloo International railway station, Grimshaw Architects, 1993, closed Novemver 2007.

[5] B. Kolarevic, Architecture in the Digital Age: Design and Manufacturing, 2004

Regardless of the tools “the ability to define, determine and reconfigure geometrical relationships is of particular value” (Kolarevic, 2001). This can be recognized in the work of Antoni Gaudi and Frei Otto, whose work although analogue was essentially parametric. The method that they used — both applying the catenary phenomenon to their experiments in order to achieve minimum energy state of material systems — is known as “analogue computing”[6].

Figure 05. Hanging chain model by Antonio Gaudi, for Sagrada Familia in Barcelona.

[6] Kolarevic, 2004

Algorithmic Design and Creativity

> “The determinism of algorithms leads to an indeterminism. In this way the character of creativity is an open horizon, even though it is degenerated through a finite number of rules. Creativity means algorithmic design, which needs to be conceived in a way that even the unpredictable, that is occurrences far beyond the subjective horizon of design, can be designed for. Creativity entails algorithmic design plus the complementary designing additional system components. (...) The creative figure will be both the designer of algorithms and the interpreter of their outcomes.”[7]

[7] P. Weibel, Algorithm and Creativity

Digital Fabrication

"…it seems entirely likely that ultimately the inanimate machine can behave more rapidly and more reliably in response to situations that are far too complex for the animate machine to handle."[1]

The advancement and amount of digital tools available for architects and designers are also a result of the need to provide Computer Numerically Controlled (CNC) machines with instructions in order to mill curved geometries. CATIA [2] is the first CAD software initially developed for the aerospace industry, which had all the necessary modelling capabilities and allowed for interface with structural programs.

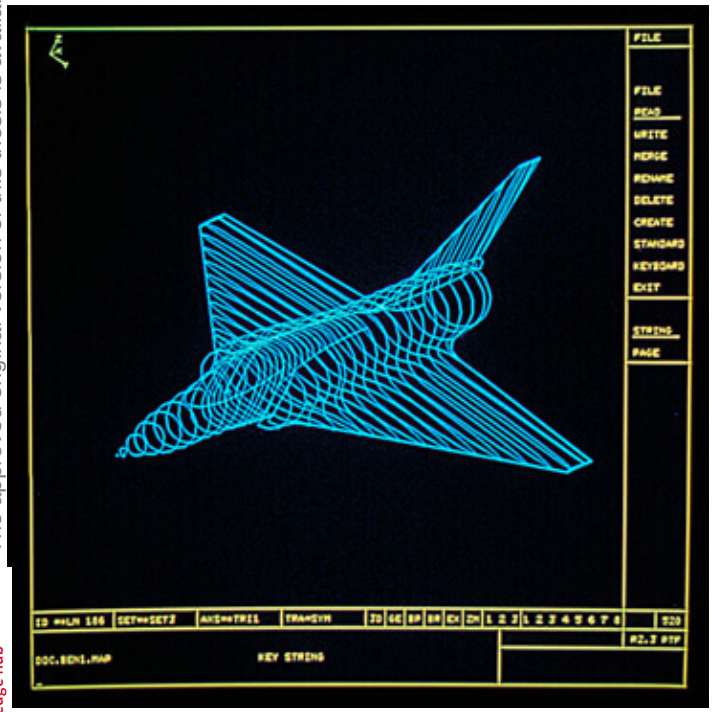


Figure 06. CATIA, Fighter Plane 3D, 1988

[1] - Steve Coons as quoted by Negroponte (1975)

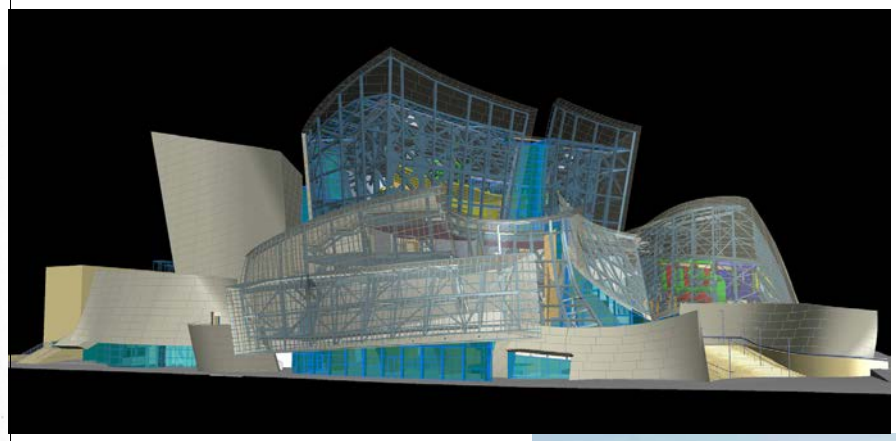
[2] CATIA started as an in-house development in 1977 by French aircraft manufacturer AVIONS MARCEL DASSAULT. It was later adopted by the aerospace, automotive, shipbuilding, and other industries. From CATIA: https://en.wikipedia.org/wiki/CATIA

Digital fabrication as a form of analogue manufacturing controlled by digital means.

"having abandoned the discourse of style, the architecture of modern times is characterized by its capacity to take advantage of the specific achievements of that same modernity: the innovations offered it by present-day science and technology."[3]

[3] Solà-Morales and Whiting, 1997

Gehry's Guggenheim in Bilbao is an example of a design idea that has profited from digital tools for its realisation and not for its shape. Due to its precise control over the position of each point of the numerically controlled model, CATIA was the perfect tool for the realisation of Frank Gehrys Guggenheim Museum in Bilbao. It is the first architectural project that illustrates the potential of the software, suggesting the application of digital tools for architectural purposes. Digitizing Gehry's complex free-form design was crucial for providing digital data for manufacturing.



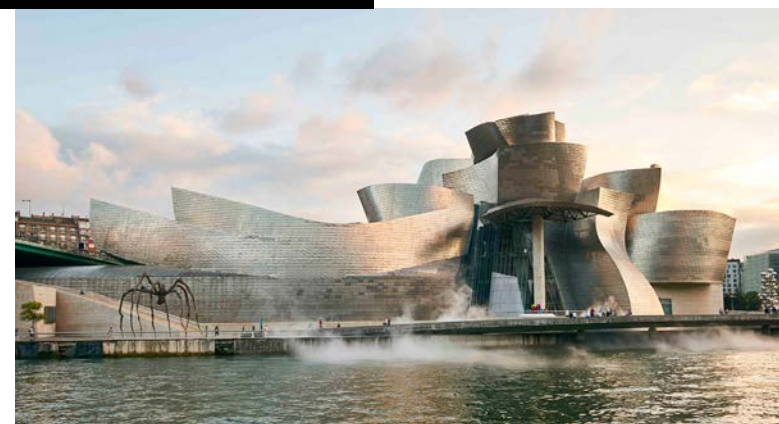


Figure 07. Frank Gehry, Guggenheim Museum, Sketch
Figure 08. Frank Gehry, Guggenheim Museum, Digital Model
Figure 09. Frank Gehry, Guggenheim Museum.

Computational Systems

“Every aspect of form, whether piece-like or pattern-like, can be understood as a structure of components. Every object is a hierarchy of components, the large ones specifying the pattern of distribution of the small ones, the small ones themselves, though at first sight more clearly piece-like, in fact again patterns specifying the arrangement and distribution of still smaller components.”[1]

[1] Christopher Alexander, Notes on The Sythesis of Form, 1964, p.130

Approaches

“The age of mechanical production of linear processes and the strict division of labour, is rapidly collapsing around us.”[1]

[1] Toshiko Mori

Computational Fabrication

Following the ideas of industrialisation and automation the architectural profession was “optimized” through partition into small instances/ processes being conducted by individual specialists. This presuppose a chain of multiple processes separated from each other and defined by a closed frame restricting and interference from precursory or successive processes. This separation created a gap between designing and fabricating which now could be bridged by the means of technology.

Current techniques for “file-to-factory” design and fabrication are executing processes in a closed feedback loop[2]. However novel fabrication and construction strategies, inspired by novel tools and devices for extracting information from the physical environment, explore the possibilities of a continuous feedback loop between analogue and digital processes. This means that data is constantly exchanged between physical and digital models and computation happen throughout the whole process. In this context, computational fabrication refers to the integration of computational design tools and digital fabrication processes — to the convergence of digital and analogue design processes.

Through digital technology, e.g. on site robotics, augmented fabrication, designers and architects can participate fabrication and construction processes. The digital advancements of CAD/CAM technologies are still rather abstract but novel approaches and customized processes seek a direct link between digital and analogue and reinvention of the forgotten idea of craftsmanship: “digital craft”, “today craft medium need not have a material substance, and the craftspersons need not touch the material directly”[3]

[2] Feedback occurs when outputs of a system are routed back as inputs as part of a chain of cause-and-effect that forms a circuit or loop. From: Andrew Ford, Modeling the Environment, 2010

[3] Kolarevich, 2008, p.127

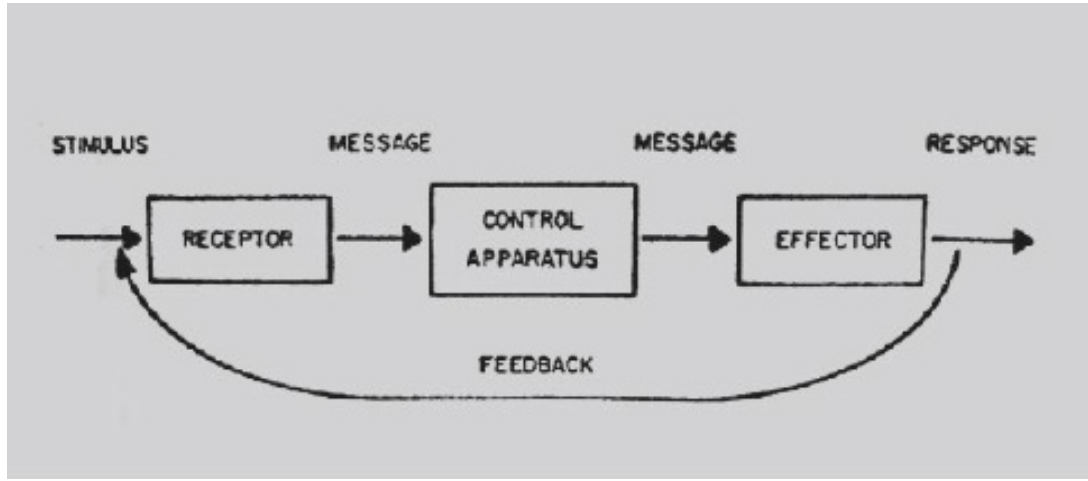


Figure 10. Simple Feedback Scheme (Ludwig von Bertalanffy, 1969)

Figure 11. Augmented Robotics: Mixed reality environments for on-demand fabrication, assembly and analysis of complex steel structures.

Figure 12. Brass Swarm, Shanghai, installation, 2015

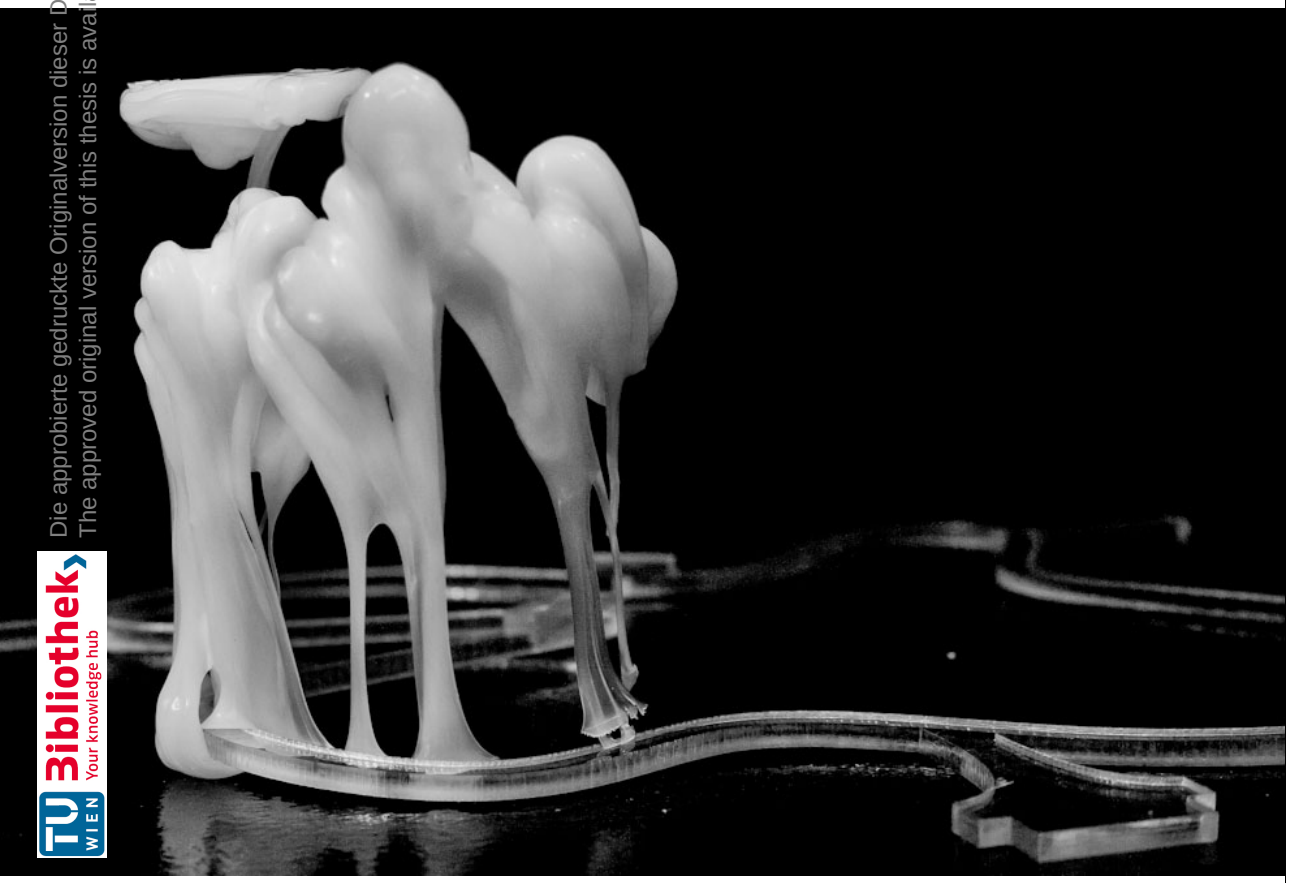


Figure 13. Robotic Plastic Extrusion Workshop with Francois Roche

Material Behaviour

> **“Steel, especially mild steel, might euphemistically be described as a material that facilitates the dilution of skills… Manufacturing processes can be broken down into many separate stages, each requiring a minimum of skill or intelligence… At a higher mental level, the design process becomes a good deal easier and more foolproof by the use of a ductile, isotropic, and practically uniform material with which there is already a great deal of accumulated experience.”**

In his essay “Uniformity and Variability. An Essay in the Philosophy of matter”, the philosopher Manuel DeLanda discussed the problematic of the labour division that resulted from the industrialisation and how it affected the way we use and shape materials. Prior industrialisation material including its imperfections was given and the process (handcraft) resulted from it. Later materials have been modified in order to program their properties and behaviour and to fit them to an automated process part of the fabrication chain. Furthermore DeLanda suggests that form could be either homogeneous or heterogeneous. Homogeneous forms implies that shapes are a product of a design and conceptual idea detached from the way of materialisation. One could understand this approach as similar to the one of the industrial fabrication. Heterogeneous form on the contrary sees materials as “as not inert receptacles for a cerebral form imposed from the outside, but active participants in the genesis of form”.[1]

Unlike industrial fabrication where processes are reduced to simple routines, craftsmen had to be skilled and experienced. This reduction was only possible when complex material behaviour was replaced by scientifically modified predictable one in order to fit the fabrication. In contrast “Craftsmen did not impose a shape but rather teased out a form from the material, acting more as triggers for spontaneous behaviour and as facilitators of spontaneous processes than as commanders imposing their desires from above”.[2]

DeLanda defines the problem as follows: “despite the availability of new materials with complex behaviours, our design skills may now lag behind.” So despite the great variate of advanced but predictable materials that has already contributed for the extinction of craftsmanship it might lead to the reduction of the design process to yet

[1]
[2]

another routine. Realising the complexity of material in its pure form without being modified in order to do what designers want is a crucial step in developing the understanding of a material informed fabrication processes.

> **“Furthermore, in a paradoxical way, the new techniques and methods of digitally enabled making are reaffirming the long forgotten notions of craft, resulting from desire to extract intrinsic qualities of material and deploy them for particular effect. As such, interrogating materiality is fundamental to new attitudes towards achieving design intent.”[3]**

[3] Branko Kolarevic, Kevin Klinger, Manufacturing Material Effects: Rethinking Design and Making in Architecture, 2013

Figure 14. Between shit and architecture, Anish Kapoor, 2011

Robotic Fabrication

“In robotic fabrication, complex custom processes of material deposition, manipulation and assembly are carried out by industrial robots. While the industrial robot is truly generic as a mass-produced machine, the fabrication process is project specific. The physical manipulation on the material becomes a constructive part of the process, where it demands as much attention, care and creativity as the design of the robots’ actions.“[1]

Industrial robots are complex machinery that in the scope of the automated industrialized fabrication are predominantly executing repetitive tasks. Nevertheless they are programmable machines that could be assigned an infinite variety of tasks and perform rather custom instead of dull standardized fabrication processes.
Being able to design custom fabrication processes, traditional material systems can be now updated by new ones utilising heterogeneous processes, that could adapt their logic to the project or site specific conditions.

Figure 15. RobArch 2018, Gramazio Kohler Research, Robotic Fabrication

[1] F. Gramazio, Matthias Kohler and Michael Budig, The Tectonics of 3D printed Architecture, 2013

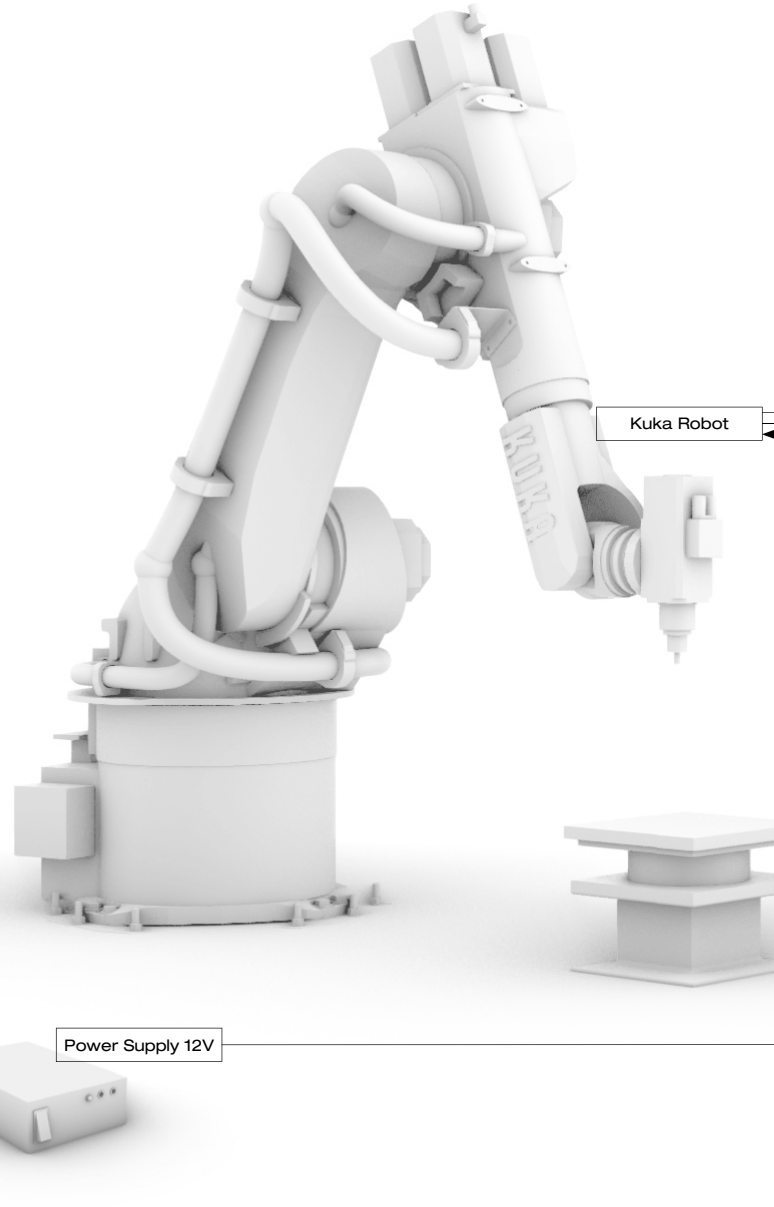
Implementation

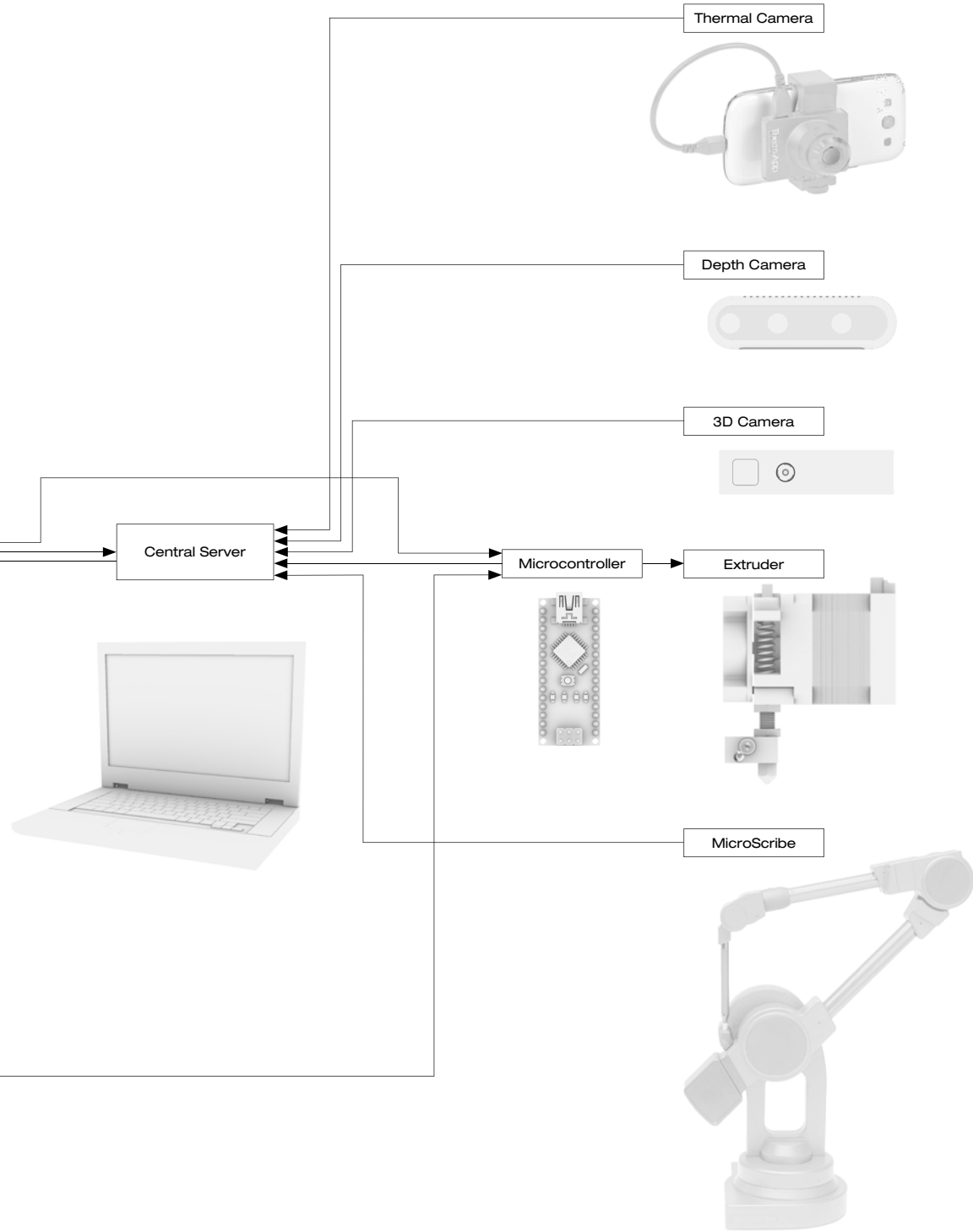
Following chapter covers the technical background for the executed experiments and case studies. It describes the experimental setup used in the work and gives a detailed overview of the used hardware as well as its programming and controlling.

A plastic extrusion head deposits material in 3D space guided by a 6-axis industrial robot that is been remotely controlled. This setup has been extended by additional devices inputting information from the physical environment to the external control system. Due to the unpredictable material behaviour feedback - position of structure parts in space - from the deposited layer has been introduced to the process defining the subsequent tool-paths for the robotic end effector and hence the areas of material accumulation. This facilitates the customisation of a robotically controlled additive manufacturing where the design is not predefined and a result of automated fabrication, but the material is affecting the robotic tool-paths and thus participating to the genesis of form.

Current material practices in architecture invest immense efforts in exploring new materials and determining their behaviour. Nevertheless the way of manipulating them remains unchanged, imposing materials a given shape rather than triggering its spontaneous behaviour. This results primarily from the division between design and fabrication processes. Decisions concerning the manufacturing are taken in advance and the fabrication is a mere execution of machine instructions, requiring that materials perform in a predictable fashion. However the processes could be responsive and adapt to different conditions and react to changes.

A dynamic data exchange between physical and digital environments facilitates a design process with digital matter that carries information from the physical environment and its physical characteristics are informing the fabrication. This is enabled by the integration of devices and software that stream, translate and manipulate data.



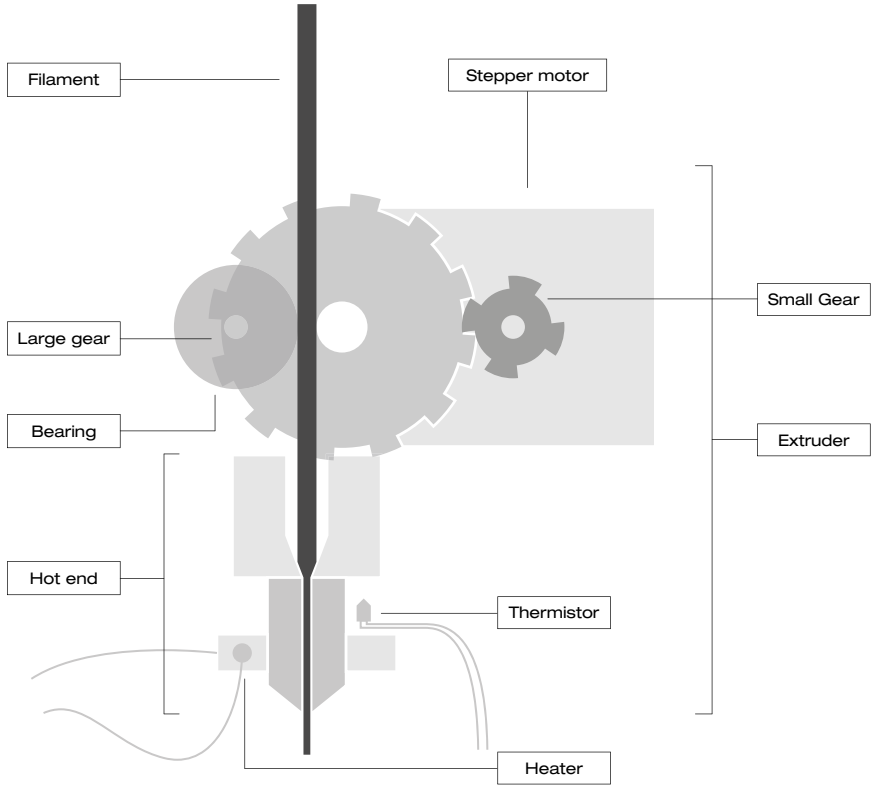


Experimental Setup

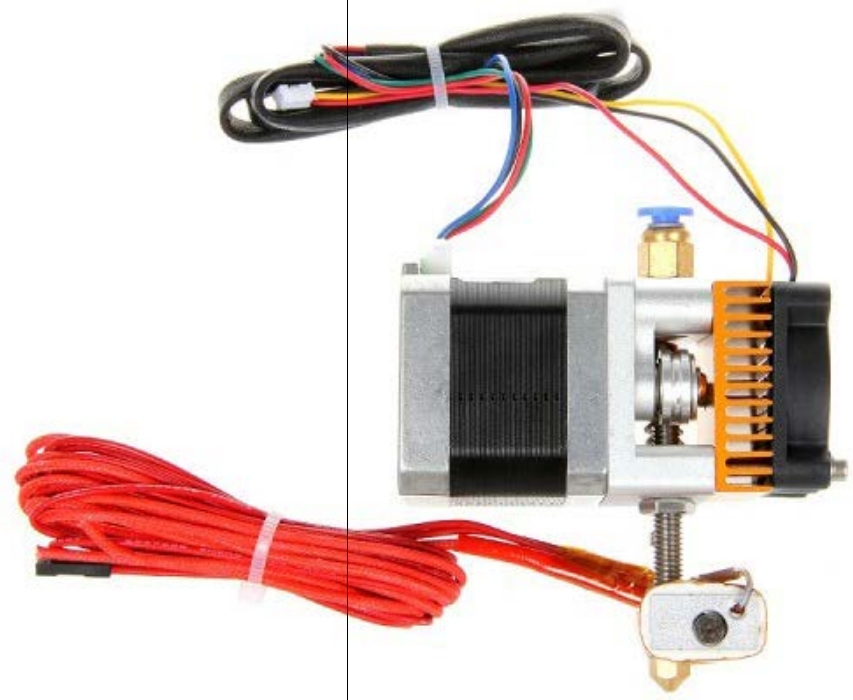
In the chapter the major modules required for the robotic extrusion and feedback are introduced and discussed. The experimental setup essentially consists of a Kuka six-axis industrial robot with an end effector – 3D printer extruder – a computer and a five-axis coordinate measuring machine, MicroScribe. While these components were applied throughout the whole practical part of the project, others additional devices for extracting information from the physical environment like Intel RealSense depth camera and Picoflexx 3D camera could not be successfully integrated to the fabrication process due to insufficient information output. Nevertheless they proved essential and delivered valuable results and therefore are documented in this chapter.

In addition, some components have been extended by custom software applications allowing for better integration during the process development as well as effortless upgrade of the system on a later stage. The was achieved by writing custom programs in various languages (Python, C++, C#, KRL) on several different platforms (Windows 10, ATmega, Arm M4F, KSS). In total about 10000 lines of code were written to tie all of these different hardware and software components together. While direct point-to-point links between two partners is used in some cases, the most important data exchange between the robot and Rhino/Grasshopper is managed by a Python script, the Central Control Script (CCS).

Central Server
see p.137



FDM extruder, parts and function



Plastic Extruder

The extruder and the hard- and software controlling is a critical component of the experimental setup. Starting from a simplistic design it was interactively improved and adapted to the needs imposed by the different experiments and case studies. In hindsight two main extruder generations can be identified. A first, rather simplistic, version, mainly based on off-the-shelf components and a second generation – incorporating several new developments, most notably a very slim hot-end/tip design.

Generation 1

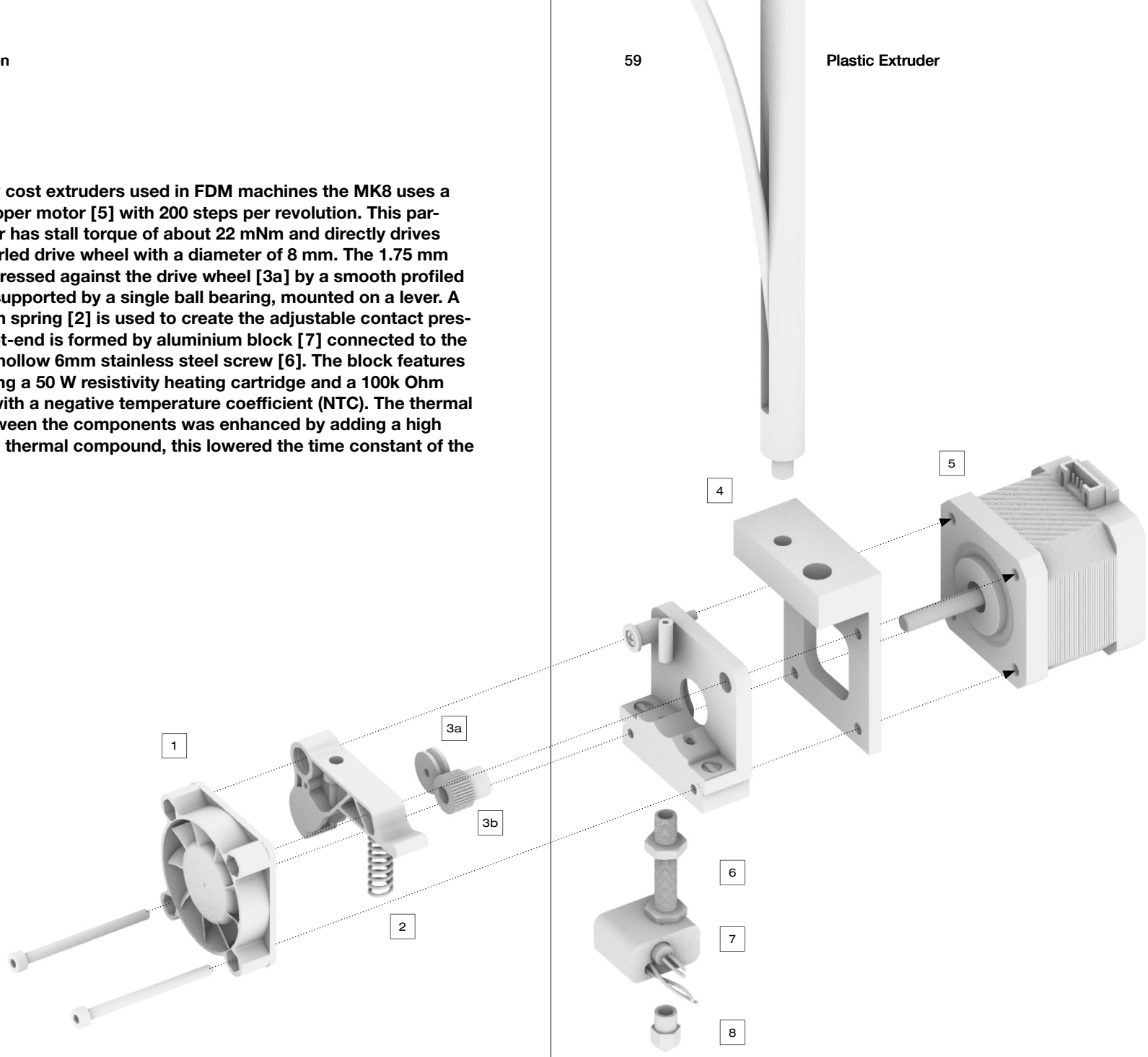
The first extruder was not developed from scratch but borrowed heavily from existing extruder designs used in low cost FDM machines [1] . This facilitated the use of off-the-shelf components and reduced the time from design to experiment significantly.

Figure 16. Assembled MK8 extruder by Geeetech

The MK8 extruder by Geeetech soled as a drop-in replacement of the Prusa I3 FDM machine was used as basis of the extruder generation 1. Only minor modifications to the mechanics were done to properly mount the extruder to the robot.

