Measuring and optimizing the 3D concrete printing process, using real time feedback

van Strien, E.C.F.

Award date:
2017
MEASURING AND OPTIMIZING THE 3D CONCRETE PRINTING PROCESS USING REAL TIME FEEDBACK

Graduation Thesis

E.C.F. (Emiel) van Strien
e.c.f.v.strien@student.tue.nl
Colophon

GENERAL
Report : Measuring and optimizing the 3D concrete printing process, using real time measurements
Date : March 8, 2017
Place : Eindhoven

STUDENT
Author : E.C.F. (Emiel) van Strien
Student number : 0744043
E-mail : e.c.f.v.strien@student.tue.nl / emielvanstrien@gmail.com
University : Eindhoven University of Technology
Faculty : Department of the Build Environment
Master Track : Architecture, Building and Planning (ABP)
Chair : Information Systems in the Built Environment (ISBE)

GRADUATION COMMITTEE
Chairman : Prof.dr.ir. B. (Bauke) de Vries
Supervisor : Dipl.-Ing. J. (Jakob) Beetz
Supervisor : R.J.M. (Rob) Wolfs MSc
Preface

This graduation thesis is the result of my graduation project, carried out at the department of the Build Environment at Eindhoven University of Technology. The graduation project is performed to complete the master Architecture, Building and Planning. A free-master program has been composed in collaboration with Prof.dr.ir. B. de Vries and the Information Systems in the Built Environment (ISBE) Group. The topic of this report is Measuring and optimizing the 3D concrete printing process, using real time feedback.

When the 3DCP-project was started in the summer of 2015 I was immediately interested in performing my graduation research within this project. For me it was the perfect combination of developing something completely new with a very practical approach. A topic was selected within the 3DCP-project in which I could incorporate the knowledge I gained from all other projects performed at the ISBE group as well as interest in exploring unknown terrains.

This graduation project was performed under supervision of Prof.dr.ir. B. (Bauke) de Vries, Dr. Dipl.-Ing. J. (Jakob) Beetz and R.J.M. (Rob) Wolfs MSc. I would like to greatly thank all of my supervisors for their expertise, confidence, understanding and wonderful collaboration. I am very proud and pleased with the end result of this graduation project and without them I could have never achieved this.

Additionally I would like to thank several other people. Firstly, all the students of the 3DCP-group and ir. Z.Y. (Zeeshan) Ahmed for all their effort and help with the development of my research and the print sessions. Secondly Dr. Ir. T.A.M. (Theo) Salet, as the founding father, and the partner companies who together created the 3DCP-project, which allowed me to take part in this great development.

Last but not least, I would like to thank my family and friends, specifically my brother Joeri, for their help and support during this entire project.

I hope you will enjoy reading this report.

Emiel van Strien
Content

COLOPHON ................................................................................................................................................. 2
PREFACE ..................................................................................................................................................... 4
CONTENT .................................................................................................................................................... 5
SUMMARY .................................................................................................................................................. 8

1. INTRODUCTION .................................................................................................................................. 10
   1.1 3D CONCRETE PRINTING .................................................................................................................. 10
   1.2 PROBLEM DEFINITION ..................................................................................................................... 13
   1.3 RESEARCH QUESTION(S) ................................................................................................................... 14
   1.4 EXPECTED RESULTS ......................................................................................................................... 16
   1.5 RESEARCH DESIGN .......................................................................................................................... 16
       1.5.1 METHODOLOGICAL JUSTIFICATION ...................................................................................... 16
       1.5.2 RESEARCH MODEL ................................................................................................................ 19

2. GLOSSARY .............................................................................................................................................. 22
   2.1 FIGURES .............................................................................................................................................. 22
   2.2 TABLES ............................................................................................................................................... 23

3. MEASUREMENT SYSTEM .................................................................................................................... 24
   3.1 SELECTED MEASUREMENT SYSTEM AND DEVICE ......................................................................... 24
   3.2 PLACEMENT OF THE MEASUREMENT DEVICE .............................................................................. 25
   3.3 MEASUREMENT SYSTEM SETUP .................................................................................................... 29
   3.4 DEVELOPED MEASUREMENT SYSTEM ......................................................................................... 32
   3.5 PERFORMANCE OF THE MEASUREMENT SYSTEM ......................................................................... 34

4 LOGGING AND ANALYZING OF VARIABLE DATA .............................................................................. 36
   4.1 STEP 1 - PERFORM MEASUREMENTS AND TRANSFER DATA ......................................................... 38
   4.2 STEP 2 - EXECUTING THE G-CODE PRINT PATH ............................................................................. 38
   4.3 STEP 3 - PERFORM THE TRACE FUNCTION AND EXPORT DATA .................................................... 38
   4.4 STEP 4 – IMPORT DATA IN EXCEL AND ANALYZE DATA USING GRAPHS ................................... 39
   4.5 STEP 5 – IMPORT AND CONVERT TRACED/EXTERNAL DATA AND ADJUST DISPLAY SETTINGS ...... 41
   4.6 STEP 6 – VIEW AND ANALYZE THE DATA USING THE 3D VISUALIZATION .................................. 43

5 OPTIMIZING USING REAL TIME ADJUSTMENTS .............................................................................. 46
   5.1 PRINTER VARIABLES FOR OPTIMIZATION ....................................................................................... 46
       5.1.1 THE COORDINATES OF THE PRINTER ..................................................................................... 46
       5.1.2 THE MOVEMENT SPEED OF THE PRINTER ........................................................................... 47
       5.1.3 THE PUMP-FREQUENCY OF THE CONCRETE PUMP ................................................................. 47
       5.1.4 SELECTED VARIABLE FOR OPTIMIZATION .......................................................................... 48
   5.2 REAL TIME OPTIMIZATION METHOD ............................................................................................. 48
       5.2.1 SYNCHRONIZED ACTIONS ......................................................................................................... 48
       5.2.2 REALTIME OPTIMIZATION MODULES .................................................................................... 50
   5.3 BASIC OPTIMIZATION MODULE .................................................................................................... 52
   5.4 TESTING THE BASIC OPTIMIZATION MODULE .............................................................................. 54
5.4.1 TEST SETUP........................................................................................................................................54
5.4.2 TEST RESULTS....................................................................................................................................57
5.4.3 ANALYSIS OF THE TEST RESULTS .................................................................................................59
5.5 IMPROVEMENTS TO THE BASIC OPTIMIZATION MODULE .............................................................64
5.5.1 DELAYED RESPONSE..................................................................................................................64
5.5.2 USE AVERAGE OR FILTERED VALUE FOR RESPONSE .................................................................73
5.5.3 USING PID-CONTROL................................................................................................................83
5.5.4 CONCLUSION OF THE IMPROVEMENTS TO THE OPTIMIZATION MODULE...............................92
6 USE CASES ..............................................................................................................................................94
6.1 DETERMINE PID-FILTER RESPONSE MODULE SETTINGS FOR THE SECOND USE-CASE.................96
6.2 USE-CASE 1: PRINT CYLINDER WITH NO PREPROGRAMMED Z-MOVEMENT ......................................98
6.3 USE CASE 2: PRINT ON AN UNEVEN 3D SURFACE WITH NO PREPROGRAMMED Z-MOVEMENT.....100
6.4 USE CASE 3: LOGGING AND ANALYZING VARIABLE VALUES IN A 3D VISUALIZATION ..............102
7 CONCLUSION AND RECOMMENDATIONS ...........................................................................................106
7.1 CONCLUSION........................................................................................................................................106
7.2 RECOMMENDATIONS .......................................................................................................................108
7.2.1 IMPROVE THE CURRENT MEASUREMENT SYSTEM......................................................................108
7.2.2 EXTEND THE VARIABLES ADJUSTED TO CREATE A REAL TIME OPTIMIZATION.......................108
7.2.3 EXTEND THE MEASURED OUTPUT PARAMETERS .........................................................................109
7.2.4 USE A THIRD GENERIC BOX TO STORE DATA AND TO PERFORM COMPLEX CALCULATIONS AND SIMULATIONS ..............................................................................................................109
8 REFERENCES ..........................................................................................................................................110
9 APPENDICES .........................................................................................................................................112
APPENDIX A MEASUREMENT SYSTEM .................................................................................................113
APPENDIX A1 MEASUREMENT SYSTEMS MATRIX .................................................................................113
APPENDIX A2 ARDUINO 1 SCRIPT – TRANSMITTER ARDUINO .................................................................114
APPENDIX A3 ARDUINO 2 SCRIPT - RECEIVER ARDUINO ....................................................................116
APPENDIX A4 CONNECTION SCHEMATICS MEASUREMENT SYSTEM PART 1 .......................................120
APPENDIX A5 CONNECTION SCHEMATICS MEASUREMENT SYSTEM PART 2 .......................................121
APPENDIX A6 HOW TO ATTACH AND USE THE MEASUREMENT SYSTEM .............................................122
APPENDIX B LOGGING AND ANALYZING OF VARIABLE DATA ..............................................................124
APPENDIX B1 HOW TO USE THE TRACE FUNCTION ON THE SINUMERIK ................................................124
APPENDIX B2 HOW TO USE THE SINUMERIK OFFLINE TRACE TOOL ...................................................130
APPENDIX B3 HOW TO IMPORT AND XML-FILE INTO EXCEL ..............................................................139
APPENDIX B4 GRASSHOPPER MODEL ..................................................................................................141
APPENDIX B5 PYTHON SCRIPT BLOCK 2 ...............................................................................................146
APPENDIX B6 PYTHON SCRIPT BLOCK 4 ................................................................................................149
APPENDIX B7 CREATE AN EXTERNAL DATA FILE USING EXCEL .........................................................150
APPENDIX C OPTIMIZING USING REAL TIME ADJUSTMENTS ..................................................................151
APPENDIX C1 BASIC OPTIMIZATION MODULE (SPF) ..............................................................................151
APPENDIX C2 START AND STOP TRIGGER SETTINGS TEST SETUP .......................................................152
APPENDIX C3 PRINT PATH SCRIPT OF THE TEST SETUP (MPF) ............................................................153
APPENDIX C4 DELAYED RESPONSE MODULE (SPF) .............................................................................153
APPENDIX C5 FILTERED RESPONSE MODULE (SPF) ..........................................................................154
APPENDIX C6 AHV RESPONSE MODULE (SPF) .....................................................................................155
APPENDIX C7 PID RESPONSE MODULE (SPF) .......................................................................................156
APPENDIX D USE CASES .......................................................................................................................157
APPENDIX D1 PID/FILTER RESPONSE MODULE ON (SPF) .....................................................................157
APPENDIX D2 PID/FILTER RESPONSE MODULE OFF (SPF) .................................................................157
APPENDIX D3 PRINT PATH SCRIPT USE CASE 1 – WITH PREPROGRAMMED Z-MOVEMENT (MPF) ....158
APPENDIX D4 PRINT PATH SCRIPT USE CASE 1 – WITH PID-FILTER RESPONSE MODULE (MPF) .....159
APPENDIX D5 PRINT PATH SCRIPT USE CASE 2 (MPF) ............................................................................160
Summary
Similar to other disciplines, the construction industry is working on the development of integrating 3D printing in their line of work. One of the techniques being developed is 3D concrete printing (3DCP). In line with this development the department of the build environment at the Eindhoven University of Technology, together with ten industrial partner companies, have recently constructed a 3D concrete printer. The main components involved in 3DCP are the printing technique, the material used and the printed shape which are all related to each other. Little is known about the about the relationship between those components, because of this technique being relatively new. The 3DCP process is controlled by adjusting the printer variables like the used material, print path, shape of the nozzle and printing speed. The used variable settings influence the output of the printer. Example of output parameters are the layer height, shape of the layer, bond strength between the layers and many more. Since little is known about the exact relationship between the variables and the output parameter, a time consuming trial and error process is used to optimize the output by adjusting the variable settings. A more efficient way to optimize the printing process is to optimize the output during printing. This can be achieved by measuring the output parameters and automatically adjust the variables to improve the output in real time. In this report a proof of concept is developed that shows that real time measurement can be used to automatically adjust the variables of the 3DCP process, in order to optimize the output of the printing process in real time.

The layer height is used as the output parameter on which measurements are performed. A measurement system is developed that measures the distance from the nozzle of the printer to the previous layer, which is used to derive the layer height. The measurement system transfers the data wirelessly to the operating machine of the printer. The printer variable that is adjusted to optimize the output of the printer is the height of the printer. The actual layer height is often smaller than the expected layer height as a result of deformations in the printed element. This causes the distance from the nozzle to the previous layer to increase, which results in inaccurate placement of the concrete. This affects the stability properties and increases the deformations in the printed element. An optimization module is created within the operating machine that uses the real time measurements to adjust the height of the printer to maintain a constant and desired distance between the nozzle and the previous layer, also in real time. A cycle of tests is performed to determine the performance of the optimization module and to create improvements to the module. This resulted in a final optimization module which is used to perform two use case studies. The two case studies deliver a proof of concept that show that the developed module is capable of successfully using real time measurements to optimize the output of the printer. Beside the real time optimization, it became clear during the research that creating a method to effectively log and analyze is very important to fully understand the printing process. A method is developed in this research to log data from the operating machine and effectively analyzing it using 2D graphs or a 3D visualization of the printed element.

The proof of concept delivered in this research can form the base for more research regarding this subject. The report is written in a way that allows future researchers to understand and use the techniques developed in this research to continue researching this topic. Recommendations for future research are provided to give a possible direction for this research.
1. Introduction

1.1 3D concrete printing
The process of 3D printing is developing very fast and is widely integrated in many disciplines. More and more different materials can be printed, and the range of possibilities provided by 3D printing is increasing. More recently, the construction industry is working on the implementation of 3D printing in their line of work as well. In Amsterdam, a canal house is currently being 3D printed with a bio-based Table 1: Example spreadsheet by importing and XML-trace file into Excel material that is filled with Eco-concrete (DUS Architects). Another development is the 3D printing of concrete structures. Multiple techniques for 3DCP (3D concrete printing) are being developed and are the subject of recent research efforts. Examples of these techniques are: Contour crafting (CC), Concrete Printing (CP), D-Shape. These methods have their own advantages as well as disadvantages. CC is the oldest still existing technique in the area of 3DCP. This method uses layered manufacturing to create large objects with a smooth finish. This smooth surface finish is achieved by constraining the output flow in the vertical and horizontal direction. Multiple nozzles have been used simultaneously side by side to create a double-layered structure. CC allows the use of a large variety of materials with large aggregates and additives (Khoshnevis, Hwang, Yao, & Yeh, 2006). CP is similar to CC, but has a smaller resolution of deposition to achieve higher 3-dimensional freedom. This results in a ribbed surface finish. The resulting finish completely depends on the layer thickness used (Lim, Buswell, Le, Austin, Gibb, & Wackrow, 2011). A third 3DCP technique is D-Shape. This technique differs from the two aforementioned techniques. Applying a binder material to specific locations selectively hardens a powder deposition. In order to achieve this, a printing head composed of 300 nozzles is used. The print head moves over the entire printing area and deposits the binder to areas that need to be solidified. The unhardened sand acts as a temporary support for the layers above. This can result in shapes that cannot be achieved with the other techniques. The unhardened sand is removed when the entire process is completed. A disadvantage of this method is that, for each layer, the sand has to be spread and compacted. (Lim, Buswell, Le, Austin, Gibb, & Thorpe, 2012)

Many different components are involved in every 3DCP technique. The printed shape, the material used and the printing technique are the main components involved in this process. Due to this technique being relatively new, little is known about the different components and specifically about the relation between those different components. (Figure 1)

![Diagram](image)

Figure 1: Relationship between the main components of 3DCP (Wolfs, 2015)
The department of the Built Environment at the Eindhoven University of Technology, together with ten industrial partner companies, have recently constructed a new 3D concrete printer (Eindhoven University of Technology). This printer is 12x6x4 meters and is able to print large- and small-scale concrete objects. The printer creates thin layers of concrete, from millimeters up to multiple centimeters, that are placed on top of each other to create one big structure. This method is similar to other 3DCP techniques that are currently being used and developed. Using this technique, a structure can be created with high precision. This printer is specifically designed and created to study 3DCP techniques. This project is in its early stages and much more research and tests are necessary before the printing process is fully understood and can be controlled at desirable levels.

The 3D concrete printer developed in this project consists of three main components. A gantry robot (1), a concrete mixing pump (2), and an operating system (3). These three components combined allow for 3D concrete printing. The entire printer setup can be seen in Figure 2. The concrete mixing pump mixes a dry cement mixture with water creating the fluid concrete. This fluid concrete is then forced through a rotor stator pump driven by a frequency motor. The rotor stator pump pumps the concrete through a hose to a nozzle connected at the bottom of the vertical arm of the gantry robot. The gantry robot can move in X-, Y-, Z-direction and rotate along the vertical Z-axis. By simultaneously pumping concrete through the nozzle and moving the gantry robot, concrete material can be deposited at a desired location in the printing area. The operating system is a SINUMERIK 840D sl (Siemens, 2011). This is a CNC (Computer Numerical Control) control system developed by Siemens. The operating system controls the motion of the gantry robot, but is also connected to the concrete mixing pump to be able to control the frequency of the rotor stator pump. This allows the operating system to indirectly influence the pressure with which the concrete is forced through the nozzle.

Figure 2: The 3D concrete printer setup including the three main components
In 3DCP multiple components, variables and output parameters are involved. These components interact and influence each other, which is visualized in Figure 3. The involved components (A) in the 3DCP process result in many different variables (B) that heavily influence each other. Those variables influence the output parameters (C) of the 3D concrete printer. Parameters (C) are the shape of the printed layer (height, width, sectional shape), the strength of a layer, the bond strength between the different layers, and the overall stability of the object, amongst others. Examples of variables (B) that influence those parameters are the composition of the concrete mixture, the nozzle (shape and system), the printing speed, the hardening time, printing properties (height and angle from which the next layer is printed), environmental influences (temperature, humidity, sunlight etc.) and the shape of the printed object. All of those variables potentially influence each of the output parameters (C). The printed concrete is the output and has multiple different parameters. Examples of those output parameters (C) are the shape of the layer, which can be decomposed into the height/width of the layer or the sectional shape, the strength of a layer, the bond strength between the layers, and more. To fully understand and control the process of 3DCP it is important to be able to control and influence these output parameters. To ensure that the actual output meets the expected output it is important to measure the output on those different parameters (D). When the actual output differs from the expected output, changes can be made to different variables. Changes to these variables will influence the printing process to optimize the output (E). By changing the variables separately, the impact of each variable on the different parameters can be determined. In this early stage of the project, the actual output will often differ from the desired result. Accurate measurements will help to improve the analysis of the output and determine the impact of variables. Extensively testing while continuously adjusting variables will help to control the process of 3DCP and will make it increasingly able to achieve the desired results.

**Figure 3: Relationship between components, variables, and output parameters of 3DCP**
1.2 Problem definition

The main problem of the measuring process of 3DCP is that presently, the measurements are performed manually and visually after a print session is completed. This is very time-consuming and does not allow for real time optimization. With this method, the measurements can be used to change variables for the next run. The impact of changing the variables can then be determined. A more efficient way of 3DCP would be to use real time measurements to be able to determine parameter values of the output during printing. This will create the possibility to change the input by altering variables during the process. A similar method is used by (E. Lloret, R. Shahab, M Linus, J. Flatt, F. Gramazio, M Kohler, S Langenberg, 2015), in which a material feedback system is used to monitor the strength evaluation as concrete is extruded by an industrial robot arm. In (S. Neudecker, C. Bruns, R. Gerbers, J. Heyn, F. Dietrich, K. Droder, A. Raatz, H. Kloft, 2017) a feedback loop is proposed for robotic spraying of concrete, where 3D scanners measure the surface finish of the sprayed concrete parts. Using real time measuring data to manually alter a variable can be difficult and will create the possibility for human errors. A more efficient and user-friendly way to do this is creating a method that can automatically use the measurement data to adjust the variables and optimize output in real time (Figure 4). Aside from this optimization of the print process, this technique also allows for testing newly developed concrete mixtures, different printing strategies and printing in changing outdoor environments. With this technique, this can be achieved without the need for a time-consuming trial and error process to find the matching variable values. Therefore, it is very important, not only from a scientific point of view, but also for 3DCP in practice, to monitor, control and create an efficient printing process.

![Diagram of components and variables](image)

Figure 4: Real time optimization of the 3D concrete printing process
1.3 Research question(s)

The goal of this research is to optimize the printing process by creating a proof of concept that shows that real time measuring can be used to automatically adjust the input for the 3D concrete printer to optimize the output of the printing process in real time. In order to generate real time measurements, the optimal measurement method needs to be found for this specific construction method and research. The main research question is:

*How can real time measurements be employed in the automated optimization of 3D concrete printing technologies?*

The following sub questions divided into two groups need to be answered in order to answer the main research question:

- **Measurements**
  - Which critical output parameter(s) will be measured?
  - What are the requirements for a measurement system?
  - Is there a measurement system usable and available that meets the requirements of this research project?
  - Can usable data for this research be retrieved from the measurement data?

- **Optimization adjustments to the input**
  - Which variable(s) will be altered to influence the output?
  - Can the printer input and the variable settings be automatically adjusted in real time?
  - Can adjustments to the input optimize the output?
  - Can adjustments to the input result in the desired output?
  - Will these automatically generated adjustments lead to a more efficient printing process?
Within this project and the development of the 3D concrete printer, it became increasingly clear that to be able to fully understand the printing process, information about different variables and output parameters is very important. Logging of this data creates the possibility to review a print session and give insight into this variable information. This can help to determine the impact of different variables on the output of the printer. Little is known about the values of many variables which might have a big impact on the output of the printer. In order to gain more insight into the values of those variables and therefore better knowledge about the printing process, a secondary goal of this research has been defined. This goal is to develop a method for logging and analyzing measurement and variable data (Figure 5). For this reason, a third sub-question group is added regarding this subject.

- Logging and analyzing of variable data
  - Can variable data be retrieved from the printer?
  - Can a method be developed to log and analyze this data?
  - Is this method applicable and usable within the 3DCP-process?
  - Does the developed logging method give more insight on the impact of the different variables in the printing process?

![Diagram](image)

*Figure 5: Secondary goal, Logging and analyzing of data*
1.4 Expected results

The expected result of this research is a proof of concept that uses measurements to real time optimize the printing process. This optimization means that variables will be automatically adjusted based on measurements on an output parameter. Adjustments to the variables will be with the goal optimize the output and maybe even to achieve the desired output. As mentioned before, the relationships between variables and the measured parameters of the 3D concrete printer are relatively unknown. Because of this, actual optimizations to the printing process by changing specific variables are difficult to predict, and therefore not guaranteed. However, the goal is to deliver a proof of concept in which the printing process will be real time influenced by automatically changing variables. The chosen strategy for this research project: Measuring and optimizing the 3DCP process using real time feedback, will be evaluated and discussed on its future value.

Beside this main goal of real time optimization of the printing process, a secondary goal has been defined to create a method for logging and effectively analyzing variable data. This will help gain more insight and better control the 3DC printing process.

1.5 Research Design

1.5.1 Methodological justification

Multiple tests with the 3D concrete printer have been performed at the TU/e. The system is functioning and some significant results have already been achieved (Eindhoven University of Technology). Concrete structures of up to eighty layers stacked on top of each other have been printed so far. Often, after this amount of layers have been printed, stacking problems occur which result in layers toppling off the previous ones. These results have been achieved by manually changing variables in a trial and error process. Even though promising results have been achieved, this process was very time-consuming. Preparing and cleaning the printer can take up to one hour for each printing session. When the printing process is stopped because of problems, the whole systems needs to be cleaned immediately before the wet concrete dries within the system.

It is possible to measure all output parameters at the same time, but it is impossible to determine the impact of all printer variables at the same time. Many parameters are related to each other and are influenced by different variables. The impact of the variables on the parameters is still unknown to a large extend. Hence, this research is focused on determining the impact of one or more variables on one specific parameter. This parameter will be measured so the impact of the variable(s) on this parameter can be analyzed.

To gain knowledge about the impact of a variable, tests have been performed to determine the impact of a single variable on an output parameter. In Figure 6 the impact of changing the printing speed on the sectional shape of the layer is shown. All variables were set to fixed values except for the printing speed, which was increased over time. Increasing the print speed had a big impact on the section of the printed layer. Both the layer height and width are influenced as
well as the sectional shape. This test shows the ability to test and influence the output by changing one variable at the time while observing a parameter.

There are multiple boundary conditions that need to be met when choosing a specific parameter for this research. Firstly, the parameter should be measurable in real time in order to be able to automatically use the measurement data to alter the input. The second condition is, that a parameter should be chosen which could otherwise interrupt the printing process, since the goal of this research is to create an efficient printing process. Based on these two boundary conditions a choice for a specific parameter can be made.

The strength of a layer, the bond strength between the different layers and the total strength of the object are three parameters for which it is hard to perform real time measurements. Measuring these properties will often require instruments that need to be physically attached to the concrete output. This is possible after printing, but it is very complicated to do this while printing. The shape of the printed layer, and specifically certain properties of this parameter, like layer height and width, can be measured without the need of attaching physical instruments to the concrete output. For example: using lasers to determine the distance from a known location to the concrete object could be used to determine the height of a printed layer. These measurements can be made parallel to the printing process, which allows for direct use of this data to alter the input during printing. Another reason for choosing the layer height as parameter to measure is that when the expected layer height differs from the real layer height this could lead to having to interrupt the printing process. This is explained in the following section using Figure 7.
When the actual layer height is smaller than the expected layer height the distance between the printing nozzle and the previously deposited layer of concrete will increase with every layer printed. This will affect the actual distance between the nozzle and the previous layer. When a small number of layers is printed the effect will be small, but when the amount of layers increase, the difference between the desired distance and the actual distance to the previous layer will increase. Eventually this will cause printing in “open air” instead of printing the next layer exactly on top of the previous one. This results in dropping the concrete mixture, which influences output parameters of the printing process. This directly causes changes to the layer shape and strength of the printed object. When the actual layer height is bigger than the expected layer height the actual distance will decrease. This will also influence output parameters and might even cause the printer to get stuck or break due to collision with its previously laid concrete. Measuring the height of each layer and using this data to automatically change the input of the 3D printer can prevent this. This could help to create an efficient printing process. For this reason, real time measurements on layer height is the parameter of choice for this project. These measurements will then be used to optimize the printing process in real time. There are measurement systems that can generate data about the entire top profile of a printed layer. When such a measurement system is available and usable within this project, information about the layer width and top profile can also be used for optimizing the printing process.
1.5.2 Research model

The research model of this project is visualized as a BPMN diagram, shown in Figure 8. The vertical lanes represent the different actors and the rectangles are the actions that need to be performed. Four actors are involved in the research model: The 3D printer (the gantry robot and the concrete mixing pump), the measurement system, the operating system and the researcher. The researcher is placed in the fourth lane, since it will not be part of the result of this research, which is a method to optimize the printing process in real time. The actions that need to be performed by the researcher create the base of the optimization tool. The researcher only supplies the original input for a print session once, which will then be automatically optimized. In the future when the method is fully developed a user can utilize this method and only needs to supply the original input. This input is the starting point of each printing session in which the variable values are chosen and the desired output is formulated. The other three actors are part of the optimization method of this project, which is a continuous process. This method will optimize the printing process in real time and automatically to achieve the desired output by using real time measurements.

The operating system executes the provided printer input. This causes the concrete mixing pump to start pumping the fluid concrete and causes the gantry robot to move, which will result in a printed layer (the output). The measurement system will then perform measurements on the printed layer and send the data to the operating system. The operating system generates usable data from the measurement data and gathers system variable data. When the actual output of the printer is the expected output, no adjustments to the input are made and the printing process continues unchanged. However, when the actual output differs from the expected output the real time optimization takes place within the operating system. The operating system changes the input (the variables) until the desired output is achieved. This optimization within the operating system is performed by using synchronized actions (SA). How they work and how they are used to create a real time optimization during printing, is explained in chapter 5.

The real time optimizations are performed directly from the SINUMERIK NC (Numerical control) in the operating machine, without the use of a third “generic box” to perform calculations. This decision is made in collaboration with Siemens, the producer of the operating machine and the supervisors of this project. Using a third generic box would complicate the process of real time transferring both the measurement data and other variable data, since data would need to be transferred from the operating machine to the generic box and then back again to the operating machine. For this reason, the real time optimizations will be performed directly from the operating machine to increase the chance of successfully delivering a proof of concept within the time frame of this project. However, such external calculations are generally possible and might be desirable in other situations (Asgarpour, 2016).
Figure 8: Research model in a BPMN swim lane diagram
Besides the decision to create the real time optimizations directly from the operating machine, an important part of this research project is the design of the measurement method and system. The measurement systems should meet multiple requirements. The requirements have been formulated in collaboration with R.J.M. Wolfs, based on experience gained from multiple tests performed with the 3D concrete printer. In chapter 3: “Measurement system”, the requirements for a measurement system are determined, a decision is made for a specific measurement system and a design is created based on these requirements.

To be able to meet the secondary goal regarding the logging and analyzing of printer variable data, data needs to be retrieved and stored during printing. Similar to the real time optimizations, this retrieval and storing of data will be performed directly on the operating machine. A method to retrieve and store variable data, called the “trace function” is available within the operation machine. How this function is used to create a method to retrieve and analyze variable data is further elaborated in chapter 4.
2. Glossary

2.1 Figures

Figure 1: Relationship between the main components of 3DCP (Wolfs, 2015).................................10
Figure 2: The 3D concrete printer setup including the three main components.................................11
Figure 3: Relationship between components, variables, and output parameters of 3DCP .....................12
Figure 4: Real time optimization of the 3D concrete printing process...........................................13
Figure 5: Secondary goal, Logging and analyzing of data ...............................................................15
Figure 6: Impact of printing speed on sectional shape ........................................................................17
Figure 7: Effect of a difference between the actual and the expected layer height ............................18
Figure 8: Research model in a BPMN swim lane diagram ...............................................................20
Figure 9: Three concept designs for a measurement setup ...............................................................26
Figure 10: New measurement setup design ....................................................................................26
Figure 11: Incorrect measurements when moving around corners ...............................................28
Figure 12: Measurement system setup ............................................................................................29
Figure 13: Nozzle allows for infinite rotation around the Z-axis ..................................................30
Figure 14: The first part of the developed measurement system which is connected to the nozzle ...32
Figure 15: The second part of the developed measurement system that transfers the data to the SINUMERIK NC.................................................................33
Figure 16: The logging and analyzing process ..................................................................................37
Figure 17: Example Time vs Variable value graph created from the imported XML-trace file ........40
Figure 18: Overview of the developed Grasshopper model to create a 3D visualization ..................41
Figure 19: Example 3D visualization with the Path Velocity variable projected as a black and white gradient, the path velocity differs a lot throughout one layer, but is similar at the same X- and Y-coordinates of each layer..........................................................43
Figure 20: Adjusting Z-offset to maintain the desired distance between the nozzle and the layers ......47
Figure 21: Schematic diagram of synchronized actions (SIEMENS AG, 2011) ...............................49
Figure 22: Example of a synchronized action .................................................................................49
Figure 23: Real time optimization modules using the MPF-SPF structure ......................................51
Figure 24: Basic optimization module ............................................................................................52
Figure 25: Visualization of the basic optimization module ..............................................................52
Figure 26: The three different test scenarios used to determine the performance of the real time optimization module .........................................................................................54
Figure 27: Photo of the test setup with the third scenario ...............................................................54
Figure 28: Baseline graph of the actual situation of the three test scenarios .................................56
Figure 29: Scenario 1 - Z-movement of the basic optimization module compared to the actual situation .................................................................57
Figure 30: Scenario 2 - Z-movement of the basic optimization module compared to the actual situation .................................................................58
Figure 31: Scenario 3 - Z-movement of the basic optimization module compared to the actual situation .................................................................59
Figure 32: Transformation of the raw measurement values .............................................................60
Figure 33: Baseline graph versus measurement scan test: scenario 1 ..........................................61
Figure 34: Baseline graph versus measurement scan test: scenario 2 ..........................................61
Figure 35: Baseline graph versus measurement scan test: scenario 3 ..........................................62
Figure 36: Using a delayed response .............................................................................................63
Figure 37: Delayed response based on the distance between the nozzle and the measurement device (D) .................................................................64
Figure 38: Visualization of a ring-buffer with six elements ...........................................................65
Figure 40: Delayed response based on travelled distance .............................................................65
Figure 40: Storing X- and Y-coordinates in the Ring-Buffer to calculate the travelled distance ........66
Figure 41: Calculating the Distance travelled (DT) using Pythagoras ..........................................66
Figure 42: Delayed response module .............................................................................................67
Figure 43: Ring-buffer setup used in the delayed response module ...............................................68
2.2 Tables

Table 1: Example spreadsheet by importing an XML-trace file into Excel .......................................................... 39
Table 2: Raw and delayed measurement values of the fictional scenario .......................................................... 72
3. Measurement system

3.1 Selected measurement system and device

A very important step in this research project is to create a measurement system to perform and transfer the measurements needed to perform a real time optimization. In order to choose and create a measurement system the following requirements have been formulated:

- the measurement system needs to be able to generate data about each layer height
- the accuracy of the measurement system should be around 1mm for the system to be able to improve the printing process
- the measurement range should be sufficient
- the data needs to be generated and processed real time
- the system needs to be affordable
- the system should be applicable and usable with the 3D concrete printer used in this research.

Aside from these requirements, optional requirements are formulated that influence the decision for a measurement system. In this research project the first priority is to use height data. However, in the future more parameters can be used to optimize the printing process. By choosing a system that can create a 3D point cloud or a top-profile of a layer, information about other parameters like shape of the section and layer width can be retrieved. A great advantage of a system would be if it were applicable in outdoor environments. This would enable this system to be used in an outside setting. Multiple measurement systems are available from which the best suited system for this project are weighed against each other on different properties, based on the requirements, in a matrix shown in Appendix A1. From the seven measurement systems, six have the possibility of creating a 3D point cloud, which yield information of the entire sectional shape of a layer. The 1D/2D measurement system is not capable of doing this.

The Structured light, Stereo and 3D laser scanner systems are not capable of delivering real time data because of high software complexity and computationally intensive algorithms. Photogrammetry can deliver real time measurement data, but since high precision is required (+-1mm accuracy) many cameras are necessary which increases the complexity of the algorithms, makes real time data generation impossible and increases the costs of this system. Time of flight cameras, RGB-D cameras and a 1D distance measurement system is able to deliver real time data. RGB-D cameras like the Xbox Kinect are already used to deliver real time distance data for interactive computer games. However, the range of the Kinect is only 5m with a precision that is too low for this project. The time of flight system is the system that can produce real time measurements at a decent range with the highest precision. In a controlled and fixed setting, it can reach a precision of 4-6mm. At this stage of this project, this is enough to optimize the printing process. However, when the printing process becomes increasingly controlled, this accuracy will be too low to create usable measurement data to optimize the printing process. Aside from this precision problem, the cost of this system is too high. A measurement system that can deliver real time data by creating a 3D point cloud that meets the accuracy requirements is not achievable within this project. In other projects where less
accuracy is required, this would be possible. If in the future, with more research and budget, the accuracy of the 3D Time of Flight cameras is improved, this measurement system could become viable for in other similar projects. The measurement system that will be used for this project is the 1D/2D distance measurement system, since it meets all requirements. This measurement system will not be able to generate a 3D point cloud, but if a 2D measurement device is chosen that can generate a top-profile or the system is expanded with multiple measurement devices it can also generate, beside data regarding the height of a layer, data about the width and top profile of a layer.

As mentioned before, the aim of this project is proving the concept of measuring and optimizing, not the development of an optimal measurement setup. Therefore, the choice has been made to use The Sparkfun ToF Range Finder Sensor – VL6180. Information about the performance of the sensor, how it needs to be connected and example scripts can be found on (SparkFun Electronics, 2017)

The relatively simple Sparkfun ToF sensor uses Time of Flight technology to deliver 1D measurements. This sensor is often used in combination with the Arduino microcontroller. Examples of multiple working setups of this sensor in combination with an Arduino are available online (SparkFun Electronics, 2017) and show that this sensor is able to perform measurements real time, with an accuracy of approximately 1mm and a range up to 200mm. A maximum range of 200mm is sufficient to cover the distance between the measurement device and the concrete layers. Due to the ToF (Time of Flight) technology used, this sensor is able to deliver measurement data in different lighting conditions, on different types of surfaces including an uneven concrete surface. This sensor is not able to generate profile data of the entire width of a layer, but this profile data is unnecessary at this stage of the project. With very low costs of only 25 euros, this sensor is very well suited for this project.

### 3.2 Placement of the measurement device

The goal of this project is not to develop the perfect measurement system setup, but to develop a measurement setup that can be used to create a real time optimization. For this reason, a choice has been made for a measurement setup that is rather simple and can be built and used within the time frame of this project. Three concept designs for a measurement system are shown in Figure 9. All three of those concepts were developed with the requirement that measurements can be generated from the layer at all times, even when moving around corners. Why measurements with a simple fixed 1D measurement device can cause incorrect measurements while moving around corners is explained this section.
Figure 9: Three concept designs for a measurement setup

The first two concepts cannot be used since the choice has been made to use a simple 1D sensor. The first two concepts use a more sophisticated measurement device which can either create measurements within an angular range (option 1) or can create measurements along a linear line (option 2). The third option uses a simple 1D sensor, which can either rotate around the axis parallel to the printed layer or move alongside a linear line that is perpendicular to the axis of the printed layer. During the project the decision is made that a rotating or moving components are too complicated in this stage of the project and therefore a simpler setup should be used.