[1] Fused Deposition Modelling (FDM) is the most common extrusion-based additive manufacturing process developed by Stratasys

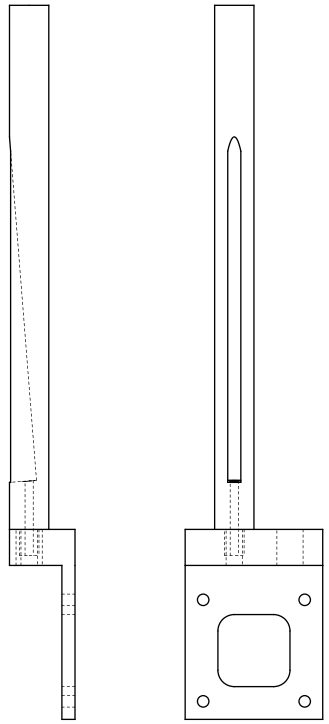
As most low cost extruders used in FDM machines the MK8 uses a bi-polar stepper motor [5] with 200 steps per revolution. This particular motor has stall torque of about 22 mNm and directly drives a single knurled drive wheel with a diameter of 8 mm. The 1.75 mm filament is pressed against the drive wheel [3a] by a smooth profiled wheel [3b] supported by a single ball bearing, mounted on a lever. A compression spring [2] is used to create the adjustable contact pressure. The hot-end is formed by aluminium block [7] connected to the feeder by a hollow 6mm stainless steel screw [6]. The block features bores housing a 50 W resistivity heating cartridge and a 100k Ohm thermistor with a negative temperature coefficient (NTC). The thermal contact between the components was enhanced by adding a high temperature thermal compound, this lowered the time constant of the



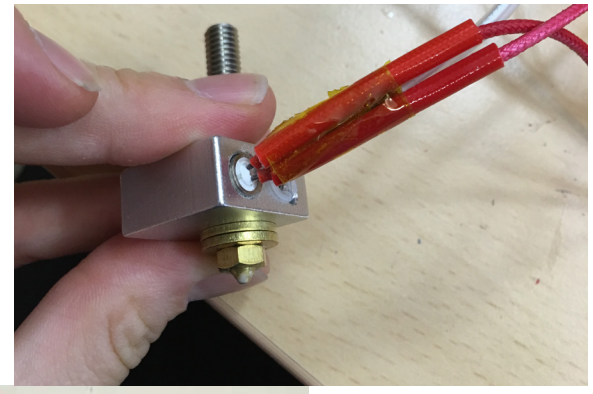
Exploded drawing of the first extruder:

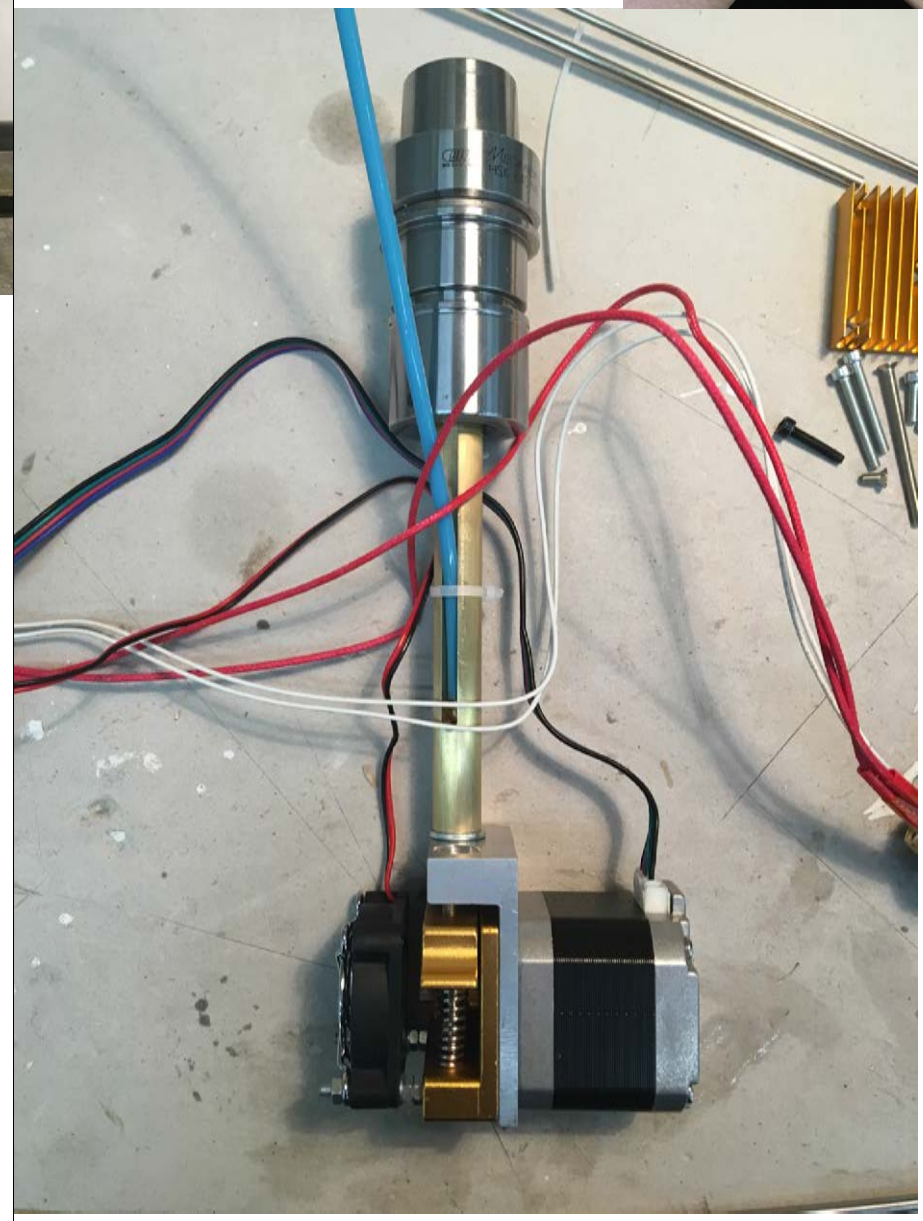
1 fan
2 compression spring
3a drive wheel
3b profiled wheel
4 L-shaped bracket
5 stepper motor
6 hollow steel screw
7 aluminium block with heating cartridge and temperature sensor
8 brass nozzle

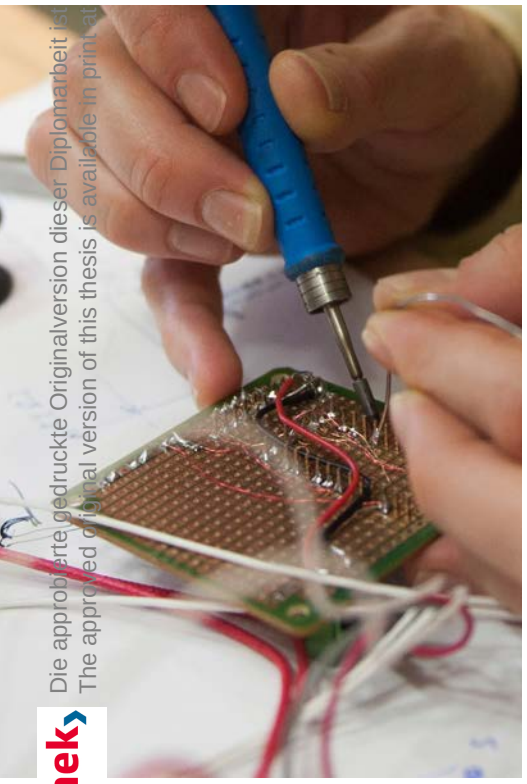
feedback loop and significantly improved controller performance. At the bottom of the aluminium block a short brass nozzle [8] with a 1 mm bore is mounted in a female M6 thread. Between the stepper motor and the feeder frame a customized machined L-shaped bracket [4] was inserted to mount the assembly to the robot.

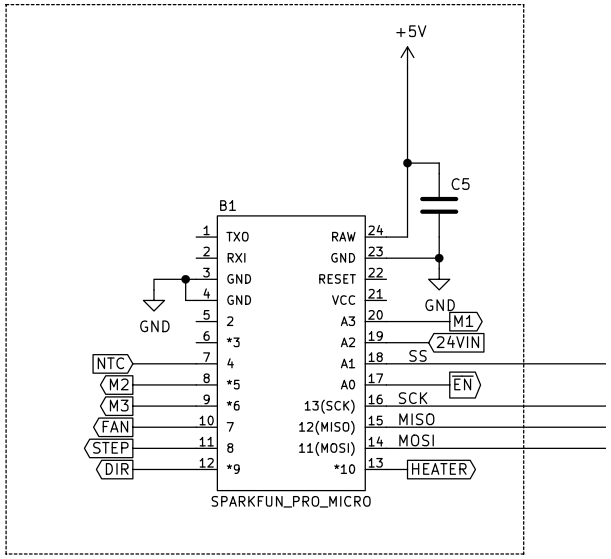


..-bracket and custom
..ube adapter for mounting
..he extruder on the robot
..with a slit and a hole for
..he plastic filament to be
..ed.



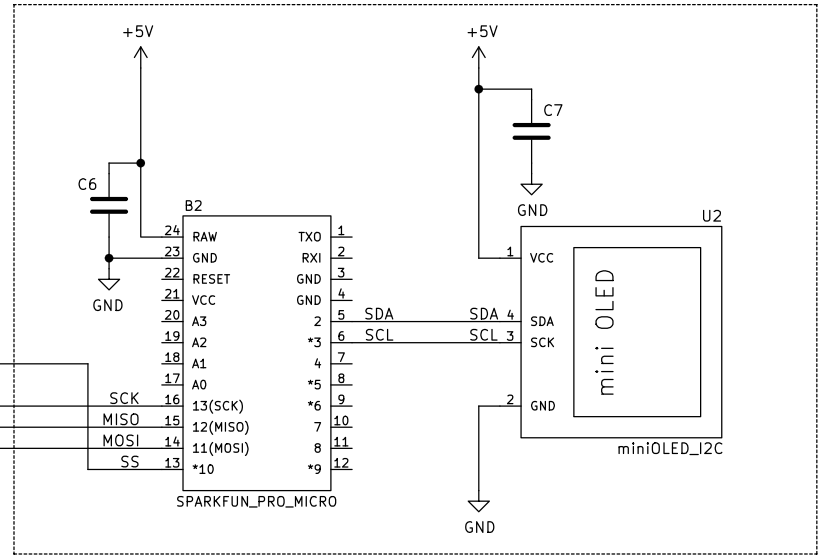




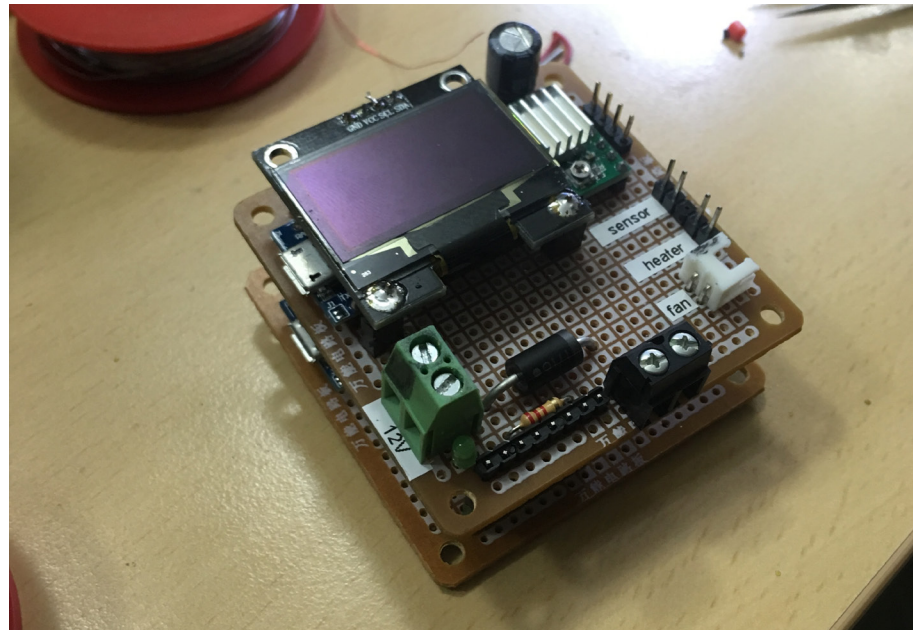


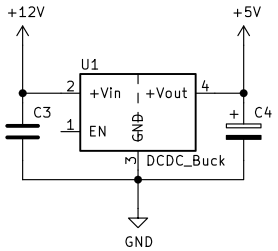
master controller

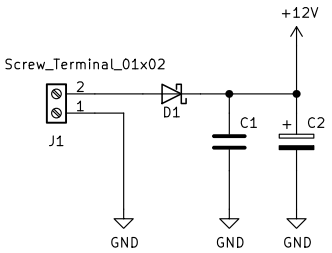
The controller is a custom development specifically designed for this project. The main tasks, controlling the temperature and the stepper motor and communication with a PC via USB is performed by an 8 bit microcontroller from Microchip Technology (ATmega32U4), a second ATmega32U4 acts as display controller for a 0.96” OLED Display.

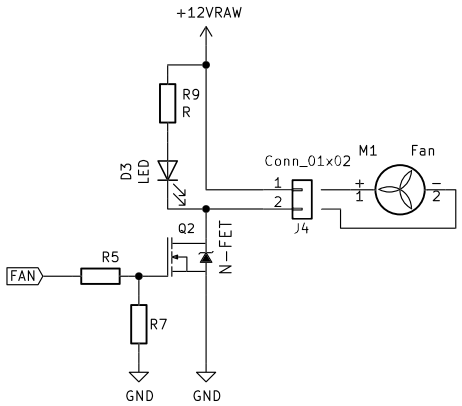


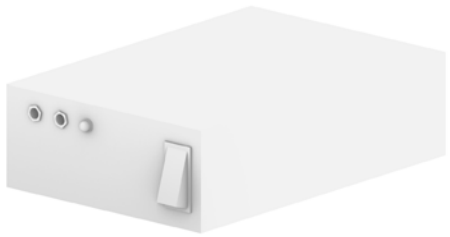
display unit

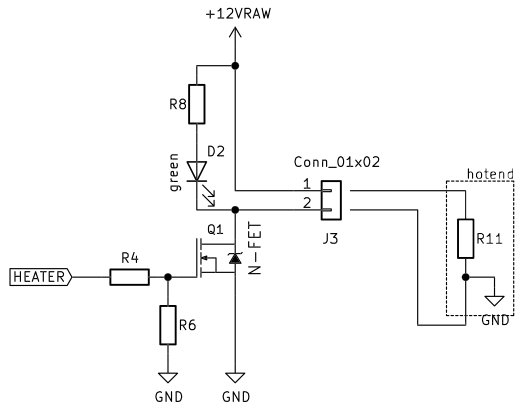


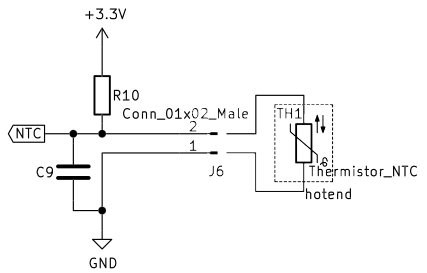


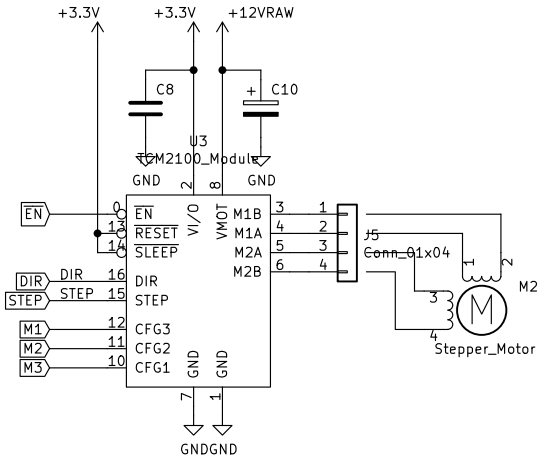


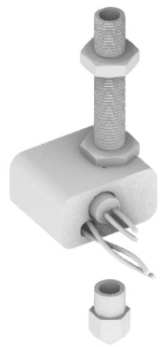


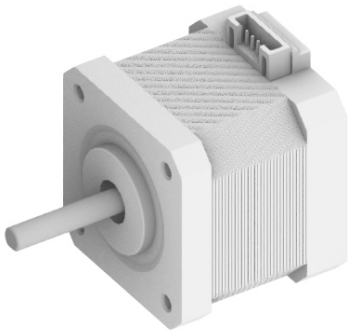


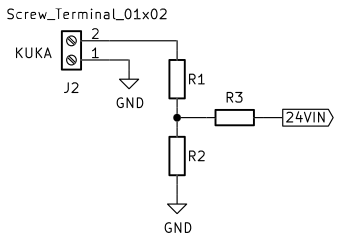


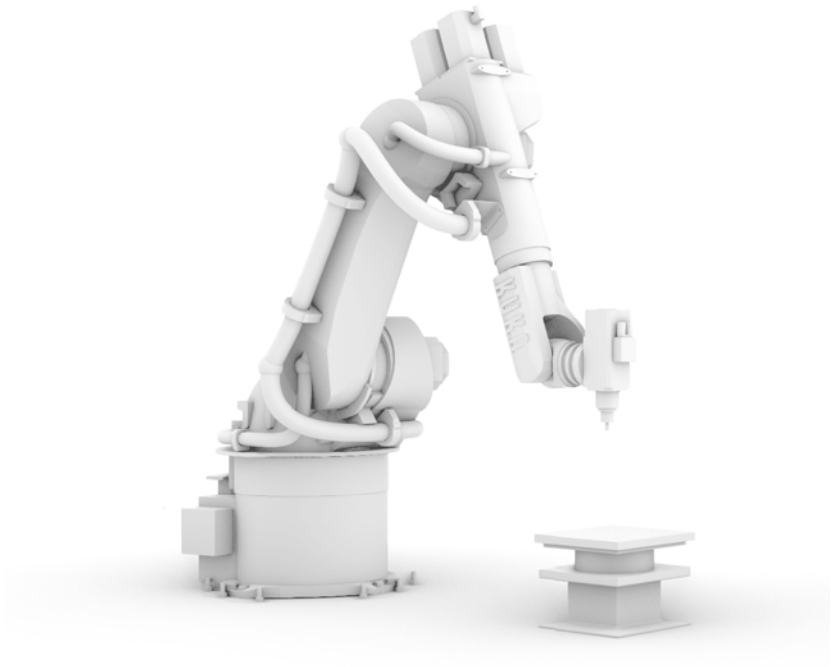


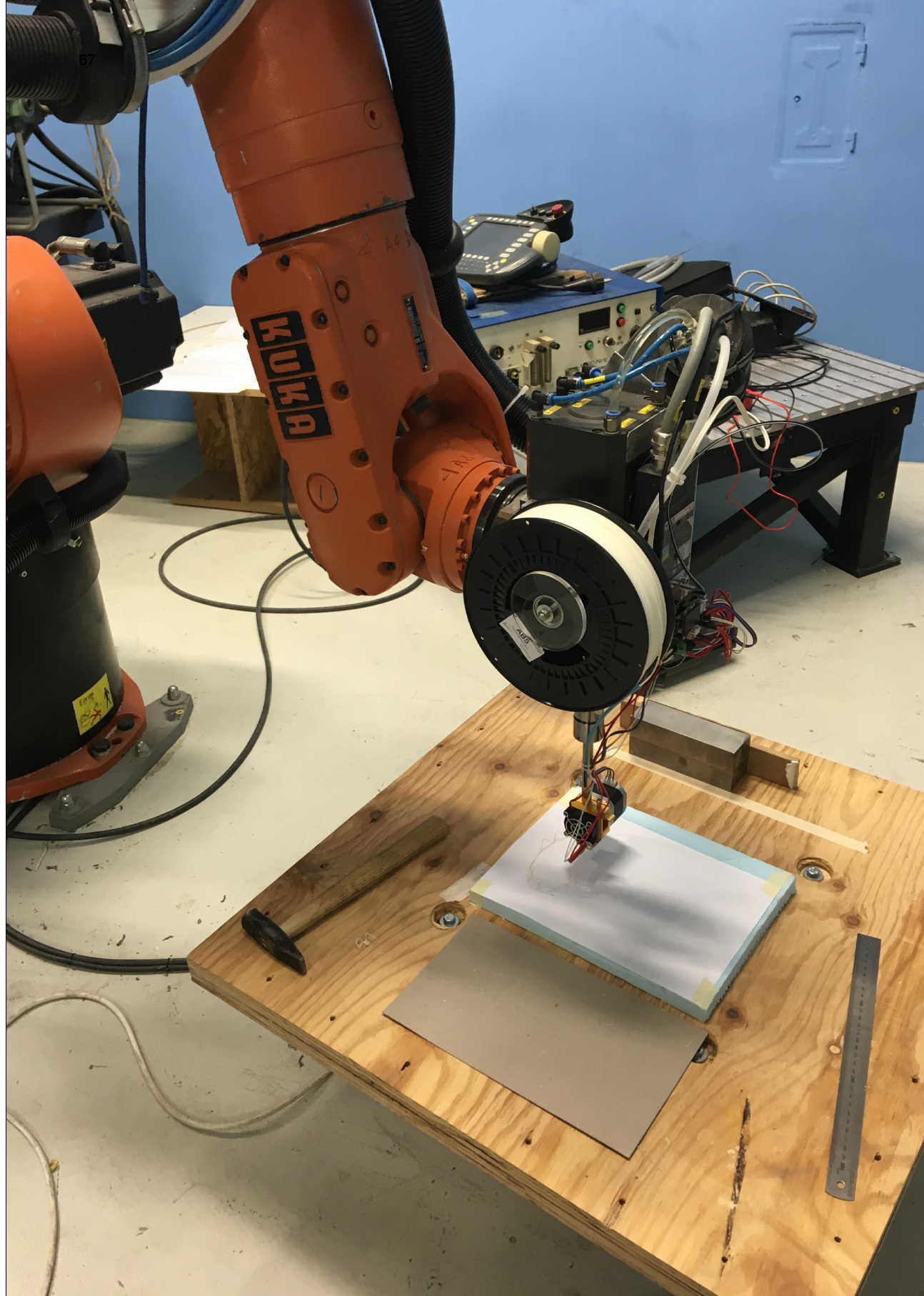


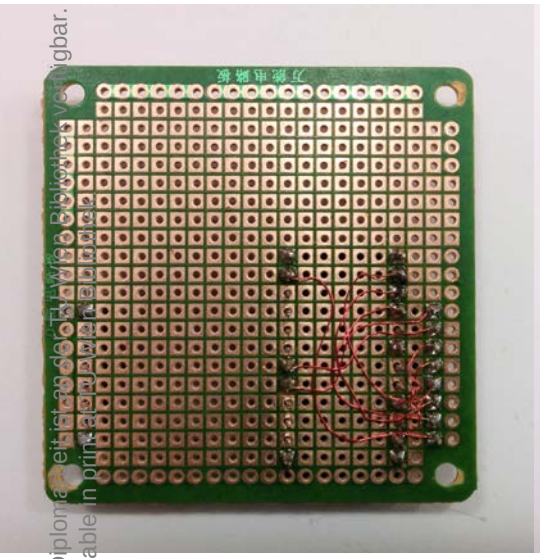


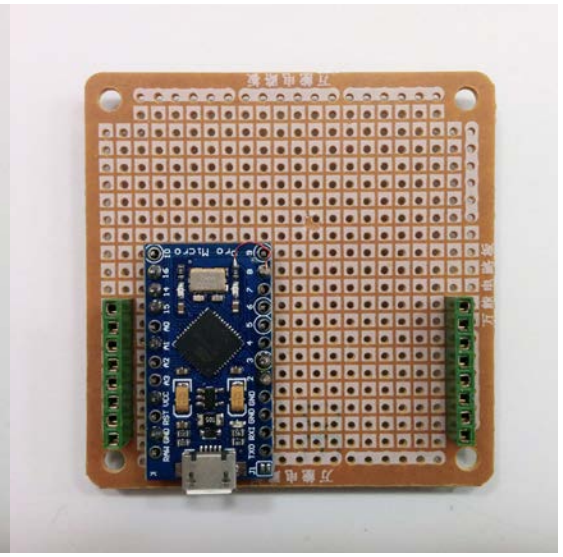




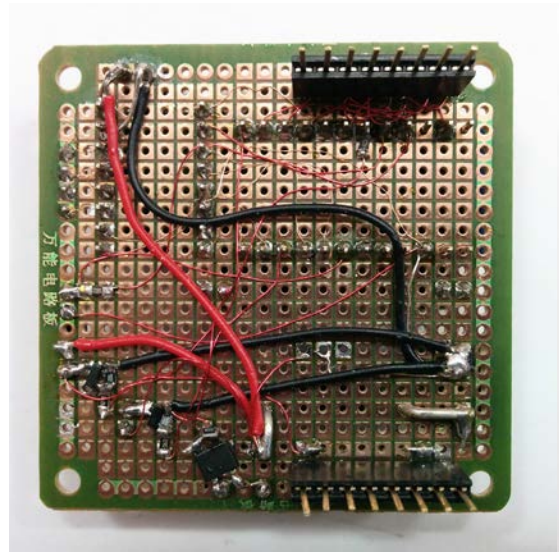


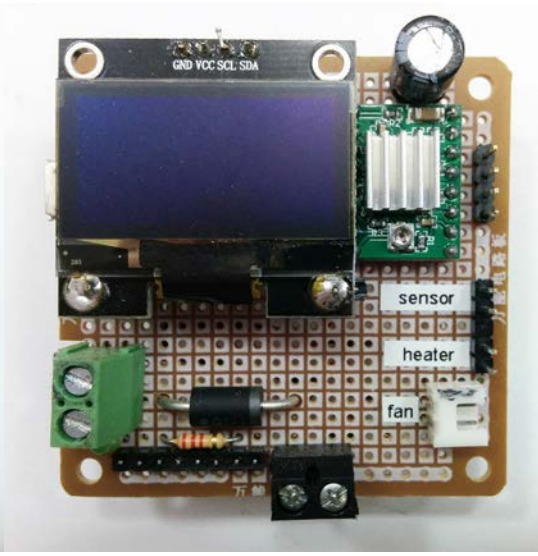


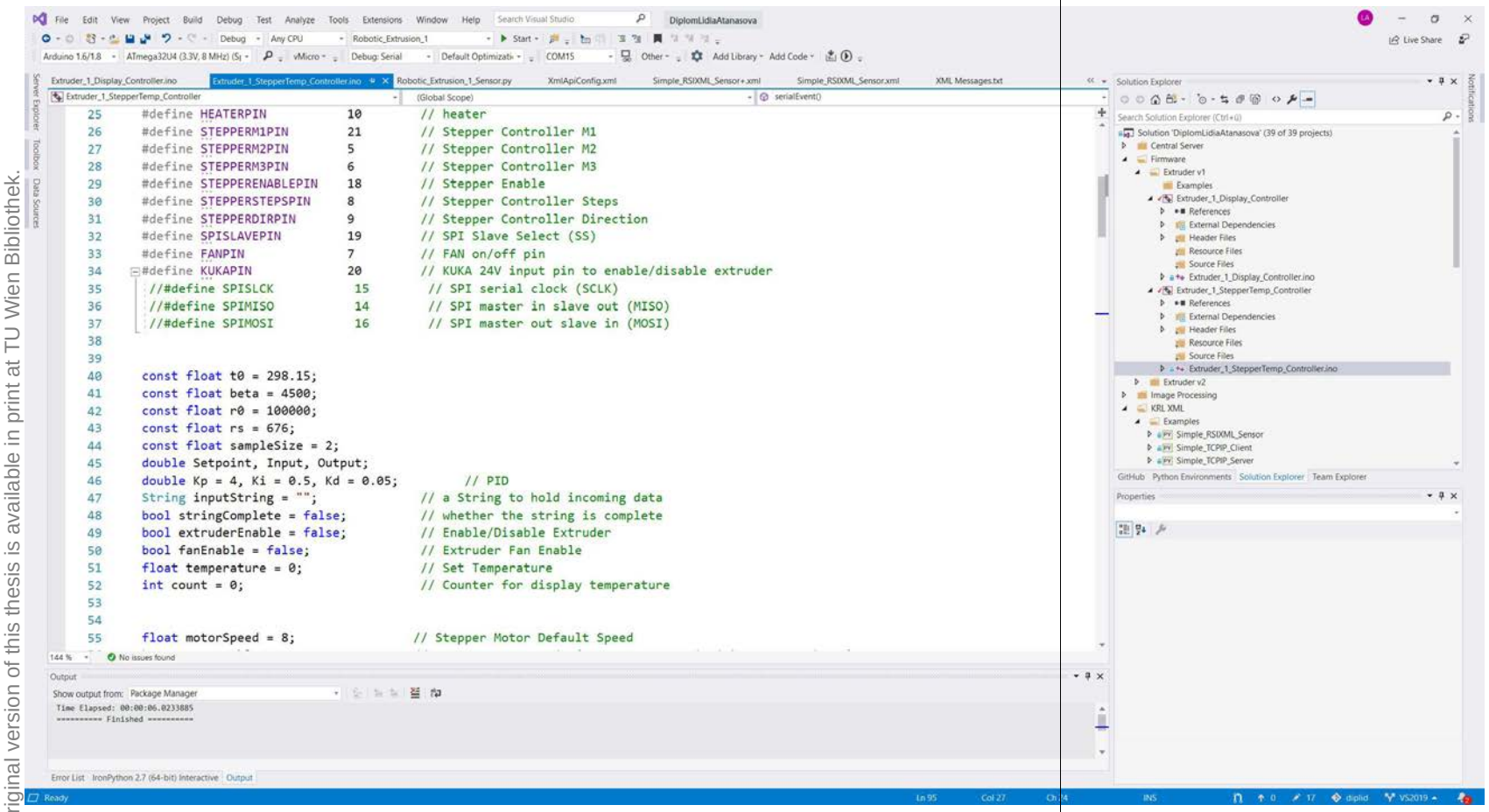




The electronics are built on double sided prototyping board, using individual components as well as per-fabricated modules (microcontroller board, stepper driver, display), the connections are manually wire wrapped using 0.16 mm Verowire.







Arduino Logo

The Arduino framework is used to simplify and speed up firmware development, the Arduino IDE however was abandonment in favour of Visualmicro's Arduino IDE for Visual Studio. The rich library support by the Arduino community allowed to realize the needed functionality in just 300 lines of C++ code. This firmware allows to change the set-point of the temperature PID collector, the filament feed rate, and other parameters via a COM port over USB from a PC and to start and stop the feeder from the robot via a 24V digital input (controlled via DeviceNet[2] on the robot side).

[2] DeviceNet is a network protocol used in the automation industry to interconnect control devices for data exchange.

Firmware

The first extrudes firmware is simple. On the main controller the main loop takes care about the different tasks the controller has to perform one by one:

[3]

```
while (true){
      updateTemperatureController()
      runFeederMotor()
      readSerialCommands()
      if mills()>oldTime+dispDelay {
         sendDataToDisplayController()
         oldTime=mills()
      }
}
```

The time critical part of the mail loop is the runFeederMotor() method. It has to be called at least at twice the desired maximum step rate. For a feed rate of 25 mm/min at 16 fold micro stepping the motor has to do 53 steps per second, so one complete iteration of the main loop has to finish in less than 10 ms. Given the inefficiency of several of the used Arduino libraries this is already close to the minimal time achievable without any hand-coded optimization. To avoid unnecessary delay in the processing of the serial commands have to be as short as possible.

As a consequence short yet still human readable commands are used, for example:

```
s100      //set feeder speed to 100 micro steps per minute
t175      //set temperature controller set-point to 175°C
f1        //switch on the cooling fan
m1        //override external feeder control and run the feeder
m0        //disable the feeder
m2        //enable external feeder control via 24 V input
```

During normal operation temperature and feed rate are set as desired and operation mode ‘m1’ is activated. Now a KRL program running on the robot can start and stop the feeder with a simple command:

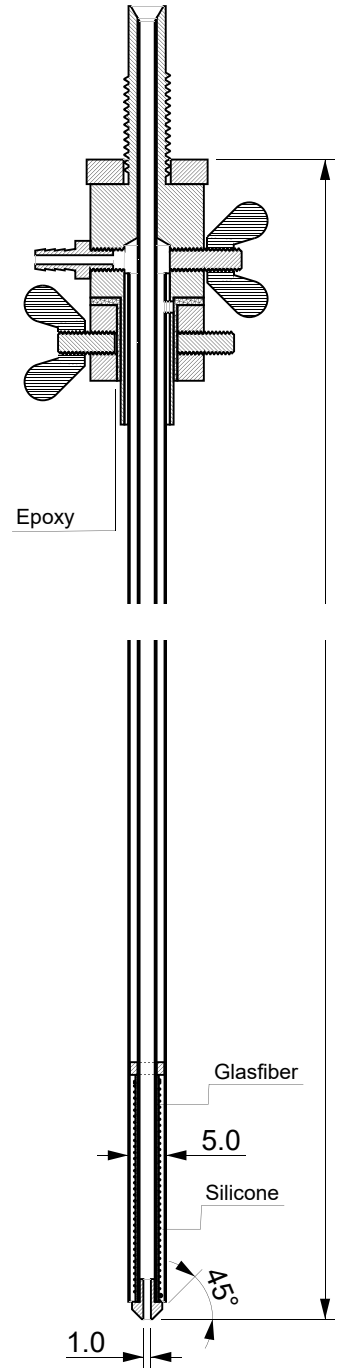
```
$OUT[extruderOnPort]=TRUE          ; turn on extruder feeder motor
$OUT[extruwderOnPort]=FALSE        ; stop feeder motor
```

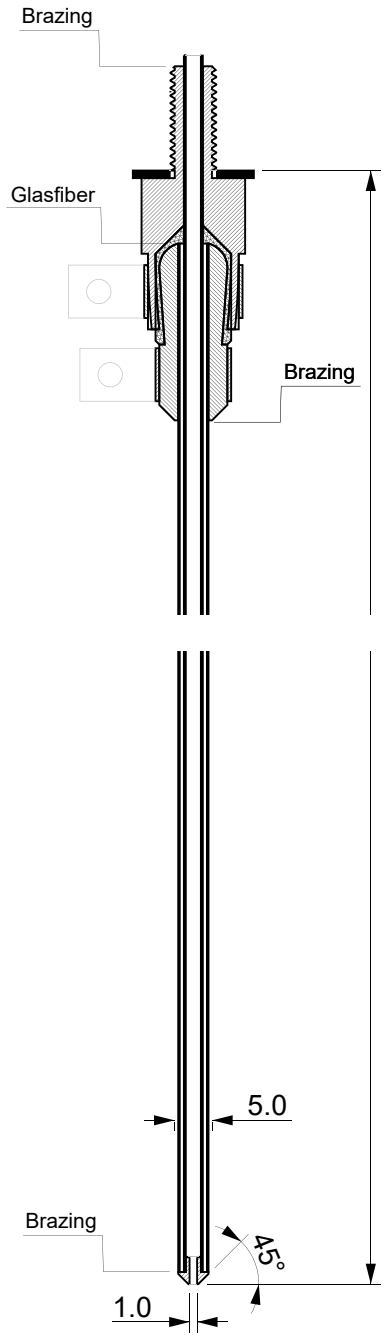
To quickly stop the flow of material the extruder firmware does not only stop the motor, but run it backwards several steps quickly. On start of the motor a number of fast forward steps is inserted by the firmware to instantly restart material flow.

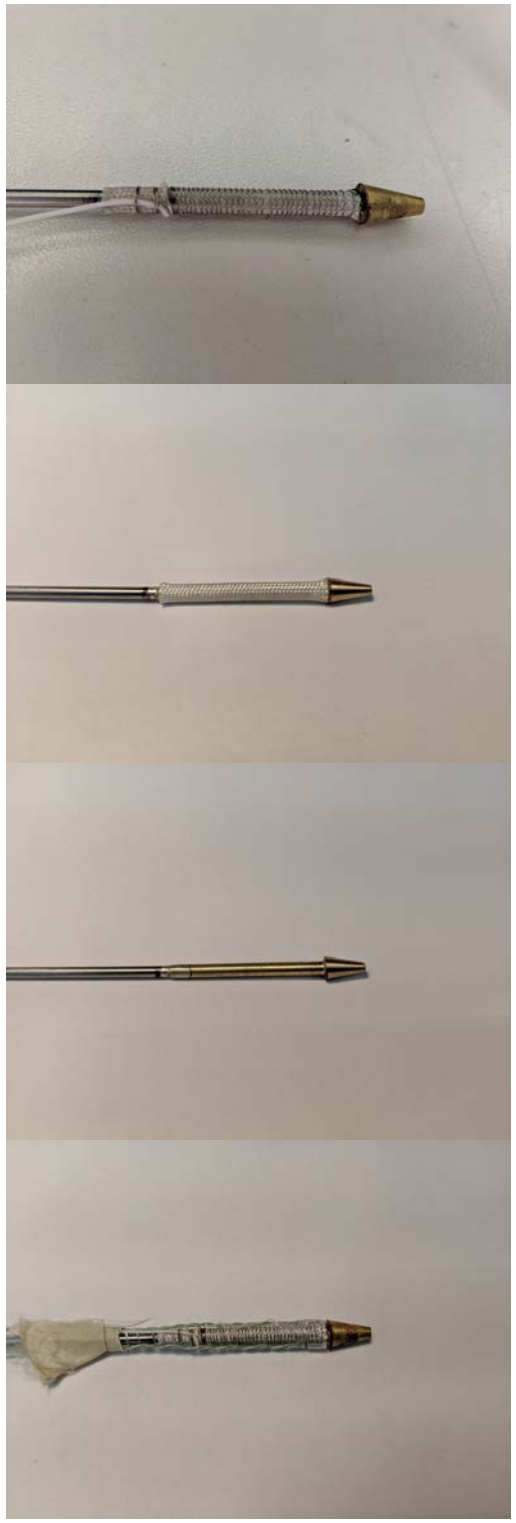
[3] pseudocode

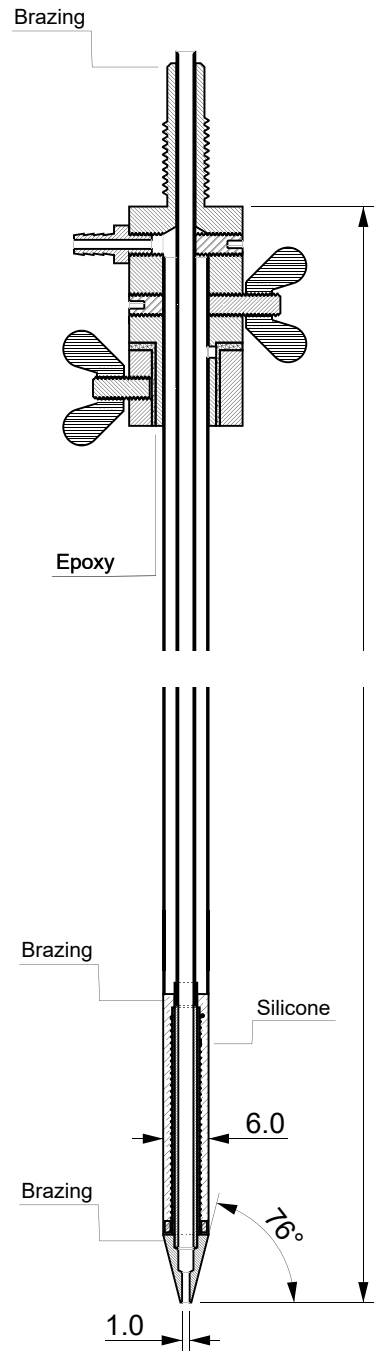
The main controller can send information to the display controller. Any null terminated string set via the SPI bus to the display controller will be displayed on the small OLED display. During normal operation the displays shows the current temperature to allow easy and quick monitoring without the need to start a serial terminal to retrieve information from the controller via the serial interface.

The firmware on the display controller constantly monitors the SPI bus and prints any received characters on the OLED display. As mentioned earlier this dedicated display controller is needed because a single refresh of the displays frame buffer takes about 50 ms. Sending a few characters over the SPI bus takes the main controller about 3 orders of magnitude less time, this ensures uninterrupted step generation.









Thin Hot-End

Layer bases extrusion processes usually do not impose size or shape restrictions on the hot-end assembly. In spacial extrusion the fabrication and design possibles can be seriously limited by the hot-end geometry. In general a rotary symmetric, long and thin hot-end with a pointy extrusion nozzle is desired. While there are may different hot-end designs for layer based processes available and prior spacial extrusion projects modified these design to better fit the specific needs, no new, from scratch designs are available.

One main task of the hot-end is to heat up the material. This can be done in multiple ways, large scale screw extruders for example generate most of the heat by working the material, creating heat from inner friction. While this can most probably not be scaled down, it would be interested to look at direct or indirect microwave heating as used often in plastic welding. Yet, the explored designs stick to restive heating as used in all 3d printer hot-ends.