A new and simple concept is developed and is shown Figure 10. The 1D measurement sensor is either placed directly in front, or behind the nozzle. Then the sensor is aimed vertically at the centre of the printed layer. In this setup no top-profile information can be generated, but the height is only measured at one specific point along the width of the layer. The height along the entire width of the layer is, in this case, assumed to be similar to the measured height. In the future the setup can be extended and improved with more sophisticated sensors that can deliver top-profile data.

There are multiple options to position the measurement sensor on the nozzle. The most important decision is whether the sensor is placed in front of the nozzle (option A) or behind the nozzle (Option B). Both of those options are visualized in Figure 10. These options create different possibilities regarding the processing
and storage of the data and the possibilities to either directly respond to measurements or to respond based on stored data.

If option B is used, the height of a layer is measured directly after it has been placed. A major advantage of using this option is that the entire layer of the print path needs to be completed before the nozzle returns to the point where the next layer will be placed and where adjustments to the variables are needed. This means there is more time to prepare and perform calculations for this response. However, there are also multiple downsides of option B compared to option A. The goal of this project is to be able to respond real time to performed measurements. The advantage of having more response time in option B is also a disadvantage. The measurement data needs to be stored and linked to specific coordinates of the position of the printer. This requires a lot of data storage on the SINUMERIK machine and creates more complex synchronized actions, which need to be linked to the data storage in order to be able to respond to the measurements by adjusting variables. There is another disadvantage to option B that can cause inaccuracies in the generated measurement data: With option B there is a relatively large time frame between the measurement and the response at certain coordinates. Measurements are performed on the concrete that has just been printed. This means that this concrete is still in an early stage of curing and is still relatively wet. Even though ‘zero-slump’* concrete is used, in this early stage of curing, there is still a chance of minor deformations in the recently printed layer. This means that when the printer returns to the original coordinates, deformations will have taken place and the measurement can differ from the actual situation. This can be further amplified by the fact that not only deformations can take place in the previously printed layer, but as a result of the increasing weight, the underlying layers can deform during the time until the next layer is printed as well. Since the goal of this project is to respond to these deformations, it is important that those deformations are known as accurately as possible at the time of the actual response. Another advantage of option A is that no previously generated data is necessary to be able to respond to a situation. This creates the possibility to, for example, print directly on a previously printed element, an uneven surface or on a curved mould about which no data is yet available. The SINUMERIK is used to create the response to measurements by using synchronized actions. In deliberation with an employee from SIEMENS, the producer of the SINUMERIK, the conclusion was drawn that the SINUMERIK machine is best suitable for creating direct responses based on data gathered in real time. Therefore using the positioning of the measurement device of option A, in front of the nozzle, would give the biggest chance to respond to measurements in real time and being able to prove the concept of this project. This led to the choice to use option A and place the measurement device in front of the nozzle.

The chosen 1D measurement setup can cause incorrect data when the printer is moving around corners. This is visualized in Figure 11. Two different scenarios are shown with a different Distance between measurement device and nozzle (D). In the scenario depicted in the left-hand side of Figure 11 the D is smaller than in the right scenario. When the printer is moving around the corner the measurement device does not stay exactly above the centre of the printed layer. In the left scenario the measurement line stays within the boundary of the printed layer, so no incorrect measurement data is generated. In the right scenario D is bigger which causes the
measurement line to exceed the width boundaries of the printed layer. This means that incorrect height data is generated at certain points of the corner. For this reason, the distance between the nozzle and the measurement device, should be kept as small as possible in the design of the measurement system.

Figure 11: Incorrect measurements when moving around corners
3.3 Measurement system setup

In order to control the measurement sensor and to be able to process and transfer the data a microcontroller is needed. This microcontroller is a small computer on a single integrated circuit. These computers are often used in wireless interactive objects. The type of microcontroller used for this project is the Arduino Uno. Arduino uses open-source hard- and software which means that example scripts and hardware connections can be found online for free. Since Arduino is extensively used worldwide a lot of example projects with connections and used scripts are available. This creates the possibility to develop a working measurement system within the time frame of this graduation project, without the need for extensive knowledge about mechanical engineering and electronics. The Arduino forms the base of the measurement system. All other hardware parts, like the Sparkfun ToF Distance sensor, are selected based on whether they are usable within the Arduino system. More information about can be found in (Arduino AG, 2017). The design of the complete system that will perform measurements and transfer the data to the SINUMERIK NC can be seen in Figure 12.

![Figure 12: Measurement system setup](image-url)
The system can be split up in two separate parts. The part that will be connected to the nozzle of the printer (Part 1) and the part that will not be connected to the moving parts of the printer, but which will form the connection to the SINUMERIK machine (Part 2). The system is split up in two parts because the part connected to the nozzle needs to be wireless. The reason that this part needs to be wireless, see Figure 13.

The nozzle of the printer and the connection of the concrete input hose are designed in such a way that the nozzle is able to rotate infinitely around the Z-axis, while the concrete input part (blue component) can remain at a fixed position. This makes sure that the concrete input hose does not entangle around the printer when the nozzle is rotating. In order to maintain this freedom of rotation the measurement system (red component), which needs to positioned underneath the concrete input hose and connected on the rotating nozzle, needs to be connected wirelessly so the wires will not entangle around printer when rotating.

A second Arduino is used to be able to receive the wireless data and transfer it to the SINUMERIK machine. The wireless connection between the Arduinos is created with two 2.4 G Wireless nRF24L01p transceivers (Nordic Semiconductor ASA, 2006). These transceivers transport data through a radio signal on 2.4 GHz bandwidth. With a successfully tested range of up to 100m, these wireless modules have enough range to transfer data across the entire concrete printer field. An advantage of these transceivers is that it is possible to use multiple transceivers to send data to one single receiving transceiver. Because of this, when in the future measurements are performed on multiple output parameters, this data can be transferred to the one receiving transceiver which is connected to the second Arduino. This second Arduino will then send the data to the SINUMERIK NC. This creates a flexible system, which can easily be expanded upon in the future.

The data needs to be transferred to the Siemens PLC, to be able to use the data in the SINUMERIK software. Data can be transferred to the PLC with an Ethernet connection. In order to make the second Arduino capable of transferring data through an Ethernet connection to the Siemens PLC, an Ethernet shield is connected on top of the Arduino. A connection to the PLC could also be made with a PROFIBUS (Process Field Bus) connection, but an Ethernet connection is faster which increases the possibility of transferring and processing data in real time (Nardella, settimino.sourceforge.net, 2017). The PLC uses the data format Little Endian while the raw measurement data in the Arduino has the Big endian format. In order to transform the data from Big Endian to Little Endian and to transfer the data from the Arduino to the PLC, the Settimino library is used. This library is developed by Nardella. More information about how to

*Zero slump concrete is concrete which has no almost no deformations in the hardening process
Transfer data from an Arduino to a Siemens PLC can be found on his website (Nardella, settimino.sourceforge.net, 2017). The developed scripts of both Arduinos are added and explained in Appendix A2 and Appendix A3.

During printing, it is very convenient to be able to see the distance from the nozzle to the previous layer. For this reason, an LCD-display was added to the first Arduino. The schematics of the connections between the different components of the measurement system can be found in Appendix A4 and Appendix A5.
3.4 Developed measurement system

A picture of part 1 of the measurement system that is connected to the nozzle is displayed in Figure 14. This figure also shows that this part of the measurement system is split into two separate parts, with a detachable connection. The ToF distance sensor is separated from the other electrical components. This choice is made for three reasons. First of all because the space on the actual nozzle is limited. In this setup, the distance sensor is the only element that needs to be directly connected to the nozzle. The other elements are placed in a box that is positioned directly underneath the concrete input hose. An on/off-button is located on the outside of the box. This button is connected between the system and a 9V battery, to power the system and the Arduino itself. This button makes it easy to turn the device on and off without the need to open the box. The LCD display shows the measurement value on the outside of the box. The second reason to split this system into two parts is that it separates all electronic parts in the box from the wet part, which is at end of the nozzle where the concrete is extruded. This reduces the chance of errors caused by moisture and concrete getting in the measurement system. The third reason to split the system is to create the possibility to attach a different type of sensor or to change the position of the sensor without the need for adjustments to the box. Only the mount for the actual sensor needs to be adjusted.

Figure 14: The first part of the developed measurement system which is connected to the nozzle
A picture of the second part of the measurement system is displayed in Figure 15. The Ethernet Shield is connected directly on top of the Arduino. From the Ethernet Shield, a connection is made to the SIEMENS PLC through an Ethernet cable (grey cable). The NRF wireless transceiver is connected through the pins of the Ethernet Shield to the Arduino below. The entire module is powered through a USB cable (black cable). No LCD-display is added to this second part, since the measurement data can be directly seen from the SINUMERIK operating machine.

![Figure 15: The second part of the developed measurement system that transfers the data to the SINUMERIK NC](image)

The final step is to transfer the data from the Siemens PLC to the Numerical control (NC) of the SINUMERIK. In this step the measurement data in the PLC is assigned to a Data Block (DB). A DB is a sequence of bytes or bits with a maximum length, which can be used to store data. In this project DBs are used to store data values from the PLC. The DBs can be accessed directly from the NC. This creates the possibility to transfer data from the PLC to the NC. Specific DBs are available for different data types. The measurement data transferred from the second Arduino to the PLC is BYTE data. Because of this the DB-type DBB is used. Multiple DBBs are available in the SINUMERIK, but the first DBB (DBB0) is used to store the height measurement data. In the future, when more real time measurement data is required, other DBBs can be used to store extra data and transfer it to the SINUMERIK NC. More information about Data Blocks can be found in (Siemens, 2011).

In order to write the measurement data from the PLC to the DBB, programming needs to be performed directly in the PLC using STEP 7. STEP 7 is a standard software package used for the programming of SIMATICS PLCs, which is the PLC used in the SINUMERIK operating machine. Programming directly in the PLC is complicated and requires extensive knowledge about STEP 7 software. Therefore, a Siemens employee programmed the connection between the measurement data from the Arduino and the DBB. More information about programming with STEP 7 can be found in (Siemens, 2006).
3.5 Performance of the measurement system

The developed measurement system is able to perform measurements and wirelessly transfer the measurement data to the SINUMERIK NC. The transfer of the data is never instantaneous. A small delay is caused by the elements in the developed measurement system through which the data needs to travel. It appeared to be hard to determine the exact duration of the delay. However, based on experience gained during this research by working with the measurement data, the conclusion is drawn that the delay does not greatly affect the ability to create a real time optimization and will therefore be neglected. To not further complicate the report, the term “real time” is used to describe the data transfer time. In the future, when this research topic is beyond the stage of a proof of concept, it is important to know the delay and incorporate it within the responses. The range of the measurement system is sufficient to successfully transfer data from the entire print bed to the second Arduino. One battery can supply enough power to the measurement system to perform and send measurements for multiple hours. The data is transferred with a rate of 50 measurements per second, which is a sufficient transfer rate to perform a real time optimization. Beside all the positive points, there is one area where aspects for which the sensor does not meet its expectations. While measurements are displayed at an accuracy of 1mm, noise is present in the measurements. When the sensor is placed at a fixed distance from an object, the measurement values fluctuate up and down with a range of up to 5mm constantly. Using a more accurate sensor can improve the performance of the real time optimizations developed in this research. Because of the limited time-frame of this research, the decision is made to use the current measurement setup rather than spending a lot of time on redeveloping the setup.
4 Logging and analyzing of variable data

As mentioned in the introduction logging and analyzing variable data is very important to gain more insight and knowledge of the 3DCP-process. For this reason, a method is developed to log and analyze variable data. The following requirements for the developed method are defined:

1. The method needs to be usable in combination with the current 3DC-printer setup
2. The method should not require many extra proceedings during printing, which will complicate the 3DCP-process.
3. The method should be flexible in which additional logged variables and external measurement data can be imported
4. The methods require a 3D visualization of the printed element in which information can be displayed
5. In this 3D visualization variable data needs to be derivable at specific coordinates on the print path

The goal of the 3DCP research group is to further develop the current 3DCP-printer setup as a stand-alone system. Therefore, it is desired that the developed logging method can be used in combination with the current 3DC-printer setup, without additional required hard- or software to retrieve and store the variable data. The current process of 3DC-printing requires multiple operators working simultaneously to perform all required operations. For this reason, it is important to not further complicate the process with many extra proceedings during printing.

The third requirement states that the method should be flexible. The developed method needs to be usable for a variety of users and to remain usable in the future. Certain variable information might be important in this stage of the project or to a specific user. In the future, or to other users, different information might be required. Additionally, some of this variable data might not be gathered through the operating system, but through an external device. For these reasons, the method needs to be flexible, extendable and able to import external measurement data. Analyzing the logged data in a 2D graph makes it difficult to directly relate the logged data results to a specific location of the printed element. For this reason, it is important to create a 3D visualization of the printed element from which variable information can be derived at specific locations on the element. This allows for very effective and efficient analysis of the data.
Figure 16 shows a visualization of the developed method for logging and analyzing of variable data. The method consists of six main steps. The first two steps are general steps of the printing process used in this research but are also important in the logging process:

1. Gathering measurement data and transferring it to the operating machine.
2. Executing the G-code print path.

The next four steps are specifically developed to be able to log and analyze variable data. Within these steps, two different methods are provided to analyze the data. Using a spreadsheet (step 4) or using Grasshopper and Rhino (step 5&6)

3. Perform the trace function and export data
4. Import data in excel and analyze the data using graphs
5. Import and convert traced/external data and adjust display settings
6. View and analyze the data using the 3D visualization
4.1 Step 1 - Perform measurements and transfer data
In this step, the measurements are performed, the data is gathered and is transferred to the operating machine as described in chapter 3. How the measurement system is turned on and connected to the operating machine is explained in Appendix A6.

4.2 Step 2 - Executing the G-code print path
The G-code print path is loaded into the operating machine and can be executed, which results in the movement of the gantry robot and the flow of concrete by turning on the concrete pump. However, the actual program cycle is not started until the trace settings are correct and the trace has been started (Step 3).

4.3 Step 3 - Perform the trace function and export data
Within the SINUMERIK software of the operating machine, there is a specific function that can be used to log variable data. This function is called the “Trace-function” (Siemens AG, 2002). Using this function makes this logging method easily applicable for our 3DCP-process and does not require additional hard and software. Using this build-in function does not increase the difficulty of the already complex print process. Performing the Trace function consists of a few sub-steps. It starts with the trace settings. These settings consist of: which variables should be traced, the name of the variables, at which events and at what interval the trace needs to be performed and lastly the start and stop trigger. Setting an automated start and stop trigger creates the possibility to start the logging of data exactly at a desired coordinate or event. Using this automated trigger reduces the extra proceedings being added to the current printing process. When the trace is performed, the data gathered during the trace needs to be stored to an XML file to be further analyzed. The X-, Y- and Z-coordinate variables always need to be traced to be able to create a 3D visualization of the printed element later on. A more detailed explanation of how to use the Trace function on the SINUMERIK can be found in Appendix B1.

There is a limitation in using the Trace function. During a trace, which regularly collects data during a time-frame, a lot of data is gathered. The operating machine does not have enough computational power to properly perform a long lasting trace. When tracing for over 5 minutes the operating machine will get stuck and sometimes even crash. Adjusting the storage interval reduces the amount of data gathered and therefore might increase the maximum trace time. However, this is not possible with this trace function. For this research project, a lot of tests are performed in which many traces are performed with a time-frame of less than 1 minute. For these tests the Trace function works perfectly. However, when used to log data during a longer print session this method becomes unusable. In order to be able to perform a long lasting trace an offline software package can be used. The software package is developed by Siemens and is called SinuCom Trace. This software package can be installed on a secondary computer from which a live connection is made to the SINUMERIK operating machine using an Ethernet cable. Performing a trace using this software package works in similar way as the Trace function available on the SINUMERIK. The goal is to develop a method without using external hardware. However, due to the limitations of the trace function, the decision is made to use a secondary computer to create the possibility to perform a long lasting trace. A brief explanation how to perform a trace using the SinuCom Trace software package is available in Appendix B2.
4.4 Step 4 – Import data in excel and analyze data using graphs

Besides the method developed to analyse the logged data in a 3D visualization (step 5&6), a method is also developed to analyse the data using a spreadsheet application. Using the XML-import function, a baseline excel file has been created to be able to easily import a XML-trace file. The setup is able to automatically organize the data of up to 16 logged variables in separate columns. This allows the user to easily scroll through the logged data in a table (Table 2), but most importantly it allows the user to view the variable data in a 2D graph. Individual trace files can be analysed by viewing multiple variables in a Time vs. Variable value graph (Figure 17). Furthermore, multiple excel files, created from logged XML-files, can be combined to be able to compare the results logged data of multiple printed elements or performed tests. The developed XML-import method is used extensively within this research to compare the results of multiple tests performed within chapter 5 and 6 of this report. A more detailed explanation how to import an XML file in the baseline excel file can be found Appendix B3. Creating a baseline file which can automatically organize up to 16 variables regardless of the type of selected variables and their names makes this method very flexible and easy to use.

<table>
<thead>
<tr>
<th>time</th>
<th>Required Z-offset</th>
<th>X-axis</th>
<th>Y-axis</th>
<th>Measurement value</th>
<th>Z-axis</th>
<th>C-position</th>
<th>Path Velocity</th>
<th>Program runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>4899.89</td>
<td>926</td>
<td>99</td>
<td>-1580.24262</td>
<td>180</td>
<td>5000</td>
<td>3.552</td>
</tr>
<tr>
<td>0.008</td>
<td>1</td>
<td>4899.223</td>
<td>926</td>
<td>101</td>
<td>-1579.93597</td>
<td>180</td>
<td>5000</td>
<td>3.56</td>
</tr>
<tr>
<td>0.014</td>
<td>1</td>
<td>4898.723</td>
<td>926</td>
<td>101</td>
<td>-1579.71819</td>
<td>180</td>
<td>5000</td>
<td>3.566</td>
</tr>
<tr>
<td>0.02</td>
<td>1</td>
<td>4898.223</td>
<td>926</td>
<td>101</td>
<td>-1579.51326</td>
<td>180</td>
<td>5000</td>
<td>3.572</td>
</tr>
<tr>
<td>0.028</td>
<td>1</td>
<td>4897.556</td>
<td>926</td>
<td>101</td>
<td>-1579.26377</td>
<td>180</td>
<td>5000</td>
<td>3.58</td>
</tr>
<tr>
<td>0.036</td>
<td>4</td>
<td>4896.89</td>
<td>926</td>
<td>104</td>
<td>-1579.0461</td>
<td>180</td>
<td>5000</td>
<td>3.588</td>
</tr>
<tr>
<td>0.042</td>
<td>4</td>
<td>4896.39</td>
<td>926</td>
<td>104</td>
<td>-1578.90681</td>
<td>180</td>
<td>5000</td>
<td>3.594</td>
</tr>
<tr>
<td>0.05</td>
<td>4</td>
<td>4895.723</td>
<td>926</td>
<td>104</td>
<td>-1578.75592</td>
<td>180</td>
<td>5000</td>
<td>3.602</td>
</tr>
<tr>
<td>0.056</td>
<td>4</td>
<td>4895.223</td>
<td>926</td>
<td>104</td>
<td>-1578.66713</td>
<td>180</td>
<td>5000</td>
<td>3.608</td>
</tr>
<tr>
<td>0.064</td>
<td>2</td>
<td>4894.556</td>
<td>926</td>
<td>102</td>
<td>-1578.57714</td>
<td>180</td>
<td>5000</td>
<td>3.616</td>
</tr>
<tr>
<td>0.07</td>
<td>2</td>
<td>4894.056</td>
<td>926</td>
<td>102</td>
<td>-1578.52815</td>
<td>180</td>
<td>5000</td>
<td>3.622</td>
</tr>
<tr>
<td>0.078</td>
<td>2</td>
<td>4893.39</td>
<td>926</td>
<td>102</td>
<td>-1578.48338</td>
<td>180</td>
<td>5000</td>
<td>3.63</td>
</tr>
<tr>
<td>0.084</td>
<td>4</td>
<td>4892.89</td>
<td>926</td>
<td>104</td>
<td>-1578.46242</td>
<td>180</td>
<td>5000</td>
<td>3.636</td>
</tr>
<tr>
<td>0.092</td>
<td>4</td>
<td>4892.223</td>
<td>926</td>
<td>104</td>
<td>-1578.44719</td>
<td>180</td>
<td>5000</td>
<td>3.644</td>
</tr>
<tr>
<td>0.098</td>
<td>4</td>
<td>4891.723</td>
<td>926</td>
<td>104</td>
<td>-1578.4425</td>
<td>180</td>
<td>5000</td>
<td>3.65</td>
</tr>
<tr>
<td>0.104</td>
<td>4</td>
<td>4891.223</td>
<td>926</td>
<td>104</td>
<td>-1578.44119</td>
<td>180</td>
<td>5000</td>
<td>3.656</td>
</tr>
<tr>
<td>0.112</td>
<td>4</td>
<td>4890.556</td>
<td>926</td>
<td>104</td>
<td>-1578.44124</td>
<td>180</td>
<td>5000</td>
<td>3.664</td>
</tr>
<tr>
<td>0.12</td>
<td>4</td>
<td>4889.89</td>
<td>926</td>
<td>104</td>
<td>-1578.44401</td>
<td>180</td>
<td>5000</td>
<td>3.672</td>
</tr>
<tr>
<td>0.126</td>
<td>4</td>
<td>4889.39</td>
<td>926</td>
<td>104</td>
<td>-1578.45085</td>
<td>180</td>
<td>5000</td>
<td>3.678</td>
</tr>
<tr>
<td>0.132</td>
<td>4</td>
<td>4888.89</td>
<td>926</td>
<td>104</td>
<td>-1578.46417</td>
<td>180</td>
<td>5000</td>
<td>3.684</td>
</tr>
<tr>
<td>0.14</td>
<td>4</td>
<td>4888.223</td>
<td>926</td>
<td>104</td>
<td>-1578.49574</td>
<td>180</td>
<td>5000</td>
<td>3.692</td>
</tr>
<tr>
<td>0.148</td>
<td>2</td>
<td>4887.556</td>
<td>926</td>
<td>102</td>
<td>-1578.5478</td>
<td>180</td>
<td>5000</td>
<td>3.7</td>
</tr>
</tbody>
</table>

Table 2: Example spreadsheet by importing an XML-trace file into Excel
Figure 17: Example Time vs Variable value graph created from the imported XML-trace file
4.5 Step 5 – Import and convert traced/external data and adjust display settings

Step 5 and 6 are used and performed simultaneously. In step 5, traced and externally generated data is imported and converted using Grasshopper. The converted data is then used to automatically create the 3D visualization of the printed element in Rhino, with variable data projected on this visualization. The data can then be analysed in step 6 using the created 3D visualization. An overview of the developed Grasshopper model is shown in Figure 18. The model consists of seven main blocks, which are numbered 1 to 7 in the overview. The red blocks are blocks that the user needs to manually adjust. The purple blocks automatically operate using the manually created settings and supplied data to create a 3D visualization. The grasshopper model is developed in such a way, that a varying amount of variables, regardless of their type and name, as well as extra external data can be displayed without any required adjustments to the model. This creates a flexible model, which is future proof and can easily be used by users with different requirements. What the different blocks of the model do is briefly explained in the following paragraphs. A more detailed explanation about each block and how a user should use the grasshopper model is available in Appendix B4.

Figure 18: Overview of the developed Grasshopper model to create a 3D visualization
In the first block “Data input”, the user needs to insert the XML-data files. The main XML-trace file is added and a secondary external data file can be optionally added. The time-interval with which the external data is logged might be different from the main XML-trace file. Therefore, two insert possibilities have been created. One for when the interval of the external data file is equal or bigger compared to the main XML-trace file, one for when the interval of the external data file is smaller.

In the second block “Convert and combine all data”, a python script is created that extracts, converts and combines all required data from the XML-data files. After the required data is extracted, the external data files are converted to synchronize the time-interval with the time-interval of the main XML-trace file. All the data is then combined and the amount of points (number of intervals on which data is logged), the available measured variable names and the new combined data are exported.

In the third block “Display settings”, the user needs to choose two different display settings. First, the user needs to select a point on the 3D visualization of which all variable information is displayed. This is done by using a “Radial Control Knob” to scroll through all available points until the desired point is selected. The 3D visualization is adjusted in real time when using the knob to scroll through all available points. In the second display setting, the user needs to choose one measured variable of which the values will be projected on the 3D visualization using a black and white gradient.

In the fourth block “Transform data for display”, the combined data from the second block is first converted to be able to display all variable information at the selected point. This converted data is then exported. Besides this, the logged X-, Y- and Z-coordinates are used to create a single 3D point for each interval on which data is logged. All the created 3D points are exported and will be used in block 5 to create the 3D visualization. Lastly, the data of the selected measured value, which will be displayed in a black and white gradient, is exported.

In the fifth block “Display 3D visualization and variable color gradient”, the actual 3D visualization is created. This is done by displaying all the created 3D points exported from block 4. Using the exported data of the selected measurement created in the fourth block a color is assigned to each of the 3D points. The point with the lowest variable value is white and the point with the highest variable value is black. All 3D points with an intermediate variable value receive a grey-scale based on their variable value. This creates a gradient over the entire 3D visualization based on the variable value. This allows the user to have an overview of the differences in the values of one specific measured variable value by projecting it on the 3D visualization.

In the sixth block “Display marker”, a red spherical marker is displayed at the location of the selected point. This allows the user to see the location of the selected point from block 3.
In the seventh block “Display all variable values at marker”, the value of each of the measured variables on the selected point is extracted. All of those extracted values including the variable name are then displayed in Rhino. This allows the user to see the value of each of the measured variables at the selected point (which is marked by the red spherical marker).

4.6 Step 6 – View and analyze the data using the 3D visualization

In this step, Rhino is used to analyze the logged data in the 3D visualization created in the previous step. Adjustments to Block 3 “Display Settings”, in the grasshopper model are directly displayed in the 3D visualization in Rhino and the two steps are therefore executed simultaneously. Figure 20 shows an example in which the developed method is used to analyze the data from a XML-trace file. Three different windows are shown in this figure. The center window shows the 3D visualization of the printed element. The top left window shows all logged variables with their value at the selected point. The window on the right shows the display settings box from the grasshopper model. All available measured variables are automatically displayed in the display settings. From the available measured values, one can be selected to be displayed as a black and white gradient on the 3D visualization by entering the number of a specific measured variable. In this example, value of “1” is entered which refers to the Path Velocity variable. Point 1730 is selected and the location of this point is shown by the red spherical marker. The value of the variables at that location is shown in the top-left window. The temperature variable was not logged using the trace function, but was added as an external data file with a different measurement interval. The interval is automatically synchronized with the interval of the main XML-trace file.

Figure 19: Example 3D visualization with the Path Velocity variable projected as a black and white gradient, the path velocity differs a lot throughout one layer, but is similar at the same X- and Y-coordinates of each layer.
The 3D visualization allows for direct comparison between the data and the printed element. The slider creates the possibility to scroll through the entire element and view the variable information at any point. The black and white gradient creates an instant overview of one variable projected on the entire element. In the example shown in Figure 20, the path velocity differs a lot throughout one layer, but is similar at the same X- and Y-coordinates of each layer.

Overall, the developed method meets the requirements stated in this research project and provides a method to efficiently log and analyze variable data of the 3DCP-process.
5 Optimizing using real time adjustments

In this chapter the generated measurements are used to make the printer automatically adjust variable values to optimize the output. The output parameter measured in this project is the distance from the nozzle to the previous layer, which can be used to derive the height of the layers. As explained in the Research Design chapter, the actual layer height can vary from the expected layer height or the total height of multiple layers can change because of the weight of the concrete. This can lead to undesired results or even lead to an interruption of the printing process. Therefore, adjustments to printer variables based on the height measurements will be made, to optimize the printing process. In this chapter, this optimization is further elaborated.

5.1 Printer variables for optimization

As explained in the introduction of this report, there are multiple printer variables that influence the layer height. Not every variable can be adjusted directly from the SINUMERIK machine. Only the variables that can directly be adjusted from the SINUMERIK can be used in this project to create a real time optimization because the responses will be preprogrammed in the SINUMERIK machine. For this reason, only variables that can be regulated and adjusted by the operating machine itself can be used. There are over 100 variables which meet this criterium. From these, a selection has been made that will be used in this research to optimize the printing process. In the future, the amount of variables used for optimization can be extended. The first selection of printer variables for this research are:

1. The coordinates of the printer
2. The movement speed of the printer
3. The pump-frequency of the concrete pump.

If and how each of those variables can be used to optimize the printing process will now be further elaborated upon in the following sections.

5.1.1 The coordinates of the printer

During the printing process, the printer follows a predefined print path. This print path is saved in a G-code file, inserted into and then executed by the SINUMERIK. The printer moves from a certain X,Y,Z,C-coordinate to the next X,Y,Z,C-coordinate. The C-value is the rotation in degrees along the vertical Z-axis. During printing no actual adjustments can be made to the print path, but an offset from the print path can be created for each of the axes. The Z-coordinate of the printer directly relates to the distance between the nozzle and the previously printed layer. When the actual layer height differs from the expected layer height, the difference between the desired distance (DD) and actual distance (AD) from the nozzle to the previously printed layer increases with the amount of layers printed. In Figure 20, an example is shown in which the actual layer height is smaller than the expected layer height. With the first few layers this difference will be small, but when more layers are printed this difference will become substantial and will directly affect the results of the printing process. By creating an offset from the Z-axis based on the generated height measurements, the distance between the nozzle and the previously printed layer is kept at a desired distance. This means that the nozzle will always be at the desired distance to the previous layer.
5.1.2 The movement speed of the printer

The movement speed of the printer (the print speed) along the print path is a variable that needs to be defined in the print path script inserted into the operating system. The movement speed does not need to be constant during the entire print path, but different speeds at different sections of the print path can be pre-programmed. Similar to the coordinate-variables of the printer, the print speed cannot be directly adjusted, but an offset can be given to the print speed variable. As shown in Figure 6 the print speed directly affects the sectional shape of the layer and thus the height of a layer. This is due to the fact that the same amount of concrete per time-unit is printed, but since the movement speed of the printer is adjusted, more or less concrete is deposited along the print path. Therefore, the measurements performed on the height of a layer can be used to alter the print speed variable in real time with the goal to achieve the desired layer height.

5.1.3 The pump-frequency of the concrete pump

Similar to the print speed and the coordinates of the printer, the pump-frequency of the concrete pump is a variable that is pre-programmed in the print path script. This means that alterations to this variable can be achieved directly from the operating system. Increasing the pump-frequency results in a bigger pressure with which the concrete is pumped through the hose to the nozzle of the printer. This results in a speed increase of the concrete flow and thus the amount of concrete printed per time-unit is increased. No specific tests have been performed that show that the pump-frequency directly affects the height of a layer, but based on the test performed in Figure 6, adjusting the pump-frequency will most likely affect the layer height parameter. Pump-frequency differs from the two other variables: Coordinates of the printer and The print speed, in the sense that increasing the pump-frequency will most likely not instantly affect the layer height, but will do so with a delay. A long hose is used to transport the material from the concrete pump to the nozzle of the printer. When the pump-frequency is adjusted the pressure at the concrete pump is instantly increased or decreased, but it takes more time to adjust the pressure along the entire length of the hose, thus at the end of the
nozzle. Therefore, adjusting the pump-frequency will not instantly affect the layer height, but will most likely gradually do so over time.

5.1.4 Selected variable for optimization
In this research only one of the three variables is selected for real time optimization to create a focus point.

The pump-frequency variable is not likely to instantly affect the layer height as a result of the delayed effect of the pressure increase. The real time optimization will take place directly within the SINUMERIK operating machine and this machine can be best used to instantly respond to a specific measurement. The delayed effect of adjusting pump-frequency requires more complicated real time responses and will make it impossible to instantly respond to a certain measurement and effectively achieve the desired output. For this reason, this variable is not used within for the real time optimization.

The choice is made to focus on adjusting the coordinates of the printer and not on the print speed, since this adjusting the coordinates of the printer is relatively easy to do real time during the printing process. Also previous experience has shown that having a fixed distance from the nozzle to the previous layer is very important in this stage of this project to be able to control the output of the printer.

5.2 Real time optimization method
As explained in chapter 1.5, the choice has been made to create real time responses directly from the NC of the SINUMERIK operating machine and no “third generic box” is used to perform calculations. In order to achieve the real time adjustments to the printer variables, synchronized actions are used. It is important that the real time optimizations can be used on different print paths without extensive programming in each of the print path scripts. This is not only important for implementation of this method in the future, but also to be able to efficiently perform different tests within this research. Therefore, a method is developed in which real time optimization modules are created. These modules can be applied to any print path without the need of programming these responses within the actual print path script. How synchronized actions work and how the real time optimizations modules are created is further elaborated in the following paragraphs.

5.2.1 Synchronized actions
In order to achieve the real time optimization, the SINUMERIK operating system needs to be able to react to measurement data while running. This is achieved by using synchronized actions, which are directly programmed in G-code. The structure of a G-Code program is build up in different program levels. The main program level is referred to as level 0. Different subprograms, which are programmed on other program levels, can be used from the main program. The Numerical Control Kernel (NCK) executes those part programs. This is a component of the NC that coordinates the motion operations for the machine tool. Synchronized actions are instructions programmed by the user. Those instructions are evaluated in each interpolation cycle of the NCK. Conditions can be programmed in each synchronized action. If those conditions are met, then actions assigned to the instructions are activated in
synchrony with the remainder of the part program run. Those conditions may depend on and involve real time events and values (Figure 21)

Figure 21: Schematic diagram of synchronized actions (SIEMENS AG, 2011)

In this research the real time events and values are the measured distance values. This data can be used to create conditions, test those conditions and make alterations to the input. Aside from the measurement data, values of system variables, the coordination of the 3D printer, are required to optimize the input as well and will be altered from the G-code. Information about system variables combined with the measurement data create the ability to perform an automated real time optimization.

A synchronized action consists of seven components (Figure 22):

1. Identification number
2. Frequency
3. G-code for condition and action
4. Condition
5. Action code word (DO)
6. G-code for action
7. Action

Some of those components are optional. More detailed information about programming with synchronized actions can be found in (Siemens AG, 2011). An example of a simple synchronized action is shown in Figure 22.

Figure 22: Example of a synchronized action

In this example the optional components 3 and 6 are not used. These components define the settings for the evaluation (metric or inch dimension). When nothing is used, the settings of the
The entire program are used instead. The identification number (ID) of the synchronized action is 5 (1). This number should be unique, this synchronized action remains operative until a synchronized action with an identical ID is used or until it is cancelled. This synchronized action creates an offset to the Z-axis with -1mm (7) whenever (2) the variable $R_1 > 15$ (4). Synchronized actions can also be used without any frequency (2) and condition (4). This means that the synchronized action runs constantly and simultaneously to the script.

During this research it became clear that programming complex calculation and responses in G-code is possible, but also more complicated compared to most other programming languages. G-code has limited possibilities regarding complex calculations and actions since it has been developed specifically for the control of automated machine tools. The basis of G-code is for users to tell a computerized tool how to make something by telling the machine where to move, how fast to move and which path it should follow. Because of this the developments regarding the real time optimizations are difficult to achieve and are sometimes limited. The limitations are mostly related to the storage possibilities of data and impossibility to program looped actions within a synchronized action. The behavior of concrete material is very hard to predict and for this reason complex optimizations are required. When working with other materials, from which the properties and behavior can be better predicted, less complex optimizations will most likely be required. For optimizations regarding those materials the “simple” basis of G-code might be better suitable. More information about programming in G-code can be found in (Siemens AG, 2002)

5.2.2 Real time optimization modules
As mentioned in the introduction of this section it is important to create an efficient method for applying real time optimizations to different print path scripts. Therefore, a method is developed which used the MPF - SPF structure (Main Program File - Sub Program File structure), which uses the different program levels described earlier. This can be explained using a simplified visualization, shown in Figure 23. The MPF is a single file used for the print path script, the printer settings and the used synchronized action modules. The MPF is programmed on the main program level 0. Alongside this MPF file, multiple SPF-files can be created, each of which is a single real time optimization module. The SPF is programmed in a lower level. Within a single module, multiple synchronized actions can be programmed which together form the real time response module. Both MPF and SPF files need to be contained in one Workpiece Directory (WPD). By referring to a specific module (SPF-filename), within an MPF file this module will be activated automatically when the MPF is executed by the operating system. This makes it very easy to use modules on different print path scripts without extensive scripting in those MPF-files. In the future a library could be created and extended with more and improved real time optimization modules to optimize the printing process.

In Figure 23 two simplified example module are shown: BasicresponsemoduleOn and BasicresponsemoduleOff. The first modules activates the basic response module and activates two synchronized actions. The second one turns the basic response module off by canceling the two synchronized actions. This allow you to turn a module on and off for specific parts of the print path.
Figure 23: Real time optimization modules using the MPF-SPF structure
5.3 Basic optimization module

As explained in Figure 20 the printing process is optimized by real time adjusting the Z-coordinate of the printer based on the performed height measurements. The goal of this optimization is to keep the nozzle at a fixed distance from the previously printed layer. The basic module developed to perform this real time optimization can be seen in Figure 24. This module responds to the performed measurements by creating an offset to the Z-position of the printer until the nozzle is at the desired distance from the previously printed layer. Text in the module script which is placed behind “;” are comments which explain what the script does, but have no function when the module is executed. R-variables are variables within the operating machine which are used to store data and are referred to by scripting R[variable_number]. The script of the basic optimization module with more detailed explanation is added in Appendix C1.

The first two parts of the module “Desired distance settings” (red) and “Initializing values” (orange), are only performed once at the start of the printing process. The third part “Synchronized actions” (yellow) are performed repeatedly and simultaneously during the entire printing process in which this module is activated. How the basic optimization module works is explained using Figure 25.