The wast majority of hot-end designs use a heating-cartige to heat up a block of metal and melting the material inside a chamber or a bore in this block and measuring the temperature of the block. This design can not be shrunk down substantially, but all these components can be replaced by a single thin capillary tube. The tube can be directly heated by running high current thru the tube. The resistance of most materials is temperature depended, so a dedicated temperature sensor could be replace by measuring the tubes residence. A first prototype following this idea was built, while the principal worked as expected the hot-end was was nonfunctional, because the long (about 200 mm) thin tube restricted the flow of material. The heated zone has to be short.

In a second prototype only about 35 mm of the thin capillary are heated by an external coil of nickel chrome wire. A sharp temperature gradient towards the cold part is achieved by activly cooling the capillary using compressed air flowing between the inter capillary and an outer, thicker tube. The temperature is sensed by a thermocouple. This hot-end worked well until a short made the thermonuclear unusable.

A third prototype was build following the same principal as the second one, but incorporating several improvements, for example it features two thermocouples and a pointy nozzle.

Generation 2

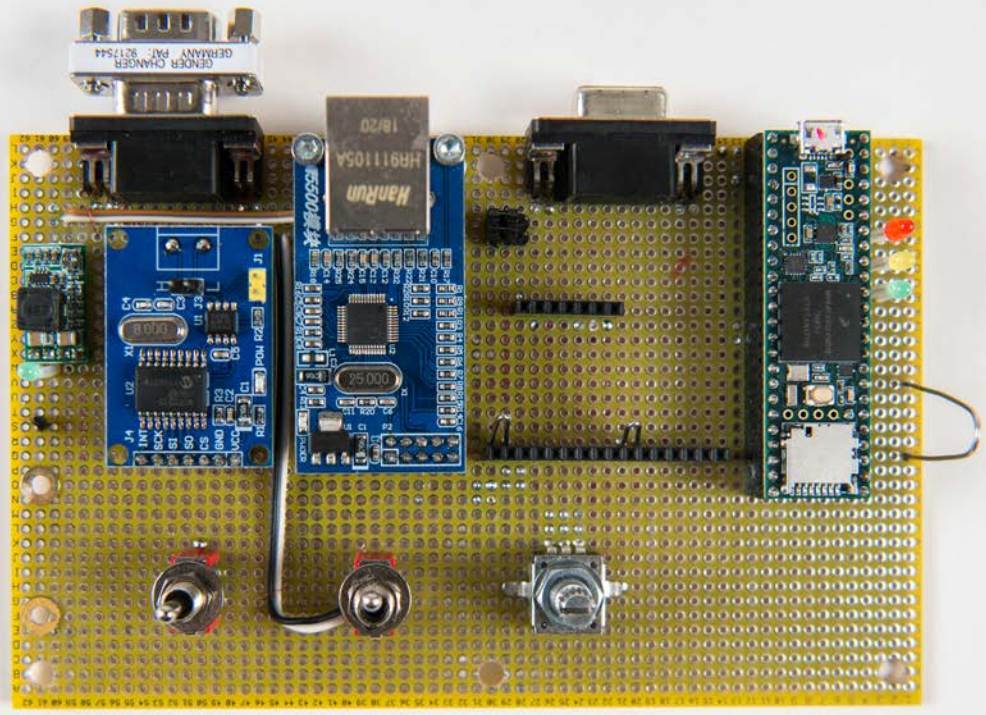
Increasing complexity of the experiments done with the first extruder revealed that it falls short in several aspects. So a second generation was developed with this main design goals in mind to overcome flaws of Generation 1:

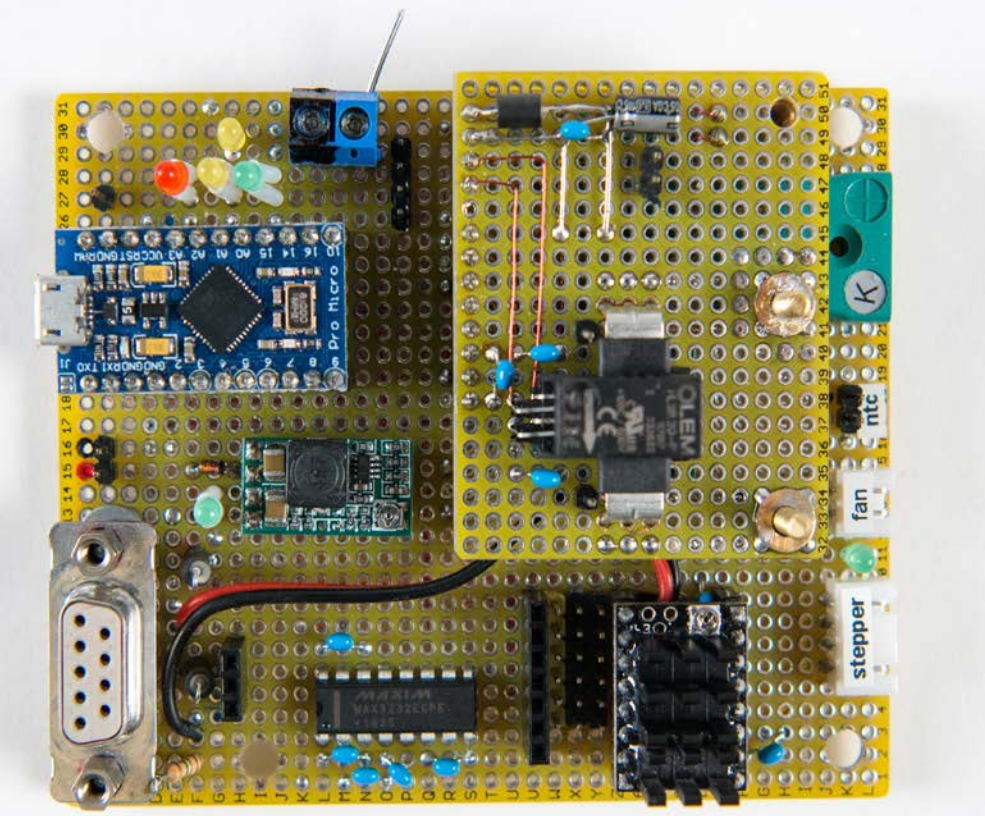
- full integration into the Ethernet/XML based control network
- physical user interface to override programmed settings
- improved display
- increased precision and accuracy of the temperature control
- thin tip with long reach

The two controller concept was developed further, again featuring the small 8 MHz 8 bit microcontroller ATmega32U4 as a temperature and stepper motor controller, but now assisted by a powerful 120 MHz 32 bit ARM M4F core with dedicated floating point unit. The ARM M4F handles Ethernet communication, controls the display and reads user input from a rotary encoder with push button and provides a RS232C serial and a CAN bus interface[4]. The later is used to link the ARM 4M to the ATmega32U4. The use of the CAN bus (instead of SPI[5] as in version 1) now allows spacial separation of the two controllers while maintaining a stable and reliable connection even in the harsh environment close to the robot. The now separate units are refereed to as **Ethernet Controller Module (ECM)** and **Stepper Temperature Controller (STC).**

[4] Controller Area Network is designed to allow microcontrollers and devices to communicate with each other without a host computer.

[5] Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. In this case it was used for the communication between the two microcontrollers.





The STC consists of the ATmega32U4 controller, a CAN bus controller with CAN transceiver, a stepper motor driver, a high resolution 24 bit delta-sigma analogue to digital converter with an ultra low noise power supply supply, a thermocouple-to-digital converter, a level line driver/receiver to provide a RS232C compatible serial interface and a finally not used high current Hall current sensor. The heater is controlled by a high side-switch (P-channel MOSFET), the heater current can be monitored via a shunt. A low-side switched output is provided to control a solenoid valve.

The ECM is built around the PJRC Teensy 3.5 ARM M4F module. The main peripherals are an Ethernet module, a CAN bus controller with transceiver, a line driver/receiver to provide a RS232C serial interface, a small TFT display a few LEDs and a rotary encoder with push-button for user input.

Again, no PCBs were design, but the circuitries are built on prototyping bards and connections are done by Verowire. To speed up development modules were used wherever possible but significant parts had to be realized by using individual components.

STC Schematic description

The STC is connected to the ECM via 9-pin D-sub connectors and a standard serial cable. This connection provides all needed power and communication links. An additions 24V tolerant digital I/O is used to allow the Robot to quickly switch on and off the extruder without the overhead of Ethernet communication.

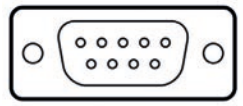
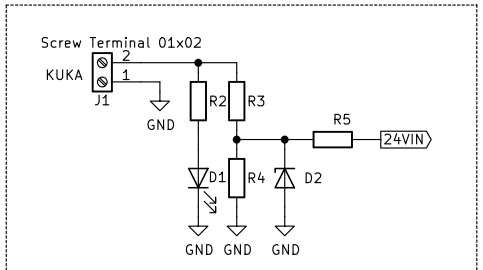
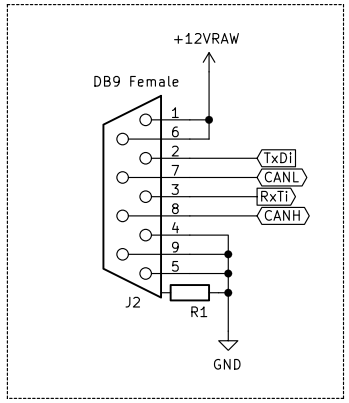


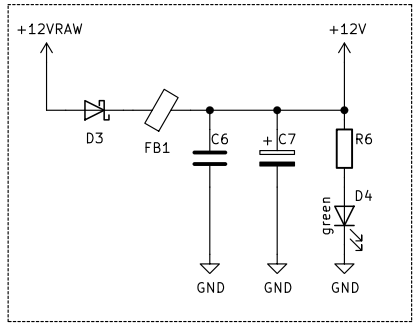
Figure xx. D-sub connector with 9 pins



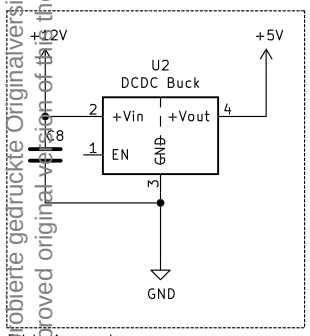
24V digital input



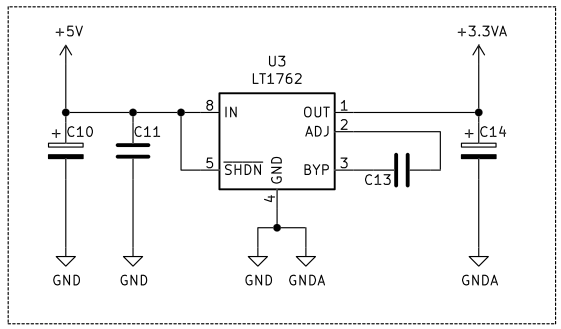
sub-d 9 main power and digital i/o connector



power supply input filter

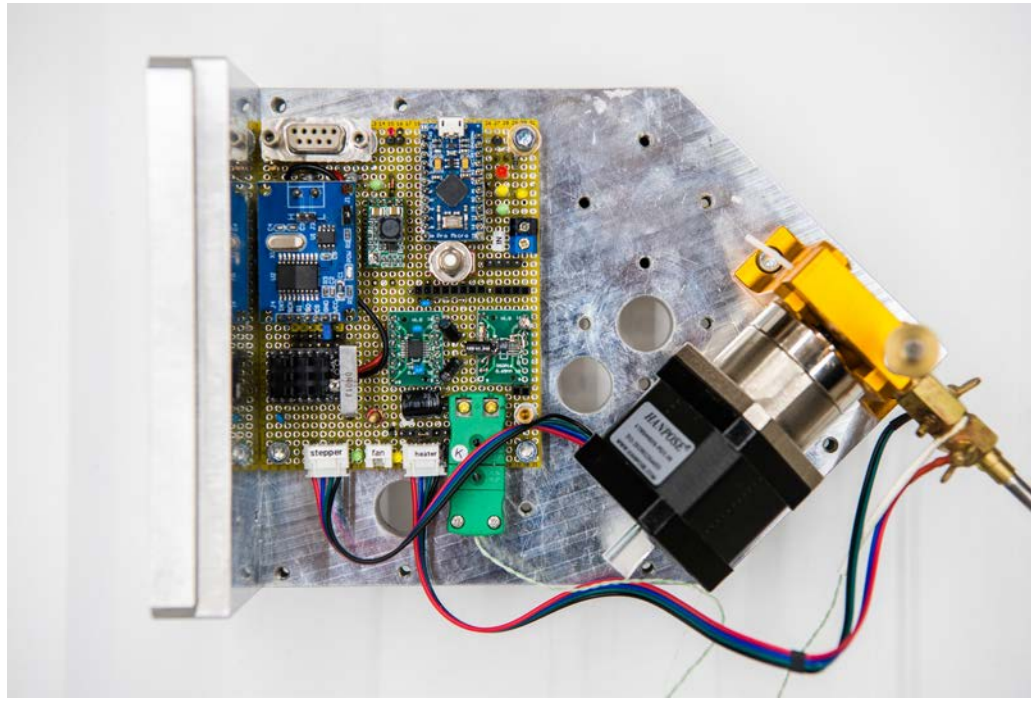


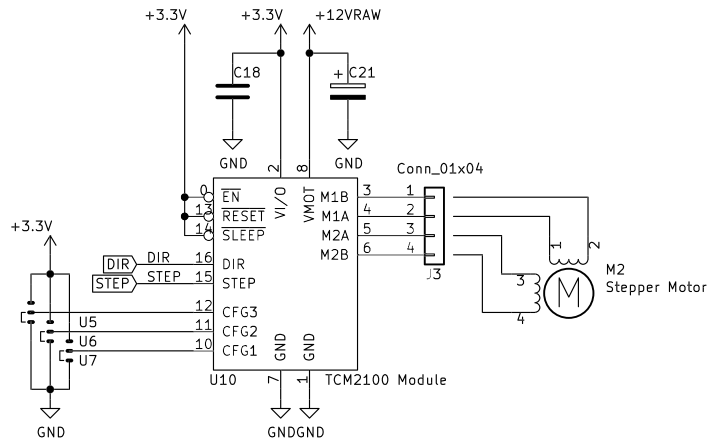
5V logic supply



ultra low noise 3.3 V supply

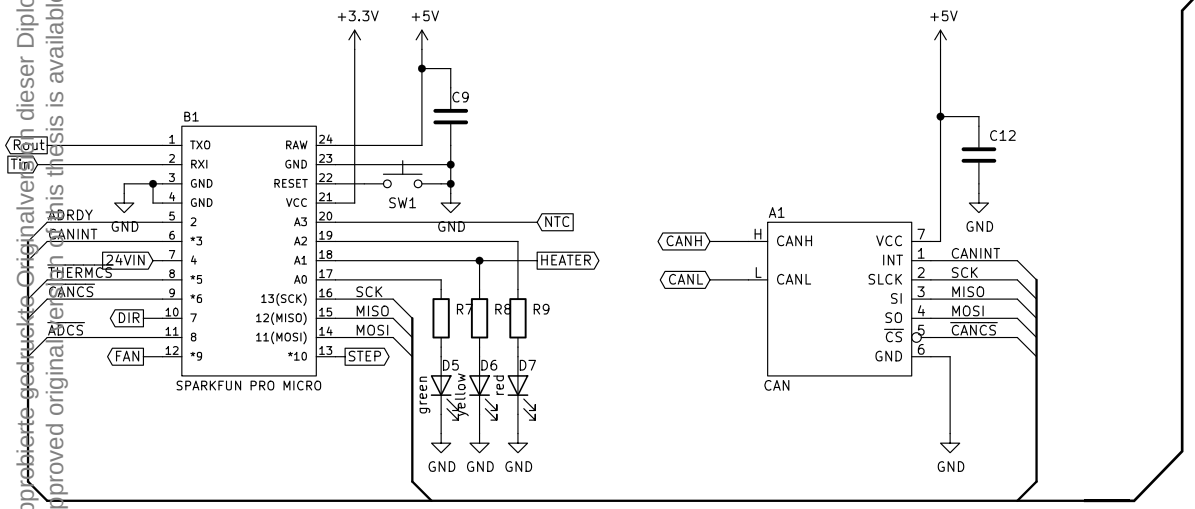
The STC uses four different power rails and GND topology is split into two parts to reduce noise in the analogue part of the circuitry. Power and GND are provided via 2x2 pins on the 9-pin D-sub connector. The 12V rail is used to supply the heater and the solenoid valve, a DCDC converter provides 5V, a simple linear regulator is used to generate the 3.3V used for all digital components (this regulator is part of the microcontroller module and therefore not directly visible on the schematic), a ultra low noise linear regulator is used to generate a clean 3.3V supply for the analogue section of the analogue to digital converter and the thermocouple-to-digital converter.

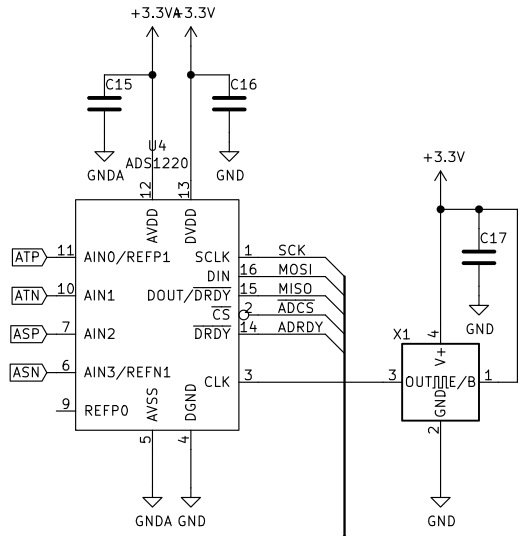




A SparkFun Arduino Pro Micro module, providing the minimum periphery (quartz, voltage supply, USB connection, etc.) for the ATmega32U4 microcontroller is used to simplify development. The 18 I/O pins broken-out by the module are just enough for the desired periphery. The microcontroller runs at a clock frequency of 8 MHz.

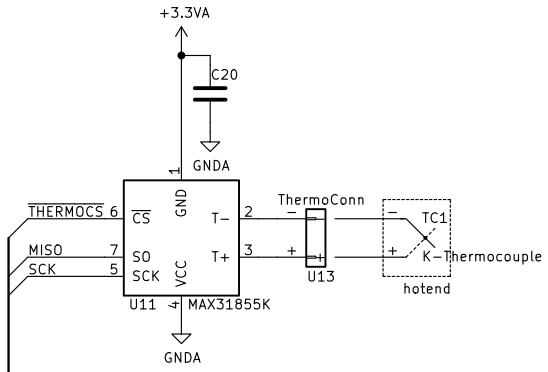
The CAN controller MCP2515 handles most of the CAN protocol, so sending and receiving messages is simple. The controller is connected to the microcontroller via the SPI bus shared by the other digital periphery.

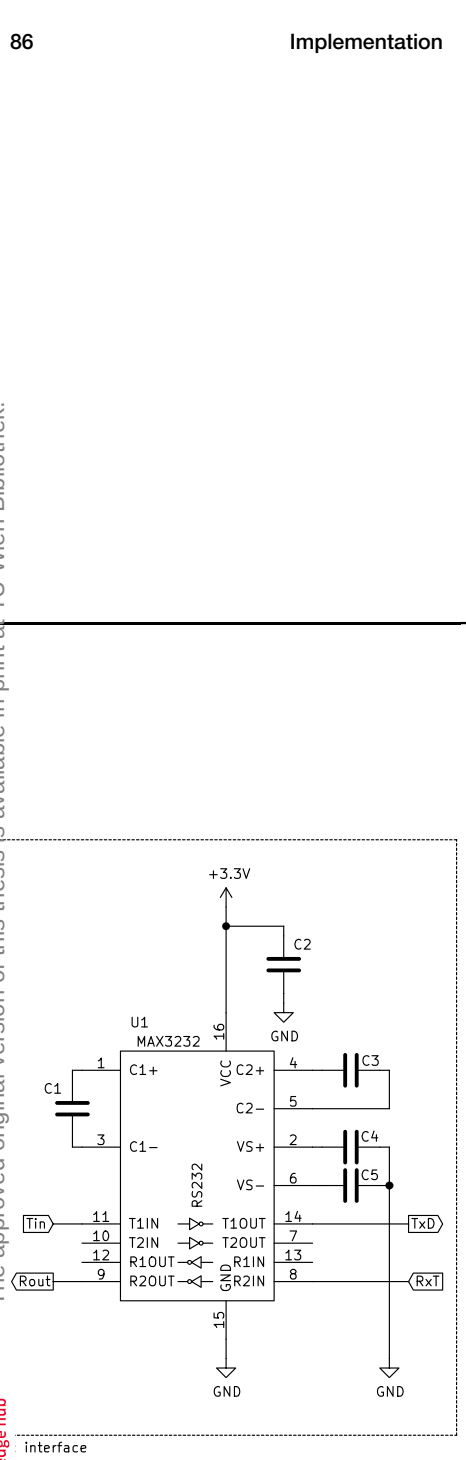




To do more precise measurements the high resolution ADC AD1220 is used. This is dual channel differential 24 bit delta sigma converter. To power its analogue section a low noise 3.3V rail is provided using a LTC1762. While this combination theoretically could profile very precise and accurate AD conversion the breadboard wiring limits the actual performance. However results are still better by a factor of about 100 compared to the integrated AD converter of the ATmega32U microcontroller. The ADC is used to measure current flowing though the heater and to measure the voltage drop at the thermistor (in case this method is used).

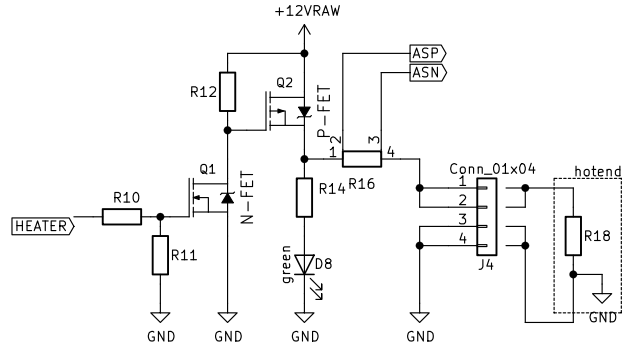
Using the external ADC to measure temperature via an thermistor increased resolution of the measurement and reduced noise which positively affects the regulator characteristics, but it could not solve the accuracy problems. These are inherent to the use of thermistors until they are specifically selected for interchangeability. As a results tow hot-ends with two thermistors are usually not interchangeable without calibration as errors in the order of 5% are expected for cheap thermistors.

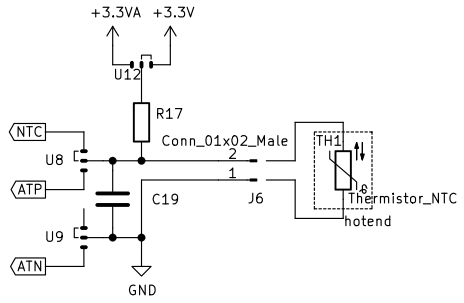




For serial communication a serial universal asynchronous receiver/ transmitter (SUART) module with FIFO on the ATmega32U4 is used, so just a line driver/receiver is needed to realize a serial interface similar to RS232C. Though available hardware flow control is not used, mainly because the primary communication partner this interface is designed for, does not support it either.

To overcome this difficulty the thermistor used in the low cost 3D printer hot-ends was replaced by a K-type (chromel–alumel) thermocouple. Thermocouples are generally interchangeable without calibration (for the used thermonuclear wire the expected tolerances from the wire translate to +/- 1°C at 200°C or 0.5%) and a second advantage is, they can be fabricated from thermocouple wire in exactly the length needed. A hydrogen/oxygen micro welder was used to weld the thermocouple wire. A disadvantage however is the output signal is very small, 8.138 mV at 200C° temperature difference between the cold and hot junction for K-type. Luckily integrated circuitries exist that integrate the necessary amplifier and an analogue to digital converter and could-junction compensation, the used MAX31855 is an example for such an IC. This makes it very easy to accurately and precisely read the temperature using thermocouples.





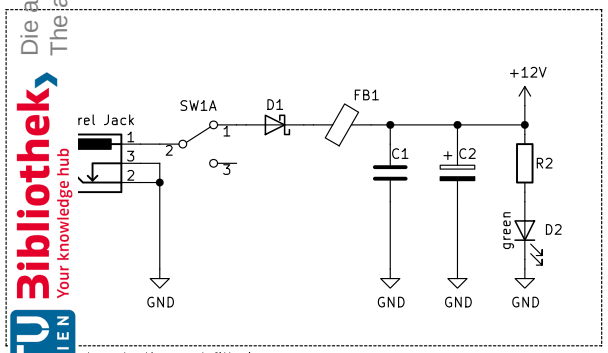
ECM Schematic description

The ECM connects to the STM via 9-pin D-sub connectors and a standard serial cable. It is used to power the STC from the ECM and exchange data via CAN and/or RS232C.

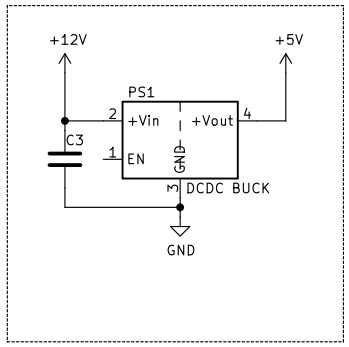
The ECM is powered from an external 12 V, 50 W switch mode power supply, built for this purpose. This 12 V rail is passed on the the STM and to power a DCDC converter generating 5 V powering the Teensy microcontroller module, the Ethernet module and the TFT display. The other digital components are powered from 3.3V generated by a linear voltage regulator on the Teensy module.

The Teensy module contains the MK64FX512VMD12, a 32 bit microcontroller based on the ARM M4F core from NXP/Freescale clocked at 120 MHz, featuring 256 kB of RAM and 512 kB of program memory.

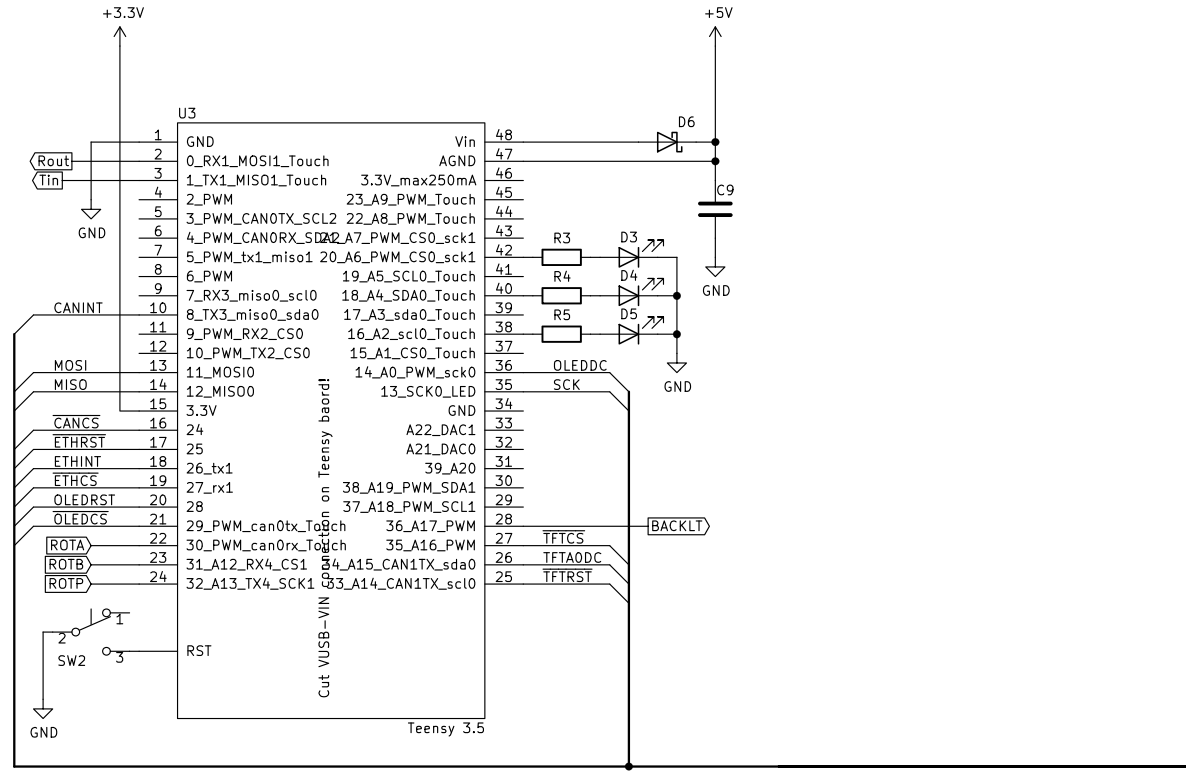
The microcontroller provides a SUART. Line transceivers are added to get RS232C compatibility to communicate with external serial periphery, mainly a modified programmable high current switch mode laboratory power supply (Manson HCS-3400, 1-16V, 40A), but ultimately this was not used used as the corresponding hot-end design proved to be non-functional. The interface itself however can be used to realize a simple, low-latency communication with the KUKA robot that can replace the simple digital I/O used to start and top the extruder from the robot.

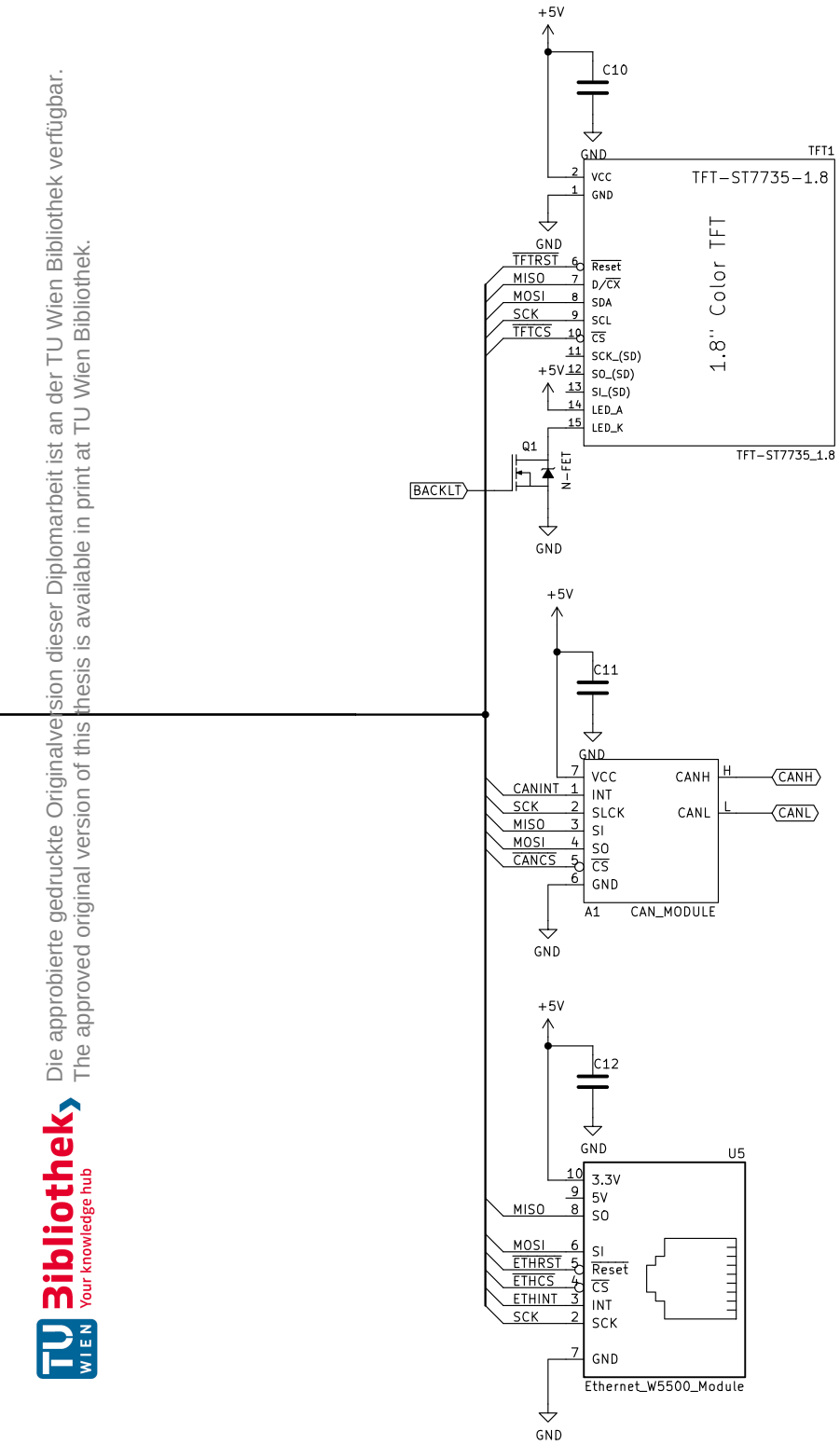


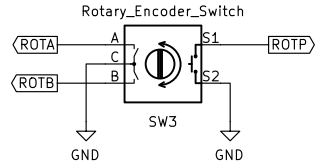
ut protection and filtering



5V logic supply







A WIZnet W5500 Ethernet controller linked to the microcontroller via the SPI bus is used to provide a 100Mbit/s Ethernet interface to connect the ECT a central controller or the KUKA robot directly via TCP/IP. The MK64FX512VMD12 provides built-in Ethernet interface, but the connections are not easily accessible on the Teensy 3.5 module, so it was committed in favor of the external controller.

The same CAN controller as on the STM is used to realize CAN on the ECM. The CAN interface integrated into the MK64FX512VMD12 is not used to be able to use the same CAN library on the STC and ECM.

The display connected via SPI to the microcontroller is a low-cost 128x160 pixel colour TFT with a simple and quite outdated twisted nematic (TN) panel and LED back-light controlled by a Sitronix ST7735S display controller/driver. Contrast and colour rendering are not really up to current standards, but the display is easy to program and inexpensive.

Firmware Stepper Controller Module (STC)

Conceptually the STC firmware is similar to the one of the first generation extruder but its more complex. After initializing the various peripheral components, like the temperature-to-digital convener, the high resolution AD converter, the CAN controller and others the controller enters a main loop running the PID controller, generation

the step and direction signals for the motor driver and handling the communication with the ECM via CAN.

```
while (true) {
      // flash LED to indicate the main loop is running
      blink(GREENLED);

      // receive commands (if new commands are available)
      if (can.available()) {
         can.receive(frame);
         cmdData=parseComand(frame);
         processComands(cmdData);
         reData=prepareAnswer();
         frame=encodeAns(reData);
         can.send(frame);
      }
      // temperature controller
      temp=readThermocouple();
      cmdValue=updatePID(temp, setTemp);
      setHeaterOutput(cmdValue);

      // stepper controller
      stepperUpdate();
      stepper.run();
}
```

The most interesting part here is the CAN bus communication. The CAN controller handles most of it, so the microcontroller only has to prepare a CAN bus frame and send it, or parse a received frame. A CAN frame can have a length of 0-8 bytes. The transport of longer messages requires an additional software layer, for example ISO-TP. For the application here the 8 byte are enough, if used carefully, and they even allow to include a simple checksum to ensure data integrity.

The received frame simply contains a structure as a byte array. In C++ this can be declared as a union of the array and the structures:

```
typedef union {
      byte arr[8];
      struct member {
      unsigned int temp : 12;         // hot-end set point tempera-
                                         ture [deci °C]
      unsigned int feedrate : 12;     // feed rate [steps/hecto
                                         seconds]
      unsigned short stpsRetract : 8; // nr. of deca steps of fast
                                         retract feed
      unsigned short stpsAdvance : 8; // nr. of deca steps of fast
                                         advance feed
      bool feed : 1;                  // enable feeder
      bool feedOverride : 1;          // override run feeder signal
                                         on 24V input
      bool heat : 1;                  // turn heater on
      bool fan : 1;                   // turn fan on
      unsigned short : 4;             // reserved for future use
      unsigned int chksm : 16;        // Fletcher-16 checksum
      };
} ComandData;

ComandData *cmdData;
```

This gives access to the byte array and allows to send it:

```
can.send(&cmdData.arr);
```

But it also give easy access to the members:

```
cmdData.member.temp = 175;
```

The reply sent by the STC to the ECM is similar:

```
typedef union {
   byte arr[8];
   struct member {
      unsigned int temp : 12;  // measured hot-end temperature in
                                  deci °C
      bool feed : 1;           // feeder status (running --> true)
      boo fan : 1              // fan status (on --> true)
      bool err : 1;            // general error
      unsigned int volt : 16;  // heater supply voltage in mV
      unsigned int curr : 16;  // heater current in mA
      unsigned int chksm : 16; // Fletcher-16 checksum
   };
} ReplyData;

ComandData *reData;
```

Firmware Ethernet Controller Module (ECM)

While the small microcontroller in the STC only handles efficient single frame CAN communication and the necessary control tasks the powerful ARM M4F on the ECM can process XML data and send communicate with others on the control network via TCP/IP or UDP, handle a graphical TFT display and user input. The ECM can also forward serial serial communication between the STM to a programmable high current lab power supply, or manage this communication on its own. This was implemented to be able to power and control the very low resistance (ca. 200 mOhm) heating element in on of the slim hot-end prototypes. As later evolution of the slim hot-end use a different heater this feature became obsolete.

Depending on the setting of a jumper the ECM can communicate via TCP/IP or UDP. The TCP/IP mode is primarily implemented to allow the Kuka robot to communicate directly with the ECM by using the Kuka Ethernet RSI XML package. Due to the fact that the available RSI XML package version 1.1 dose not allow to keep multiple sensor connection open at the same time this mode was never used productively as it would require to constantly open and close the connections to the ECM and the central control server CCS.

In the second mode of operation the ECM sends and receives XML messages via UDP. This mode is used to link the ECM to Rhino/Grasshopper by utilizing the gHowl plug-in by Damien Alomar et.al. Just like TCP UDP build upon IP but it is a much simpler protocol. This has some advantages, the receiver does not have to acknowledge received packages which speeds up communication and allows for asynchronous operation, however UDP packages lack a sequence number so there is a possibility that they are received out-of-order. In addition to that, while a checksum I each packages allows to detect corruption of the data during transport, corrupted packages are simply doped and not resent as in TCP. In the LAN consisting of a single layer 2 switch, the Kuka robot, the ECM and two PCs no drop or out-of-order events were observed during the experiments.