![Figure 24: Basic optimization module](image)

![Figure 25: Visualization of the basic optimization module](image)
The module starts with setting the desired distance settings (red). The height difference between the measurement system and the nozzle (R25) and the desired distance from the nozzle to the previous layer (R26) need to be defined. Those values are used to calculate the total distance from the measurement device to the previous layer (R22).

In the next part the important variables are created and initialized (orange). The required offset (R16) is initialized at 0 and the previous measurement value. Since there is no previous measurement value at the start of the script, is initialized at 999.

The synchronized action part (yellow) consists of four separate synchronized actions (1 to 4). In the first synchronized action: ID=1, the R20 variable is updated with the latest measurement value (A_DBB[0]). In the second synchronized action: ID=2, the difference between the latest measurement value and the total desired distance (R21) is calculated. In the third synchronized action: ID=3, the required offset (R16) is calculated by taking the negative of R21 and the previous measurement value (R23) is updated with the latest measurement value. This third synchronized action is only performed whenever R20 is not equal to R23. This means that the required offset is only updated when a new and changed measurement value is updated. This synchronizes the frequency with which a new offset is calculated with the frequency with which new measurement values are imported. Otherwise, the required offset would constantly be updated for no reason, since no new measurement information is available yet. The final synchronized action: ID=4, performs an offset to the Z-axis equal to the required offset calculated in the previous synchronized actions. These synchronized actions are repeated during the entire print script and the required offset is constantly updated and applied to the Z-axis. This causes the printer to optimize the printing process in real time by automatically adjusting the Z-axis position to keep a fixed distance between the nozzle and the previously printed layer.
5.4 Testing the basic optimization module

5.4.1 Test setup

In order to determine the performance of the basic optimization module tests are performed in a fixed test setup. Three different test scenarios within this fixed setup are used which are shown in Figure 26. A picture of the test setup with the third scenario is shown in Figure 27.

![Figure 26: The three different test scenarios used to determine the performance of the real time optimization module](image)

![Figure 27: Photo of the test setup with the third scenario](image)

The printer moves in a straight line across the print bed in the X-direction for a distance of 1800mm with a fixed movement speed. There are no pre-programmed Z-coordinate movements. All Z-coordinate movements are based on the generated real time measurements and are performed by the basic optimization module. No actual concrete is printed, but the responses of the printer to the three different test scenarios are monitored and analyzed to determine the performance of the basic optimization module. The desired distance from the measurement device to the print bed (R22) is programmed at 100mm, which consists of 20mm height difference between the nozzle and the measurement device and 80mm distance from
the nozzle to the print bed. This high distance is used in order to safely perform the tests with minimal risk of crashing the printer on the print bed. When this real time optimization is brought to practice in the use-cases, the distance from the measurement device to the print bed will be reduced to approximately 30mm, which is the desired distance between the measurement sensor and the previous layer.

The first scenario, flat surface, is a flat surface without any obstacles. This test simulates the scenario when the real time optimization is used for printing on a previous layer that is almost flat. The first scenario also gives a baseline test to determine whether any measurement noise is present, since a completely flat surface is used.

The second scenario, transition surface, consists of three horizontal plateaus with different heights. Between the plateaus there are two slopes which create the transition from one plateau to the next. This test simulates the transition from one layer to the next layer of a 3D concrete printed object. Two different layer height transitions are simulated. With this test the capability of the real time optimization to perform a smooth transition from one layer to the next is tested.

The third scenario, 3D surface, consists of a flat surface with two trapezoid shapes with different heights, lengths and slope angles. Trapezoid shapes are used since their geometry can be easily translated into a 2D graph baseline as explained further on in this paragraph. This test simulates the scenario when the printer is used to directly print on a 3D surface without any pre-programmed Z-coordinate movements. Directly printing on a 3D surface is chosen as one of the final use-cases for this research. Therefore, this test has been developed to determine the performance of the real time optimization module towards directly printing on a 3D surface. It is important to keep in mind is the fact that the third scenario is a specific 3D surface. This means that two objects are created, which each have a specific slope and height. This means that the results of the tests to determine to performance of the real time optimization and the conclusions drawn from those tests are only applicable and transferable on 3D surfaces with similar elements and slopes. Different scenarios, with different slopes and element heights might yield different results and might lead to different conclusions and optimizations.

Every test is recorded on video and important variables are logged using the trace-function method developed in this project. The video recording combined with the logged results help effectively analyze the test results. It is important to be able to compare the results of different tests with each other. For this reason, the trace-function for each of the tests is started and ended at fixed X-coordinates. By starting each trace at the same X-coordinate, the time stamps of the traced information are instantly synchronized with the time-stamps of the other tests. The start and stop trigger settings are shown in 0. The most important variable used for analyzing the performance of the real time optimization is the Z-coordinate of the printer. The logged results are imported in excel and a graph of the different variable values plotted against the time stamps is used to analyze the results. The print path script (the Main Program File) is shown and explained in Appendix C3.
It is important to compare the movement of the printer in Z-direction with the actual Z-coordinates of the surface. In order to do this the Z-coordinates of the printer are logged using the trace-function and are displayed in a graph on the vertical axis. The time of the print session is displayed on the horizontal axis. For each of the three scenarios, a baseline graph is created which shows the Z-coordinates of the actual situation of the scenario’s. By comparing the z-movement of the printer with the baseline graph of the actual situation the performance of the optimization module is efficiently analyzed. The desired distance from the surface below in these tests is set at 100mm. This means that the Z-coordinates of the printer are 100mm above the surface below. For this reason, the Z-coordinates of the baseline graphs are increased with 100mm. This enables a direct comparison of the baseline graphs with the Z-coordinates of the printer. The baseline graphs of the actual situation for all three scenarios are shown in Figure 28.

![Figure 28: Baseline graph of the actual situation of the three test scenarios](image-url)
5.4.2 Test results

Figure 29, Figure 30 and Figure 31 show the Z-coordinate movement of the printer caused by the basic optimization module for each of the scenarios (red line). They are compared to the baseline graphs of each of the scenarios (blue line). These figures are used to determine the performance of the basic optimization module for each of the three scenarios.

Overall, the test results show that the printer responds to the performed measurements by altering the Z-coordinate of the printer in real time. This means that the concept of real time optimizing the printing process based on real time measurements is achieved and that this principle works. In the three different tests the printer responds to the three different scenarios in which the nozzle is moved up and aims to keep the nozzle at a desired distance from the surface below, in this case 100mm. In the ideal situation the red line would follow the blue line perfectly, which means that the distance from the nozzle to the surface below is constantly 100. However, in the results it becomes clear that this is not the case and that there is room for improvement to the basic optimization module.

In Figure 29, the printer responds to the first scenario, a flat surface. The printer moves up and down a lot while moving across the flat surface. It deviates from the baseline graph up to a distance of 5mm. The average height of the printer remains almost equal to the baseline graph, which means that the deviations of the printer do not increase and the nozzle is kept, with some deviations, at the desired distance from the flat surface.
In Figure 30 the printer responds to the second scenario, transition surface, which simulates a transition between layers. Again, similar to the previous test the printer keeps fluctuating up and down on the horizontal planes. Aside from this, the height of the printer nozzle is not increased enough to keep a fixed distance from the two horizontal planes. On the first horizontal plane (from 7.5 to 10s) the average height of the printer nozzle is around 4mm lower than the baseline graph. On the second horizontal plane (from 10.5 to 14.5s) the average height of the nozzle is around 10mm lower compared to the baseline graph. This means that when used during printing the nozzle would be too close to the previous printed layer and might even crash into the previous layer. The printer also starts moving up a fraction to early (at 7.5s and at 10s) and also starts moving down to soon (at 14.5s). At 14.5 the baseline graph is completely vertical while the printer moves down with a slope.
In Figure 31 the printer responds to the third scenario, the 3D surface. Similar to the previous test the printer fluctuates up and down on the horizontal surfaces. However, the average height of the printer at low horizontal surfaces is almost equal to the baseline graph. Even more so than in the previous test, the printer moves up and down too soon at both surface elements (at 6, 9, 11 and 14s). Again, like in the previous test, the height of the printer is not increased enough to keep a fixed distance from the two horizontal planes. In this test the difference between the average height of the printer and the horizontal planes is even larger than in the previous test. On the first horizontal plane (from 7 to 8s) the average height of the printer is around 30mm lower than the baseline graph. On the second horizontal plane (from 12 to 13s) the average height of the printer is around 20mm lower than the baseline graph. These tests seem to indicate that an increase in element height results in a larger difference between the printer nozzle and the baseline graph.

### 5.4.3 Analysis of the test results

A point for improvement is the that the printer fluctuates a lot in the Z-direction even when moving across a flat surface. Fluctuating up and down can cause problems during printing and can greatly affect the output. Why the printer is fluctuating can be explained using Figure 33, Figure 34 and Figure 35. In these figures the result of three extra tests are displayed. In these tests, the printer is moved across the three different scenarios without adjusting the Z-coordinates of the printer. The measurement values are logged using the trace-function, which creates a scan of the three different scenarios. Aside from these measurement values, the baseline values of each of the scenarios are added to the graph. In order to be able to directly compare the baseline graph with the measurement values, all measurement values transformed
through multiplying by -1 and then decreasing with 1479, which plots the measurement graph directly on the graph of logged Z-coordinate values. To explain this, the transformation of the scanned surface values of the third scenario is demonstrated in Figure 32. The measurement values get smaller when the height of the surface increases. This means that when the measurement values are plotted in a height-time graph an increase would be shown as a decrease. For this reason, the measurement values are multiplied by -1. Secondly, the starting Z-coordinates of the printer in the test setup are -1579 and the measured values start around 100, since this is the desired distance from the nozzle to the surface below. Therefore, after multiplication, 1479 is subtracted from the measured value.

Figure 32: Transformation of the raw measurement values
Figure 33: Baseline graph versus measurement scan test: scenario 1

Figure 34: Baseline graph versus measurement scan test: scenario 2
The results of the measurement tests of the three different scenarios all show that there is noise present in the measurement values. Even when the printer keeps a fixed distance from the surface below, the measured distance values differ up to 5mm from the actual distance. When the basic optimization module is used to optimize the printing process in real time, it will respond to those incorrect measurement values and this will cause the printer to fluctuate up and down.

The most obvious solution to solve this problem is to find a new and more accurate sensor with less noise. This will reduce the fluctuating caused by the basic real time optimization module. However, changing the sensor would require changes to the entire measurement setup. An improved sensor will still have some noise and therefore fluctuations will always be present. This means that, to improve the real time optimization further, another solution to prevent the printer from fluctuating and creating fluent movements will be necessary. Due to the time frame of this project not to change the entire measurement setup, but attempt to find a solution by optimizing the real time optimization module instead. By achieving this, the optimized modules can also be used in the future to improve the real time optimization when a better sensor is used or when measurements are performed on other output parameters to optimize the printing process.

A solution to reduce the fluctuations of the printer is to not directly use the measured values in the optimization module, but to constantly calculate a filtered or average value and respond to this value. This will result in a more stable value thus less fluctuating, but will also affect the ability for the printer to respond adequately and fast enough to the surface below.
A second solution is to use a PID-controller. This is a control loop feedback mechanism often used in industrial control systems. A desired set point is entered into the controller, which in this project is the desired distance from the surface below. The controller constantly calculates an error value, which in this case is the difference between the desired set point and the process variable (the measured height value). The controller tries to minimize this error over time by constantly adjusting the control variable (the Z-coordinate of the printer) (Control Solutions Minnesota, 2017)

Aside from the fluctuating, resulting from the noise in the measurements, the test results show that the response action is performed too early. The reason for this can be explained using Figure 34 and Figure 35. In these figures you can clearly see that the height elements used in the second and third test scenario are measured around 0.4 seconds before the printer actually arrives at that specific location. The height elements are measured too early because the measurement sensor is placed in front of the nozzle. Therefore, the measurements are not taken at the exact location of the nozzle, but at a distance of around 30mm in front of the nozzle. This causes the printer to respond too early. A possible solution for this is to use a delayed response. In Figure 36, a hypothetical scenario is shown in which there is a height increase in the previous printed layer. In the basic optimization module the printer moves towards this height increase and when the sensor, which is in front of the nozzle, measures the height increase the printer directly responds by adjusting the Z-offset. However, ideally the printer uses a delayed response which causes the printer to respond exactly when the nozzle arrives at the height increase and not when the sensor measures it.

A third problem derived from the results is that the printer does not move up enough to keep the desired height above the height elements in scenario 2, the transition surface (Figure 34) and scenario 3, the 3D surface (Figure 35). This does not cause any problems in the tests since the desired height is set at 100mm above the surface below, but when used during printing the desired distance will be much smaller. In that case the printer can crash into the surface below. The optimization module needs to be adjusted in such a way that the printer moves up and down faster over time when an element is encountered. PID-control might be a solution to improve this. By adjusting the settings of the PID-controller the intensity of the response over time can be influenced.
All of these possible optimizations, using an average or filtered value and using a PID-controller to reduce the fluctuations, using a delayed response to perform an action at the correct location and using a PID-controller to increase the height response over time are further elaborated in paragraph 5.5. In this part the performance of the different optimized modules is investigated to determine which settings and modules will be used to perform the use-cases for this research.

5.5 Improvements to the basic optimization module

5.5.1 Delayed response
The test results in paragraph 5.4.2 show that the printer responds too early to measurements, due to the distance between the measurement system and the nozzle of the printer. It is impossible to measure the height of the layer precisely underneath the nozzle since at that point the concrete is flowing from the nozzle. The distance between the measuring device and the nozzle is kept as small as possible, but can never be brought down to zero. Therefore, it is important to improve the optimization module to be able to perform a delayed response, so the printer can respond at exactly the right coordinates. How much the response needs to be delayed depends on \( D \), the distance between the nozzle and the measurement device. This is explained using Figure 37. In this figure, the fictional scenario from Figure 36 is used to illustrate how the delayed response works. When the sensor measures the height increase in the layer, the printer should not respond directly but only when the nozzle is at the exact location of the height increase. When the printer travelled the distance \( D \) since the height increase is measured, the nozzle is at the location of the height increase. This means that the printer needs to change the \( Z \)-offset when the distance \( D \) is travelled.

![Figure 37: Delayed response based on the distance between the nozzle and the measurement device (D)](image)

This is achieved by adjusting the optimization module to not respond to the most recent measurement value, but to a previous measurement value. To be able to use a previous measurement value, measurement values need to be stored. This storage of previous measurement data is done using a Ring-buffer. This is type of data-storage with a fixed-size buffer. This type of storage is often used to buffer data streams, which in this project is the constant stream of measurement data (Wikipedia, 2017). Using the ring-buffer enables you to use previously buffered values, but does not require extensive storage capability since only a
limited amount of previous data is stored. This is convenient, since there is only a limited amount of R-variables available on the SINUMERIK to use for storing data.

A ring-buffer has a predefined length of elements and starts empty. The buffer is filled up starting at the first element and continues to the next element when a new measurement value is available. This continues until all elements of the buffer are filled. At that point the system returns to the first elements and overwrites the data in this element with the new measurement data. Then the buffer continues to the next element and overwrites the present data with new data until all elements of the buffer are overwritten and the process starts again.

In Figure 38 a visualization of a ring-buffer with six elements is shown.

![Figure 38: Visualization of a ring-buffer with six elements](image)

The next step after filling a ring-buffer is to select the correct and desired element from the buffer. A possible way is to do this is to not only store the measurement values, but to also store a time-stamp for each of the measurement values in the ring-buffer. The time the printer takes to travel D (the distance between the measurement device and the nozzle) can be calculated using the speed of the printer. By subtracting the added time-stamps linked to the measurement values from the current time-stamp, the time that has passed can be determined. Using this calculation, the correct measurement value can be selected from the buffer using the added time-stamps. However, the problem with this method is that the movement speed of the printer can vary during a print session. When the speed changes this means that the time the printer takes to travel distance D changes as well. The wrong element from the buffer would be selected and the printer would still not respond with a correct delay. Therefore, another method is developed which is independent of the speed of the printer.

This method is to constantly calculate the distance travelled (DT) during printing, instead of the time that has passed. When the printer travelled the distance D, the delayed response will take place (Figure 39). This is achieved by storing not only the measurement values, but also storing the X and Y coordinates of the printer in the ring-buffer (Figure 40). This is possible, since the X- and Y-coordinates of the printer are known and can be directly accessed within the SINUMERIK.

![Figure 39: Delayed response based on travelled distance](image)
Figure 40: Storing X- and Y-coordinates in the Ring-Buffer to calculate the travelled distance

With Pythagoras the distance travelled can then be calculated using the current X- and Y-coordinates and the stored X- and Y-coordinates in the buffer (Figure 41). When the DT of one of the elements in the buffer is equal to D, distance between and the nozzle and the measurement device, the measurement value of that element should be selected. The selected measurement value can then be used in the response module and a delayed response will take place. \(X_{\text{travelled}}\) is calculated by subtracting \(X_{\text{stored}}\) from \(X_{\text{current}}\) and \(Y_{\text{travelled}}\) is calculated by subtracting \(Y_{\text{stored}}\) from \(Y_{\text{current}}\). Pythagoras is then applied to \(X_{\text{travelled}}\) and \(Y_{\text{travelled}}\) to calculate the DT on the horizontal plane. The Z-coordinates are not used in this calculation since the delayed distance only relates to the horizontal surface, thus the X and Y-coordinates.

\[
\begin{align*}
X_{\text{travelled}} &= X_{\text{current}} - X_{\text{stored}} \\
Y_{\text{travelled}} &= Y_{\text{current}} - Y_{\text{stored}} \\
DT^2 &= X_{\text{travelled}}^2 + Y_{\text{travelled}}^2 \\
DT &= \sqrt{X_{\text{travelled}}^2 + Y_{\text{travelled}}^2}
\end{align*}
\]

Figure 41: Calculating the Distance travelled (DT) using Pythagoras

Delayed response module

In Figure 42 the delayed response module is shown. A more detailed explanation of the module can be found in Appendix C4. A ring-buffer is created with a length of 795 elements. The elements in the buffer are the R-variables R200 until R995. The major difference with storing of data shown in Figure 40 is that on an R-variable only one value can be stored. Therefore, instead of one, three elements are used to store the measurement value, X-coordinate and Y-coordinate at a specific point. Instead of taking one step to the next element with every iteration three steps are made in every iteration between one location and the next (Figure 43).
The module starts with the distance settings (red) and the delayed response settings (orange). In the delayed response settings, the \( D = \) Distance from measurement device to nozzle is set (R196) and the Setting sensor collecting distance is set (R197). This is the interval with which the ring-buffer is filled. This is necessary since otherwise the buffer would be filled too quickly and the buffered-time would be too small. When a smaller sensor collecting distance is used or the distance between the measurement device and the nozzle increases, the required buffer size increases. The size of the used buffer should always have an equal or bigger size than the required buffer size. After the delayed response settings are created all the used variables are created and initialized (yellow). Afterwards, two markers are created to point out the location in the buffer where the next value needs to be filled and where the delayed measurement value is that should be read (green).

After the two markers are created the part of the module, in which the synchronized actions are activated, starts. This part of the script is continuously executed during printing. This starts with storing the X- and Y-coordinate and the measurement value in the buffer (blue). When this is done, the travelled distance is calculated and the correct delayed measurement value is read from the buffer (purple). In the final step (brown), the required Z-offset is calculated to keep the nozzle at the desired distance from the surface below and this Z-offset is performed.

After the two markers are created the part of the module, in which the synchronized actions are activated, starts. This part of the script is continuously executed during printing. This starts with storing the X- and Y-coordinate and the measurement value in the buffer (blue). When this is done, the travelled distance is calculated and the correct delayed measurement value is read from the buffer (purple). In the final step (brown), the required Z-offset is calculated to keep the nozzle at the desired distance from the surface below and this Z-offset is performed.

![Figure 42: Delayed response module](image)
**Test results**

The performance of the delayed response module is tested on the three different scenarios. The desired distance from the surface below is again set at 100mm. The distance between the nozzle and the measurement device is 35mm. Therefore, the delayed distance in the module is set at 35mm. The sensor collecting distance is set at 0.3mm. These settings make sure that the buffered-time is more than enough to be able to use delayed measurement values with a DT of 35mm. The performance of the delayed response module is compared to the baseline graph of the scenarios and to the performance of the basic optimization module.

The test results of the three different scenarios are shown in Figure 44, Figure 45 and Figure 46. Overall, the results show that the fluctuations of the printer in the Z-direction is increased compared to the basic optimization module. Especially in the second and third scenario, the 3D surface, when the printer responds to the height elements the fluctuations increase up to almost 25mm. The fluctuations are not constant but very a lot along the test path.

In the second scenario, the transition surface, when the first height element is encountered (6.5s) the delay in the response is visible. Instead of responding too early, which was the case with the basic optimization module, this module starts moving up with a delay. When the module moves the nozzle up, the fluctuations increase. When the second slope starts (10s), the printer moves up with a small delay and when the height element ends (15s) the printer moves down with a delay. The average height of the printer is similar to the average height of the printer with the basic optimization module.

In the third scenario, when the first height element is encountered (6s), the printer starts moving up with a small delay compared to the basic optimization module. It starts exactly at the point where the baseline graph starts increasing and almost perfectly follows this line until it is at the Z-coordinate of -1540. This is another indication that the delayed response works. After this, the delayed response movement is almost similar to the basic optimization module, but with increased fluctuations and some delay.
Figure 44: Performance of the delayed response module on scenario 1

Figure 45: Performance of the delayed response module on scenario 2
**Analysis of test results**

Overall, the test results show the principal of a delayed response works. The delayed response graph was expected to be similar to the basic optimization module, but with a delay. However, from the test results it becomes clear that this is not the case and the graphs shape differs a lot from the basic optimization module graph.

Using the trace-function, both the raw measurement value and the delayed measurement value used to calculate the response were logged during all the tests. In Figure 47, both the raw measurement and the delayed measurement values are shown in a graph. A small time-frame between 4 and 6 seconds is shown. The delayed measurement value (red line) is identical to the shape of the measurement value (green line), but with a small and constant delay. This means that the delayed measurement value is correctly calculated and the difference between the graphs of the basic and delayed response module is not caused by an incorrect calculation of the delayed measurement value.
The reason that the delayed response graph is so different from the basic optimization graph can be explained using Figure 48 and Table 3. In the case of the basic optimization module the printer directly responds to a measurement and directly tries to keep the nozzle at a fixed distance by altering the Z-offset of the printer. When this Z-offset is adjusted, the next measurement values are directly influenced by this Z-offset, because when the printer directly starts moving, the measurement device, attached to the nozzle, also moves up or down with the printer. This means that, depending on the speed with which the printer can alter the Z-position, the measurement values will always remain close to 100mm. In Figure 48, a fictional scenario is used in which the printer encounters a height increase in a layer. The desired distance from the measurement device to the layer below is set at 100mm and the height increase is 20mm. The path of the printer is divided into time-steps from 0 to 10s. The horizontal distance between the measurement device and the nozzle in this scenario is the equivalent of 2s. This means that the printer is responding to the measurement values with a delay of 2s. Table 3 shows a table with time-steps of 0.5 seconds in the first column. The second column shows the raw measurement values, which is the actual distance from the printer to the surface below. The third column shows the delayed measurement values. These values are similar to the raw measurement values, but the values in the column are shifted down by 2s, since this is the desired delay.
Between the first position and the second position of the printer (from 0 to 4s), the measurement values are 100 and no response is made. After that, the height increase is measured by the measurement device, but the printer does not yet respond due to the delayed response. This can be seen in the table since the delayed measurement value is still 100 and does not decrease yet. Therefore, the raw measurements change from 100 to 80 between the 4 and 5 seconds time-steps. However, as can be seen from the table, the printer will only start responding after the 6s time-step, since then the delayed measurement value will start reducing. This means that between the 5s and 6s time-step, the raw measurement value will remain 80mm. At the third position of the printer (6s) the printer will start moving up until the raw measurement is 100mm at the 7s time-step. This means that theoretically the printer perfectly kept the desired height above the height increase and followed the slope. At this point, the printer is at the desired distance of 100mm from the top surface of the layer. After this 7s time-step, the problem of a delayed response becomes clear. Since the delayed measurement value after the 7s time-step is still 80mm, the printer keeps moving up because the delayed measurement is 20mm lower than the desired height. It will keep moving up until the delayed measurement value reaches 100mm which is at the 9s time-step. However, by this time the printer has reached a height compared to the surface of about 125mm. After the 9s time-step the delayed measurement values increase, so the printer starts moving down. But similar to the movement upwards, the delayed measurement values cause the printer to move down too far and will cause the raw-measurement value to get below 100mm. Then the cycle starts again and the printer starts moving up again, but moves up too far. This effect causes the fluctuations which can be seen in the test results of the delayed response module. This happens because the raw measurement values are influenced by the delayed response.

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Raw measurement value (mm)</th>
<th>Delayed measurement value (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>0.5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1.5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>2.5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3.5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>4.5</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>5.5</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>6.5</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>80</td>
</tr>
<tr>
<td>7.5</td>
<td>110</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>120</td>
<td>80</td>
</tr>
<tr>
<td>8.5</td>
<td>125</td>
<td>90</td>
</tr>
<tr>
<td>9</td>
<td>125</td>
<td>100</td>
</tr>
<tr>
<td>9.5</td>
<td>115</td>
<td>110</td>
</tr>
<tr>
<td>10</td>
<td>105</td>
<td>120</td>
</tr>
</tbody>
</table>
Even though the concept of responding to a delayed measurement works, this is not usable within this project, since the measurement values are influenced by the by the delayed response itself. This influencing of the raw measurement values causes the printer overreact and move up and down. Furthermore, as can be seen from the test results, the overreacting increases when the height of an element increases. When an entirely different measurement setup is used in which the measurement device is not connected to the moving arm, the measurement values will not be influenced by the delayed response itself. In this case, the delayed response module would be usable to optimize the printing process. However, since this measurement setup is chosen, using a delayed response with this method is not possible and will not be used in this research.

5.5.2 Use average or filtered value for response
As explained before, using an average or filtered value can be a solution to reduce the fluctuations of the printer. It became clear that calculating an actual average value of a certain amount of previous measurements is fairly complicated to script in a G-code module. Using a filtered value is far less complicated to achieve. For this reason, the filtered value is used to reduce the fluctuations of the printer and is used to create the final use-cases. However, in order to be able to compare the performance of responding to the filtered value to the performance of responding to an actual average value, developing a module that responds to an average height value (AHV) is investigated parallel to the rest of the project. Towards the end of the project, a working module that responds to an AHV was created and therefore a few simple tests are performed to compare the filtered value response with the AHV response.

A filtered value or an average height value needs to be continuously calculated during a print session to be usable in real time. The basic response module is adjusted to not use the raw measurement values to determine how to respond, but to use the filtered value (FV) or average height value (AHV) instead. The improved response module is programmed in such a way that the Filter Factor (FF) used to determine the average or filtered value can be changed. In the calculation of the average height value (AHV), the FF is the amount of previous measurement values used to determine the AHV. In the calculation of the filtered value (FV), the FF is the value with which the newest measurement value is filtered. Creating an optimization module in which the FF can easily be adjusted allows for performing tests with different FF values to determine the optimal and desired settings. It also makes the module usable in the future when a better sensor is used and a less filtering is required, or when the module is used for measurements on different output parameters and thus different filter settings are required.

Filtered response
A filtered value (FV) is calculated without the need of storing previous measurement values and is for this reason easier to program in G-code than the AHV, for which previous measurement values are needed. The following formula is used to calculate the FV.

\[
FV_{\text{new}} = \frac{FV_{\text{previous}} \times (FF - 1) + MV}{FF}
\]
FV<sub>new</sub> = New filtered value
FV<sub>previous</sub> = Previous filtered value
FF = Filter factor
MV = Measurement value (the latest measurement value)

The principal of the FV is that the new measurement value (MV) only counts for a small part in the calculation of the FV<sub>new</sub>. The impact of the MV is dependant on the FF. When a FF of 10 is used, the new MV contributes for 1/10<sup>th</sup> in the calculation of FV<sub>new</sub>. The FV<sub>previous</sub> contributes for the remaining 9/10<sup>th</sup> of the new FV. This FV is then used to create a response instead of the raw new measurement value.

**Filtered response module**

In Figure 49, the filtered response module is shown. This module is similar to the basic optimization module. However, instead of directly using the latest measurement value, the filtered value (FV) is used to calculated the required Z-offset. The filtered response module with explanation can be found in Appendix C5.

Firstly, the desired distance settings need to be entered (red). This is identical to the basic optimization module. Then the important variables are created and values are initialized (orange). Two variables are added. The Filter Value (FV) = R20, which is initialized with the newest measurement value (MV) and the Filter Factor (FF) = R24, which in this example is set at 25. This means that the new MV contributes for 1/25<sup>th</sup> in the calculation of new FV. After this, the synchronized actions part starts (yellow), which is performed constantly during printing. Only the first synchronized action (ID=1) is changed compared to the basic optimization module. Instead of using the latest MV, the FV is calculated and this value is then used to calculate the offset to the Z-axis. The offset to the Z-axis is performed in the last synchronized action (ID=4)

Figure 49: Filtered response module
**AHV response**

The average height value (AHV) differs slightly from the FV. In order to calculate the AHV, previous measurement values are required. The AHV is the average value of an X amount of previous measurements. The amount of previous measurement values used is equal to the FF. This FF needs to be easily adjustable to be able to test different FF and make the module applicable for other scenario's. The following formula is used to calculate the AHV.

\[
AHVSUM = \text{AHVSUM} - MVold + MVnew \\
AHV = \frac{AHVSUM}{FF}
\]

AHV = Average Height Value  
FF = Filter Factor (Amount of previous measurement values used to determine AHV)  
MVold = Oldest Measurement Value (from the previous measurement values used)  
MVnew = Newest Measurement Value

First, the AHVSUM is calculated. The AHVSUM is equal to the sum of the X previous measurement values, but is calculated using a different method. The AHVSUM is calculated by subtracting the Oldest Measurement Value (MVold), and adding up the Newest Measurement Value (MVnew) from the previous AHVSUM. The value from the buffer that corresponds to MVold depends on the FF used. When a FF of 10 is used, the MVold is the 10\text{th} previous measurement value. The AHVSUM is then used to calculate the AHV. This is achieved by dividing the AHVSUM with the FF.

The method with which the AHV is calculated does not add up all X amount of previous measurements and then divides it by X to calculate the AHV, which is what normally would have been done to calculate an average value. Another method is used since it appeared to be very hard to extract the X amount of previous measurement values. Because of this the Ring-Buffer is used to store previous measurement values and because the X amount of variables used needs to be flexible and easily changed.

**AHV response module**

In Figure 50 the AHV response module is shown. The module with explanation can be found in Appendix C6. The module uses a lot of the same elements as the delayed response module. This results from the fact that the Ring-Buffer is used to store previous measurement data needed to calculate the AHV again. The module starts with the desired distance settings (red). Then the AHV settings (orange) in which the sensor collecting distance to fill the buffer and the FF, which is the amount of values used to determine the AHV, need to be set. The next part of the module is different from the other modules (yellow). In this part, the ring-buffer elements need to be reset so all values in the buffer are zero. This is because the R-variables used in the ring-buffer often still have values assigned to them in a previous print session. These values have nothing to do with the current print session and are therefore all reset to 0. When this is not done, the old incorrect values are used in the calculation of the AHV, leading to to an incorrect AHV. In the next part of the module the used variables are created and initialized (green). Two variables are added: \( R187 = \text{AHVSUM} \) and \( R192 = \text{AHV} \). These values are both initialized at 0. It is important
that the AHVSUM is initialized at 0 in order to correctly calculate the AHV. The chosen method to calculate the AHV does cause an initialization problem. This is further elaborated upon in the next paragraph. After the initialization of the variables, three markers are created (blue). The first points at the place in the buffer where the newest value should be updated. The second one points at the MVold in the ring-buffer, which needs to be subtracted from the AHVSUM to calculate the new AHVSUM. The third marker checks whether a new measurement value is added to the ring-buffer, in which case a new AHVSUM should be calculated. After this the synchronized actions start. It starts with the synchronized actions ID=1 and ID=2 (purple), in which measurement values are stored in the ring-buffer. Then in ID=3 (brown) the AHVSUM and the AHV are calculated. The final part of the module is similar to the previous modules. In this part (grey) the required Z-offset is calculated and performed. However, in this module the AHV is used to determine the required Z-offset.

Figure 50: AHV response module
As mentioned before, there is an initialization problem in the calculation of the AHV. It takes a small amount of time for the AHV to slowly increase from 0 to the actual AHV. With a sensor collecting distance of 0.3 and a FF of 25 it takes the module approximately 0.45 seconds for the AHV to increase to the actual correct AHV. This is shown in Figure 51. A test is performed in which the printer is set at a fixed height, so the measurement device is at a distance of 100mm from the flat surface below. The AHV response module is used to calculate the AHV, but the part of the module in which the response is made is turned off (ID=7). The trace-function is used to log the AHV. The graph shows the AHV over time. In the graph you can clearly see that the AHV increases gradually from 0 to approximately 100mm in around 0.45 seconds. The small fluctuations in the AHV are caused by the noise in the measurement sensor.

![Graph showing AHV over time](image)

**Figure 51: Initialization problem with the calculation of the AHV**

The initialization problem occurs because the Ring-Buffer starts empty (all values in the buffer are 0). The buffer is filled stepwise with new measurement values. A new measurement value (MVnew) is added to the AHVSUM, but an old measurement value (MVold) is also subtracted from the AHVSUM. When 25 values are used to determine the AHV, the MVold, which will be removed from the AHVSUM, is the 25th previous measurement in the ring-buffer. However, since there is no previous measurement value available yet, this value is 0 (since all the R-variables of the ring-buffer are reset to 0 before the script starts). This means that during the first 25 steps of filling the buffer and calculating the AHV, only values are added to the AHVSUM, since all MVold values that are subtracted are 0. Therefore, the AHV slowly increases. After 25 steps, the MVold values in the buffer are actual previous measurement values, the AHV remains constant and from that point the correct AHV is calculated. The initializing problem does have an effect on Z-movement of the printer. This will be discussed further in the analysis of the test results.
**Test results and analysis**

To check whether a filtered factor reduces the noise in the measurement values, a simple test setup is used similar to the test setup used Figure 51. A test is performed in which the printer is set at a fixed height so the measurement device is at a distance of approximately 100mm from the flat surface below. The filter response module is used to calculate the filtered value, but the part of the module in which the response is made is turned off (ID=4). The printer moves with a fixed speed across a flat surface (the first scenario). The filtered value and the raw measurement values are logged using the trace-function. Two different tests are performed with two different FF values, 10 and 50. By comparing the filter values with the raw measurement values the amount with which they reduce the noise in the measurement values can be determined. Reduced noise will result in reduced fluctuations of the printer when used in a respond module. The results of these tests can be seen in Figure 52. In this graph the first five seconds of the test are shown. This gives a clear view how the filtered values compare to the raw measurement values.

![Graph showing comparison between raw measurement values and filtered values](image)

**Figure 52: Comparison between raw measurement values and filtered values**

The raw measurement values (green line) fluctuate from 97mm up to 107mm which is a range of 10mm. The filtered value with a FF of 10 (red line) already has a reduced fluctuation which ranges from 98mm up to 106mm, which is a range of 8mm. The fluctuation of the filtered value with a FF of 50 (blue line) is greatly reduced compared to the other two values. This line only fluctuates from 101.5mm up to 104.5mm, which is a range of only 3mm. This means that when a filtered value with a FF of 50 is used the fluctuations of the printer will be greatly reduced.
However, besides the reduced fluctuations, it is also important to understand what the consequences to the overall performance of the optimization module are when using a filtered value rather than a raw measurement value. To determine this, two tests on the third scenario are performed with the filtered response module. Two different FF values are used, 25 and 50. The results of both of these tests are shown in Figure 53. The results are compared to the baseline graph and the performance of the basic optimization module. Testing this on the third scenario gives insight in the performance of the module. The results of this test is used to determine if and how this module can be used to further optimize the response module and whether it is usable to perform the use-cases of this research.

Aside from the fluctuations, both graphs of the filtered response tests have almost the same overall shape as the graph of the basic optimization module. No significant delay in the response is caused by filtering the measurement values. The slopes, at the position of the height elements (4s, 6.5s, 10s and 11s) as well as the maximum height the printer reaches when moving across the height elements is similar to the basic optimization module. For this reason, this module does negatively affect the overall performance of the optimization module as much as the delayed response module does. However, does not improve the overall shape either, but this is not expected from this module. What is positively affected are the fluctuations of the printer. As expected from the results shown in Figure 52, the filtered responses reduce the fluctuations of the printer. The filtered response with a FF with a value of 25 (red line) only
slightly filters the raw measurement values and therefore fluctuations are only slightly reduced and at certain points they even seem to slightly increase (10s). On the other hand, the filtered response with the FF value of 50 (blue line) greatly reduces the fluctuation of the printer compared to the basic optimization module. Figure 54 zooms in to the horizontal part between of the test between 13s and 15s. Here you can see that the filtered response module with a FF of 50 only fluctuates between −1579.5 and -1577.5, which is a fluctuation range of 2mm. The basic optimization module fluctuates from -1582 up to -1576.5, which is a fluctuation range of 5.5mm. These results show that a filtered value can be used to reduce the fluctuation while remaining the overall performance of the optimization module.

![Figure 54: Performance of the filtered response module to reduce fluctuation of the printer](image)

As mentioned earlier in this chapter the AHV response module was developed parallel to the project. To compare the performance of the filtered response module and the AHV response module the two tests shown in Figure 53 and Figure 54 are repeated with the AHV response module with similar FF’s of 25 and 50. This allows for a direct comparison between the performance of the two modules. The tests result are shown in Figure 55 and Figure 56.