The ECM can process XML messages like this one:

```
<ECMCOM>
	<TEMP>175</TEMP>
	<FEEDRATE>200</FEEDRATE>
	<RETRACT></RETRACT>
	<ADVANCE></ADVANCE>
	<FEED>TRUE</FEED>
	<FEEDOVER>FALSE</FEEDOVER>
	<HEATER>ON</HEATER>
	<FAN>OFF</FAN>
</ECMCOM>
```


And send status information back like this:

```
<ECMSTAT>
	<TEMP></TEMP>
	<FEED></FEED>
	<HEAT></HEAT>
	<ERROR><ERROR>
	<VOLTAGE></VOLTAGE>
	<CURRENT></CURRENT>
<ECMSTAT>
```

These XML messages are generated and parsed by Python scripts executed by GhPython in Rhino/Grasshopper. See section XXXX for details on these scripts.

Handling the network communication is the main task of the ECM, in addition to that it also shows the last received command and various information provided by the STC, like the hot-end temperature and set-point and the current federate on the TFT display and provides a simple rotary encoder based user interface that allows to override received settings like the set-point and feed as well to start or stop the feeder motor manually.

Figure 17. 3D Printing filament

Material

Material and its intrinsic behaviour in particular are an essential part of the presented work. Just as it affects the robotic tool-paths and consequently the whole fabrication process, its chemical composition defines the viscosity and binding properties and respectively the density, stability and aesthetic quality of the structure. Nevertheless the aim of this work is to present a material based fabrication where not the material and its modification but the adaptivity of the fabrication process to the unaltered material has been explored.

Due to their high availability on the market and affordable price two of the most common thermoplastic[1] materials for 3D Printing - ABS and PLA filament with 1.75mm diameter - were used for the robotic plastic extrusion. In addition, an extensive documentation about their characteristics and application for FDM is available on the web. This as well as the fact that both materials are easy to handle with extruder generations 1 and 2 allowed for concentrating on the practical experiments without the need to test further plastic filaments.

Polylactic Acid (PLA)

Structure	**some degree of crystallinity, depending on chemical composition (PDLLA is amorphous)**
Density	**1.2-1.4 g/cm**
Melting point	**170-180 C°**
Glass transition temperature	**60-70°C**
Yung's module	**2.7-16 GPa**

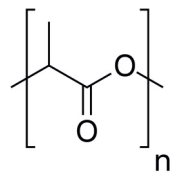


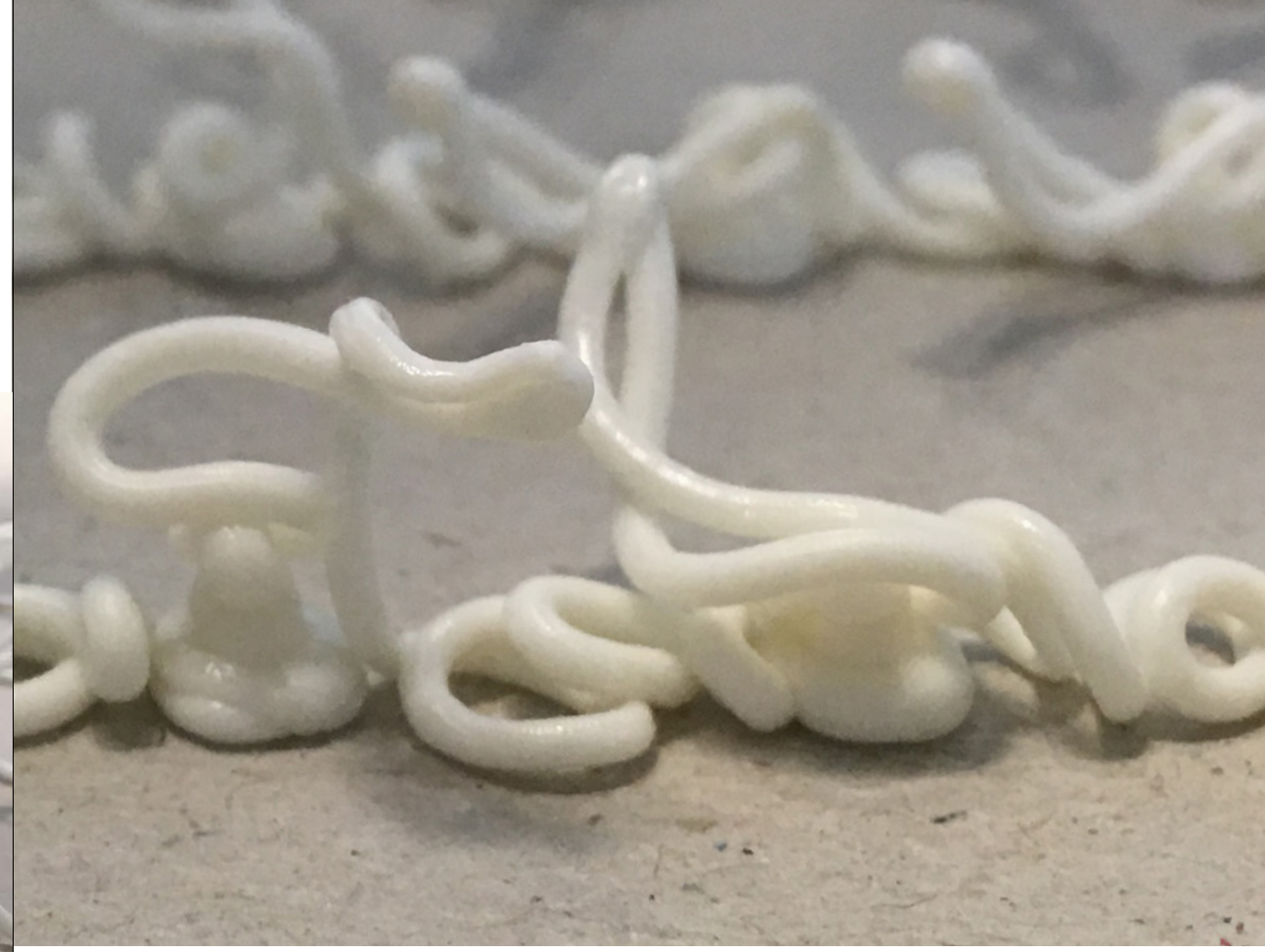
Figure 18. The skeletal formula of PLA

Polylactic Acid (PLA) is a bioplastic derived from renewable resources such as corn scratch, tapioca roots, and sugar-cane and is biodegradable.[2]

The very first tests that inspired this thesis were executed using PLA and delivered a rather abstract representation of the predetermined geometry, compared to the ABS. The unpredictable staking and substantial deflections were considered to be the challenge for the on-

[1] "Thermoplastics are a class of polymers that can be softened and melted by the application of heat, and can be processed either in the heat-softened state (e.g. by thermoforming) or in the liquid state (e.g. by extrusion and injection molding)." From: P.K.Mallick, Materials, Design and Manufacturing for Lightweight Vehicles, 2010

[2] R. Rael and V. San Fratello, Printing architecture: innovative recipes for 3D printing, 2018.



going work therefore the presented case studies were mainly realised using white PLA 1.75mm 3D printing filament. This material behaviour as well as difficulties was especially crucial for the third study “Spontaneous (Dis)order”. Besides practical issues occurred due to the amorphous structure of ABS which as well prioritized the use of PLA.

“Higher Order” on the contrary significantly benefited from the amorphous character of the ABS filament as explained in detail in the next part of this section [see p. 80].

Spontaneous (Dis)Order
see p.179

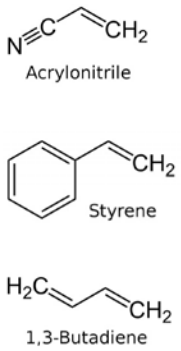


Figure 19. Monomers in ABS polymers.

Acrylonitrile Butadiene Styrene (ABS)

Structure	**amorphous**
Density	**1.1 g/cm²**
Melting point	**no true melting point**
Glass transition temperature	**105°C**
Yung's module	**1.4-3.1GPa**

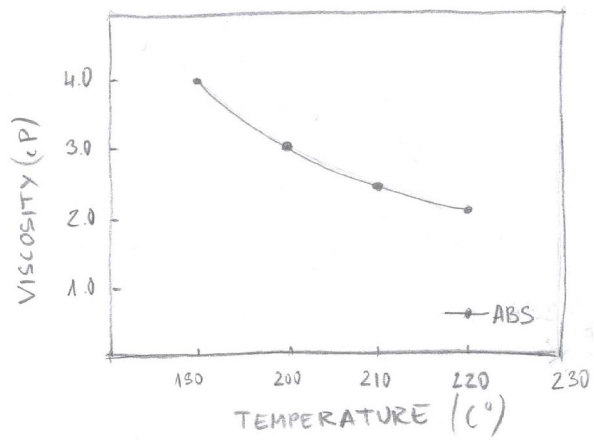
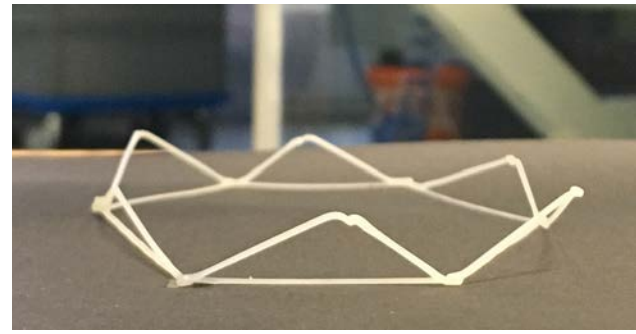


Figure 20. Viscosity of ABS at different temperatures.



Acrylonitrile Butadiene Styrene (ABS) is the most popular material for FDM. It is a an oil-based thermoplastic and owing to its amorphous structure it is more suitable for FDM than highly crystalline polymers. As amorphous polymers do not have a true melting point, with increasing temperature they soften and their viscosity[3] lowers. Nevertheless viscosity remains high enough accounting for fast solidification of the extruded material and maintenance of the extrusion shape.[4] This was confirmed by the outcome of early experiments for the second case study "Higher Order" which delivered quite exact physical representation - less stacking of the deposited "layer" and therefore less deflections - of the given computational model. Furthermore previously extruded material can bond easily to newly extruded.[5] Therefore despite the diameter of the extruded material (nozzle 1mm) as well as the type of geometry, both accounting for the small dimensions of the adjacent areas, layers connected properly to each other.

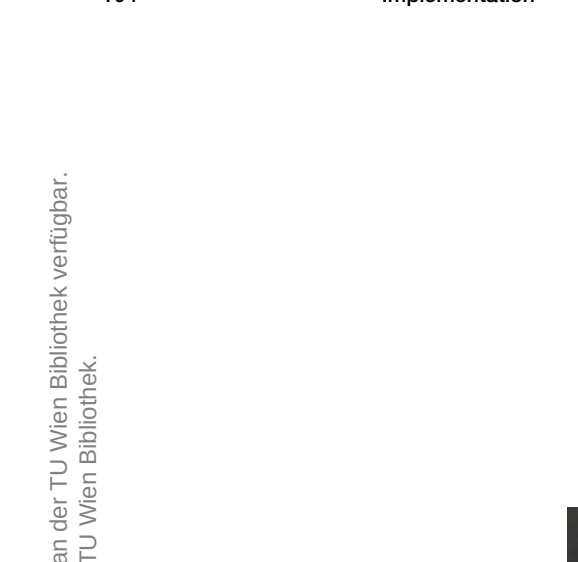
Higher Order
see p.167

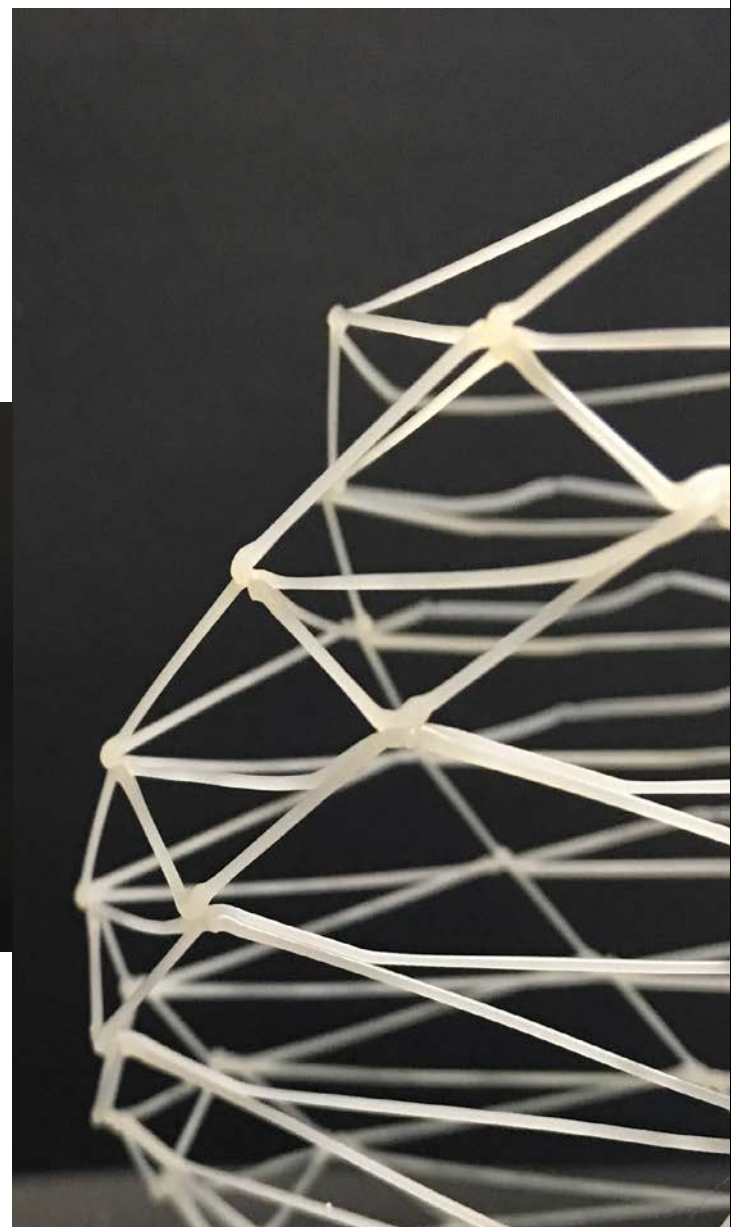
Nevertheless extruding with ABS often delivered insufficient results due to pure printing bed adhesion. Normally this is solved by a heated bed. For the purposes of the thesis this problem was considered irrelevant therefore detached parts have been manually adjusted.

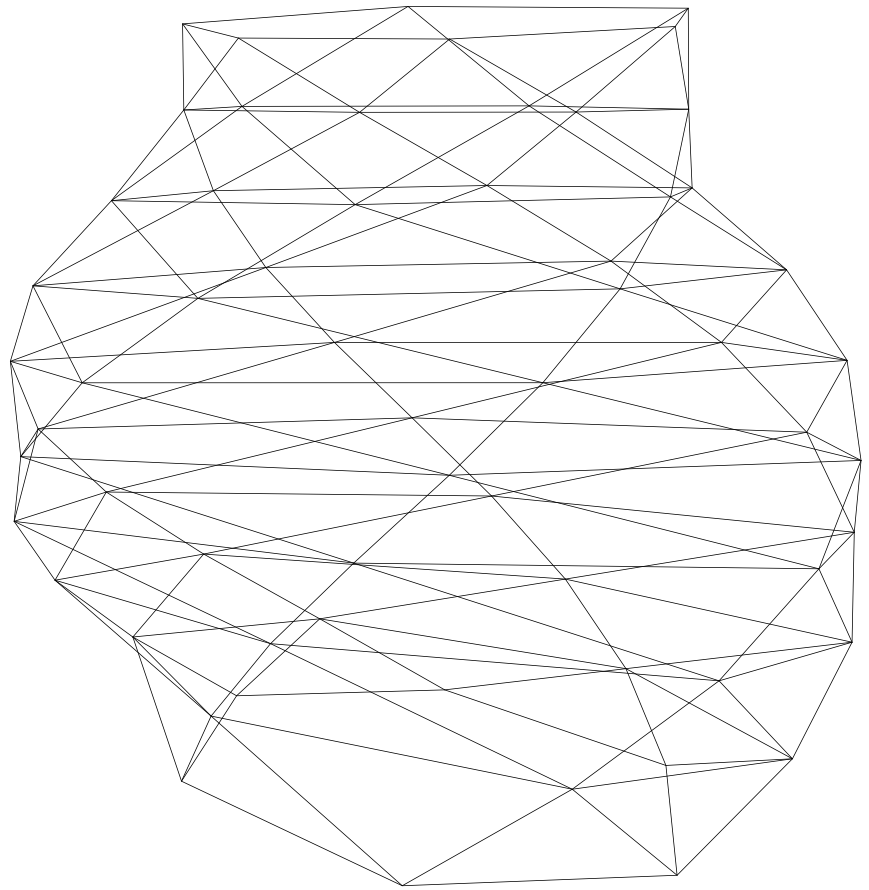
[3] Viscosity is the physical property that characterizes the flow resistance of simple fluids. From: H.F. George, F. Qureshi, Newton's Law of Viscosity, Newtonian and Non-Newtonian Fluids. In: Q.J. Wang, YW. Chung (eds) Encyclopedia of Tribology. 2013

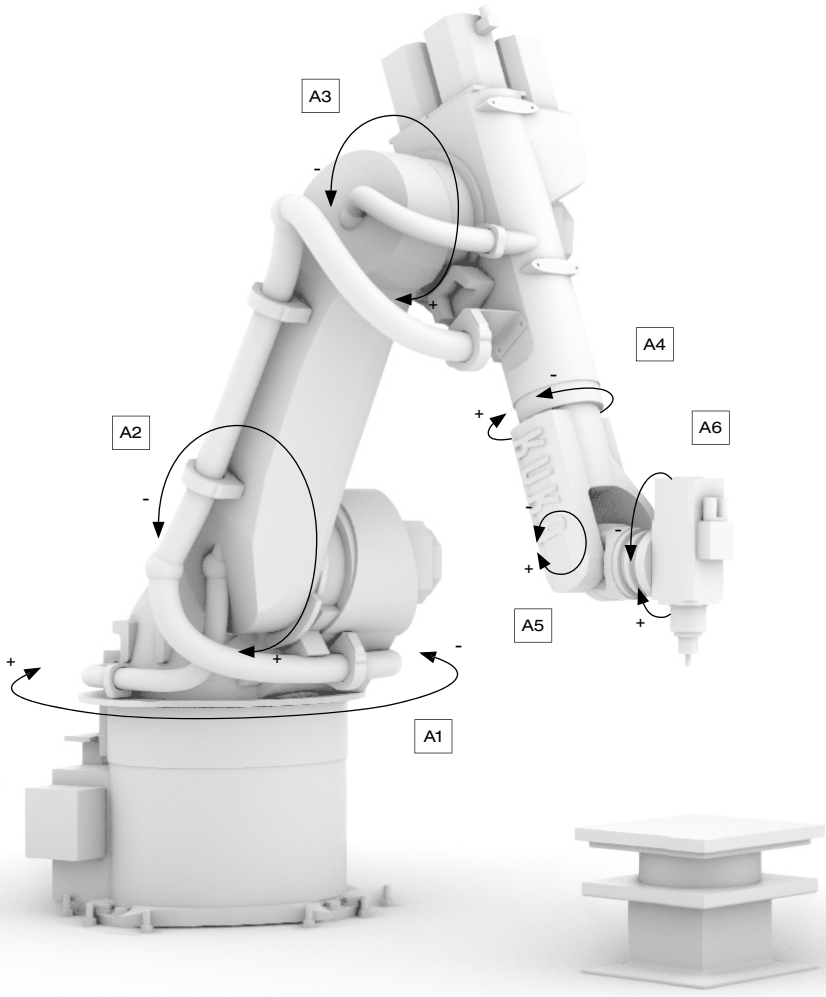
[4] I. Gibson, D. W. Rosen, and B. Stucker, Additive manufacturing technologies: 3D printing, rapid prototyping, and direct digital manufacturing. 2016.

[5] Ibid









Robot axes and directions of rotation.

Robot

The robot used in the experiments is Kuka KR60HA, a typical floor mounted 6-axes medium sized serial manipulator, extended by a single external rotary axes. The robot is rated for a payload of 60 kg and has maximum reach of 2033 mm. The arm is controlled by a Kuka KRC 2 ed. 2005 controller that also controls the external axis and can perform linear interpolation over all seven axis. It is notable that the robot used is a high accuracy (HA) model. This means an absolute calibrated was done by Kuka prior to delivery. Kuka does not give detailed information how this calibration is performed or what the results are but claims high absolute positioning accuracy. While serial manipulators usually show good reputability the generally exhibit poor absolute positioning accuracy. The main causes for this absolute errors are the deflection of the arm structure, the accumulation of joint positioning errors caused by non-linearity of the gear drives and errors in the kinematic model. It is fair to assume that the perform calibration addresses among others these sources of positioning errors.

The available controller software includes several technology packages, extending the basic functionality of the Kuka System Software. Two packages were important for this project, the technology package enabling comfortable teach-in programming and the Ethernet KRL XML package allowing KRL programs to communicate by exchanging XML messages with a server via TCP/IP.

Kuka Robot Language

Kuka Robot Language (KRL) is the programming language used to control the robot. The KRL code consists of two different files with the same name: a permanent data file, with the extension .dat and a movement command file, with the extension .scr.[1]

The KRL Client Program

A central software component are the programs and scripts used to enable direct communication between the Kuka robot and Grasshopper. This is achieved by a custom KRL program capable of sending, receiving XML documents, interpreting the commands received and executing them and a Python script running outside of Rhino, interpreted by a C-Python interpreter, routing the XML documents to and from Grasshopper.

[1] Johanes Braumann & Sigrid Brell-Cokcan, Parametric Robot Control. Integrated CAD/CAM for architectural design, 2011, p. 244.

The KRL program running on the robot uses the KRL XML packages that provides KRL functions to send and receive and XML message via TCP/IP and to parse these.

Ethernet KRL XML Technology Package

The Ethernet KRL XML package is the smaller version of the Remote Sensor Interface (RSI), which is unfortunately not available for the KRC 2 ed. 2005 controller. Just like the RSI it enables the robot to communicate via TCP/IP but not in real-time. While this is a limitation, not only in general, but also for this project, it also made things simpler but relaxing timing constraints on the eternal hard- and software used to control the robot. The KRL XML packages includes basic KRL functions to send and receive and XML message via TCP/IP and to parse these.

The available Kuka Robot Controller (KRC) is equipped with a single network interface controller (NIC) only. This interface is controlled by Windows and primarily used for file tra

nsfer. However the KRC Router package can be used to route specific network traffic between the physical NIC and the virtual NIC used by the real-time kernel vxWorks. That way a connection between a module (the low-level functions of the KRL XML package in this case) running under vxWorks and an external server system (in the nomenclature of the KRL XML package called a sensor) can be realized.

In the specific case here a TCP/IP route is configured to route traffic to IP 192.0.1.2 on port 7008 to 192.160.1.100:7008, the IP of the external system via the physical NIC. Once a socket on the external system is bound to 192.160.1.100 and listens on port 7008 the robot can open a connection by the `EKX_open()` method provided by the KRL XML package:

[2]

```
extSysName[]=”RobotExtrusionExtController”
EKX_open(extSysName)
```

[2] For better readability most KRL XML code snipped omit necessary declaration as well as error handling procedures.

The structure of the data to be exchanged is defined two markup files, one for the send one for the received XML document. For most of the experiment these definitions are used:

[3][4]

```
<!--  XML Structure of Data Send  by the Robot  -->
<Robot>
      <Position X=”” Y=”” Z=”” A=”” B=”” C=”” E1=””></Position>
      <FeedOverride></FeedOverride>
      <ExtruderTemp></ExtruderTemp>
      <Ready></Ready>
      <Tickcount></Tickcount>
</Robot>

<!--  XML Structure of Data Received by the Robot -->
<Elements>
      <Element Tag=”rC” Type=”STRUCTTAG” Stacksize=”100” />
      <Element Tag=”rC.I” Type=”INTEGER” Stacksize=”100” />
      <Element Tag=”rC.FRAME” Type=”FRAME” Stacksize=”100” />
      <Element Tag=”rC.E1” Type=”REAL” Stacksize=”100” />
      <Element Tag=”rC.V” Type=”REAL” Stacksize=”100” />
      <Element Tag=”rC.T0” Type=”REAL” Stacksize=”100” />
      <Element Tag=”rC.T1” Type=”REAL” Stacksize=”100” />
      <Element Tag=”rC.T2” Type=”REAL” Stacksize=”100” />
      <Element Tag=”rC.XM” Type=”BOOLEAN” Stacksize=”100” />
      <Element Tag=”rC.XW” Type=”BOOLEAN” Stacksize=”100” />
      <Element Tag=”rC.XT” Type=”REAL” Stacksize=”100” />
      <Element Tag=”rC.XV” Type=”REAL” Stacksize=”100” />
      <Element Tag=”rC.RUN” Type=”BOOLEAN” Stacksize=”100” />
      <Element Tag=”rC.HALT” Type=”BOOLEAN” Stacksize=”100” />
      <Element Tag=”rC.EXIT” Type=”BOOLEAN” Stacksize=”100” />
</Elements>
```

Now the robot can build a XML document by using the various `EKX_Write...()` functions, for example:

```
EKX_WriteReal(“RobotExtrusionExtController.Robot.Position.X”, $pos_act.x)
```

Once the whole document is built it can be sent to the server by evoking:

```
EKX_Send(“RobotExtrusionExtController”)
```

[3] Tags are kept short to ensure all data fits into a single TCP/IP frame, this simplifies the implementation of the system on the external system and speeds up data transfer at the expense of reduced human readability

[4] Not all fields are used, some are reserved for future use.

The document sent by the robot is received, parsed and processed by the server and an answer is prepared and sent to the robot where it received and stored in a buffer by the `EKX_WaitForSensorData()` function:

```
STRUC STR CHAR s[80]
DECL STR rectext[15]
...
EKX_WaitForSensorData(rectext[1].s[], 0)
```

Once received KRL XML provides functions to parse the data, for example a KRL fame containing potion and orientation information can be retrieved by:

```
rectext[3].s[] = “RobotExtrusionExtController.rC.FRAME”
EKX_GetFrameElement(0,rectext[3].s[],sFrame,bNew)
```

The KRL Client initially opens the connection to the external control system and than simply keeps on sending status information and receiving the next command until a command with the exit flag raised is received. Until that each command is immediately executed. For example the move and extruder on/off part by these KRL commands:

```
IF (sRun==TRUE) THEN
   $VEL.CP=sVelocity         ; set velocity to the commanded value
   $OUT[extruderOnPort]=sExtruderRunMove   ; switch on/off extruder
                                           prior move
   WAIT SEC sTime0           ; wait for extrusion to start
   LIN sFrame                ; move end effector
   WAIT SEC sTime1           ; wait at target possion
   $OUT[extruderOnPort]=sExtruderRunWait   ; switch on/off extruder
   WAIT SEC sTime2           ; wait
ENDIF
```

The used XML markup includes more advanced commands for the extruder. They are not used at the moment due to a the severe limitation of the available version 1.1 of the XML package. While it can mange up to nine different external systems, it can only connect to a single external system at a time and opening and closing a connection is rather slow. So only a single connection to the Central Control Server is used and the communication with the extruder reduced to the bare minimum, on/off of the feeder motor, realized by connecting a digital output on an external Phoenix Contact distributed I/O device directly to the STC instead of using the network capabilities of the ECM. The output can be set conveniently by a simple KRL command:

```
extruderOnPort=123              ; i/o port number of the used output
...
$OUT[extruderOnPort]=sExtruderRunMove     ; set output true or false
```


Frames, Coordinate Transformations and Euler Angles

The manipulator used in the extrusion experiments has seven degrees of freedom in total, six rotary axis in the arm itself and one external rotary axis. To define the systems position and orientation this seven axis angles are necessary and sufficient. Working with the axis angles however is in general not practically feasible and KRL offers a more convenient way by using Cartesian coordinates, Euler angles and the rotary angle of the external axis. Due to the kinematic system of the robot this information is unfortunately not yet sufficient to uniquely define the systems spacial configuration. This requires two more parameters t and s determining the flip state of axis 4/5. In many practical applications t and s can be omitted most of the time, especially if the distance between successive positions/orientations is small t and s usually are kept constant.

The KSS and KRL also offer ways to simply working with different coordinate systems or frames by providing means to define these and convert between them. This allows to define a work piece frame (for example one corner of the machine table) a tool frame (for example the extruder tip) and a rotary axis base frame. This simplifies working with Cartesian coordinates and Euler angles greatly as position and orientation of the extruder tip can now conveniently be specified by giving the relative position and orientation. This also makes is easy to account for changes in the setup by simply correcting the work piece and tool frames without requiring other changes.

KRL provides the so called geometric operator “:” to perform frame transformation.[5] This operator can, for example, be used to transform the work piece frame into the rotary axis frame. This way the rotation angle of the external axis can be changed without affecting the relative position and orientation of the extruder and the rotary table. This link is achieved by the following KRL command where `BASE_DATA[17]` is the work piece offset (relative to the rotary axis base frame):

```
$BASE=EK(MACHINE_DEF[2].ROOT,MACHINE_DEF[2].MECH_TYPE,BASE_
DATA[17]:{x 0,y 0,z 0,a 0,b 0,c 0})


MACHINE_DEF[2]={NAME[] “MDF mit 4mm Graukarton  “,COOP_KRC_IN-
DEX 1,PARENT[] “ “,ROOT {x 1395.29004,y -498.096985,z 295.505005,a
0.00281500001,b 0.590050995,c 0.546595991},MECH_TYPE #EASYS,GEOME-
TRY[] “ “}
```

Planing a robot path in Rhino/Grasshopper requires a way to specify position in Cartesian coordinates, orientation in Euler angles plus the rotation angel of the external axis. Handling the position is strait forward, x, y, z coordinates in Rhino can directly be uses to specify end

[5] KRL Expert Programming Manual p. 37

effector positions. The orientation is a bit more difficult but Grasshopper offers so called planes to define coordinate frames. This can be done in different ways, for example by a giving the origin, a unit vector specifying the x-axis direction and a point on the x/y-plane.

Often coordinates given in one frame need to be expressed in terms of another frame. In KRL the geometry operator ":" can be used to do this kind of coordinate transformations. Rhino/Grasshopper also provide convnient ways to do this transformation, hover the problem is worth taking a closer look at.

For two such Grasshopper planes, now referred to as frames, denoted by the unit vectors $\vec{u_1}$, $\vec{u_2}$, $\vec{u_2}$ and $\vec{v_1}$, $\vec{v_2}$, $\vec{v_2}$ describing their x,y,x-axis and the vectors $\vec{p}$ and $\vec{q}$ to their origins, one frame can be expressed using the other:

$$\begin{aligned}\vec{u_1} &= a_{11}\vec{v_1} + a_{12}\vec{v_1} + a_{13}\vec{v_1}\\ \vec{u_2} &= a_{21}\vec{v_1} + a_{22}\vec{v_1} + a_{23}\vec{v_1}\\ \vec{u_3} &= a_{31}\vec{v_1} + a_{32}\vec{v_1} + a_{33}\vec{v_1}\\ \vec{p} &= a_{14}\vec{v_1} + a_{24}\vec{v_1} + a_{34}\vec{v_1}\end{aligned}$$

Taking the matrix ***M*** with

$$M = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14}\\ a_{21} & a_{22} & a_{23} & a_{24}\\ a_{31} & a_{32} & a_{33} & a_{34}\\ 0 & 0 & 0 & 1\end{bmatrix}$$

can be expressed as

$$\vec{u}^T = \vec{v}^T M$$

This implies that

$$\vec{u} = M^T\vec{v} \text{ and } \vec{u} = (M^T)^{-1}\vec{v}.$$[6]

In case the frame is given by a vector $\vec{n}$ pointing int z-direction and a vector $\vec{x}$ in the zx-plane the orthonormal basis $\vec{u_1}$, $\vec{u_2}$, $\vec{u_3}$ can easily be constructed.

$$\begin{aligned}\vec{u_3} &= \frac{\vec{n}}{|\vec{n}|}\\ \vec{u_1} &= \vec{u_3} \times \frac{\vec{x}}{|\vec{x}|}\\ \vec{u_2} &= \vec{u_3} \times \vec{u_2}\end{aligned}$$

[6] E. Angel, D. Shreiner. Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL. Addison-Wesley Publishing Company, USA, 6th edition, 2011

It is generally not necessary to implement this, as these coordinate planes are well supported by Grasshopper and the RhinoCommon API. Both Grasshopper as well as RhinoCommon include functions/components to get a coordinate planes corresponding transformation matrix. In RhinoCommon the Transform structure provides the function `PlaneToPlane()` to do this:[7]

```
trans=Rhino.Geometry.Transform.PlaneToPlane(plane0, plane1)
```

Now the members M00, M10, ... M44 represent a transformation matrix M in homogeneous coordinates where R=[[M00, M01, M02], [M10, M11, M12], [M20, M21, M22]] represents the a 3D rotary matrix and p=[M03, M13, M23] a translation. The RhinoCommons API as well as Grasshopper provide functions/components to do this kind of transformation.

The Euler angles A, B, C, ϕ, θ, ψ or role, pitch, yaw as used by Kuka describe the rotation of a frame around its z-axis by the angle ϕ counterclockwise, than by the angle θ around the new, already rotated, y-axis and finally by ψ around the x-axis formed by the two previous rotations. Expressed by rotation matrices this reads:

$$\vec{p'} = M\vec{p}$$

M can be decomposed into three extrinsic rotations

$$M = R_x(\psi)R_y(\theta)R_z(\phi)$$

with

$$R_x(\psi) = \begin{bmatrix}1 & 0 & 0\\ 0 & \cos\psi & -\sin\psi\\ 0 & \sin\psi & \cos\psi\end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{bmatrix}$$

$$_z(\phi) = \begin{bmatrix}1 & 0 & 0\\ 0 & \cos\psi & -\sin\psi\\ 0 & \sin\psi & \cos\psi\end{bmatrix}$$

[7] In Grasshopper the Orient component outputs the transformation matrix in homogeneous coordinates.

Expaneded this reads:

$$M = R_x(\psi)R_y(\theta)R_z(\phi) =$$
$$\begin{bmatrix} \cos\theta\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \cos\theta\sin\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi \\ -\sin\theta & \sin\psi\cos\theta & \cos\psi\cos\theta \end{bmatrix}$$

For a given rotation matrix

$$R \qquad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{21} & r_{22} & r_{13} \end{bmatrix}$$

to get the first Euler angle θ or B note that $r_{31} = -\sin\theta$, so

$$\theta_1 = -sin^{-1}r_{31}$$
$$\theta_2 = \pi + sin^{-1}r_{31}$$

For ψ the fact that $\frac{r_{32}}{r_{33}} = \tan\psi$ can be exploited

$$\psi = \operatorname{atan2}\left(\frac{r_{32}}{\cos\theta}, \frac{r_{3_3}}{\cos\theta}\right).$$

The, at first glance, seemingly unnecessary division by $\cos\theta$ accounts for the fact that $\psi = (-r_{32}, -r_{33})$ for $\cos\theta < 0$. The special case $\cos\theta = 0$ is treated later.

Similar for ϕ or C we note $\tan r_{21}/r_{11}$ and, considering the two possible ψ, get:

$$\phi_1 = \operatorname{atan2}\left(\frac{r_{12}}{\cos\theta_1}, \frac{r_{13}}{\cos\theta_1}\right)$$
$$\phi_2 = \operatorname{atan2}\left(\frac{r_{12}}{\cos\theta_2}, \frac{r_{13}}{\cos\theta_2}\right)$$

For $\cos\ = 0$ different elements of the rotation matrix have to be chosen to get:

For $\theta = \pi/2$:

$$\psi_1 = \phi + \operatorname{atan2}(r_{12}, r_13)$$

For $\theta = -\pi/2$:

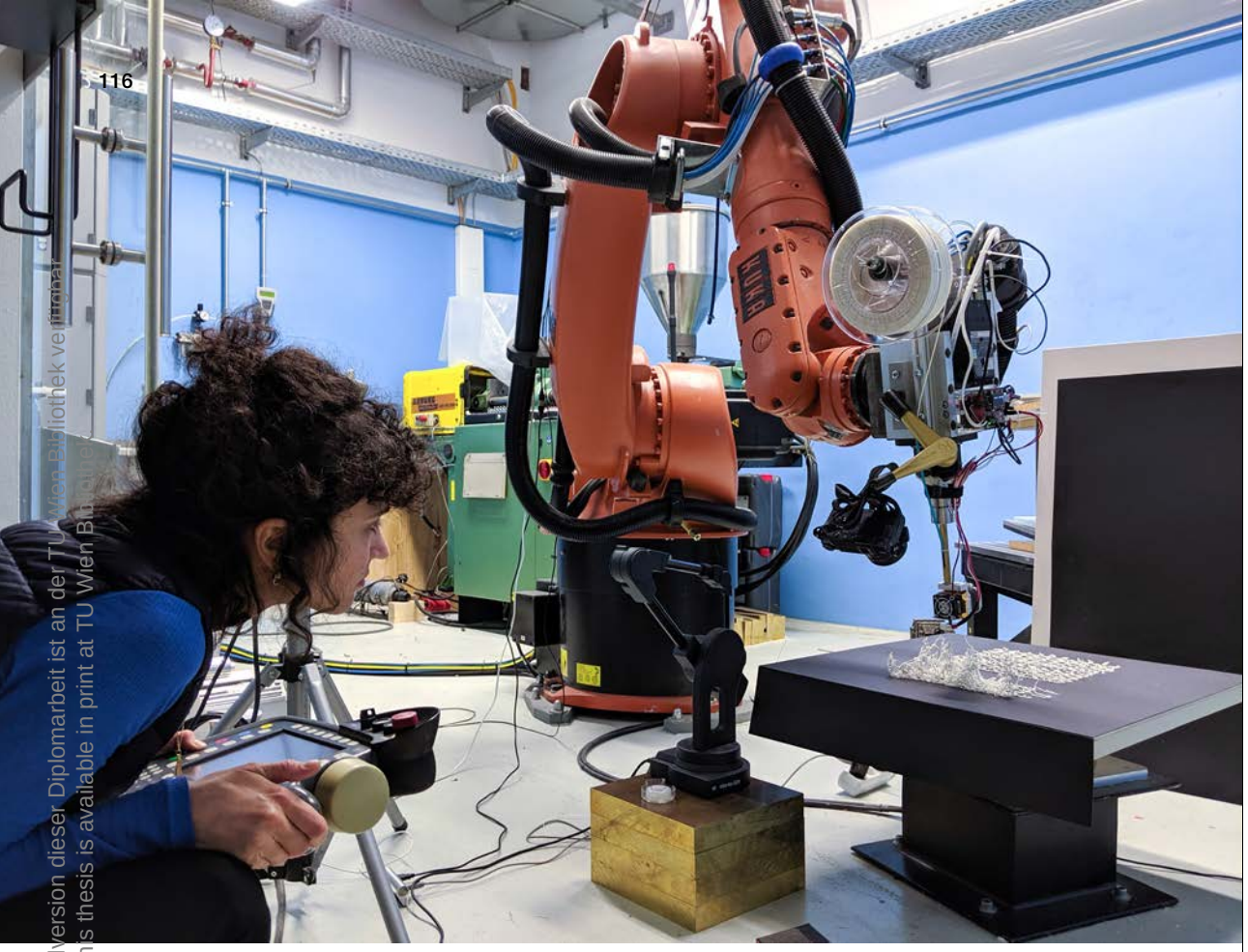
$$\psi_1 = -\phi + \operatorname{atan2}(-r_{12}, -r_13)$$

It is not necessary to implement this as The Transform structure of the RhinoCommons API includes functions to convert a rotation matrix in to Euler Angles and back. This allows to solve the problem at hand with a few lines of Python:


```
def euler(p1):
    # reference plane
    p0 = Rhino.Geometry.Plane.WorldXY

    trans = Rhino.Geometry.Transform.PlaneToPlane(p0, p1)

    # remove translation part
    trans.M03 = trans.M13 = trans.M23 = 0

    # GetEulerZYZ or GetYawPitchRoll
    rot, A, B, C = trans.GetYawPitchRoll()

    if not rot: raise Exception("No rotation from p0=WorldXY to p1
    could be calculated.")
    return math.degrees(A), math.degrees(B), math.degrees(C)

def plane((A,B,C)):
    trans = Rhino.Geometry.Transform.RotationXYZ(m.radians(A), m.ra
    dians(B), m.radians(C))
    plane = Rhino.Geometry.Plane.WorldXY
    plane.Transform(trans)
    return plane
```

The built in RhinoCommon functions give identical numerical results (up to rounding errors) when compared to the explicit calculation above. To ensure that these conversion work as expected in practice extensive testing by generating different planes and sending the corresponding commands to the robot was done.