With the exception of the first second and the fluctuating on the horizontal surfaces, the shape of the graphs in both tests are very similar to the filtered response module. Again, no huge delay is caused and the slopes and maximum height of the printer remain similar. However both graphs clearly show what the consequences of the initialization problem in the AHV response module are. In both graphs, the AHV response module causes the printer to start moving up directly from the beginning. The printer starts moving up since the calculated AHV value slowly
increases from 0 to the actual value. This means that in the beginning the AHV is lower than it is supposed to be. The module then calculates, using the incorrect AHV, that the printer is too close to the surface and thus needs to move up. When the AHV value reaches the correct AHV, the printer is too high above the surface and starts correcting this by moving down until the AHV is at 100mm. In Figure 56, where the FF value of 50 is used, it takes the module more time to calculate the correct AHV compared to the FF value of 25 and therefore the printer moves up further. The fluctuations of the printer differ slightly between the filtered response and the AHV response. In certain parts of the scenario, the filtered response fluctuates more and on other parts less. However, overall the average fluctuation is similar between the two modules.

![Figure 55: Comparison of the Filter and AHV response module (FF=25)](image)
From these results the conclusion is drawn that overall it would not make a significant difference whether the filtered response module or the AHV response module is used to reduce fluctuation. However, due to the initialization problem, the best choice is to use the filtered response module since the initialization problem will negatively affect the performance of the optimization module in the beginning.
5.5.3 Using PID-Control

**PID-controller**

A PID-controller can be used to reduce the fluctuations of the printer and to increase the intensity of the response. PID-controller stands for Proportional-integral-derivative controller. It is a control loop feedback mechanism. A desired set point (SP) is entered into the controller. The controller constantly calculates an error value (e(t)), which is the difference between the desired set point and the process variable PV(t). The controller tries to minimize this error over time by constantly adjusting the control variable u(t). (Wikipedia, 2017) The following formula is used to calculate the error value:

\[ e(t) = SP - PV(t) \]

- \( e(t) = \text{Error value} \)
- \( SP = \text{Setpoint} \)
- \( PV(t) = \text{Process variable} \)

The manner in which different variables apply to this project is explained using Figure 57. In this visualization the printer nozzle and the measurement device are shown during printing. The SP is the desired distance from the measurement device to the previous layer. The PV(t) is the actual distance from the measurement device to the previous layer, which is the measured value. The e(t) is the error value, which is the difference between the SP and the PV(t). The u(t) or the control variable, is the Z-offset which is changed. What the PID-controller does is minimizing the error value over time by adjusting the this Z-offset (u(t)). The PID-controller constantly attempts to reduces the e(t) to 0, so \( SP = PV(t) \), which means that the measured value is equal to the desired value and the nozzle is at the desired distance from the previous layer.

![Figure 57: Visualization of the PID-control variables](image)
As mentioned before, PID-controller tries to minimize the error over time by adjustment of the control variable \( u(t) = Z\) -offset of the printer. The following formula is used to calculate the control variable:

\[
u(t) = P + I + D
\]

\( P = \text{Proportional term} \)

\( I = \text{Integral term} \)

\( D = \text{Derivative term} \)

The \( u(t) \) is determined by calculating the sum of the proportional (P), integral (I) and derivative (D) term. The different terms are calculated using the following formulas:

\[
P = K_p e(t) \\
I = K_i \int_0^t e(t)dt \\
D = K_d \frac{de(t)}{dt}
\]

The P is calculated by directly using the current error value, which in this project is the required Z-offset. The I variable is determined by calculating the integral of the error value. When the result is not reduced as a result of the error value, the integral will accumulate over time. This means that the \( u(t) \) will grow over time and a more severe action is performed. The D determines the current rate of change of the error value by calculating the derivative of the error value. This allows the PID-controller to predict what the future value of the error will be (Wikipedia, 2017). For example, when the error value is quickly reducing and approaching 0 the D-term will decrease and the response will be less intense. This leads to the following formula:

\[
u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}
\]

\( u(t) = \text{Control variable} \)

\( e(t) = \text{Error value} \)

\( K_p = \text{Coefficient for the Proportional term} \)

\( K_i = \text{Coefficient for the Integral term} \)

\( K_d = \text{Coefficient for the Derivative term} \)

The PID-controller only relies on the measured process variable and has no knowledge about which situation or process the controller is used for. This makes the PID-controller usable for many different situations and applications (Wikipedia, 2017). However, the PID-controller needs to be adjusted to the specific situation of this research to achieve the desired results. The three different coefficients make you able to adjust PID-controller for a specific application. Each of the P, I and D terms have their own coefficient which can be individually adjusted. By increasing
or decreasing the coefficients, the result of each of the different P, I and D terms can be reduced or increased since the coefficient is multiplied with the result of the P, I or D. For example, when the coefficient of the P-term is set at 0, this means that the result of proportional term will always be 0 and therefore have no share in the calculated u(t). The I and D terms have a time-dependency. As a result of this time-dependency the effect of the I and D terms change when different print speeds are used, because the speed of the printer (distance per time-unit) directly relates to the time-factor.

An example of the result of adjusting the P-coefficient is shown in Figure 58. In this figure a graph is shown of the PV(t), the measured value plotted against time. In this example, the I and the D coefficient is held constant, but three different P-coefficients (Kp) are used. Kp = 0.5, Kp = 1.1 and Kp = 1.6. The blue line represents the reference situation, similar to the actual situation line used in the tests performed for this research. Adjusting the Kp has a major effect on the PV(t) over time. When a Kp of 1.6 is used (purple line), the PV(t) increases faster over time than in the cases with other Kp coefficients. However, it also overshoots the reference line positively and negatively multiple times. When the lowest Kp of 0.5 is used, the reference line is never crossed, but it takes longer for the PV(t) to increase towards the reference line. This means that the response is less intense.

![Graph showing the effect of adjusting the Proportional coefficient](image)

**Figure 58: Result of adjusting the Proportional coefficient (Wikipedia, 2017)**

Similar to this example, adjusting each of the coefficients has an effect on the development of the PV(t) over time. Similar to the results in this graph, adjusting the different coefficients can help to reduce the fluctuations of the printer and increase or decrease the intensity of the response over time. Loop tuning is adjusting the different coefficients until the settings for an optimal control response for a specific situation are found. Many different tuning methods are available, but these are very complicated and require extensive knowledge of PID-controller and
the specific tuning method. The P, I and D settings are tuned manually in a process of trial and error in a series of tests. Within these tests a fixed printer movement speed is used, which means that the results are only usable with this used speed. The goal of these tests is to optimize the PID-controller by tuning the PID-settings and improve the current performance of the basic optimization module.

Before tests with the different coefficient terms are performed, predictions are made towards the settings of the derivative-coefficient. The D-term predicts the future error value and therefore responds to the future situation. A problem occurs when the D-term is used while there is a lot of noise present in the measurement values. Due to noise in the sensor, the next measurement often differs a lot from the previous measurement. Since the D-term looks ahead, it constantly thinks that major changes are required since the current rate of change between each of the measurements is very large. This causes the PID-controller to overreact, which might make the printer fluctuate up and down even more (Control Solutions Minnesota, 2017). For this reason, the D-term is not used for this research, which means that the D-coefficient is set at 0. The P- and I-coefficients are used and manually tuned during the tests.
**PID-control response module**

The PID response module is shown in Figure 59. The module starts with the desired distance settings (red). Then the PID-coefficient settings (orange). In this example, the P-coefficient is set at 0.2, the I-coefficient is set at 1.5 and the D-coefficient is set at 0, which means the Derivative term is not active. In the next part, all variables are created and initializing values are assigned (yellow). R23 is the cycle time of the SINUMERIK, which is the dt of the PID calculation. The Setpoint (SP) is set at the desired distance = R22. After initializing the synchronized action part starts (green). In the first synchronized action (ID=1) the PV(t) is updated and the P, I and D term are calculated. Then the U(t) is calculated by multiplying each of the terms with their corresponding coefficient and adding them up. The previous measurement is then updated with the newest measurement. In the second synchronized action (ID=2) the offset is performed based on the calculated U(t). The PID response module with explanation can be found in Appendix C7.

![Figure 59: PID response module](image)

**Test results and analysis**

Many tests are performed to optimize the PID settings. The tests were performed on the three scenarios used before. The results of a few of these tests are discussed and analyzed. The results show the performance of the PID response module with four different coefficient settings. The D-coefficient is not used and is therefore set at 0, since the Derivative factor is very sensitive to noise in the measurements. Tests are performed with four different used coefficients. These settings are: (P=0.4 I=0.4 D=0), (P=0.2 I=0.2 D=0), (P=0.4 I=1.5 D=0) and (P=0.2 I=1.5 D=0). The performance of the PID response module is compared to both the baseline graph and the performance of the basic optimization module. The test results of the first scenario, the flat surface, are shown in Figure 60.
The first conclusion drawn from these results is that using PID reduces the fluctuations of the printer. This is because a relatively low P-coefficient is used. This reduces the effect of the error factor and thus reduces the contribution of the error value in the calculation of the control variable $u(t)$. This reduces the intensity of the response to direct measurements. In Figure 61 a graph is shown that shows a specific part of the previous graph from 2.5s to 4s. This graph shows that the PID modules with a P-coefficient of 0.2 (purple and orange line) have the lowest fluctuations. The purple line fluctuates up and down with a maximum of only 1mm, the orange line with a maximum of 1.5mm. The purple line is even more constant because of a low I-coefficient of 0.2. This means that the response over time is increased only slightly since the I-term remains relatively small. The modules with a P-coefficient of 0.4 fluctuate more, since the current error value has a contribution which is twice as big in the calculation of the $u(t)$. This increases fluctuations up to 2.5mm for the green line and up to 2mm for the blue line. This shows that the P-term has a great influence on the intensity of the fluctuations and this factor can be adjusted to effectively reduce them. The I-factor has a smaller effect on the intensity of the fluctuations. This can be concluded from the comparison between the purple and the orange lines. These graphs both have a P-coefficient of 0.2, but the I-coefficients differ a lot from each other, with values of 0.2 and 1.5. Even though the difference between these coefficients is big, the difference in the maximum fluctuation distance is only 0.5mm.
Figure 61: Comparison between the fluctuations of multiple PID coefficient settings

The test results of the second scenario, the layer transition, are shown in Figure 62. In this graph, clear differences in the performance of the PID response modules with the different P- and I-coefficient settings can be seen. Furthermore, they differ significantly from the performance of the basic optimization module. These results mainly show the impact of the I-coefficient. The purple lines, with P=0.2 and I=0.2, has the lowest I-coefficient. This results in the least intense response, which means that when the first height element is encountered (7s), the printer only slightly moves up compared to the baseline graph. When the height further increases (10s), the printer gradually moves up. However, the maximum height reached is approximately 20mm lower compared to the actual height of the element. The green line, with P=0.4 and I=0.4, is very similar to the purple line. However, when the height increases for the second time (10s) the printer moves up more severely and reaches a maximum height similar to the basic optimization module, but still below the baseline graph. The blue and orange line, which both have a I-coefficient of 1.5, differ a lot from the two previous lines. As a result of the increased I-factor, the intensity of the response grows over time and therefore reaches a bigger maximum height. It also differs less from the baseline graph compared to the other two PID response modules and the basic optimization module. They both reach a maximum height that is only 5mm lower compared to the baseline graph. This shows that adjusting the I-coefficient allows you to increase the intensity of the response of the printer.
Figure 62: Performance of the PID response module on the second scenario

Figure 63 shows the performance of the PID response module in the third scenario. From the graph, similar conclusions can be drawn as from the previous graph. Similar differences occur between the PID response modules with varying P- and I- coefficients. The purple and green lines, having the low I-coefficient, show moderate responses in which there is a large difference between the maximum height of the lines and the baseline graph at the locations of the height elements. However, since the elements in the third scenario are higher compared to the second scenario, the difference in height between the PID response modules and the baseline graph is bigger as well. The blue and the orange lines, with the high I-coefficient of 1.5, show a more intense response, which almost follows the baseline graph. The graph also clearly shows that the PID response module causes a delayed response compared to the basic optimization module (red line). Due to this delay, the printer does not respond too early, but starts increasing at almost exactly the correct time, when the first height element is encountered (6s). However, when the printer has almost passed the first or second height element (9s and 14s), the printer does not exactly follow the baseline graph. A smooth transition is made instead. This creates a height difference between the PID graphs and the baseline graphs. The difference is smaller when a higher I-coefficient is used.
The overall conclusion is drawn that using PID can greatly improve the performance of the basic optimization module. First of all, the fluctuations of the printer as a result of the noise in the measurement device are reduced. Secondly, a better timed response is achieved due to the delayed response caused by the PID module. Thirdly, a more intense response can be achieved in which the printer better follows the baseline graph and therefore maintains a more constant distance above the surface below. This means that using a PID-controller improves the performance of the optimization module regarding all the main problems encountered in the tests of the basic optimization module.

Figure 63: Performance of the PID response module on the third scenario
5.5.4 Conclusion of the improvements to the optimization module

After testing the basic optimization module, it became clear that there was room for improvement of the module. These improvements can upgrade the quality of the final use-cases of this project. Three main problems and optimization solutions are determined and investigated. For each of the possible solutions a working response module needed to be developed in G-code, which the SINUMERIK operating machine can execute. For each of the proposed solutions a working response module is successfully developed. Multiple tests are executed to determine the performance of each of the response module and to further improve the performance by experimenting with the settings of each of the modules.

Individually looking at the improvements and their tests result different conclusions are drawn. The delayed response module works and the module causes a delayed response. However, the raw measurement values are affected by the delayed response itself and cause the printer to overreact, which drastically increases the fluctuation of the printer. Because of this, the performance of the optimization module is reduced compared to the basic optimization module. Even though the concept of delayed response works, this proposed solution does not lead to an optimization and will for that reason not be developed any further in this project.

The second solution is to use an average or filtered value for the response to reduce the fluctuation of the printer. Two improved response modules are developed. One that calculates an actual average height value (AHV) and one which filters the measurement value and creates a filter value (FV). The AHV response module has a similar overall performance as the filtered response module. However, because of an initialization problem the performance of this module is negatively affected and is therefore less useful and effective than the filtered response module. The filtered response module is successful in reducing the fluctuation of the printer caused by the noise in the measurement system. The overall performance of the basic optimization module is not significantly negatively affected while it is able to positively reduce the fluctuations. For this reason, the filtered value can be used to optimize the basic optimization module.

The third proposed solution is using a PID-controller. This can help to reduce the fluctuations of the printer and create more intense responses which means that the Z-offset is adjusted faster over time. Again a functioning PID response module is created. Tests are performed to determine the performance of the module. The response module is optimized through adjusting the PID coefficient settings. Using a PID-controller to adjust the Z-offset greatly improved the performance of the basic optimization module. Fluctuations are reduced, more intense responses are performed and the response is delayed, which causes an almost perfectly timed response.
Important to understand is that certain PID coefficients settings and filter settings apply to a specific scenario. For example, when different height elements with a steeper slope are used to create a 3D surface, a more intense response is required. Therefore, different PID settings are required to perform an optimal response for that scenario. However, as shown in and explained from the test results, the impact of the P- and I-coefficient on the performance of the optimization module can be understood and hence predicted. For example: elements with steeper slopes require a more intense response and thus require an increased I-coefficient. This means that when a different type of scenario is used a full scenario of new tests need to be performed. However, the PID-settings can be adjusted slightly based on the differences in the new scenario. This does not only apply for the type of scenario that is used, but also for the quality of the sensor. When a sensor is used with less noise, an increased P-coefficient or a reduced filter factor can be used since smaller fluctuations need to be reduced.
6 Use cases

In order to deliver a proof of concept for this research, two use-cases are executed, which show that real-time measurements can be used to real-time optimize the printing process. Two elements are printed using the developed optimization module. A third use-case is performed that shows the end results regarding the secondary goal of this research which is to develop a method for logging and analyzing of measurement and variable data. The data of the first two use-cases is used for the third use-case.

Elements printed in the two use-cases have no pre-programmed Z-coordinate movements. All movements are based on the real-time measurements and are performed by the optimization modules developed in this research. Both the filtered response module and the PID response module improve the performance of the basic optimization module. The PID controller positively affects the overall shape, reduces the noise and causes the desired delay. The filtered response module reduces the fluctuations of the printer without significantly changing the overall shape of the response. The PID response module is combined with the filtered response module to form a new PID-filter response module. In this combined module, the fluctuations of the PID response module are further reduced by using a filtered value as input for the PID-controller. This combined optimization module is used to perform the use-cases of this research project. The PID-filter response module used for the use-cases is shown in Appendix D1. A second module is used to turn the module off during the printing session. This module is added to Appendix D2.

For the first use-case two cylinders are printed which are compared to each other. The cylinders have a radius of 250mm and a height of 50 layers. A nozzle is used with dimensions of 10x40mm. This means that layers are extruded with a width of 40mm and a height of 10mm. Theoretically, cylinders are printed that have a height of 50x10=500mm. Because of the small dimensions of the cylinders the layers are loaded relatively soon after printing which results in a short drying time before loading those layers. The cylinders are therefore expected to deform significantly due to their own weight.

The first baseline cylinder is printed without using the real time optimization module. Instead, pre-programmed Z-movements are used. Between every layer, the height of the printer is increased with 10mm (the expected layer height). The second cylinder is printed without any pre-programmed Z-movements. In this case the PID-filter response module is used to real-time adjust the height of the printer, in order to constantly maintain the desired distance between the nozzle and the previous layer. The desired distance is 10mm. As shown in Figure 7 the expected layer height will most likely differ from the actual layer height, as a result of deformations of the layers. For the first cylinder, which is printed with preprogrammed z-movements, this means that the actual distance between the nozzle and the previous layer will most likely increase over time. This will result in dropping of the concrete on the previous layer, which can cause an uneven loading of the previous layers and affect the overall stability of the object. This can even lead too collapsing of the object during printing. The second cylinder is printed using the PID-filter response module, which will keep the nozzle at the desired distance from the previous layers at all time. Since only minor height adjustments are required within a
short time frame (maximum of 1mm in the first layer) a very low P- and I-coefficient are used (P=0.2 I =0.2) to reduce the fluctuations of the printer to a minimum. These fluctuations reduced even further by using a filter factor (FF) of 50. The values of these settings are chosen based on experience gained from previous test results. The cylinders are compared to determine whether using the real time optimization module will lead to an improved end result. The print path script of the first baseline cylinder is added in Appendix D3 and the print path script of the second cylinder is added in Appendix D4.

In the second use-case, the real time optimization module is used to directly print on a 3D surface without any pre-programmed Z-movement. This 3D surface could represent a non-flat building site. Only X- and Y- coordinates of movements are pre-programmed. The 3D surface consists of multiple elements placed on the programmed print path. (Figure 64). Using the optimization module, the printer will adjust his height real time to the elements on his path. The elements used are similar to the elements used in the third test scenario in maximum height and maximum slope angles (Figure 26). Since the combined PID-filter response module is used, new tests are performed on the third scenario using the combined module to determine the desired P, I and filter factor settings. The print path script used for the second use case is added in Appendix D5.