The Grasshopper planes are also used to represent the robots work piece and tool frames, as well as the rotary axis base frame.



Teach-in

What is Teach-in?

Robot programming by demonstration is a technique that has been initially developed to teach computers or robots tasks by the means of demonstration without the need to program the instructions.[1] Since it does not require additional interface this approach was successfully adopted at an early stage in order to test the concept of the proposed custom fabrication strategy.

In order to generate adapted robot instructions, responding to deviations of the physical model, an online programming technique has been applied to enable implementing feedback data from the physical to the digital environment. In teach-in mode[2] the teach pendant is used to drive the robot to a desired positions and extract point coordinates. Normally online programming means that the programming happens directly with the robot by guiding it to predefined positions with specified orientation[3] and state of the end effector. Eventually the robot moves autonomously to the saved points in space. In this case though, the collected data has been imported to the digital model and processed.

Workflow

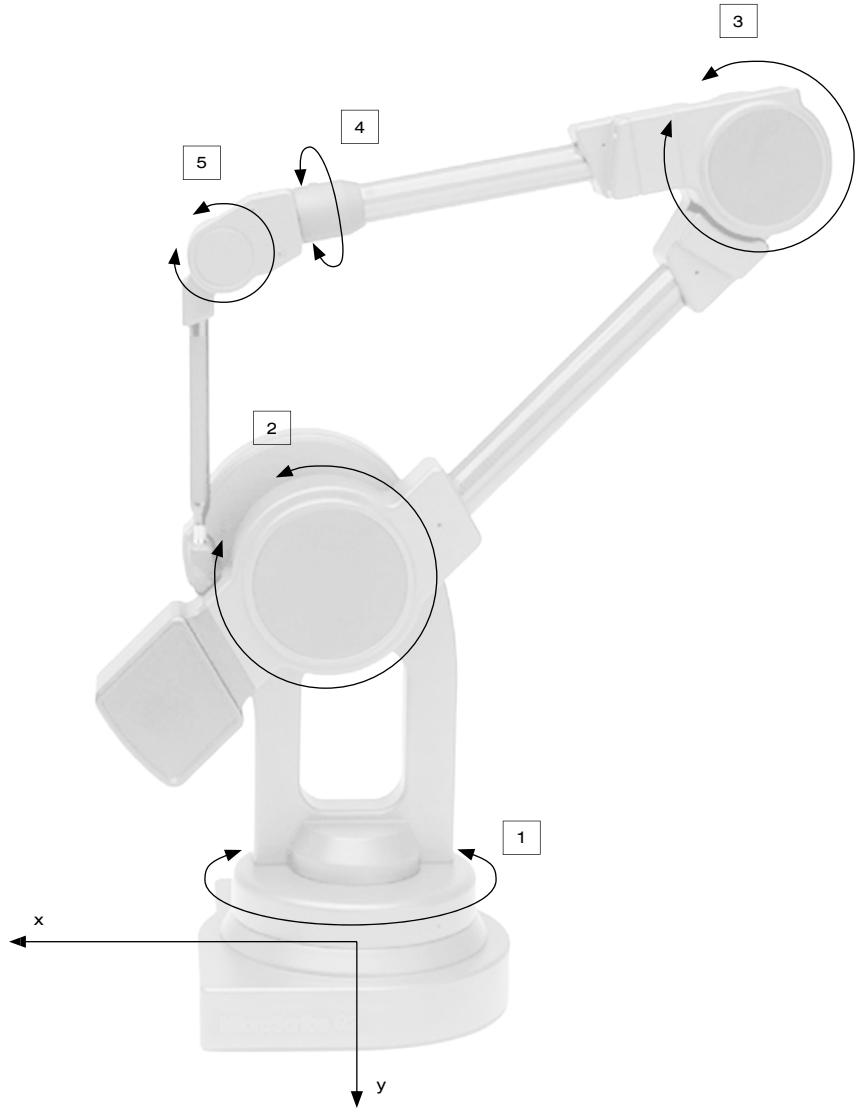
Kuka robot has been manually directed, via the KCP (KUKA Control Panel), to the nodes of the deposited layer. Subsequently coordinates of the reached points are saved to a .dat file on the robot controlling system. The file containing a list of points is then imported to the computational model in the Grasshopper environment where the text is post-processed so that old initially calculated nodes coordinates could be placed by the new actual nodes positions. New robot instructions allowing for precise material aggregation of the physical model generated are saved to a new KRL program and transferred to the robot system for execution.

Applying teach-in-programming was very useful to prove the concept of point measuring. Nevertheless the process is very time consuming, since for one experiment multiple layers are being extruded and a teach-in operation is needed after each deposited layer. Furthermore the extruder nozzle needs to cool down before measuring the points of the printed geometry otherwise the material melts and the object is damaged.

[1] Programming by demonstration: https://en.wikipedia.org/wiki/Programming_by_demonstration

[2] Teach-in is one of the first programming by demonstration strategy proposed.

[3] Stefan Hesse & Viktorio Malisa, Taschenbuch: Robotik, Montage, Handhabung, 2010

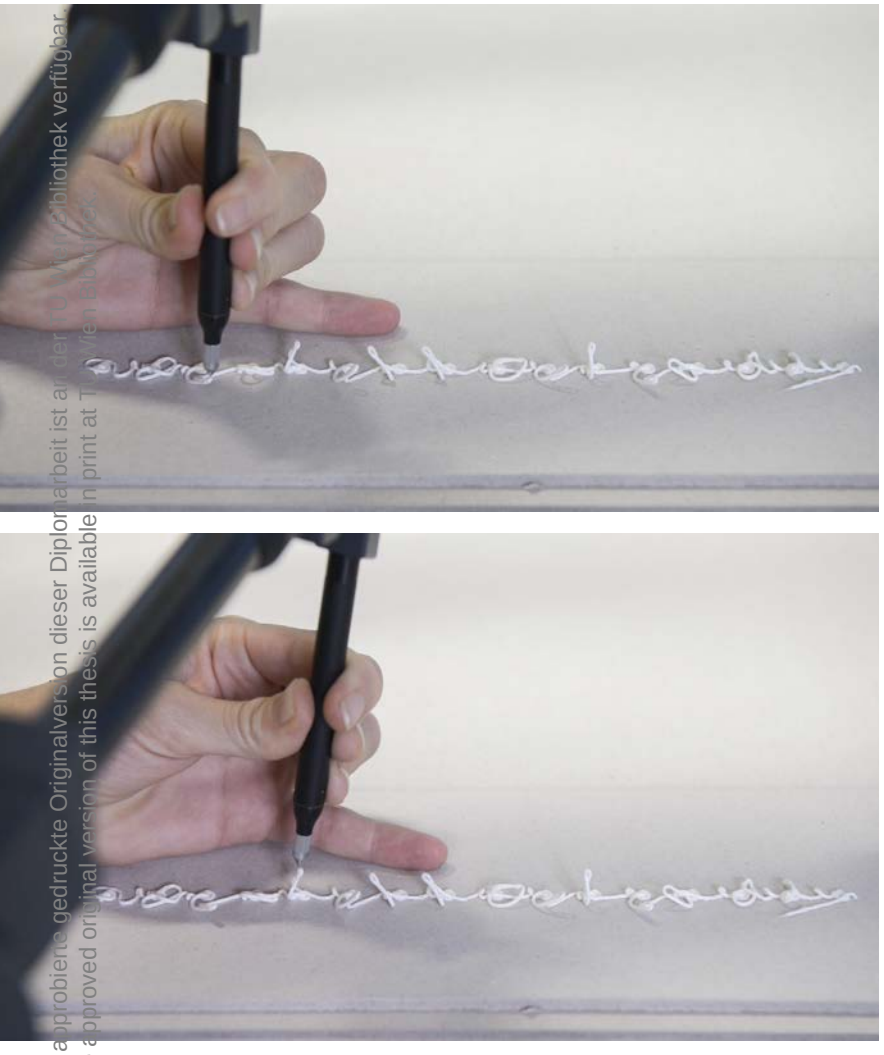


MicroScribe axis and coordinate system orientation.

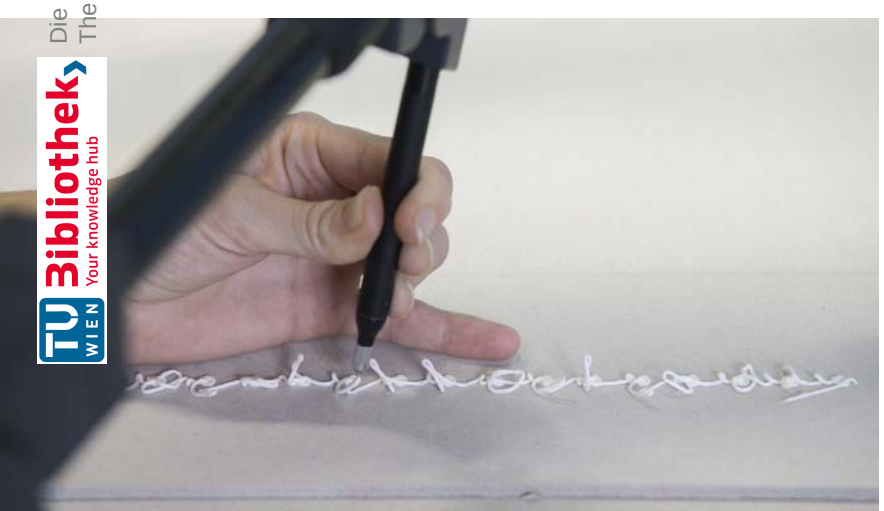
Coordinate Measuring Machine

Since the Teach-in method proved to be too time consuming an alternative method to measure the points has been successfully tested and implemented to the experimental setup. A coordinate measuring machine (CMM), an Immersion MicroScribe-3D, is used for the three-dimensional digitizing of points on the physical objects. Due to its manual control — accounting for a fast operation — the feedback process was significantly accelerated compared to the teach-in technique.

The MicroScribe is a type of small, low cost, manual coordinate measurement machine. It uses a similar serial kinematic as the Kuka robot but without any active drives. The arm is passive and follows the manual moment of the tip. By measuring the rotation angle in the five joints and calculating the forward kinematic the tips position and orientation (pitch and yaw) can be determined. It is not possible to rotate the tip independently as the arm has only five degrees of freedom. The MicroScribe is still a very useful tool for the feedback experiments as the extrusion process also only expresses five degrees of freedom (it is invariant to rotations of the extruder around the tip axis as well as of the rotary table).

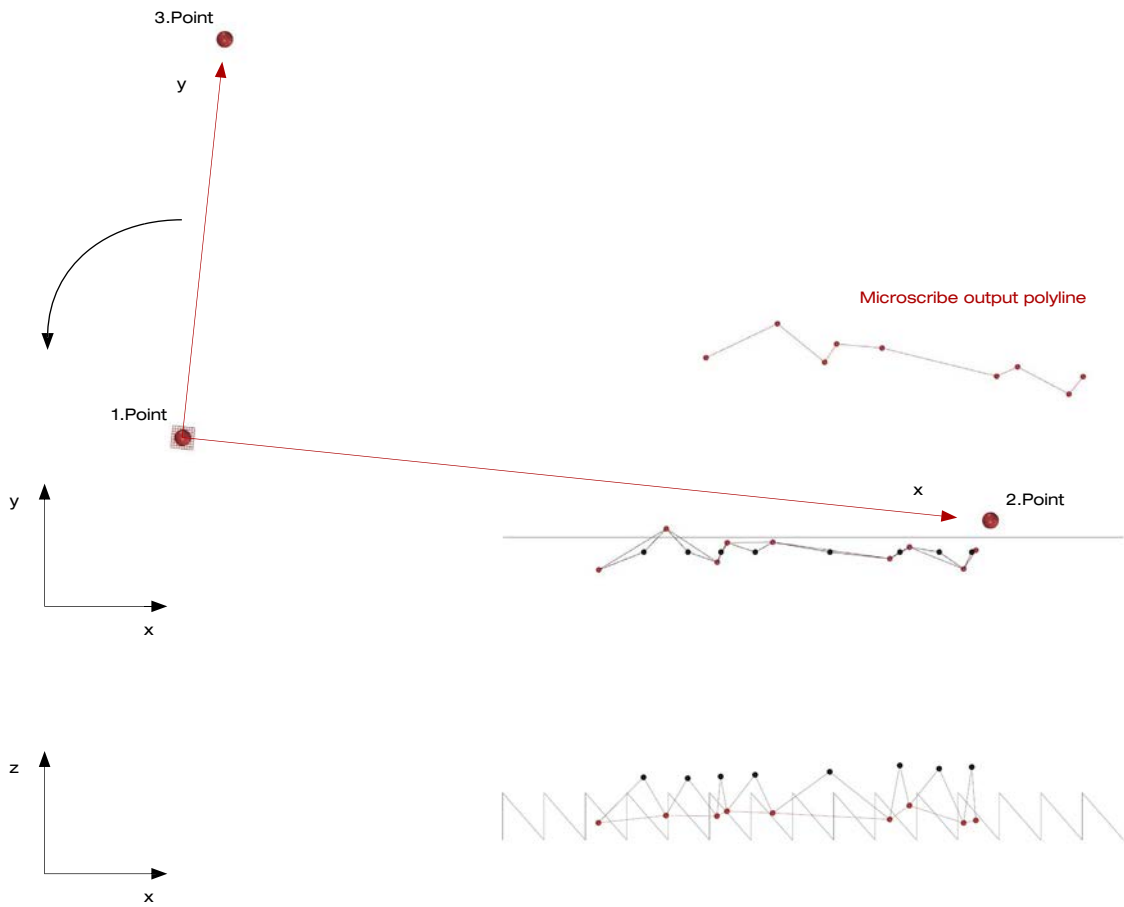


Top View



Front View

Coordinate System Orientation and Transformation
Measured coordinates are imported to Rhino and then to Grasshopper. The first three measured point are the orientations of x,y and z axes of the robot coordinate system. The transformation is performed in Grasshopper. The measured points of the printed geometry replace the calculated point of the model and therefore define the next toolpath of the robot and location of the material extruded.



MicroScribe Custom Recorder

The MicroScribe is directly supported by Rhino. With a click of a button Rhino is ready to record the tips position in space. However the default MicroScribe Rhino plug-in does not offer any way to gather information on the tip orientation. In addition to that using the Rhino plug-in every time the procedural model in Grasshopper needs information on the true location of some points is a bit cumbersome.

To overcome both problems a Windows console application was written to record the MicroScribe tip positions and orientation and write this information to a file. The program runs in the background, all user interaction is done by using the two food paddles of the MicroScribe. This allows to measure a sequence of positions and orientation and have them written to file without any user input via keyboard or mouse on the PC. In Grasshopper every modification of that file can be detected and used to trigger the re-evaluation of the sketch to generate the next sequence of robot commands and automatically stream them to the robot for execution utilizing the central control server.

C# was chosen to write this console application as it will allow to easily turn it into a Grasshopper component later (eliminating the need to start an additional program and to use a file for data exchange). Immersion, the original manufacturer of the MicroScribe, does seam to no longer exist, but Solution Technologies, Inc. has taken over sales and support and provides a Windows 10 compatible software development kit. This SDK includes C++ libraries to access the drivers API and in turn communicate with the MicroScribe hardware. This functions can be called from C# by using the platform invoke (P/Invoke) technology provided by .Net to accesses unmanaged code libraries.[1] In case of the rather simple MicroScribe libraries, just the library methods have to be imported and some of the structures used in C++

[1] https://docs.microsoft.com/en-us/dotnet/standard/native-interop/pinvoke
https://developer.rhino3d.com/guides/rhinocommon/wrapping-native-libraries

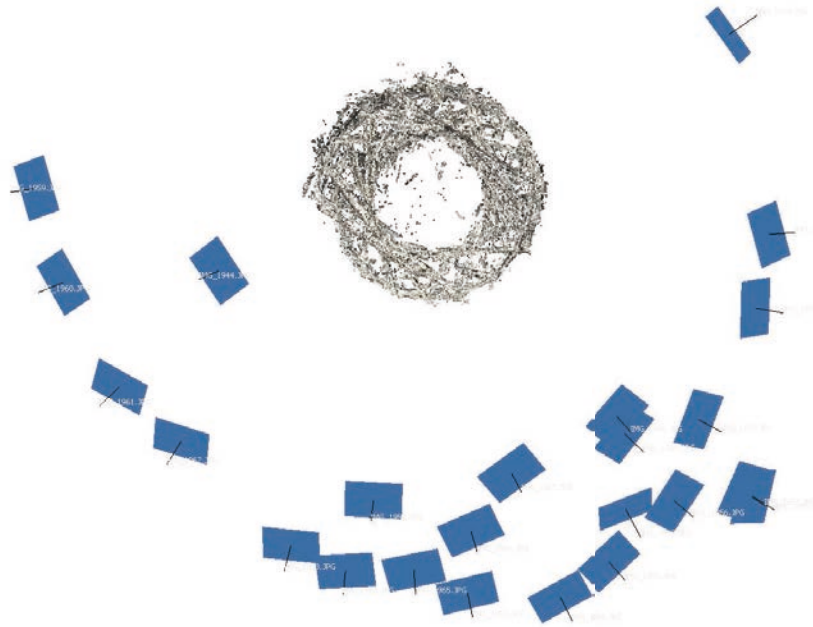
have to be reimplemented in C#. For the `ArmGetTipOrientation()` function the interesting lines of code (for 64 bit environment) read like this:

```
public class MicroScribe
      {

         [StructLayout(LayoutKind.Sequential)]
         public struct angle_3D
      {
         public float x;
         public float y;
         public float z;
         }

         [DllImport("armdll64.dll")]
         public static extern int ArmGetTipOrientation(ref angle_3D
         pAngles);
      ...
      }
      ...

      MicroScribe.angle_3D angles = new MicroScribe.angle_3D();
      int err = MicroScribe.ArmGetTipPosition(ref position);
```

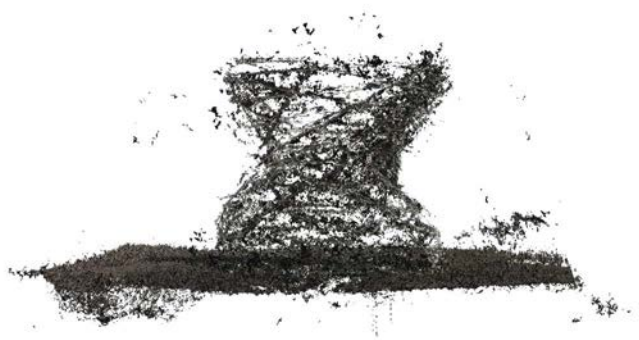
For a minimal implementation of the recorder only nine of the SDKs C++ functions and two of the structures need to be made available in C#, in total only some 200 lines of code are necessary to write the position and orientation data to disk.

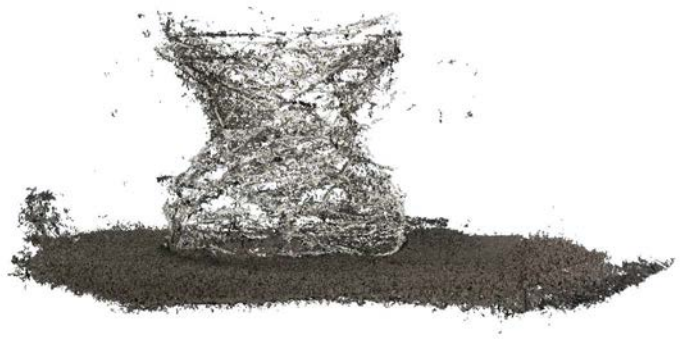
Photogrammetry

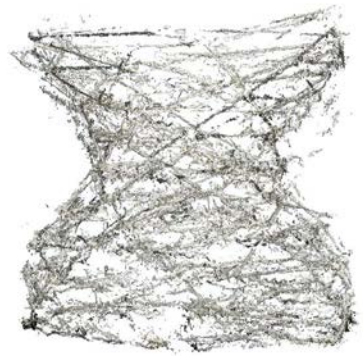
“Photogrammetry is the art, science and technology of obtaining reliable information about physical objects and the environment through the process of recording, measuring and interpreting photographic images and patterns of electromagnetic radiant imagery and other phenomena. A simplified definition could be the extraction of three-dimensional measurements from two-dimensional data (i.e. images); for example, the distance between two points that lie on a plane parallel to the photographic image plane can be determined by measuring their distance on the image, if the scale of the image is known. Close-range photogrammetry refers to the collection of photography from a lesser distance than traditional aerial (or orbital) photogrammetry. Photogrammetry is as old as modern photography, dating to the mid-19th century.”[1]

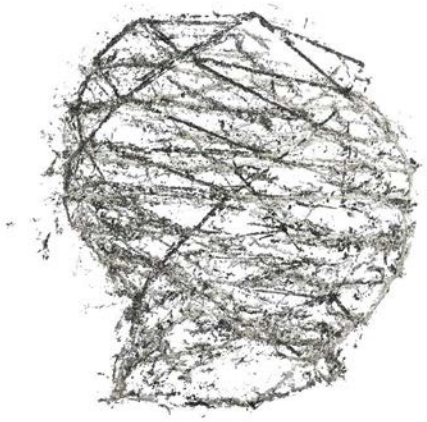


[1] Photogrametry: https://en.wikipedia.org/wiki/Photogrammetry accessed July 2019









Photogrammetry would be a quick and easy way to gain 3D information, unfortunately the thin, sparse, feature less extruded structures are difficult to capture. The resulting point could is very noisy. Due to this noise extensive post processing would be necessary to obtain accurate measurements of the node coordinates. The necessary information would be there, we as humans can easily identify the nodes, but achieving the same in a computer is difficult. One approach to tackle this problem would be using recent progress in machine learning any trying to train a deep convolution neural network to find the coordinate, but his is out of the scope of this thesis.

Time of Flight Camera

The PMDtech PicoFlexx time-of-flight camera is a small module, connected to the PC via USB. It used the Infinion IRS1145C image sensor to captures depth images by detecting the phase shift between the modulated emitted infrared light and light reflected back from the scene. PMDtec profiles an easy to work with SDK including support for C# and other .Net languages. This allows to build a simple Grasshopper component outputting the captured depth information as a high field or point cloud for further processing in Rhino/Grasshopper.

PicoFlexx Grasshopper Component

The camera can be accessed using the RoyleDotNet library. To initialize the first connected camera just a few lines of C# code are necessary:

```
private static RoyaleDotNet.CameraDevice device;
...
device = camManager.CreateCamera(connectedCameras[0]);
```

To get depth information from the camera a listener is implemented and installed:

```
class DataReceiver : RoyaleDotNet.IDepthDataListener
      {
         public DataReceiver(Picoflexx parent)
         {
         this.parent = parent;
         }
         public RoyaleDotNet.DepthData data;
         public void OnNewData(RoyaleDotNet.DepthData data)
      {
         this.data = data;
         }
      }

...

      receiver = new DataReceiver(this);
      status = device.RegisterDepthDataListener(receiver);
```

Now every time the camera captures a new frame the listener is called and the depth data is stored in a variable accessible to the Grasshopper comment.

This data can now be used in the SolveInstance() method to generate and output to Grasshopper, for example, a point cloud:

```
private List<Vector3d> points = new List<Vector3d>(4000);
protected override void SolveInstance(IGH_DataAccess DA)
      {
         RoyaleDotNet.DepthData data;
         data = receiver.data;
         points.Clear();
         for (int i = 0; i < data.points.Length; i++)
         {
         Vector3d p = new Vector3d(data.points[i].x, data.
         points[i].z, data.points[i].y)
         points.Add(p);
         }
         DA.SetDataList(“points”, points);
      }
```

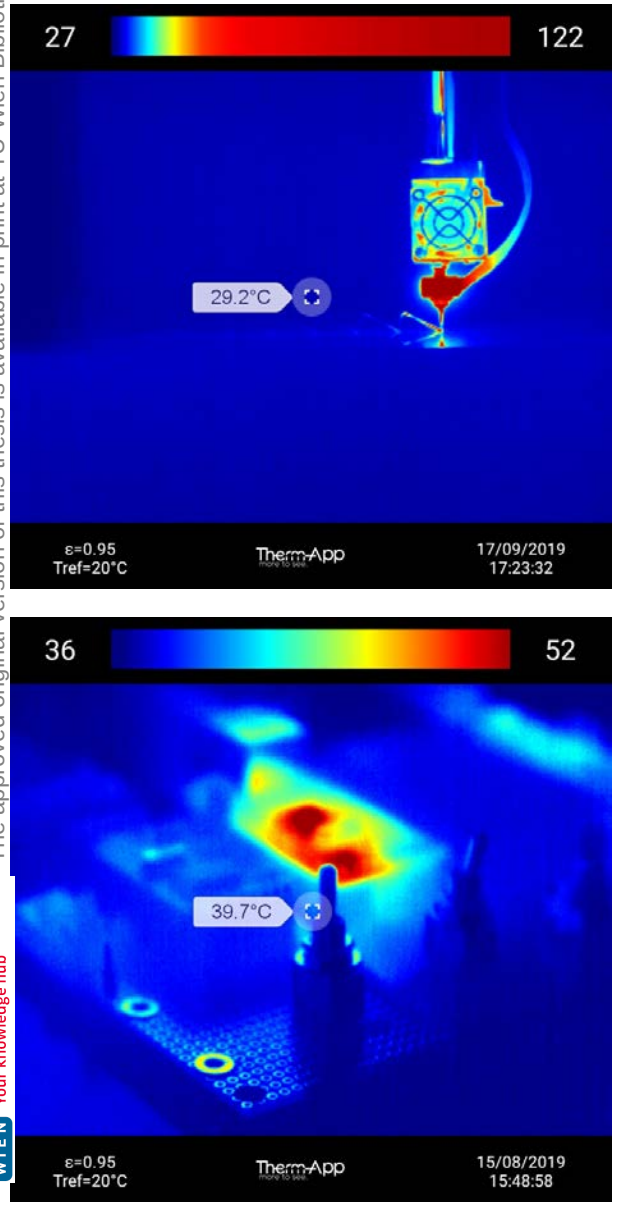
Of course the Grasshopper parameter points hat to be registered:

```
protected override void RegisterOutputParams(GH_Component.GH_Output-
ParamManager pManager)
      {
         pManager.AddPointParameter(“points”, “pnts”, “3D points
from
Picoflexx”, GH_ParamAccess.list);
      }
```

In case a depth image is needed it can easily be created inside SolveInstance() in this way:

```
bitmap = new Bitmap(data.width, data.height);
for (int x = 0; x < data.width; x++)
{
      for (int y = 0; y < data.height; y++)
      {
         double z = data.points[x + data.width * y].z;
         int c = (int)z*scalefactor;
         bitmap.SetPixel(x, y, System.Drawing.Color.FromArgb(c, c,
         c));
      }
      DA.SetData(“img”, bitmap)  // the parameter has to registered
                                    in  RegisterOutputParams()
      }
```

In the implementation of the Grasshopper plug-in these code snippets are extended by error handling, noise reduction by combining multiple frames, the possibility to filter points according to a defined a z-range of interest, automatic scaling of the colour range, false colouring and more.



Thermal Imaging

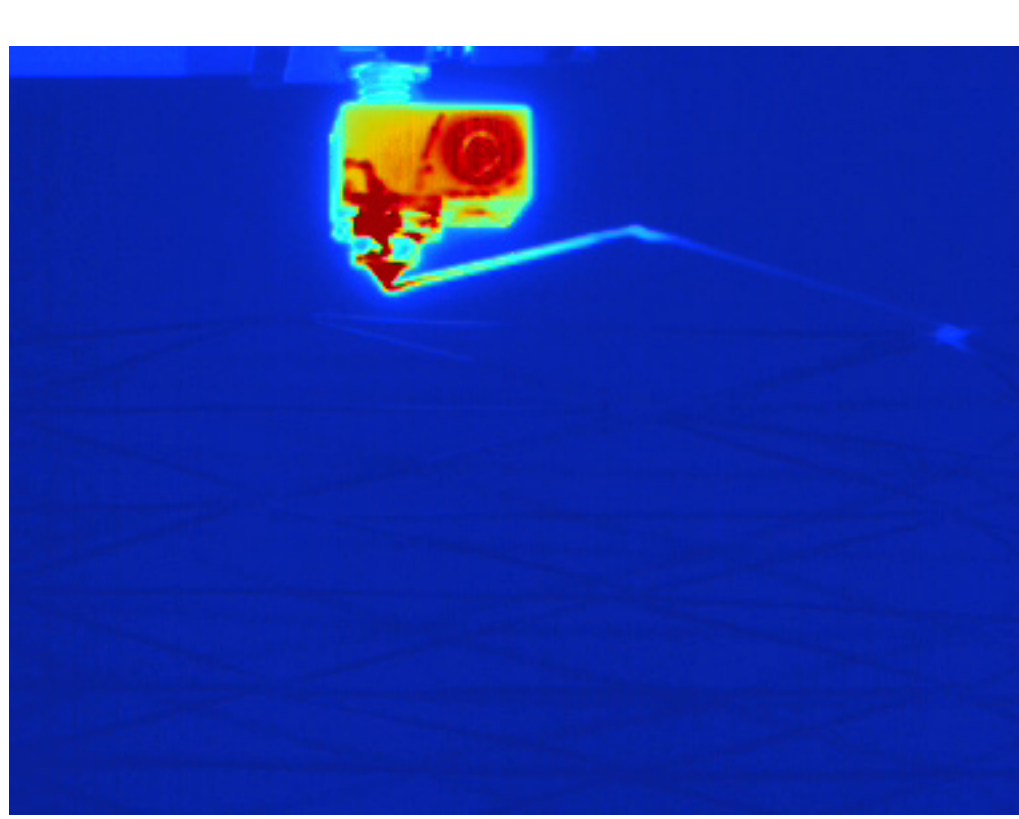
The material behavior of the used thermoplastics (ABS and PLA) is highly dependent on its temperature. Once the heated material is extruded it dissipates heat to the surrounding and cools down and solidifies. The dynamics of this process were observed using a the thermographic camera Therm-App by Opgal Optronic Industries.

For extrusion the thermoplastic is heated in the extuders hot-end, the material softens and can be extruded through a nozzle. Now the material starts to cool down to room temperature. The heat is dissipated trough conduction, convection and radiation. Depending on the shape and the spacial configuration of the extruded material conduction (for compact, layer like structures) and convection (for spares, spacial structures) are the dominant means of heat transfer. However the extruded filament also radiates heat, observing this electro magnetic radiation allows to infer the materials temperature.

For example the peak emission wavelength of an ideal black body at a temperature of 200°C is about 14 µm. At that temperatures the spectrum is already very broad and the body emits energy over a wide frequency range. To observe this emissions usually a sensor, sensitive mainly to to light in the infrared atmospheric window between 8-14µm is used. This window is chosen, because air is transparent for light with this wavelengths and therefor has only very limited influence on the measurement.

As for electro magnetic radiation of other wavelength, it is possible to build imaging optics for the Long Wavelength Infrared' (LWIR). The two most common materials for refractive LWIR optics are Zinc Selenide (which is also quite transparent for visible light) and Germanium (which is only transparent from about 2 to 22 µm).

The used thermographic camera is sensitive to light of 7.5-15µm. It features a Germanium objective with a focal length of 19 mm. With the 9.6x7.2 mm² sensor this corresponds to a field of view of 19° x 14°. The sensor is an uncooled amorphous silicon microbolometer LWIR sensor with a resolution of 384x288 pixel (25 µm pixel pitch). It shows a noise equivalent temperature difference of less than 0.07 K. While it is only calibrated for a temperature range of 5 to 90°C a test showed good accuracy up to 400°C.



Central Server

A characteristic of this project is the need to connect several quite different systems. For its wide availability and ease of use Ethernet is used to establish the needed communication links wherever possible. It is not only used to facilitate communication between hardware components but also to do remote procedure calls and exchange data between different programs. While direct point-to-point links between two partners is used in some cases, the most important data exchange between the robot and Rhino/Grasshopper is managed by a Python script, the Central Control Script (CCS).

The main task handled by this script is to buffer and forward messages between Rhino/Grasshopper, the robot and the ECM. This requires the script to run in the background all the time. It is possible to use C# to create a multi thread Grasshopper component and use background thread handling the communication — similar to the listener used in the Picoflexx component. With some limitations this is even possible by utilizing the C# Grasshopper scripting component, thereby eliminating the need to build and reload the custom component after each modification.

IronPython inside of Rhino/Grasshopper can not be used to realize the background communication thread as the interpreter runs in the same thread that also provides, among other functionality, the Rhino user interface. So as long the interpreter is running, Rhino is blocked. An external Python script, run by an external Python interpreter however, can install a background thread without affecting Rhino.

Running an external Python script is an attractive option, as this allows to use Python 3 and an up-to-date CPython interpreter, which in turn give access to the wast collection of scientific libraries for Python (may of which not available for IronPython 2.7), most importantly NumPy, SciPy and PyTorch.[1]

[1] Libary providing support for large multi-dimensional array, collections and matrices and high-level mathematical functions to operate on these arrays. https://www.numpy.org
Scientific computing library for Python provideing modules for linear algebra, optimization, interpolation and more. https://www.scipy.org
Machine learning library for Python. http://www.pytorch.org

The version of the Central Control Script used for most of the experiments does not make use of any of these libraries but is merely a message relay, but future work on machine learning or machine vision and optical tracking could be integrated nicely.

The CCS used opens a UDP socket and waits for incoming XML messages. These messages are generated in Rhino/Grasshopper and sent by use of the gHowl component. They contain, usually multiple, commands for the robot. The script parses the XML data. If a certain flag in the data is set it processes the robot commands and pushes them into a first-in-first-out (FIFO) buffer.

A second thread opens a TCP/IP socket waiting for the robot to connect to it using the functionality provided by KRL XML package. Once connected this thread waits for an XML message from the robot, when such a message is received it is parsed and processed and as an answer the next command from the FIFO is sent to the robot. Status information, for example the current robot position, is sent to Rhino/Grasshopper via UDP and received by gHowl.

It is worth noting that establishing network communication via TCP/IP (UDP is very similar) requires just a few lines of code (proper error handling and housekeeping makes things more complicated, but is omitted here):

```
tcpSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

tcpSock.bind((tcpServer,tcpServerPort))

tcpSock.listen()

tcpConnection, tcpClientAddress = tcpSock.accept()   # wait for con-
nection
```

Now data can be received:

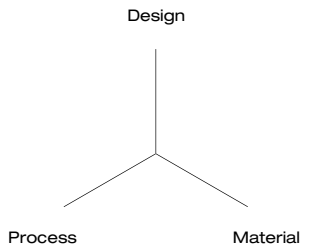
```
while True:

        rawData=tcpConnection.recv(tcpBufferSize)

        print("received raw data: %s " % rawData.decode("ascii"))
```

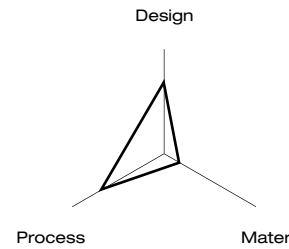
Case Studies

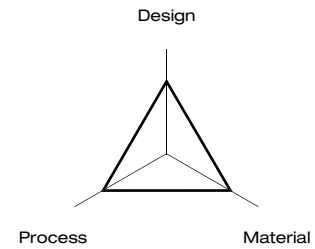
Following chapter provides a detailed description on the methodology, execution and results from the practical experiments, structured in separate case studies.

By means of physical experiments and through variation of predefined parametric values a self-organising behaviour has been deliberately triggered. To achieve “higher order” however a material feedback needs to be introduced. A link between digital design and physical object is established so that a real time control over the fabrication and necessary changes resulting from material behaviour are incorporated to the design during fabrication.

In the series of case studies the hierarchical organization of the three main parameters — design, process, material — has been explored. Material feedback proved crucial not only for correcting tool-paths and thus compensating for deviations but it enables a more general approach to materiality and formation processes. The developed design and fabrication system allows for material feedback to flow into the design process and participate in the genesis of form equally as design idea and procedure.





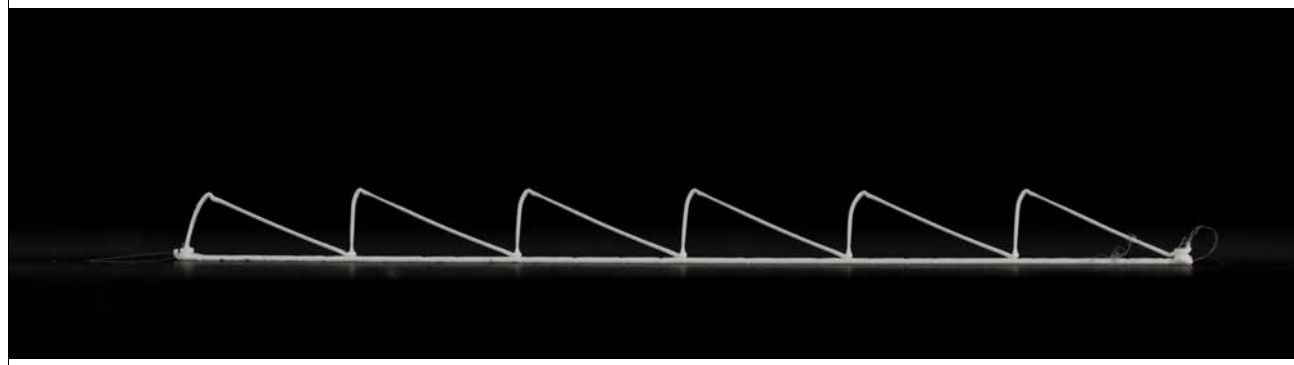


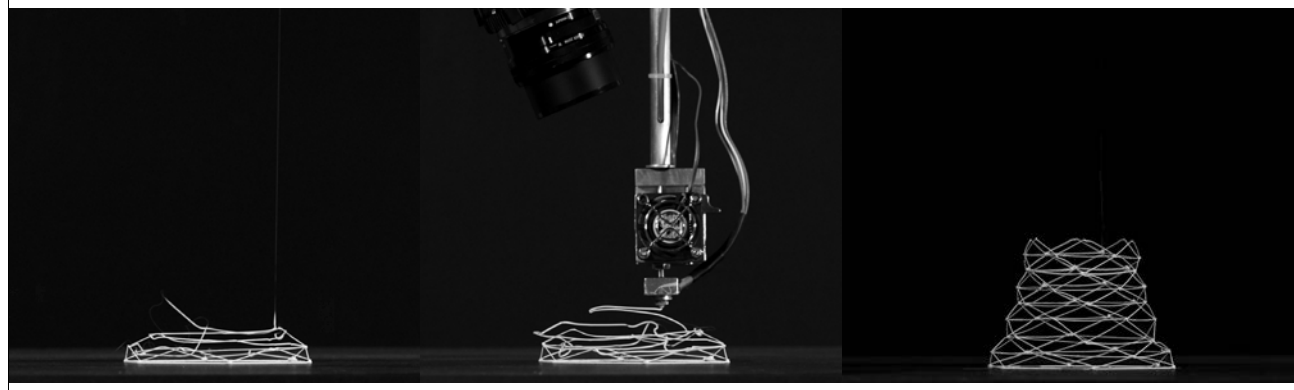
Case Study 1_Simply Complex Line

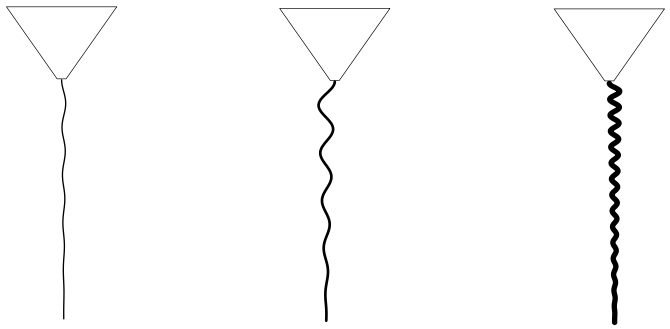
Case Study 2_Higher Order

1 Straight Forward
2 Feedforward
3 Feedback

Case Study 3_Spontaneous (Dis)Order

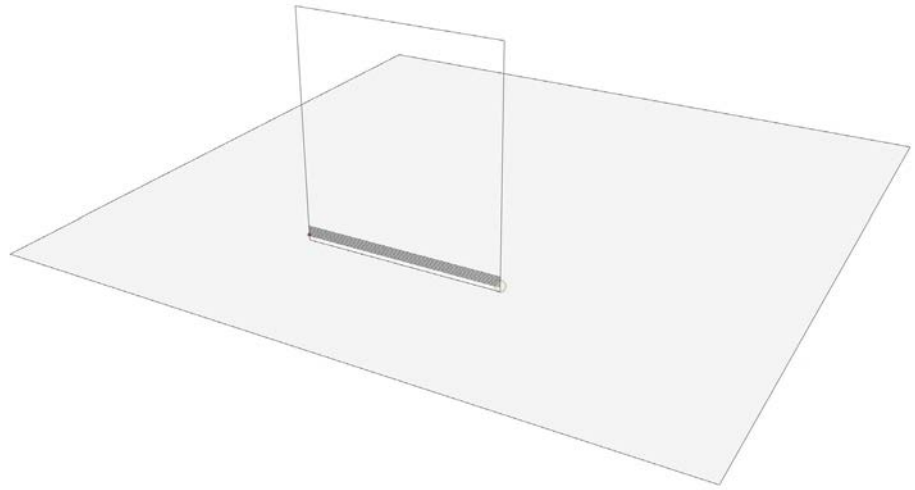


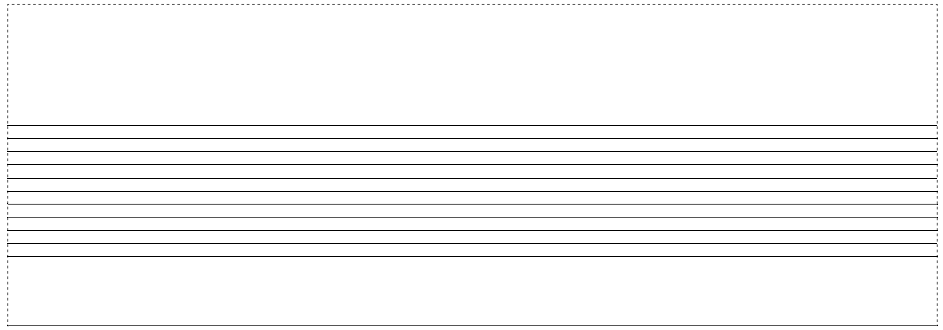




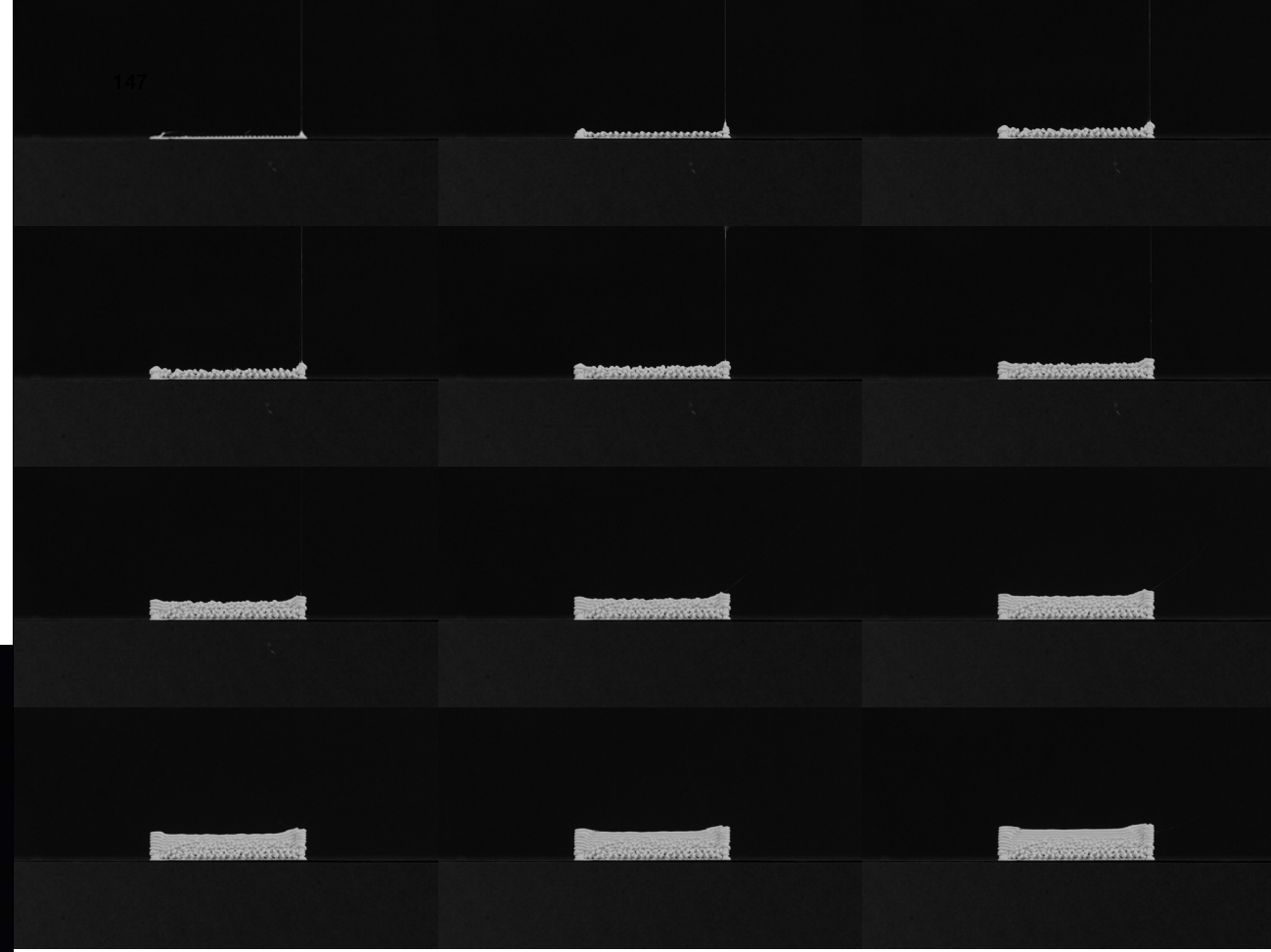
Simply Complex Line

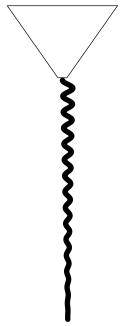
Following simple predefined tool-paths and by changing the speed and temperature of extrusion, the geometrical variation of final outcome has been explored.

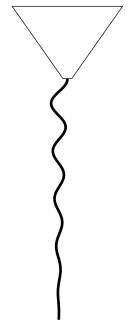


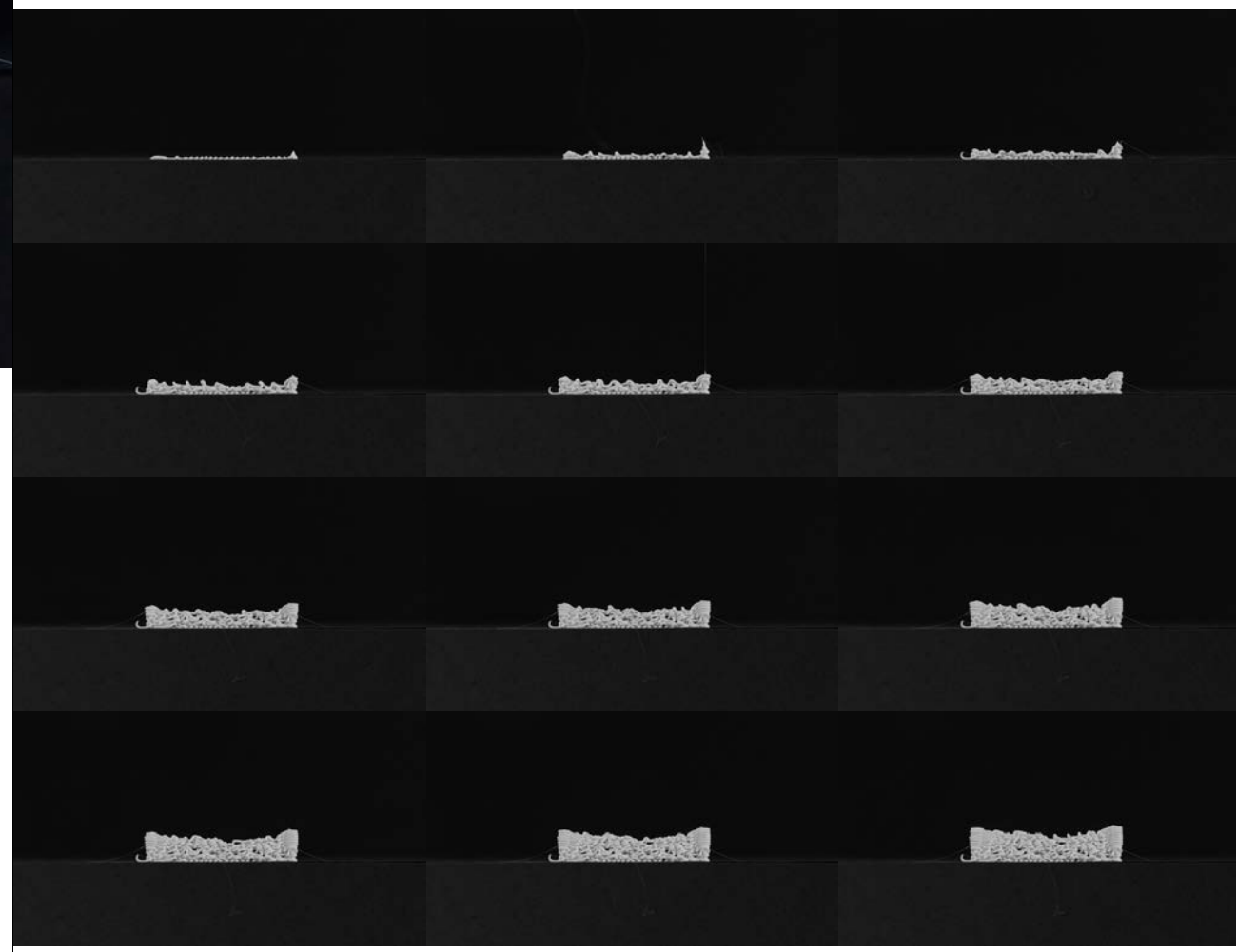


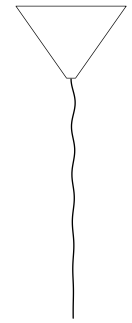
Robotic tool-paths

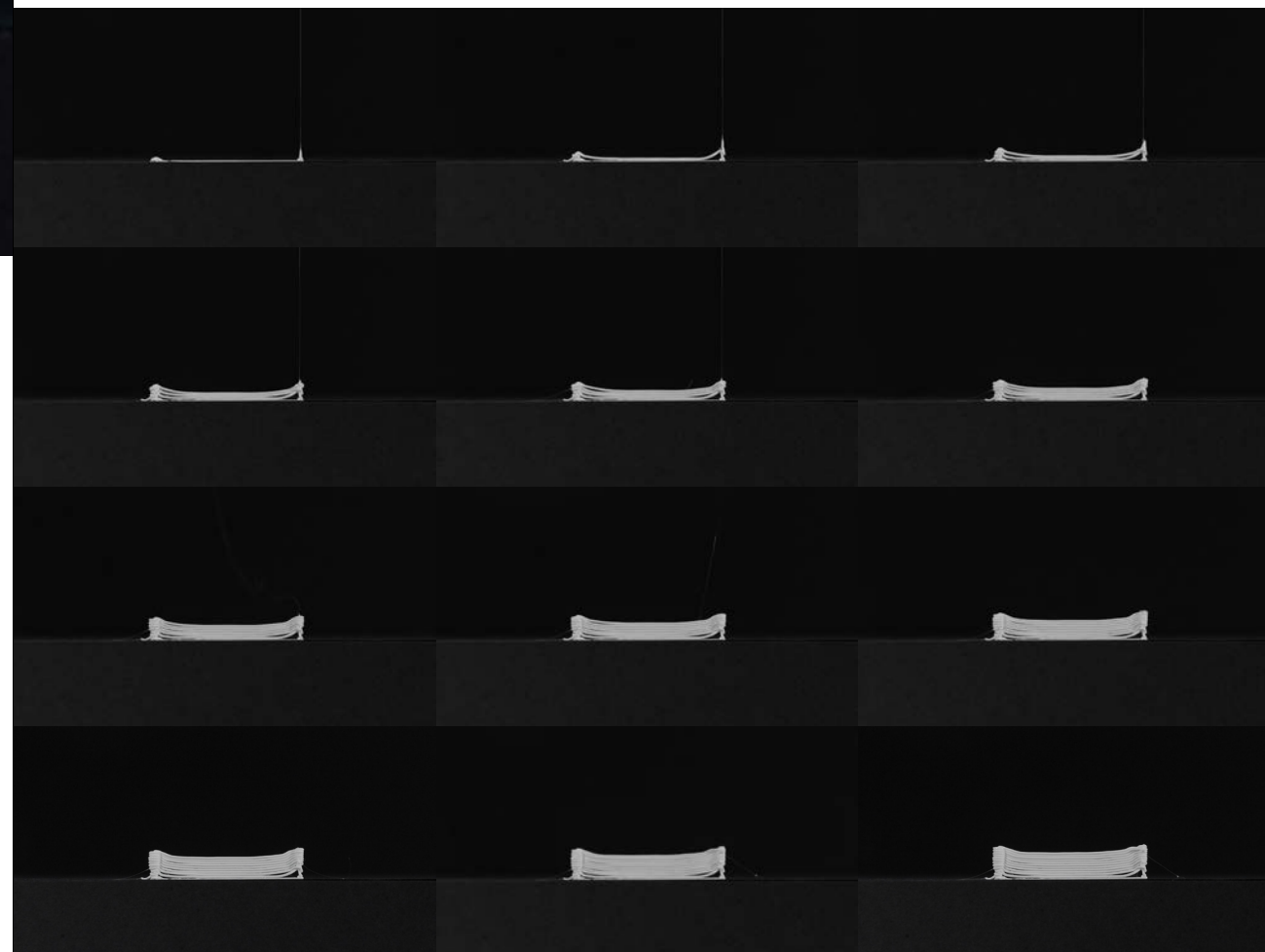


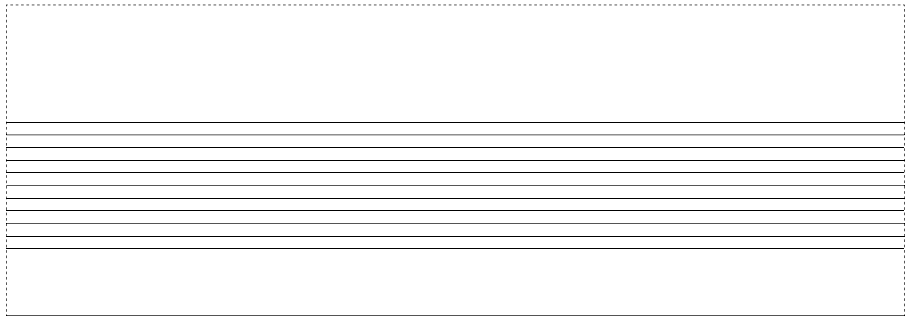


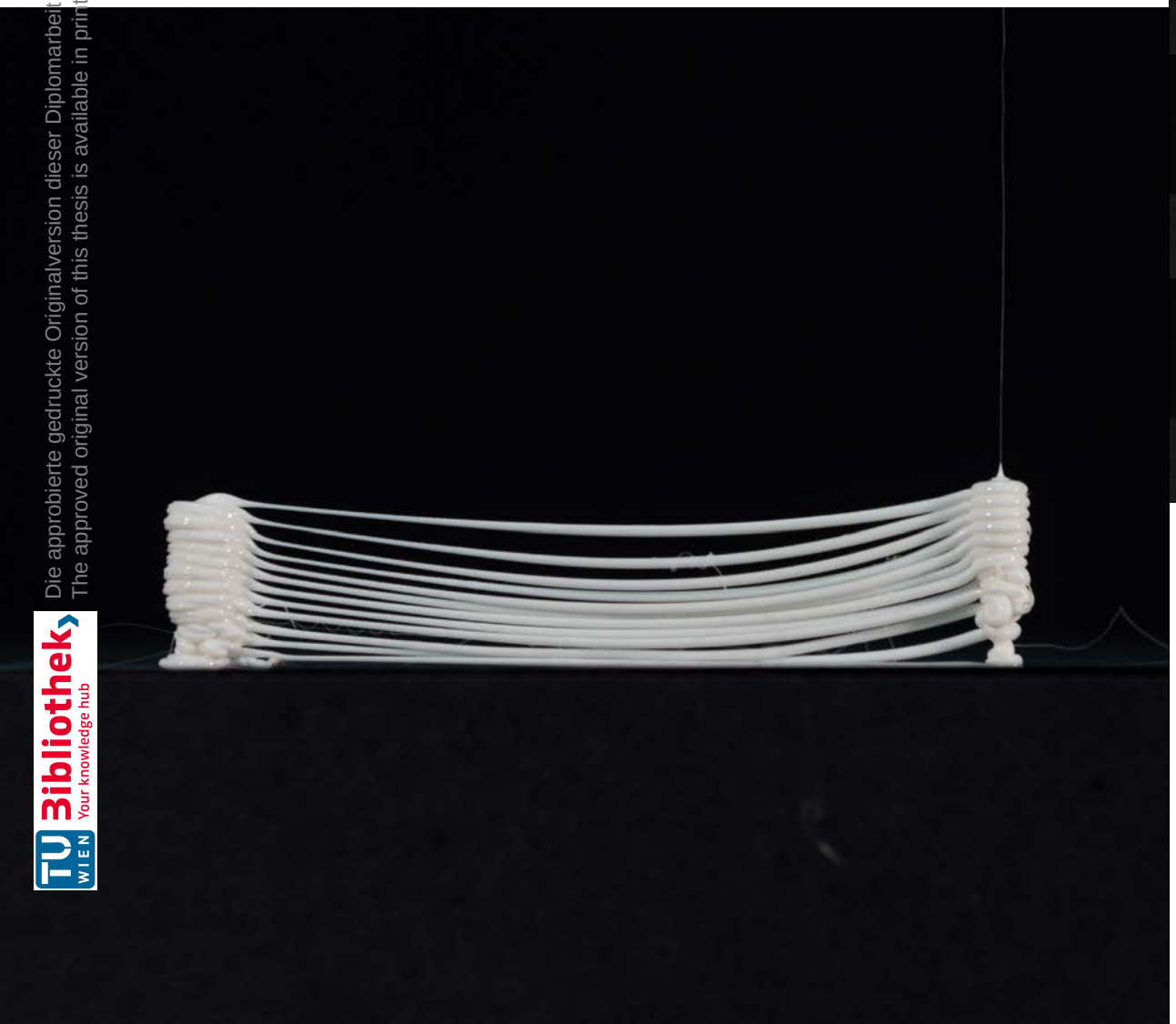


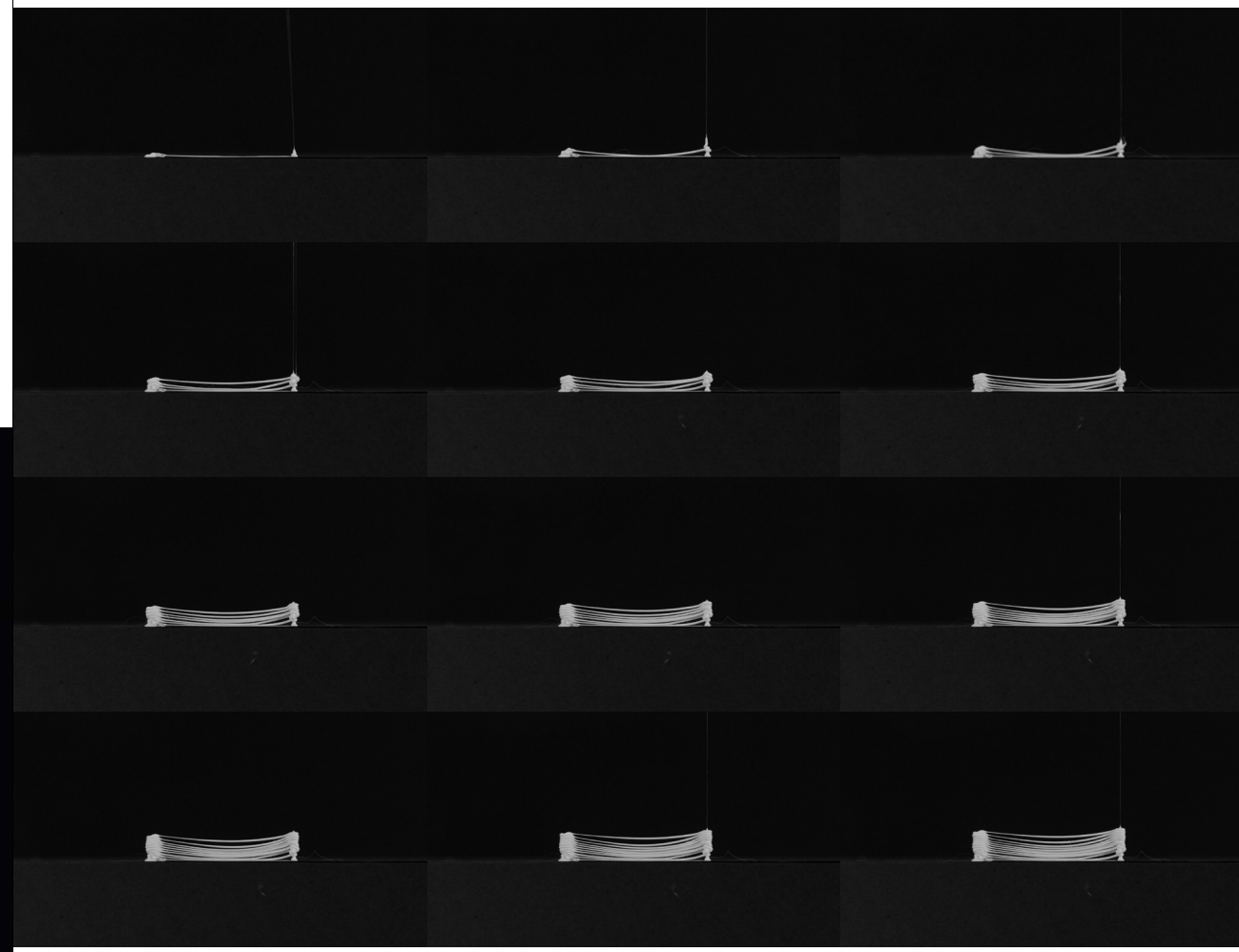


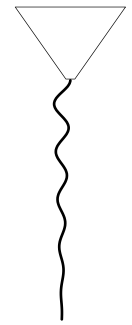


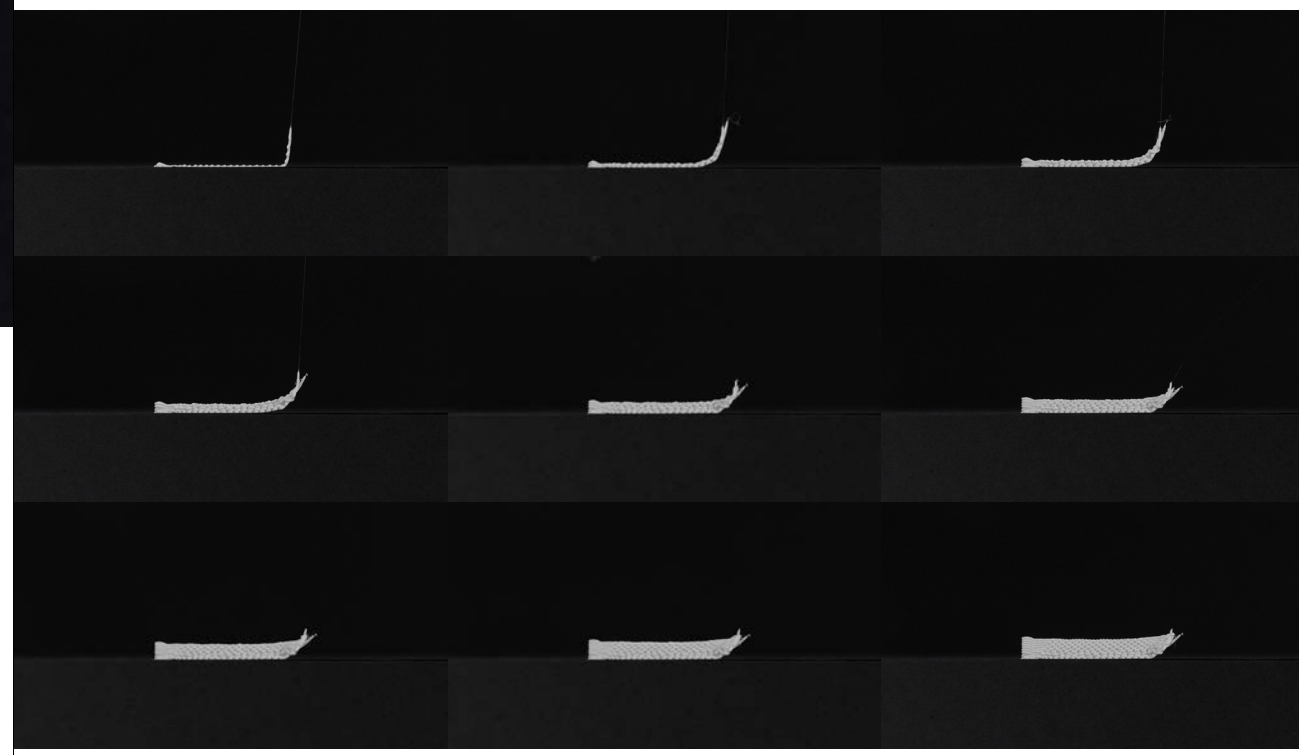


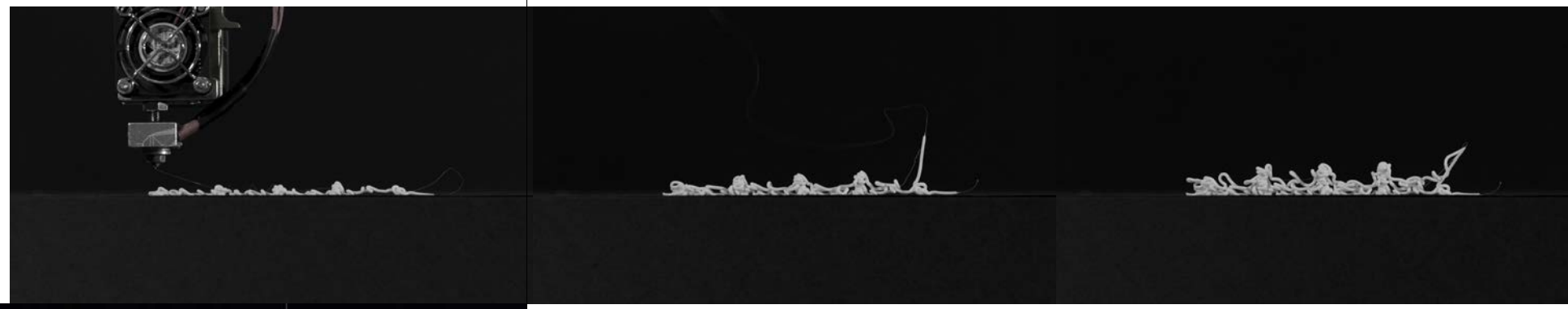


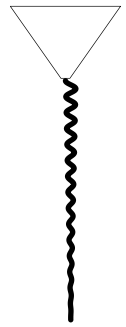


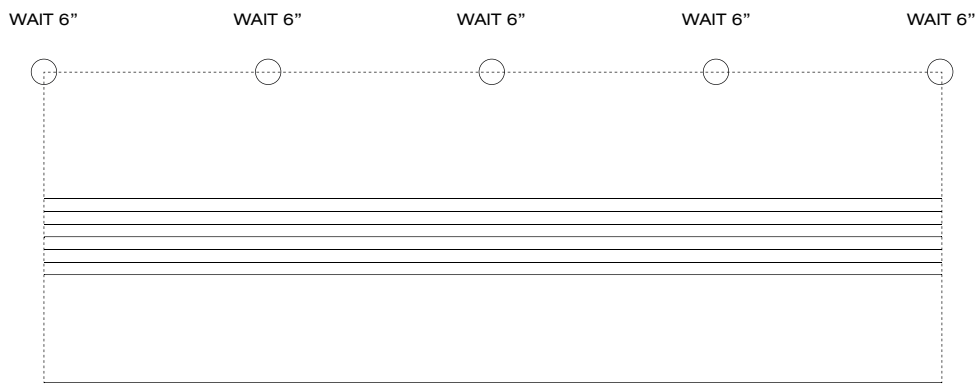


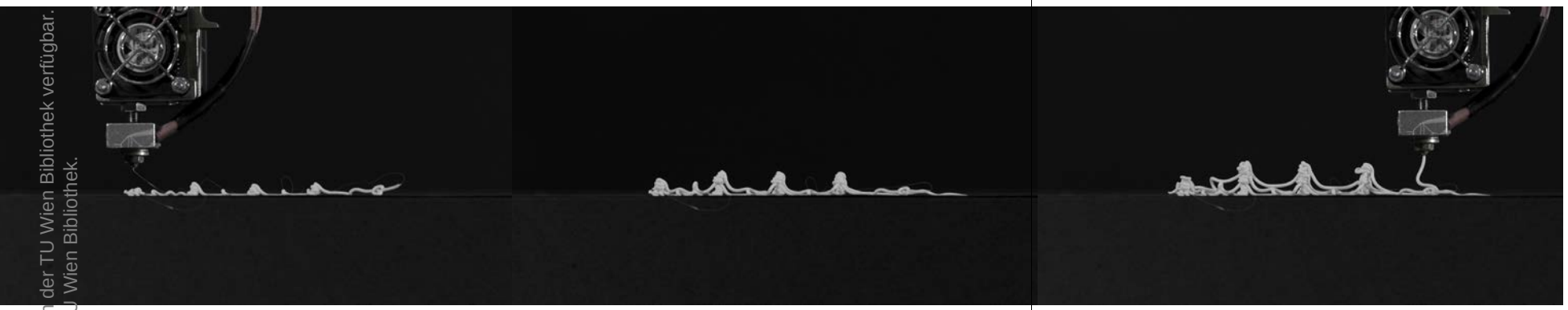


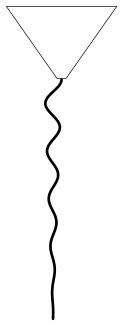


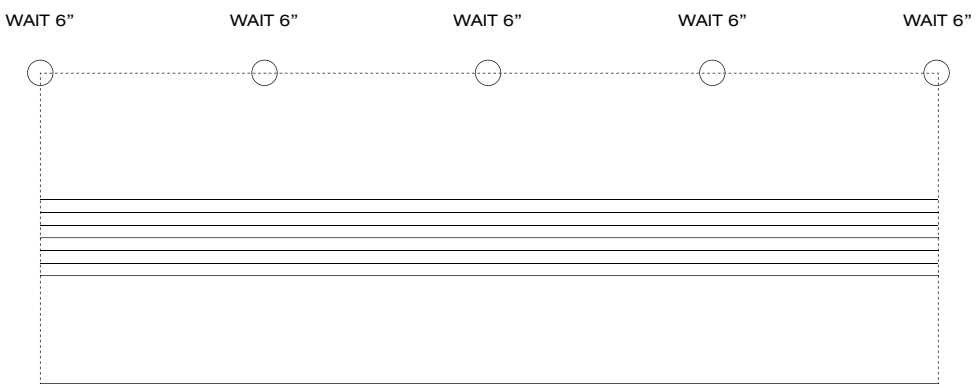


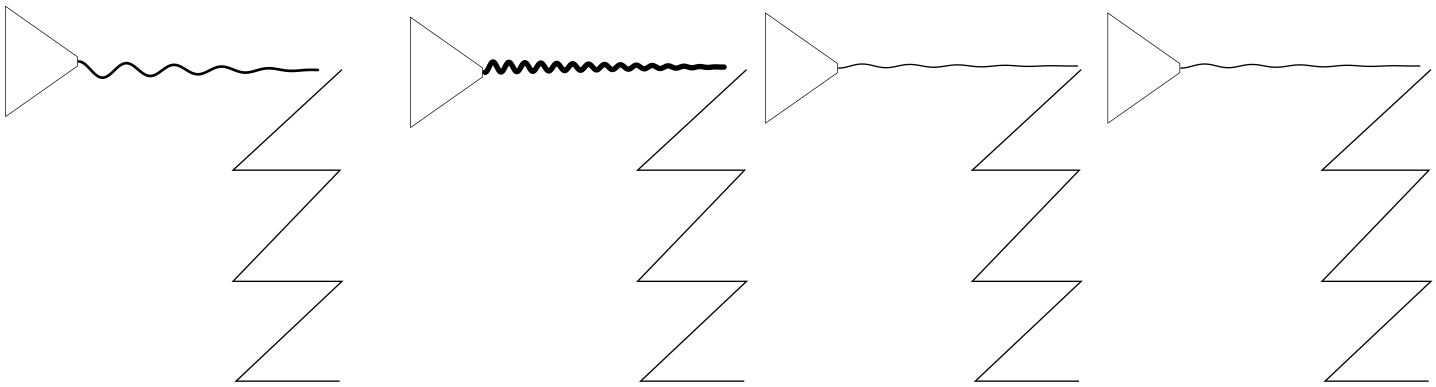


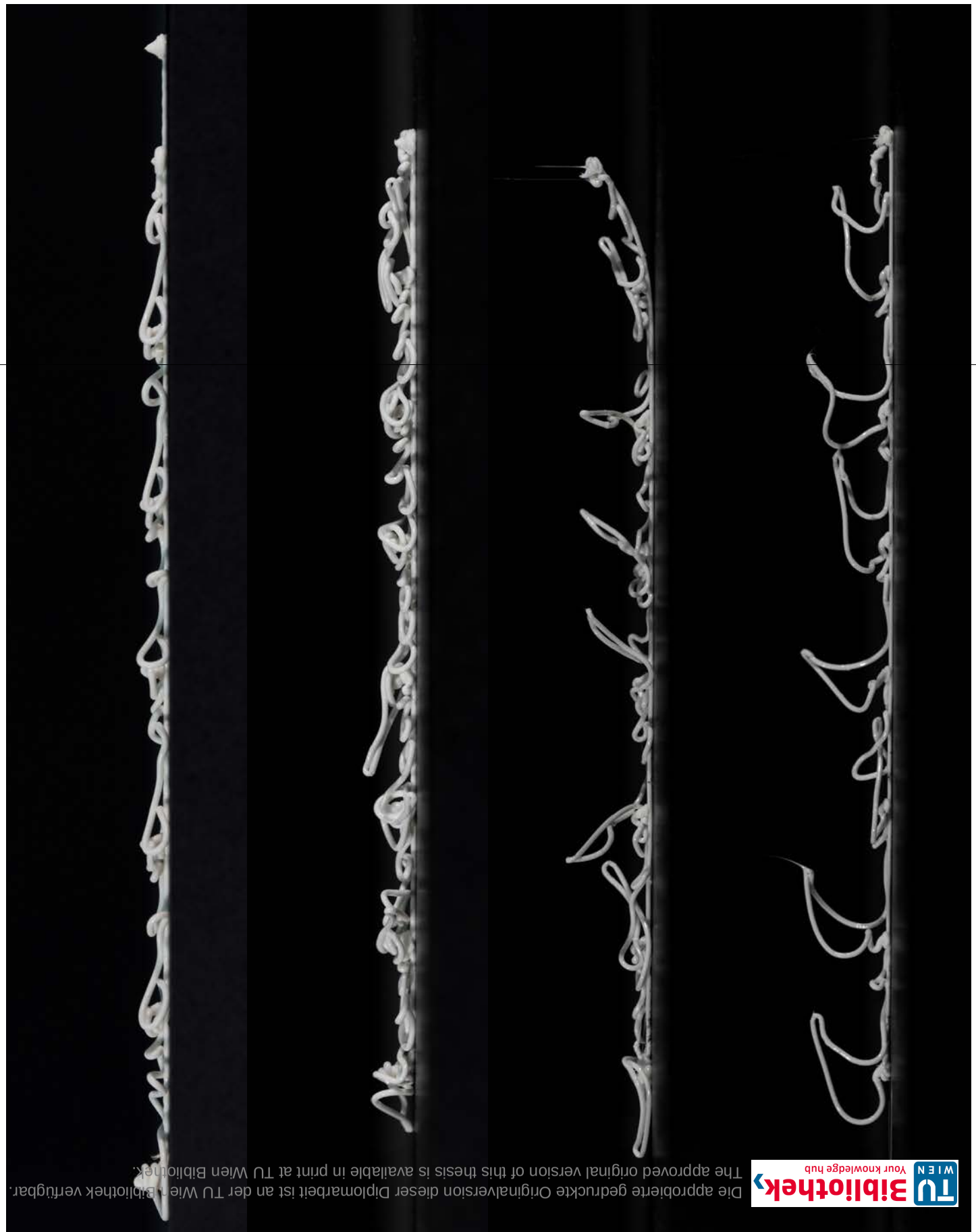


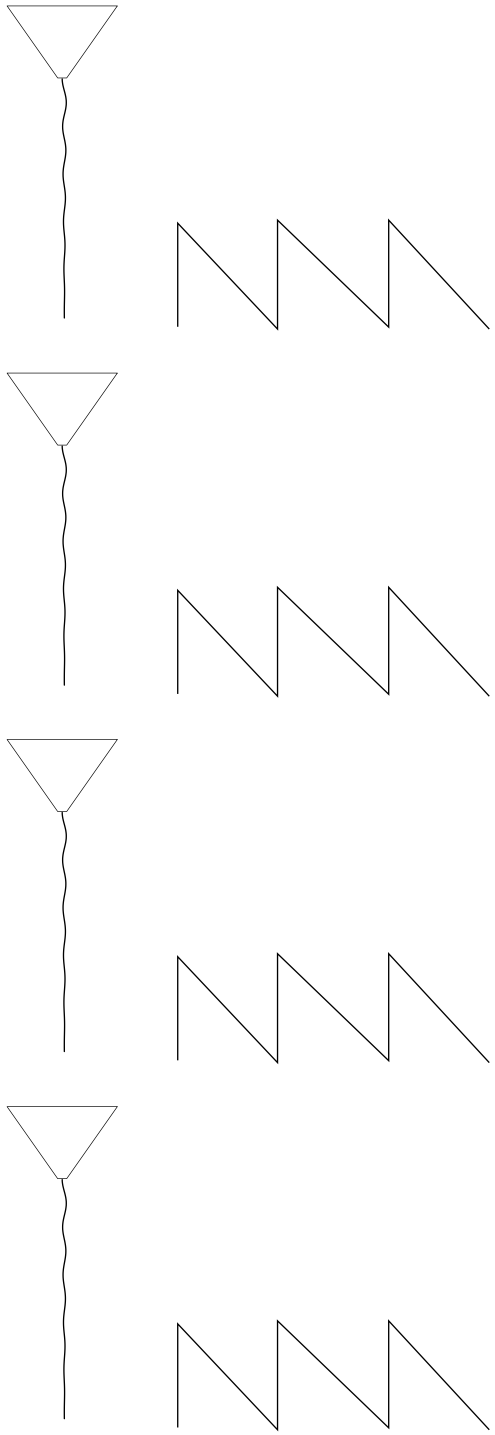


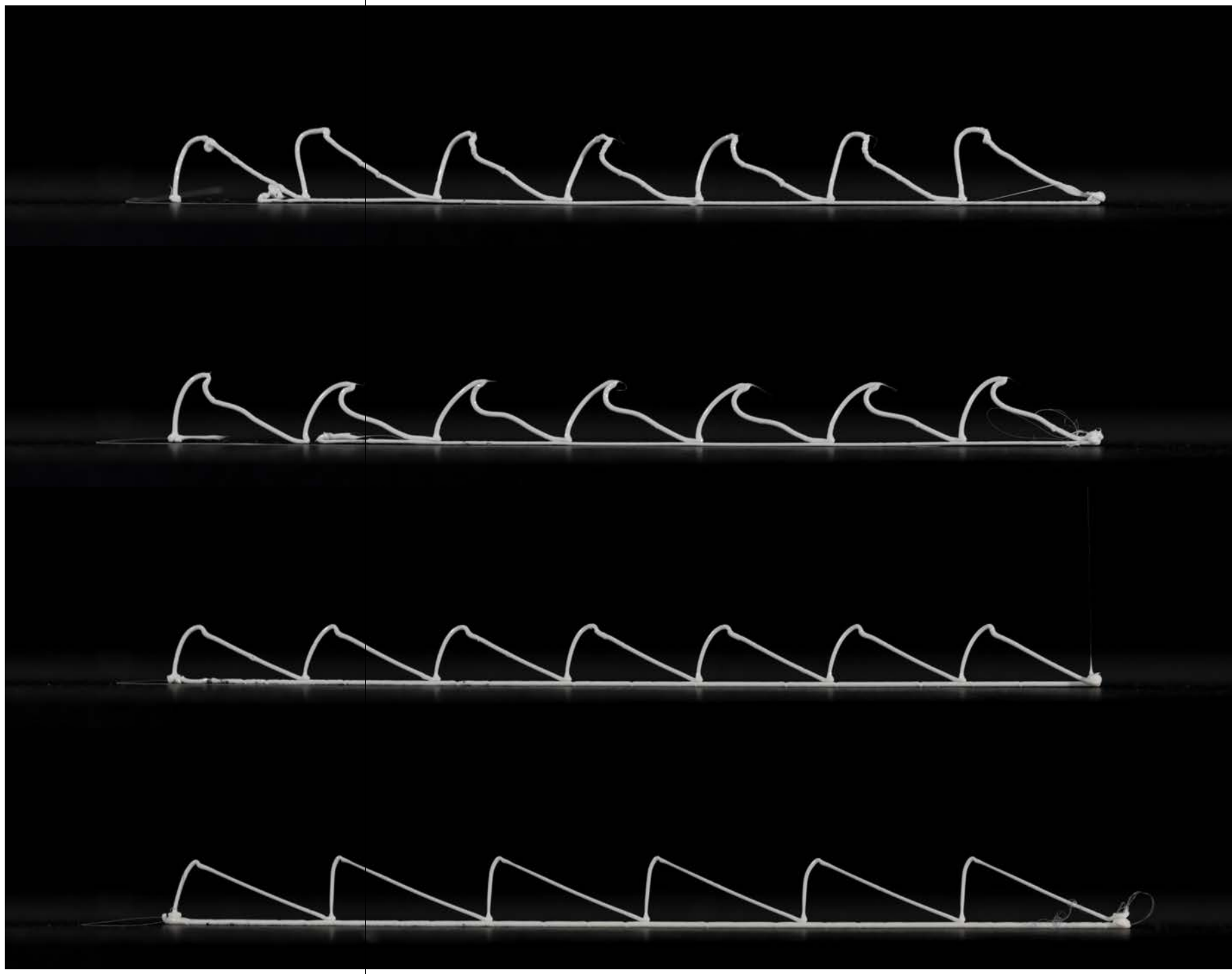


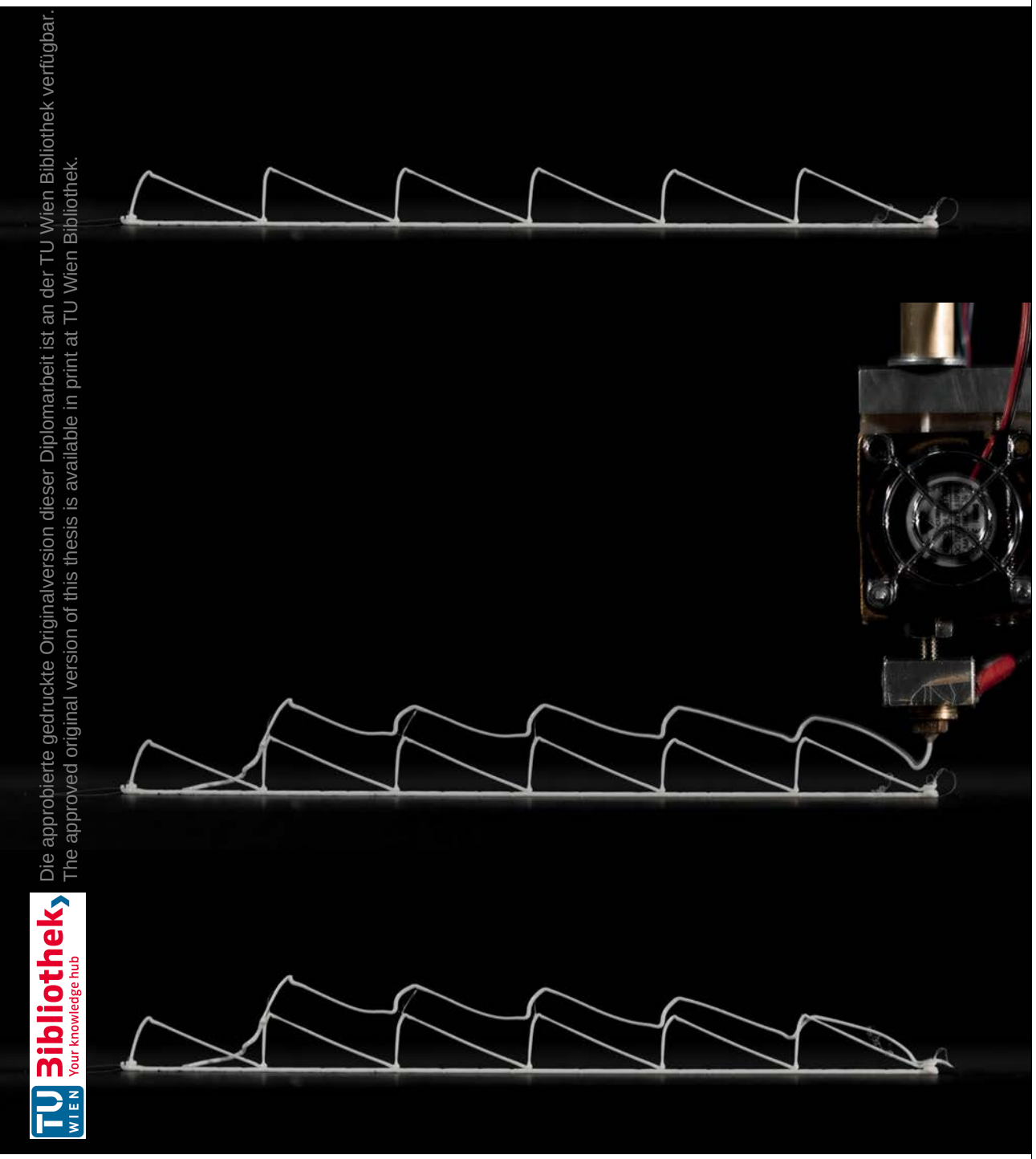


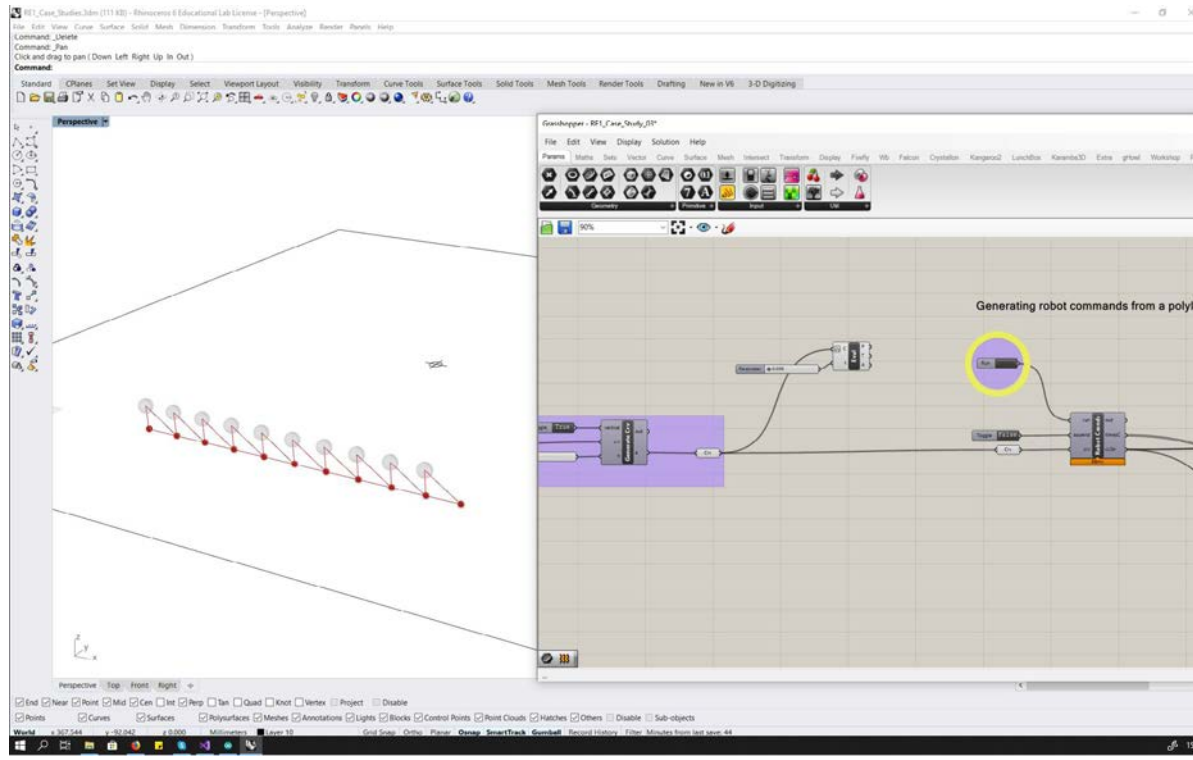












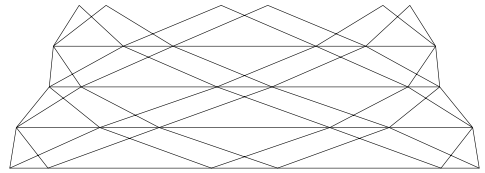
Higher Order

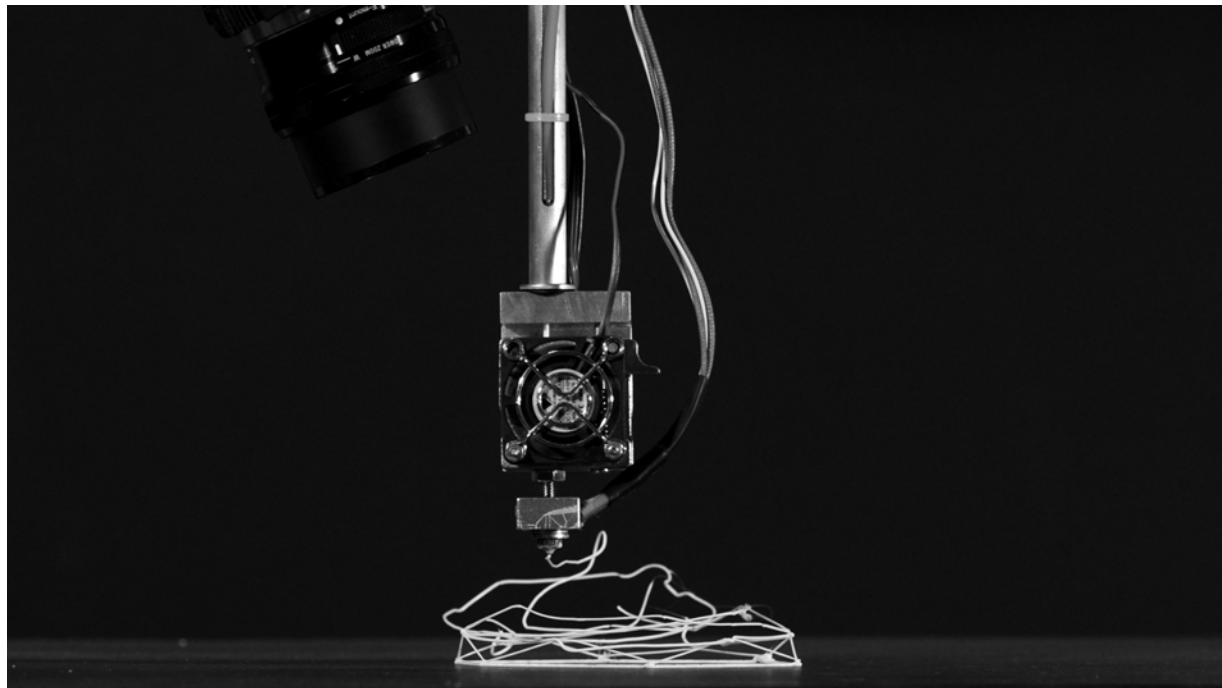
It consists of three different approaches of applying correction and feedback experiments.

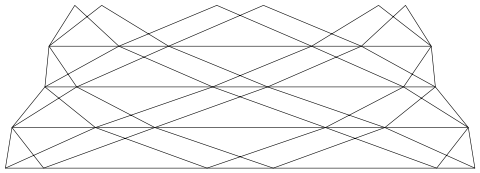
Straight Forward
The robot tool-paths are predefined and executed without been adjusted to accuring deviations while fabrication.

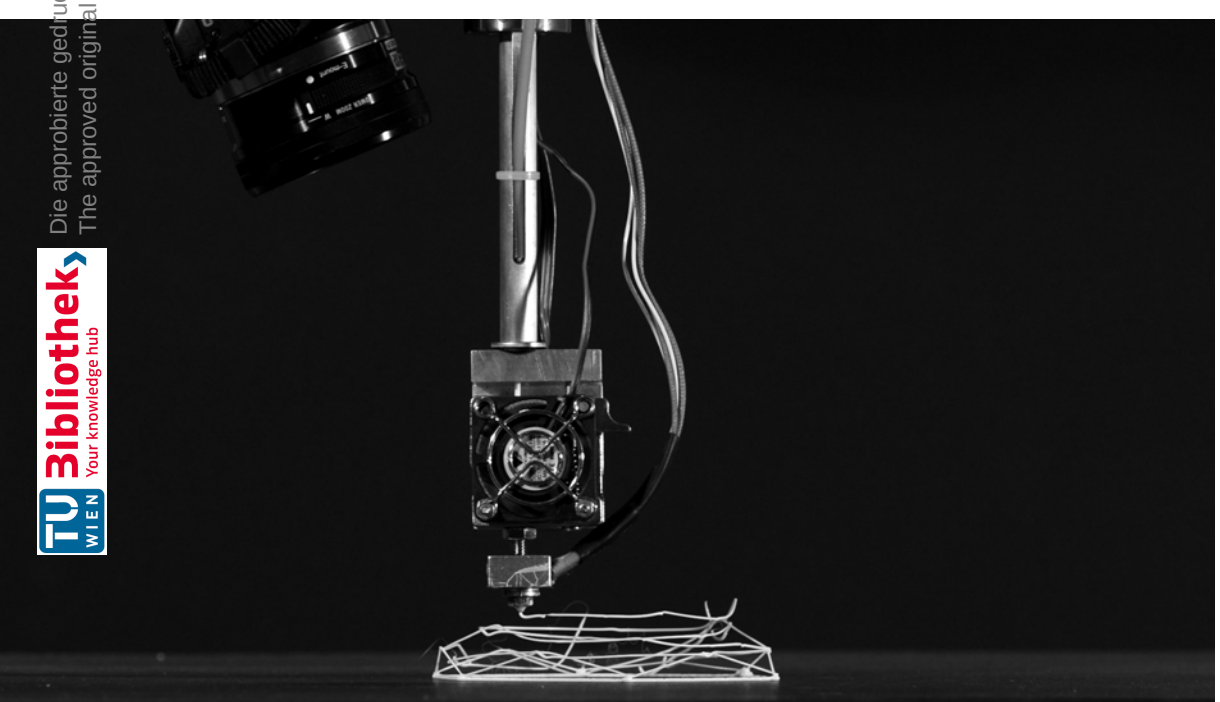
Feedforward
The robot tool-paths are adjusted by adding an offset to each “layer” in the computational model.

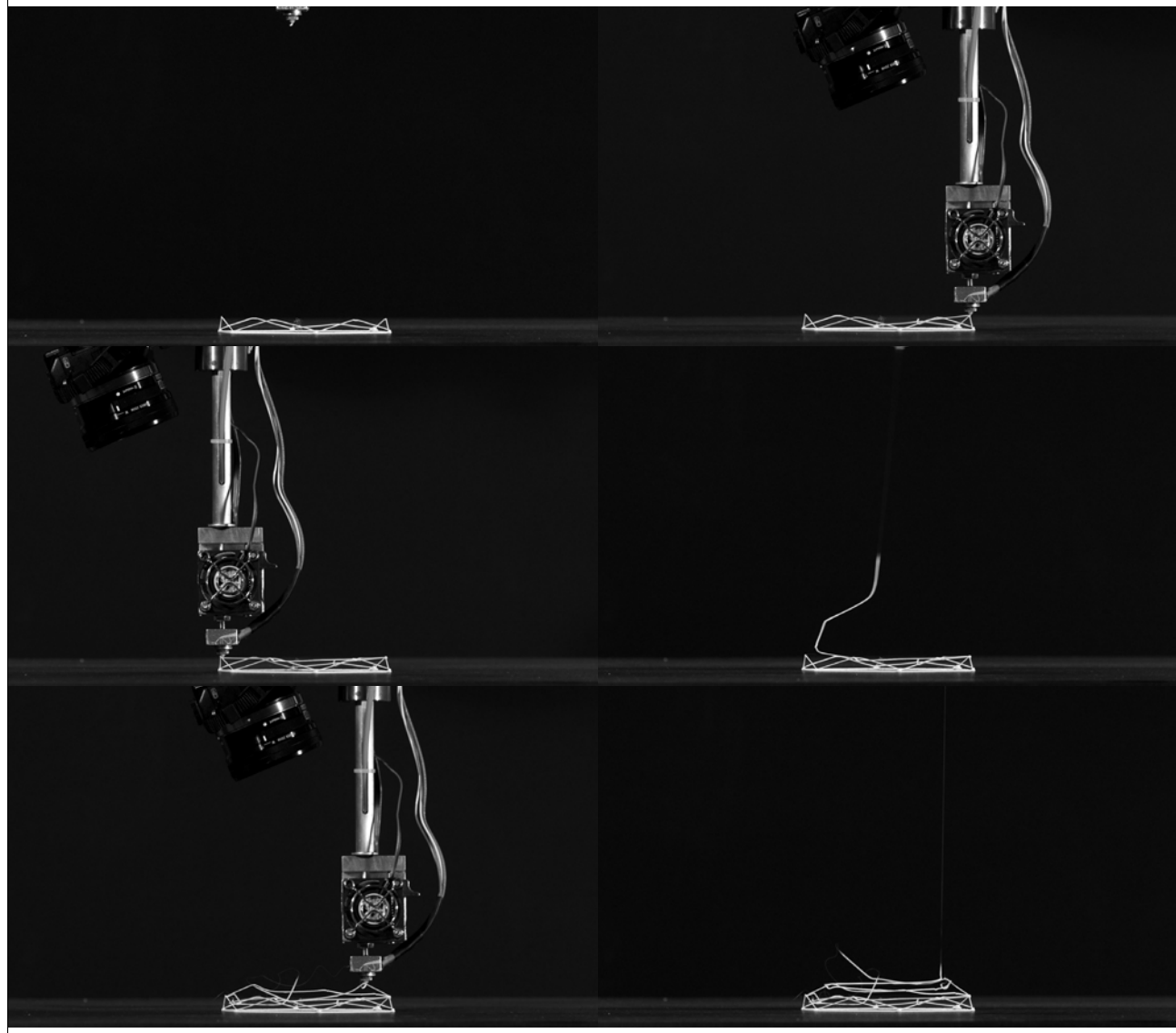
Feedback
Feedback has been introduced for better positioning of the extruded material.

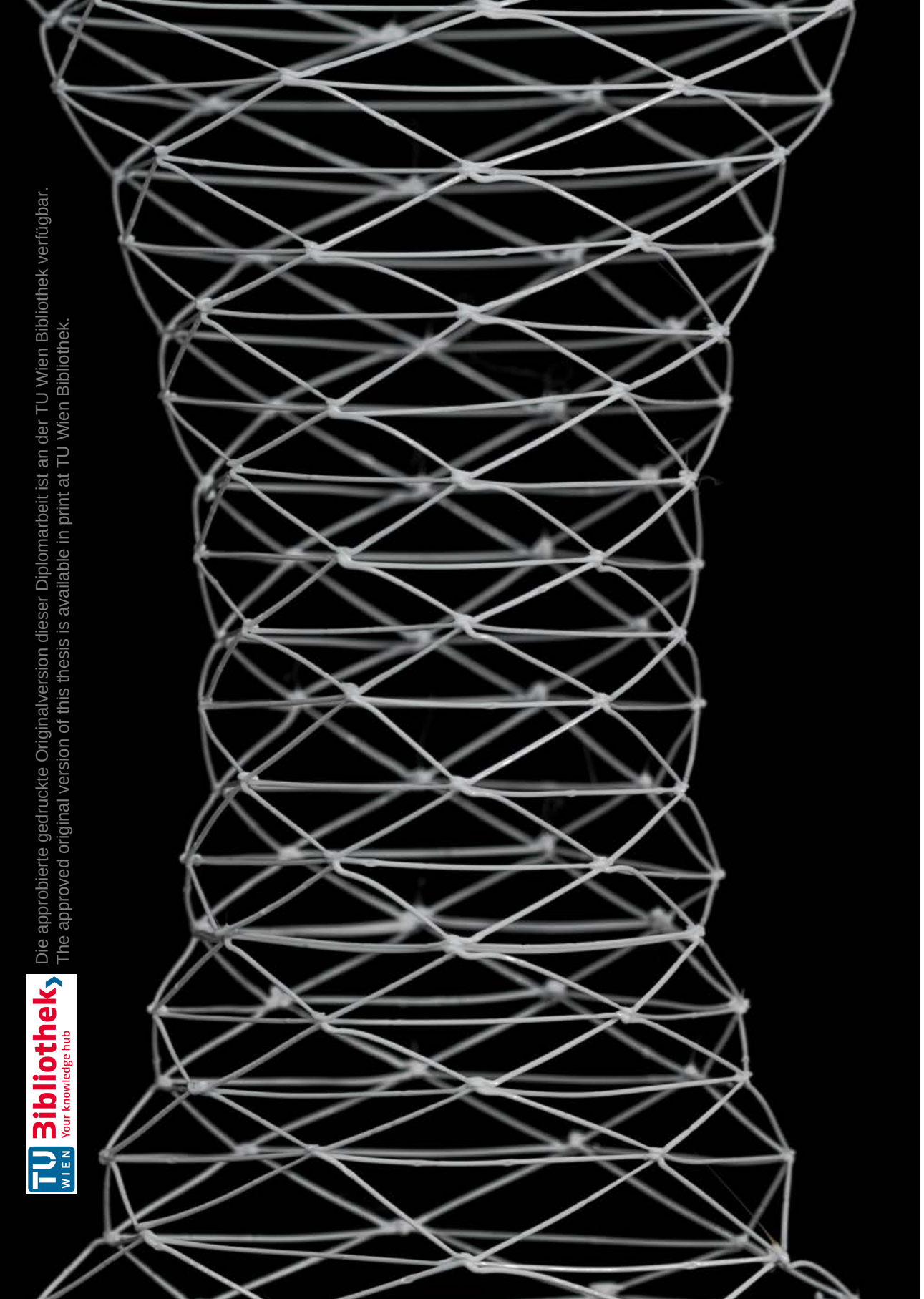


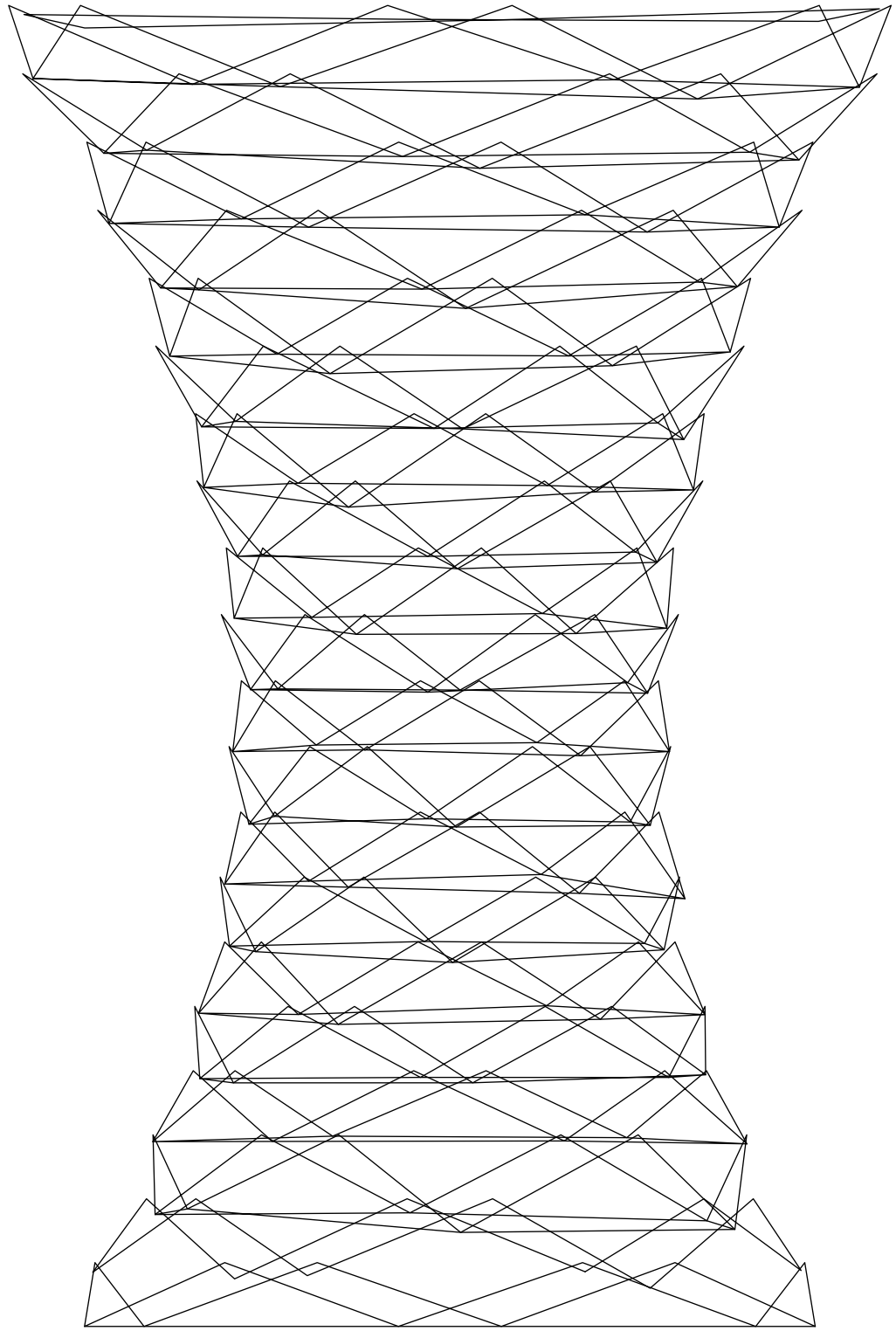


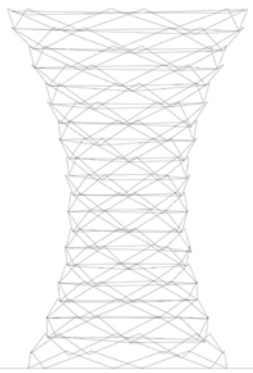


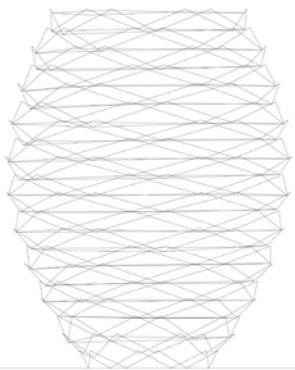


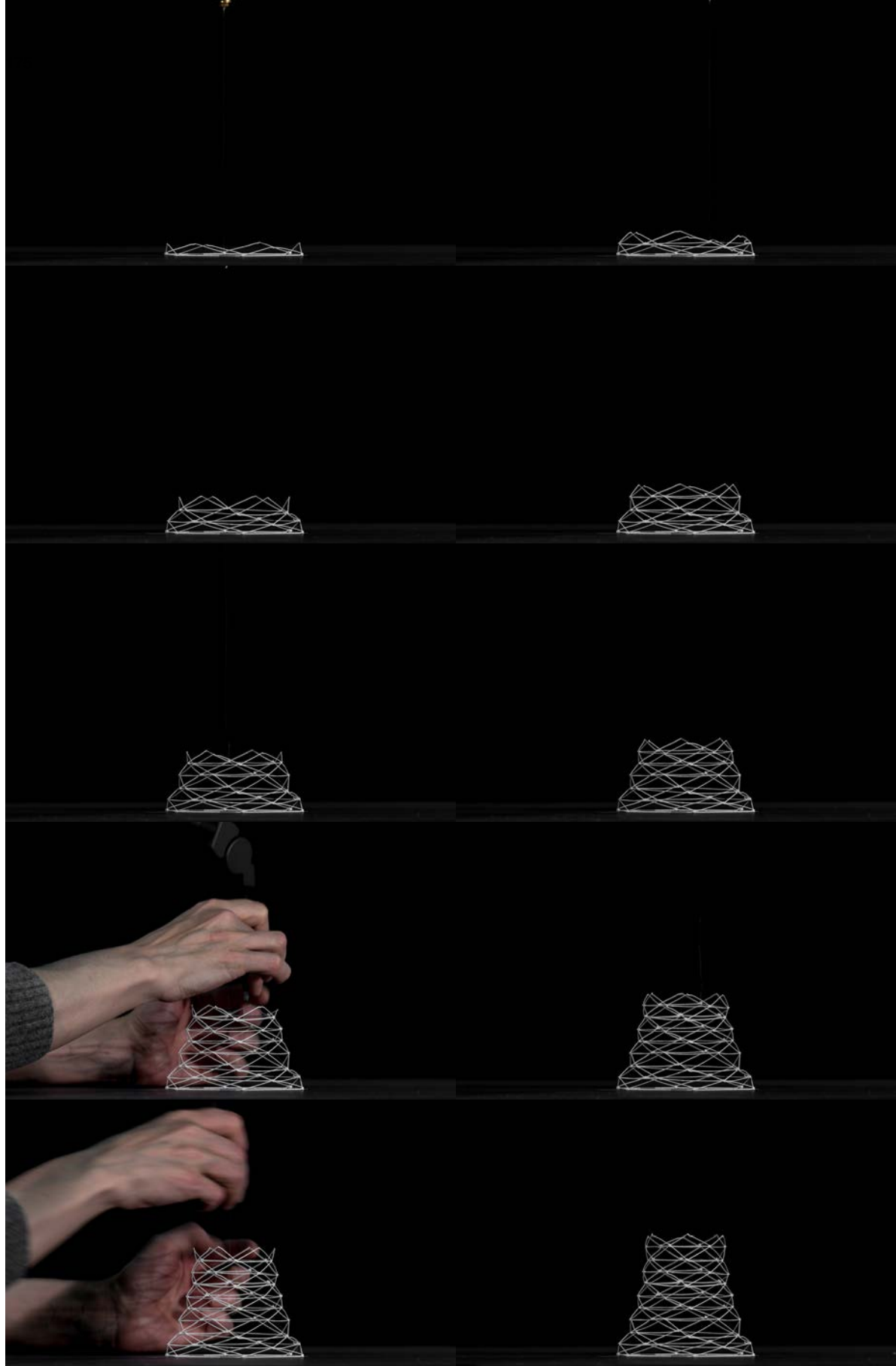


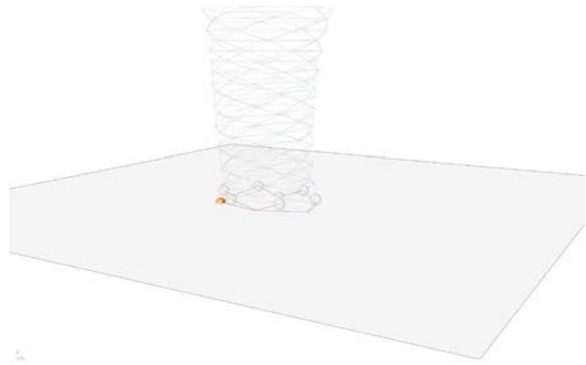


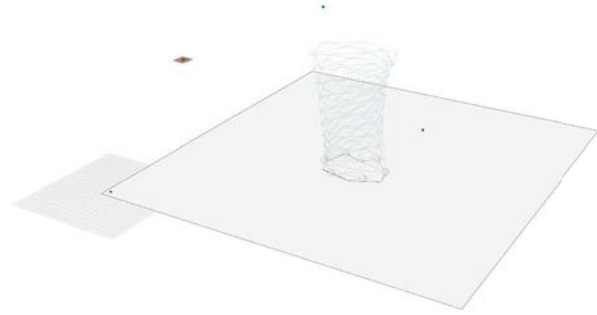


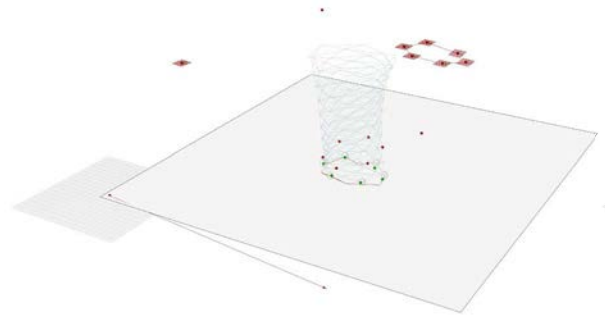


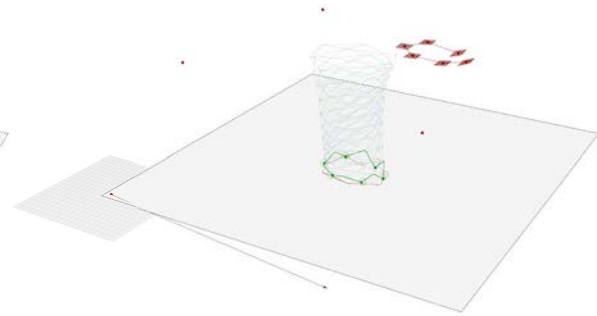


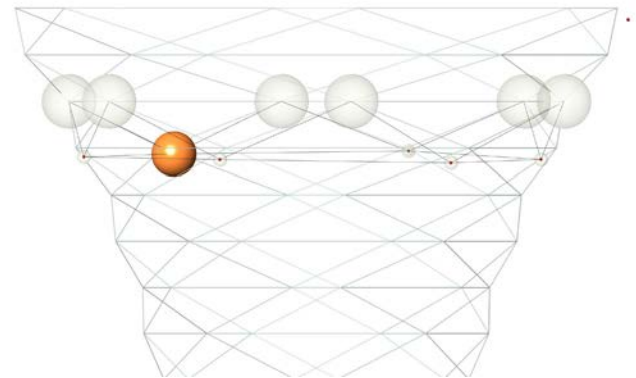




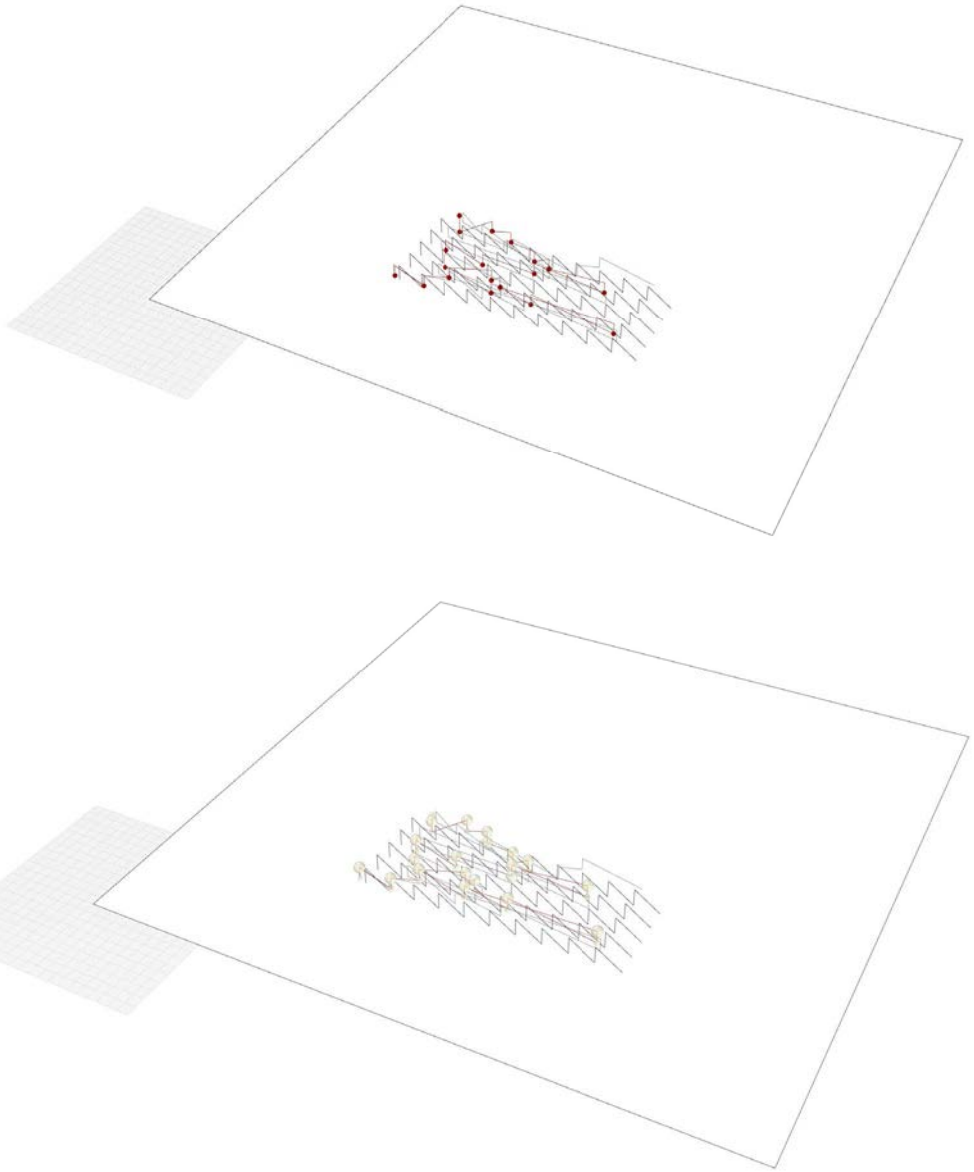


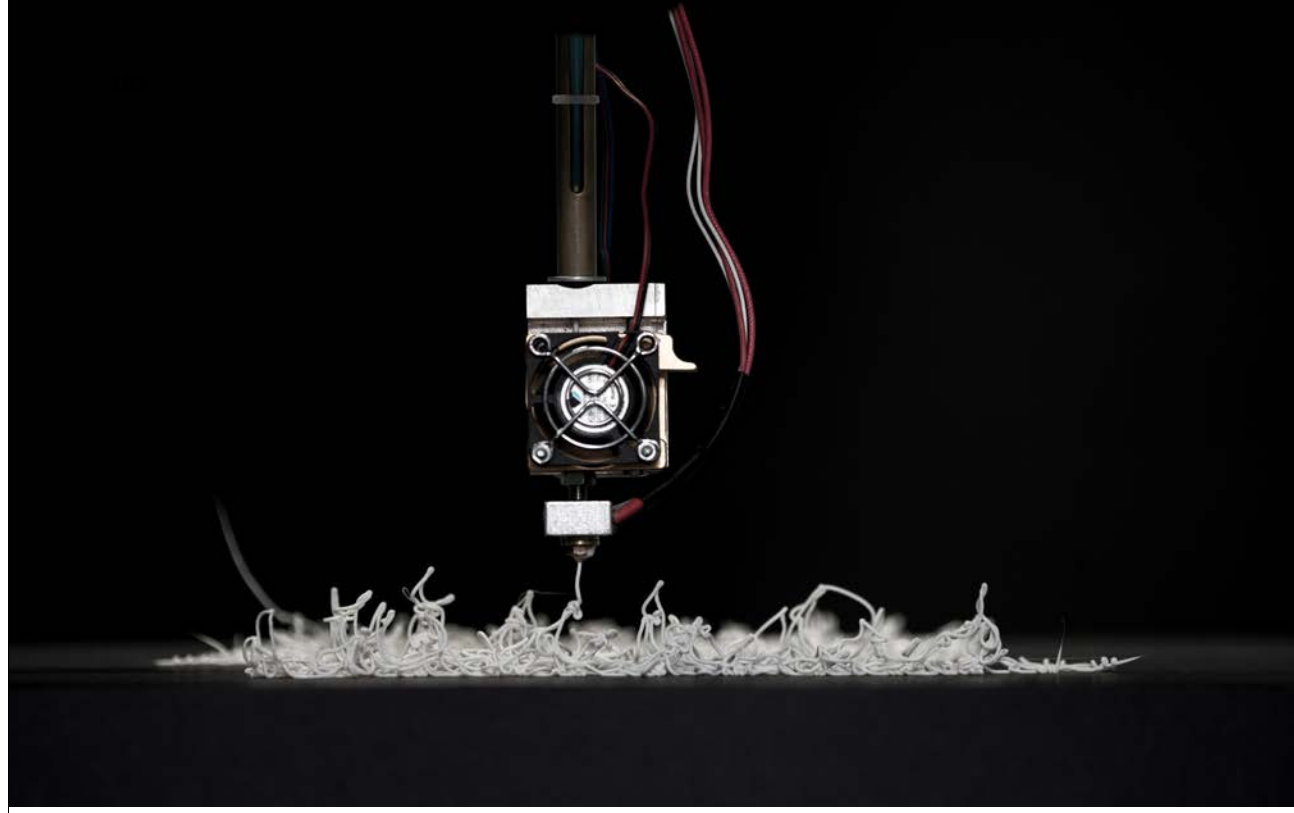


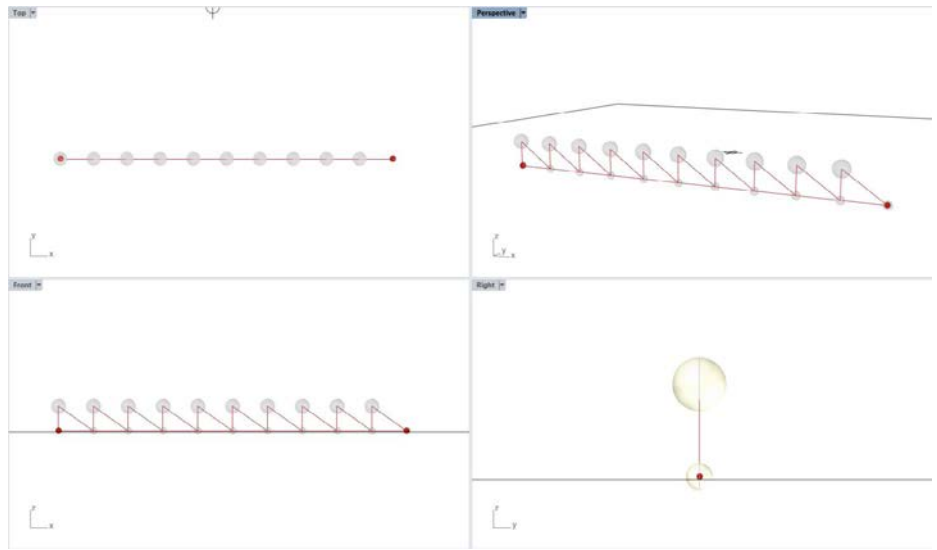


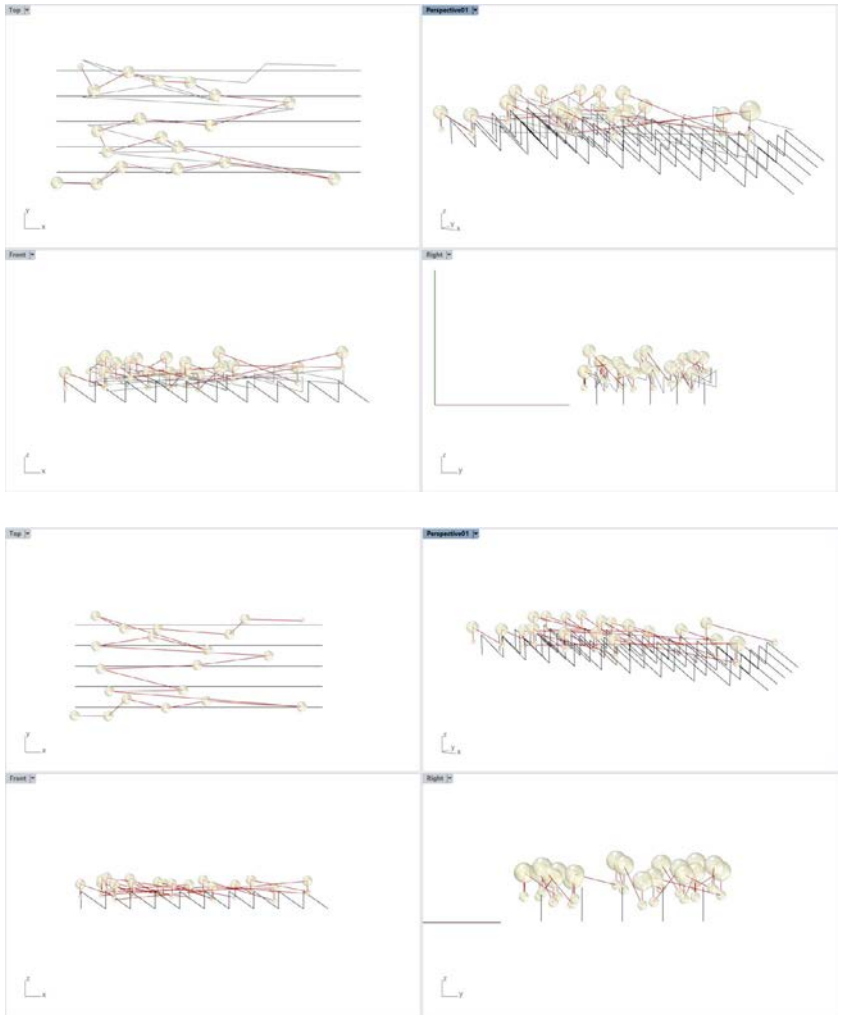


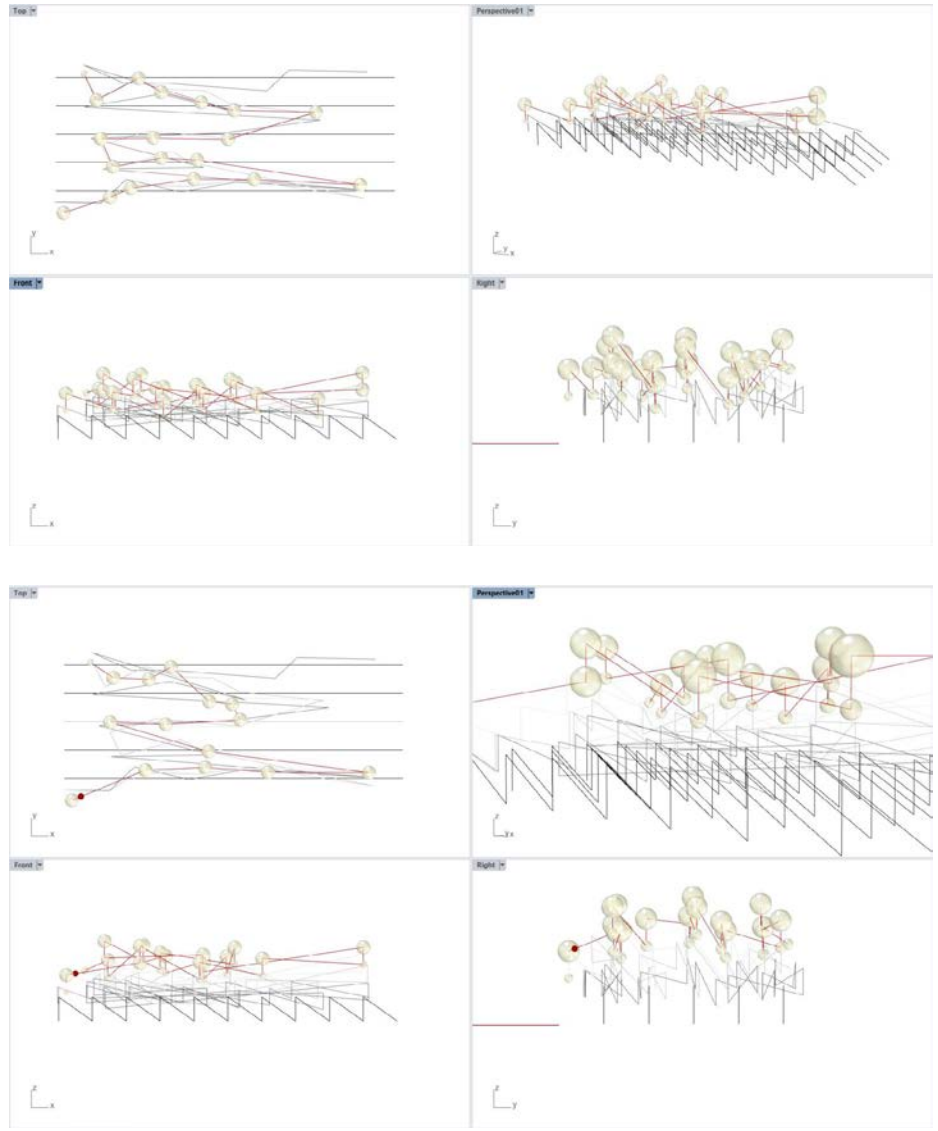
Spontaneous (Dis)order

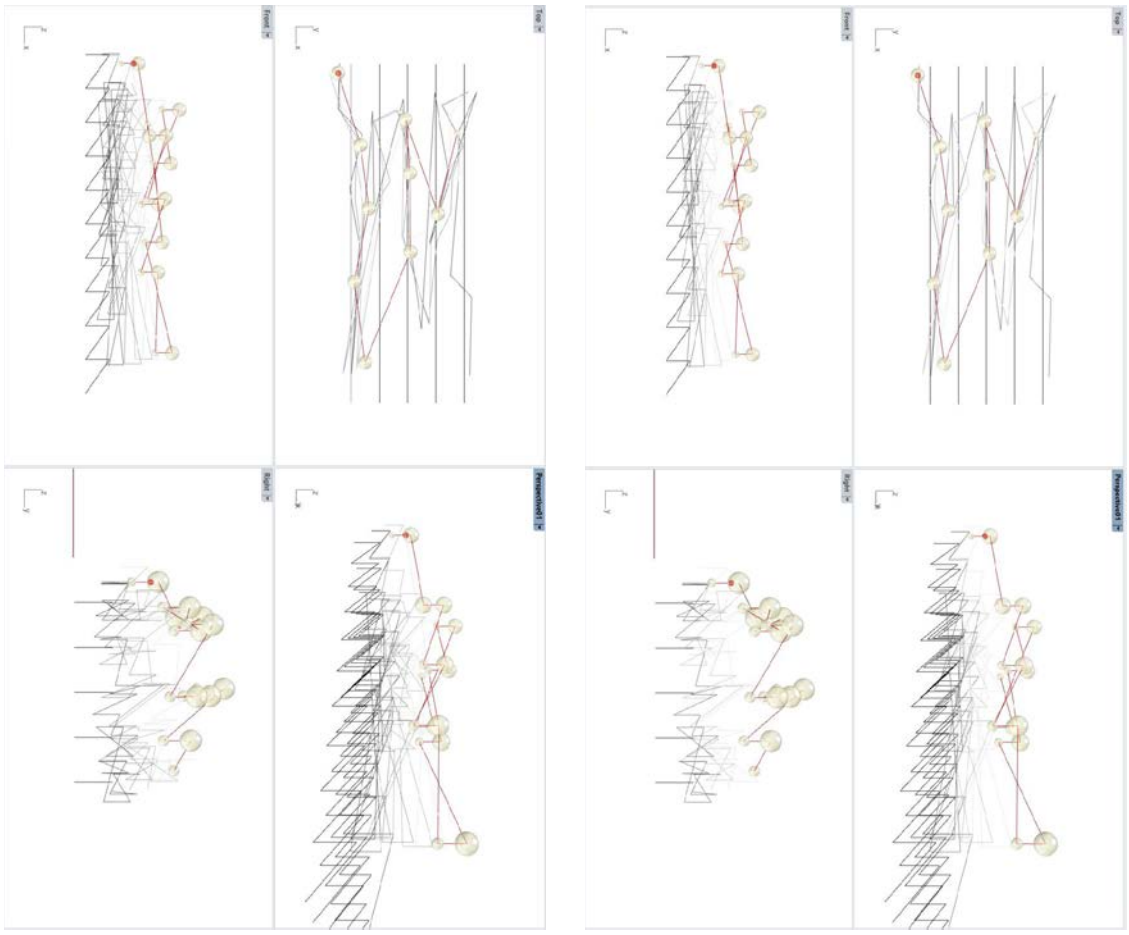


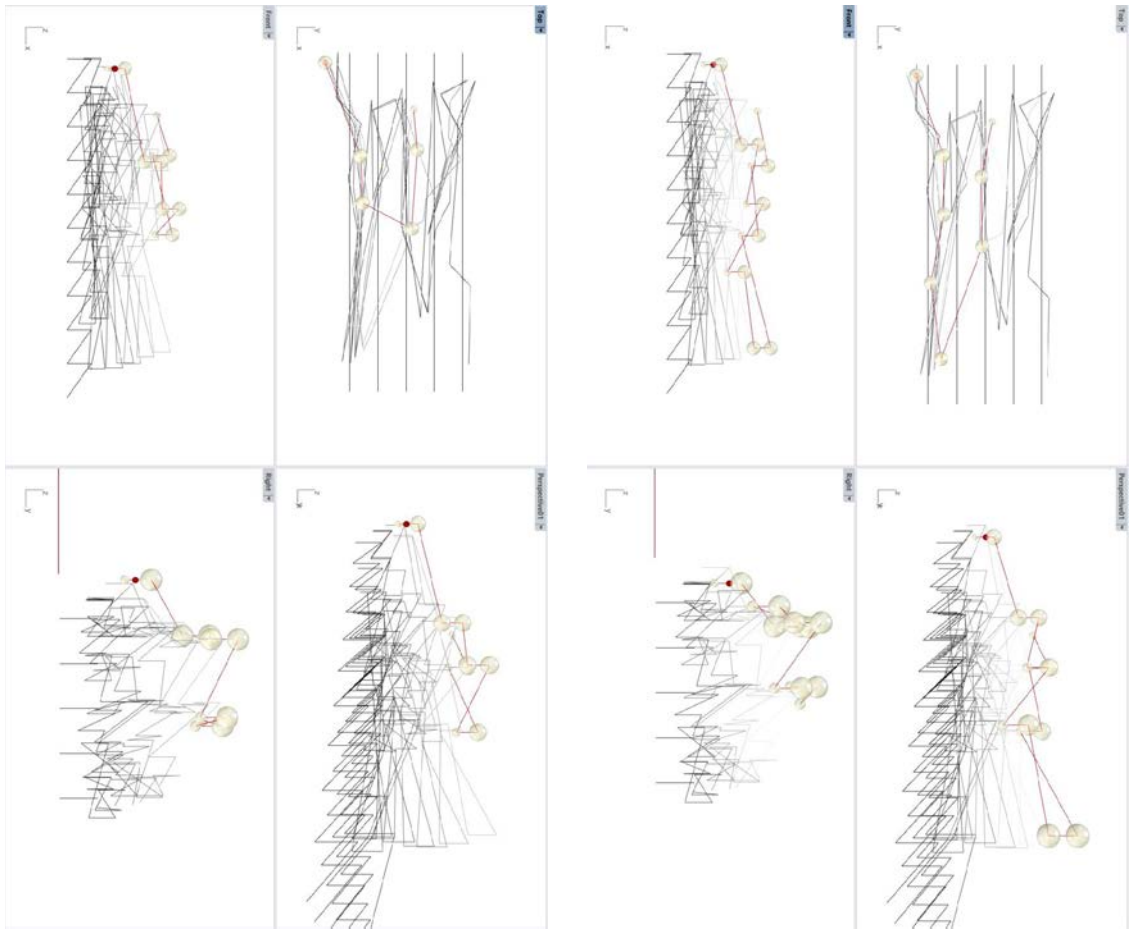


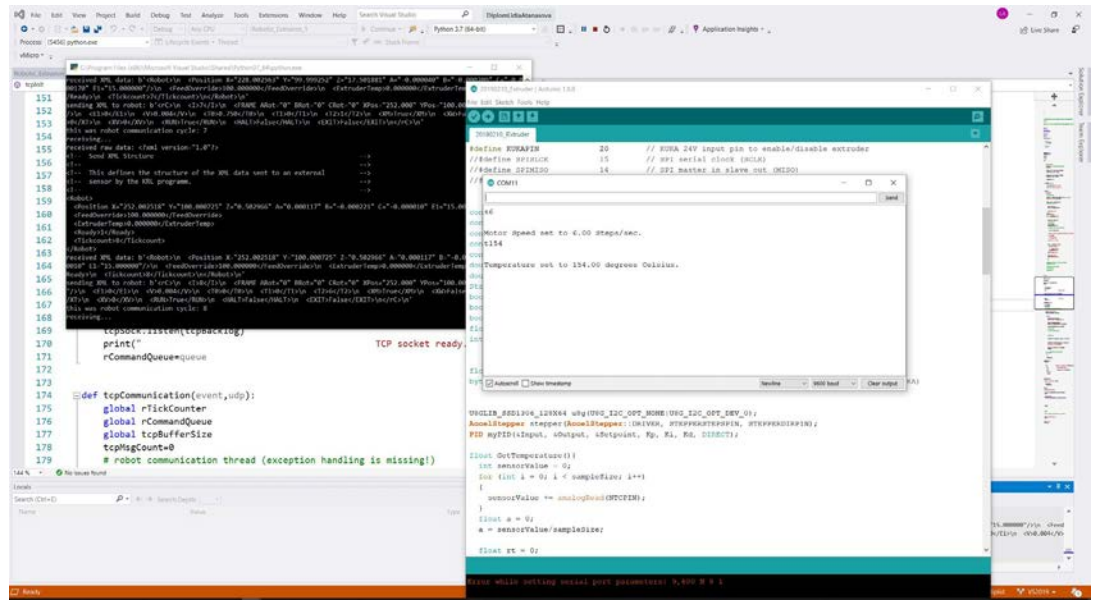


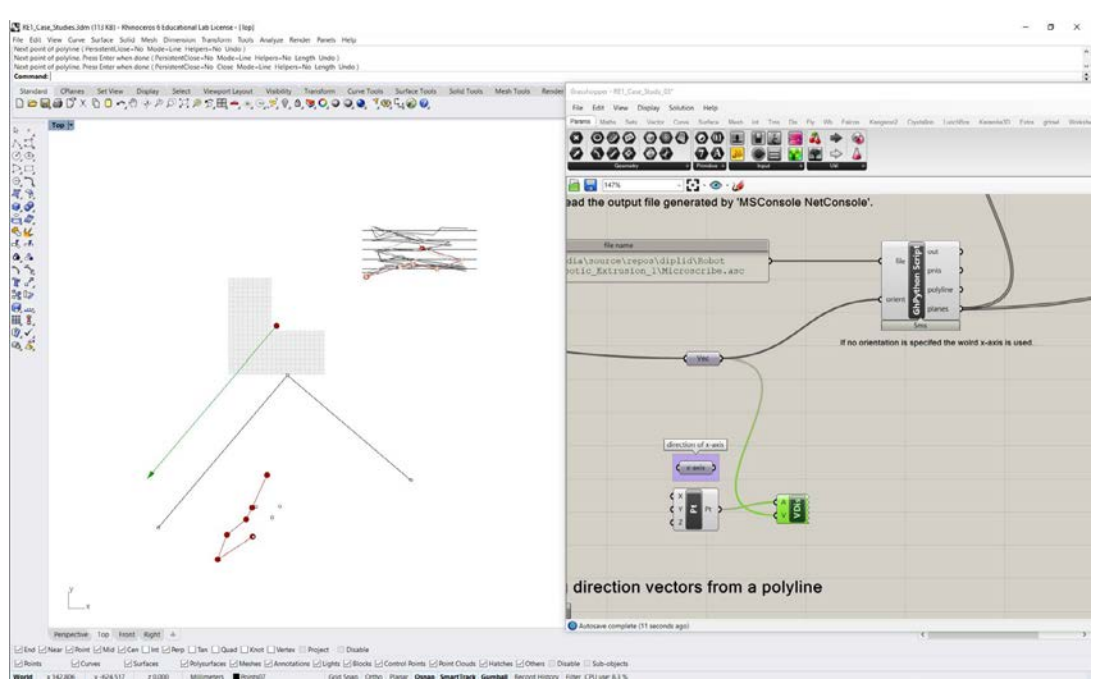












Conclusion

Current material practices in architecture invest immense efforts in exploring new materials and determining their behaviour. Nevertheless the way of manipulating them remains unchanged, imposing materials a given shape rather than triggering its spontaneous behaviour. This results primarily from the division between design and fabrication processes. Decisions concerning the manufacturing are taken in advance and the fabrication is a mere execution of machine instructions, requiring that materials perform in a predictable fashion. However processes could be responsive and adapt to different conditions and react to changes.

The present work examines the potentiality of custom robotic fabrication based on additive manufacturing for iterative material aggregation informed by physical material behaviour. In the series of case studies material feedback proved crucial not only for correcting toolpaths and thus compensating for deviations but it enables a more general approach to materiality and formation processes.

Nevertheless despite the applied fedback, it proved essential to test the material system in order to extract the range of geometrical possibilities. In this case this was done through analogue experiments. The process could be though extended by a feed forward component — extracting these parameters with simulation software. The results values are then fed to the computational model for greater precision and control over the process. Moreover the developed set up can be upgraded by additional sensing devices like cameras for 3D vision for the material feedback instead of 3D digitizing with Microscribe or additional temperature sensors for temperature control of the extruder and the extruded material. Using OpenCV 2D images could be post-processed and later applied to the computational model for digital spacial reconstruction of the physical model.

List of Figures

Figure 15. RobArch 2018,
Gramazio Kohler Research, Robotic Fabrication

Figure 16. Assembled MK8 extruder by Geeetech
https://www.geeetech.com/assembled-mk8-extruder-p-857.html
accessed June 2019

Figure 17. 3D Printing filament
https://www.pinterest.at/pin/231935449546787175/
accessed May 2019

Figure 18. 3D Printing filament materials
https://en.wikipedia.org/wiki/3D_printing_filament
accessed May 2019

Figure 19. Monomers in ABS polymers
https://en.wikipedia.org/wiki/Acrylonitrile_butadiene_styrene
accessed May 2019

Figure 20. The skeletal formula of PLA
https://en.wikipedia.org/wiki/Polylactic_acid
accessed May 2019

References

D. Balik and açalya Allmer, A critical review of ornament in contemporary architectural theory and practice, vol. 13. 2016.

M. P. Bendsøe and O. Sigmund, Topology optimization: theory, methods, and applications. Berlin ; New York: Springer, 2003.

C. Beyer and D. Figueroa, 'Design and Analysis of Lattice Structures for Additive Manufacturing', J. Manuf. Sci. Eng, vol. 138, no. 12, pp. 121014-121014–15, Sep. 2016.

M. Carpo, Ed., The digital turn in architecture 1992-2012. Chichester: Wiley, 2013.

D. Clifford, 'Project Nervi: Aesthetics and Technology'.

X. DeKestelier, 'Design Potential for Large Scale Additive Fabrication: Freeform', p. 4.

D. Fabricius, 'Architecture before architecture: Frei Otto's "Deep History"', The Journal of Architecture, vol. 21, no. 8, pp. 1253–1273, Nov. 2016.

T. Ferreira, Ed., Green Design, Materials and Manufacturing Processes. CRC Press, 2013.

J. Friedman, H. Klm, and O. Mesa, 'Woven Clay', ACADIA 14: Design Agency, pp. 223–226, Oct. 2014.

S. Galjaard, S. Hofman, N. Perry, and S. Ren, Optimizing Structural Building Elements in Metal by using Additive Manufacturing. 2015.

J. Gardan, 'Additive manufacturing technologies: state of the art and trends', International Journal of Production Research, vol. 54, no. 10, pp. 3118–3132, May 2016.

I. Gibson, D. W. Rosen, and B. Stucker, Additive manufacturing technologies: 3D printing, rapid prototyping, and direct digital manufacturing. 2016.

S. Hanna and S. H. Mahdavi, 'INDUCTIVE MACHINE LEARNING IN MICROSTRUCTURES', in Design Computing and Cognition '06, J. S. Gero, Ed. Dordrecht: Springer Netherlands, 2006, pp. 563–582.

M. Hensel, Performance-oriented architecture: rethinking architectural design and the built environment. Chichester, West Sussex: Wiley, A John Wiley and Sons, Ltd, Publication, 2013.

M. Kretzer, 'The Ever-Changing Nature of Materiality and the Meaning of Materials in Architecture and Construction', in Information Materi-

als: Smart Materials for Adaptive Architecture, M. Kretzer, Ed. Cham: Springer International Publishing, 2017, pp. 25–65.

D. L. Cohen and H. Lipson, 'Geometric feedback control of discrete-deposition SFF systems', Rapid Prototyping Journal, vol. 16, pp. 377–393, Aug. 2010.

A. Menges and S. Ahlquist, Eds., Computational design thinking. Chichester: Wiley, 2011.

N. Negroponte, 'Toward a Theory of Architecture Machines', Journal of Architectural Education (1947-1974), vol. 23, no. 2, p. 9, Mar. 1969.

P. L. Nervi, C. Chiorino, E. Margiotta Nervi, and T. Leslie, Aesthetics and technology in building, The twenty-First-Century edition. Urbana: University of Illinois Press, 2018.

F. Otto, Ed., Natürliche Konstruktionen: Formen und Konstruktionen in Natur und Technik und Prozesse ihrer Entstehung. Stuttgart: Deutsche Verlags-Anstalt, 1982.

N. Oxman, J. Laucks, M. Kayser, E. Tsai, and M. Firstenberg, 'Freeform 3D Printing: Towards a Sustainable Approach to Additive Manufacturing', Mediated Matter Group, MIT Media Lab, Cambridge, Massachusetts, USA.

A. Papadopoulou, J. Laucks, and S. Tibbits, From Self-Assembly to Evolutionary Structures, vol. 87. 2017.
G. Retsin, 'Discrete Assembly and Digital Materials in Architecture | ECAADE 2016'.

G. Retsin and M. J. Garcia, 'Discrete Computational Methods for Robotic Additive Manufacturing', ACADIA 2016.

D. Rosenwasser, S. Mantell, and J. Sabin, 'Robotic Fabrication and Digital Ceramics', p. 10, 2017.

J. D. Royo, L. M. Soldevila, M. Kayser, and N. Oxman, 'Modelling Behaviour for Distributed Additive Manufacturing', in Modelling Behaviour, M. R. Thomsen, M. Tamke, C. Gengnagel, B. Faircloth, and F. Scheurer, Eds. Cham: Springer International Publishing, 2015, pp. 295–302.

H. Salam, 'Re-thinking the Concept of "Ornament" in Architectural Design', Procedia - Social and Behavioral Sciences, vol. 122, pp. 126–133, Mar. 2014.

K.-M. M. Tam and C. Mueller, Additive Manufacturing Along Principal Stress Lines, vol. 4. 2017.

'Architektur der Sukzession. Prozesse entwerfen – Systeme entwickeln', db deutsche bauzeitung, 03-Jan-2005. [Online]. Available: https://www.db-bauzeitung.de/aktuell/diskurs/prozesse-entwerfen-systeme-entwickeln/. [Accessed: 05-Dec-2018].

'1969-70 - SEEK - Nicholas Negroponte (American)', cyberneticzoo.com, 27-Oct-2010. [Online]. Available: https://cyberneticzoo.com/robots-in-art/1969-70-seek-nicholas-negroponte-american/. [Accessed: 05-Dec-2018].

Advanced customization in architectural design and construction. New York: Springer, 2014.

Modelling behaviour. New York, NY: Springer Berlin Heidelberg, 2015.

Springer handbook of robotics, 2nd edition. New York, NY: Springer Berlin Heidelberg, 2016.

'Large-Scale 3D Printing Supports Innovation in Construction'. [Online]. Available: https://www.additivemanufacturing.media/blog/post/large-scale-3d-printing-supports-innovation-in-construction. [Accessed: 05-Dec-2018].

W. Berka, E. Brix, and C. Smekal, Eds., Woher kommt das Neue? Kreativität in Wissenschaft und Kunst. Wien: Böhlau, 2003.

M. Daas and A. J. Wit, Eds., Towards a robotic architecture. Novato, CA: Applied Research and Design Publishing, 2018.

P. Gruber, Biomimetics in architecture: architecture of life and buildings. Wien ; New York: Springer, 2011.

U. Hirschberg and Technische Universität Graz, Eds., Design science in architecture. Wien: Springer, 2005.

S. Kwinter and C. C. Davidson, Far from Equilibrium: essays on technology and design culture. Barcelona: ACTAR, 2007.

N. Leach and P. F. Yuan, Computational design. Shanghai: Tongji University Press, 2017.

H. Lipson and M. Kurman, Fabricated: the new world of 3D printing. Indianapolis, Indiana: John Wiley & Sons, 2013.

R. Rael and V. San Fratello, Printing architecture: materials and methods for 3D printing, First edition. Hudson, New York: Princeton Architectural Press, 2018.

G. Savio, R. Meneghello, and G. Concheri, 'Geometric modeling of lattice structures for additive manufacturing', Rapid Prototyping Journal, vol. 24, no. 2, pp. 351–360, Feb. 2018.

P. Weibel, 'Algorithmus und Kreativität', in Woher kommt das Neue? Kreativität in Wissenschaft und Kunst, W. Berka, E. Brix, and C. Smekal, Eds. Wien: Böhlau, 2003.

P. F. Yuan, N. Leach, and A. Menges, Digital fabrication. Shanghai: Tonji University Press, 2017.