Figure 64: 3D surface used for the second use-case
6.1 Determine PID-filter response module settings for the second use-case

To determine the desired settings for the optimization module, six tests are performed with different combinations of multiple P-coefficient, I-coefficient and Filter Factor (FF) settings. Tests with the following settings are performed: (P=0.4 I=1.5 FF=15), (P=0.2 I=2 FF=15), (P=0.2 I=2.5 FF=15), (P=0.2 I=1.5 FF=25), (P=0.2 I=2 FF=25), (P=0.2 I=2.5 FF=25). The chosen settings are selected based on experience with the previous optimization modules. The results of those tests compared to the baseline graph and the basic optimization module are shown in Figure 65.

![Figure 65: Results of the PID-filter response module tests](image)

The test results show that, as expected, the PID-filter response module in all the tests has an improved performance compared to the basic optimization module. All the lines of the different tests are very similar during the first flat surface (0 to 5s). However, the tests with the FF of 25 have a smaller fluctuation (orange, yellow and pink line). The tests with similar I-coefficients have the same overall shape and only differ slightly from each other.

All the lines remain similar on the rising slope of the first element (6 to 7s), but then differences between the lines start occurring. At the horizontal surface of the first height element (7.5 to 8s), the tests with the lowest I-coefficient of 1.5 (blue and orange line) reach a maximum height of around -1522mm, which is almost 8mm lower than the baseline graph. The lines with an I-coefficient of 2 (yellow and purple lines) reach a maximum height almost similar to the baseline graph (2mm difference). The lines with the highest I-coefficient of 2.5 overshoot the height of the baseline graph by up to 8mm. On the decreasing slope of the first height element (8 to 9s),
all the lines almost perfectly follow the line of the baseline graph. However, when approaching
the lowest point differences start occurring between the lines. The blue and orange lines (I=1.5)
start slowing down their decline earlier and more gradually compared to the other lines, which
increases the difference between them and the baseline graph (I=2). The purple and yellow lines
are more similar to the baseline graph. The pink and green line (I=2.5) undershoot the baseline
graph, which means that the printer would crash into the previous layer or the surface below
when used during printing. At the second height element similar, effects occur compared to the
first height element. However, due to the lower height of the second element all effects are
decreased in scale.

Since the lines with an I-coefficient of 2 (purple and yellow line) better follow the baseline graph
compared to the lines with an I-coefficient of 1.5 (blue and orange line), an I-coefficient of 1.5
will not be used for the use-case. The lines with an I-coefficient of 2.5 (pink and green line) over-
and undershoot the baseline graph. This will either cause the nozzle to be too far above the
surface or be too close to the surface below, which could crash the printer. Therefore, not an I-
coefficient of 2.5, but an I-coefficient of 2 will be used for the second use-case. Since the lines of
both of the test with an I-coefficient of 2 are very similar, the one with the highest FF of 25 will
be used since using this setting will lower the fluctuations of the printer. This means that the
settings used for the second use case are (P=0.2 I=2 FF=25)
6.2 Use-Case 1: print cylinder with no preprogrammed Z-movement

The two cylinders printed for this use case are shown in Figure 66. The pictures show both cylinders during printing at a point where they both reached a similar height.

The baseline cylinder, which is printed without using measurements or the real time optimization module, is shown on the left. The cylinder clearly shows deformations in the layers that have caused the middle part of the cylinder to expand to the sides. This expanding to the sides and compressing of the layers creates a difference between the expected and actual layer height. The actual layer height is smaller than the expected layer height, which decreased the total height of the element. Since the Z-movements are preprogrammed based on the expected layer height, this results in an increase in distance between the nozzle and the previous layer over time. The increased distance causes the next layer to be placed less accurately which again increases the instability of the entire element and causes even more deformations in the layers. The deformations and therefore the distance from the nozzle to the layers keeps increasing during printing until at 35 layers the deformations and weight of the concrete caused the cylinder to collapse before 50 layers could be printed.

The second cylinder printed with the real time optimization module is shown on the right. During printing, any small deformations in the layers are directly responded to by adjusting the height of the printer to maintain the desired distance from the nozzle to the layer below. This causes each layer to be perfectly placed on the previous layers, which prevents undesired deformations and reduced quality of the placed concrete caused by dropping the concrete on the previous layer. The results of a constant and desired distance between the nozzle and the previous layer can be seen by comparing this cylinder and the previous one. This cylinder shows less expansion to the sides and more printed layers, even though a similar height is reached and the nozzle is still perfectly at the desired distance from the previous layer. All of the intended 50 layers are printed without collapsing of the element. This shows that using the PID-Filter
response module improved the quality of the printed cylinder compared to the baseline cylinder.

The difference between the two cylinders is most likely not just caused by the effect of the real time optimization module. What also affected the end-result can be explained using Figure 67. The desired distance between the nozzle and the previous layer (R26) is set at 10mm in the optimization module. This distance is set at 10mm since this is the expected layer height and a similar distance is pre-programmed to print the baseline cylinder. However, during the use-case tests a nozzle switch is performed, which means that the measurement device needed to be detached and then attached to the new nozzle. In this process, the measurement device is most likely placed a little bit higher on the nozzle. This means that the height distance between the measurement device and the nozzle increased (R25). This means that the total desired distance from the nozzle to the layer (R22) needs to be increased to keep the nozzle at the desired distance from the previous layer. However, since R22 was still at the programmed value the actual distance between the nozzle and the previous layer was slightly reduced and the layers were compressed slightly. This explains the fact why more layers are printed even though a similar height is reached and this most likely also improved the strength and stability properties of the entire element. Even though this affected the end-result, this use case still shows the capability of the real time optimization module to respond to the deformations in the cylinder by maintaining the desired distance between the nozzle and the previous layer.

Figure 67: Effect of slightly changing the height position of the measurement device

Within this use case the input of the printer is constantly adjusted during printing to be able to create the desired output. All the real time adjustments are performed completely automated, which creates a very efficient printing process.
6.3 Use Case 2: Print on an uneven 3D surface with no preprogrammed Z-movement

In this second use case, the real time optimization module is used to print directly on a 3D surface without any pre-programmed Z-movement. The PID-filter response module is used with the settings determined from the previous tests (P=0.2 I=2 FF=25). Figure 68 shows a picture during the printing of the second use case. The optimization module responds very smoothly to the 3D surface by applying a Z-offset to maintain an almost constant distance between the nozzle and the surface blow. Even though height elements with varying slope angles and maximum heights are used, the layers are very accurately placed on the surface below, without any significant deformations to the printed layers (Figure 69). This use-case shows that, using the correct settings, the PID-filter response module is capable of using real time measurements to successfully adjust the Z-offset to respond to a variety of 3D elements without the need to create complex 3D print path scripts. This extends the possibilities and field of application for the 3DC-printer.

Figure 68: Printing directly on a 3D surface without any preprogrammed Z-movement
After three printed layers the printer suddenly started moving up rapidly. A screenshot of a video recording at the point where the printer started moving up is shown in Figure 70. Because of this, the print session was aborted before the five programmed layers were printed. This most likely happened because of an error in the measurement system or due to concrete getting on or in front of the measurement sensor, which caused incorrect measurement values. The measurement values were most likely reduced to 0 which means that the PID-filter response module reacts to this by increasing the Z-offset, which moves the printer up. The printer was moving up but the measurement values remained 0, so the printer keeps adjusting the Z-offset and the height of the printer keeps increasing. Even though the print session was aborted because of this, the first three layers show the optimization module was successful in printing directly on a 3D surface.
6.4 Use Case 3: Logging and analyzing variable values in a 3D visualization

The developed method to analyze the variable data in a 2D graph is used extensively throughout the research project. This method was used to analyze the results of all the tests, to determine the performance of all optimization response modules. This allowed for accurate analysis and comparison of the performance of different modules. Using this method, required improvements to the basic optimization module are determined and then realized. The method is used not only to analyze the test results, but also to identify incorrect calculations by logging and analyzing many R-variables used in the calculations during development of the modules. This shows that this method is developed not only for future analysis of data, but is also of vital importance during the development of this research.

In this use case, the developed method to log and analyze variable data is used to analyze the results of the two use cases using the 3D visualization. For both use cases, seven variables are logged and their values at the point of the marker are shown. Figure 71 shows the 3D visualization of the element printed in the first use case. The raw measurement values are projected on the visualization with a black and white gradient. The visualization shows that data is missing in the model. As explained in chapter 4, the SINUMERIK operating machine is only able to log data for a short period of time. Therefore, the SINUCOM offline trace tool installed on a separate computer is used to perform the trace. However, due to unknown reasons the SINUCOM trace tool stops tracing after a varying amount of time. This prevented it from performing one trace for the entire print session. Instead, three separate traces are performed and the XML-files are combined into one XML-file. However, since it takes time to store the XML-file and restart the Trace function, data is missing. Data of the first layers and the final layers of the printed elements are logged. Using the available data, the printed element can still be analyzed.

Figure 71: 3D visualization of the element printed in the first use case
Apart from the first three layers there is a very constant measurement value. The small differences in the black and white gradient are the result of the noise present in the sensor. The Z-axis coordinates very gradually increase, with almost no fluctuation of the printer in the Z-direction. This shows that the PID-Filter response module ensures that the printer does not fluctuate. The print speed is very constant along the entire element. In Figure 72, the as-planned 3D model is added to the 3D visualization (as-built), which has an expected layer height of 10mm and thus a total height of 500mm. This clearly shows the effect of the real time optimization. The total height of the element is much smaller because of a lower actual layer height. The first three layers show a higher difference in the measurement values. Differences occur especially at the marked point, where the transition is made from the first printed layer to the second. The measurement value at that point is only 19mm, compared to the desired distance of 32mm set in the PID-filter response module. Since very low PID settings and a high filter value are used, the printer only responds gradually to the first printed layer. This causes the printer to compact the printed layers and slightly spread them out. This effect gradually decreases over the next three layers, until the desired distance is achieved and the compacted layers create an almost flat surface. This decreases the layer height of the first few layers, and therefore further decreases the height of the printed element, which creates a steady base for the upcoming layers.
Figure 73 shows the 3D visualization of the layers printed on the 3D surface from the second use case. The shape of the printed layers, caused by the height elements of the 3D surface, is clearly visible. There is a very constant distance between the layers, which shows that the layers are placed very accurately on top of each other. Again, the measurement value variable is projected on the visualization with the black and grey gradient. The flat surfaces only show minor fluctuations in the measurement value and a very constant Z-coordinate. The arrows mark the movement direction of the printer. Every time the height of the 3D surface increases, the measurement values decrease (lighter gradient). The sensor measures the height increase, which means that the measured values decrease, since only after a small time-frame a delayed response is caused by the PID-filter response module. This is the exact opposite of when the height decreases. In this case the measured values increase, which causes the printer to start moving down again with a small delay. The measurement value at the marker is only 45, since the height increases at this point. This creates a reduced value compared to the desired distance set in the PID-filter response module of 53mm.

Figure 73: 3D visualization of the layers printed on the 3D surface in the second use case

Beside the inability of the SINUCOM trace software to perform a long lasting trace, this use case shows that the developed method of using a 3D visualization can be successfully used to analyze the printing process. In the future, more important variables and extra measured values can be logged and displayed on the 3D visualization, to gain more knowledge of the printing process.
7 Conclusion and recommendations

The main goal of this research project is determined within the research proposal and is explained in the first chapter of this report: Creating a proof of concept that real time measurements can be used to automatically adjust the input for the 3D concrete printer to optimize the output of the printing process in real time. During the research a secondary goal is added to this project: Creating a method for logging and effectively analyzing variable data. This goal will help gain more insight in and better control of the 3DC printing process. In this chapter a conclusion, is drawn whether these goals are achieved within this research. Following up on the conclusion, recommendations are given for future research regarding the topic of this research project.

7.1 Conclusion
The secondary goal, which is to create a method for logging and effectively analyzing variable data, is achieved. A method is developed that allows for effectively analyzing variable data, directly retrieved from the 3DCP operating machine, that meets the requirements defined in this research. The method creates two possible ways to analyze the data: In a 2D graph or in a 3D visualization of the printed element. All the results of the tests performed to determine the performance of the different optimization modules in chapters 0 & 6, are successfully analyzed using the 2D graph method. By analyzing the results, the weak points of the optimization are determined. Based on the weak points, improvements to the optimization module are made. The third use case and the example in chapter 0 show that analyzing the variable data through a 3D visualization of the element enables the user to directly relate the variable values to specific locations on the printed element. The developed method is able to analyze different types of variables, a varying amount of variables and import external data. This creates a flexible method that is future proof and usable for users with different requirements. Use Case 4 does show that the logging method is limited, due to the problems occurring when attempting to log data for a longer period of time.

The main goal, to create a proof of concept that real time measuring can be used to automatically adjust the input for the 3D concrete printer and use this to optimize the output of the printing process in real time, is achieved. The use cases performed in this research show that the output of the printing process is indeed optimized by creating adjustments to the Z-offset of the printer based on real time height measurements. In the first use case, using the real time optimization module, a circular element is printed with improved strength and stability properties. The second use case shows that real time optimizations improve the current 3DCP-process by creating the possibility to print directly on 3D surfaces. Both these use cases show that using real time optimization, a more efficient and improved printing process can be achieved.

Looking at the sub questions defined for this research, the conclusion is drawn that all sub questions are answered. The selected output parameter to measure: The distance between the nozzle and the previous layer, is indeed a critical output parameter that can be measured. These measurements can be used to optimize the output in real time. Based on the requirements a successful measurement system is developed, which is able to real time and wirelessly send this
measurement data to the operating machine. Using these measurements, adjustments made to the selected variable: Z-coordinate of the printer, successfully created an optimized output of the printer and extended the possibilities of the 3DC printer.

The only sub question that cannot be fully answered is whether automatically generated adjustments to the input result in the desired output. When looking at the second use case the desired output can be described as successfully printed layers on a 3D surface. This means that in this case the adjustments to the input, the Z-coordinate, resulted in the desired output. For the first use case, the desired output can be described in multiple ways. When the desired output is a circular element consisting of 50 printed layers, the desired output is achieved. However, when the desired output is a circular element consisting of 50 printed layers in which each layer has the expected layer height of 10mm, the desired output is not fully achieved. Adjustments to the Z-coordinate did not result in the expected layer height, but responded to the actual layer height by maintaining a fixed distance from the previous layer, enabling precise placement of the next layer. Even though this sub question cannot be fully answered, the main research questions: “How can real time measurements be employed in the automated optimization of 3D concrete printing technologies?”, is answered and the main goal of the research is achieved.

This research provides a proof of concept which can be used as a basis for more research regarding the subject of this report. Within this research the 3DCP process can be further optimized using the provided method, by adding more sensors and increasing the complexity of the real time optimizations.
7.2 Recommendations

This research can be used as the starting point for more research into this topic. Based on the experience gained during this project, the most important recommendations regarding this future research are given. The recommendations are grouped in four different topics.

7.2.1 Improve the current measurement system

Improvements can be made to the current measurement system, to improve and extend the possibilities of the real time optimizations. First of all, the current 1D measurement sensor can be upgraded to a sensor which has less noise and thus a higher accuracy. This will improve the performance of the response modules developed in this research. Less filtering of the raw measurement values will be required to reduce the fluctuations of the printer. This will enable to modules to maintain a more constant distance from the surface below. A delay is present in the transfer of measurement data. Reducing this delay will improve the ability to better and more quickly respond to a measured value. The current method to transfer data can be improved to reduce this delay by using more expensive and industrial equipment for the wireless transmission of the data. Both improving the accuracy of the sensor and reducing the delay are not only costly but may require extensive knowledge about these subjects. Therefore, searching for a partner company with expertise in sensors and real time transmission of data can take the topic of this research project beyond the level of proof of concept. By involving a company in this project, they can not only contribute with their knowledge, but they might also be interested in financially contributing by supplying equipment for an improved measurement system.

7.2.2 Extend the variables adjusted to create a real time optimization

Besides the Z-coordinate, other variables can be adjusted in real time to perform an optimization as well. In paragraph 5.1: Printer variables for optimization, the movement speed of the printer and the pump-frequency of the pump are already mentioned as possible variables for optimization. With a fixed pump frequency, adjusting the movement speed results in more or less material placed at one location. As shown in Figure 6, adjusting the movement speed influences the layer height. A response module, similar to the ones created for this research and using the same measurement system, can be created to adjust the printer speed to achieve the desired layer height. This will be an improvement to the current printing process, which does not require any hardware improvements to the current system. PID-control might also be usable in this optimization to reduce rapid changes in the movement speed, which can affect the quality of the printed layers. Instead of using the movement speed of the printer, the pump frequency can also be used to create a real time optimization. Increasing the pump frequency will increase the amount of material extruded per time-frame. When using a fixed movement speed, this will again result in more material placed at one location. This can be used to influence the layer height. Increasing the pump frequency will most likely slowly increase the amount of material extruded as a result of the pressure that needs to build up in the entire length of the hose. Again, PID-control can help to gradually adjust the pump frequency over time until the desired layer height is achieved, without constant overshooting of the desired layer height resulting from major changes in the pump-frequency. In both cases only minor adjustments are required to the PID-response module developed in this research.
7.2.3 Extend the measured output parameters

Extending the output parameters on which measurements are performed beyond just the height of a layer can create many new possibilities to real time optimize the printing process. By using a sensor or a combination of sensors that are capable of providing top-profile data or even a 3D data of the entire section of a layer, data about the width of a layer and even the sectional shape of a layer can be derived. Using this measurement data, variables can be adjusted to change the width or sectional shape. Influencing these output parameters can change the geometrical properties of the printed element, but also achieve a different shaped texture on the surface of a printed element. For the measurement system to be able to process data from a 3D scanner, a complete redevelopment of the measurement system will be required. However, extending the current measurement system with multiple similar 1D sensors that are placed at different positions and angles might also allow generation of data about the width of a layer. The mentioned output parameters all relate to the shape of a layer. More output parameters like the temperature of the placed concrete or the temperature and viscosity of the concrete in the nozzle can all be measured. Research can be performed to determine which output parameters can be measured in real time and which variables can be adjusted to real time influence those output parameters.

7.2.4 Use a third generic box to store data and to perform complex calculations and simulations

All real time optimizations within this research are performed directly from the operating machine of the 3DC-printer. As mentioned before, the possibilities of this system are limited regarding complex calculations and simulations. Using an external third generic box can extend the possibilities of optimizing the printing process using real time improvements. The third generic box can be used to real time create a 3D model of the printed element using coordinate and measurement data from the operating machine. Similar to the developed method to analyze variable data in the 3D visualization, information about variables can be stored at a specific location in the 3D model. This creates the possibility to not only be able to respond directly to measurement values, but also to respond to stored measurement data in the 3D model. The third generic box can be used to perform complex calculations and simulations, using information from the 3D model. This can be used to determine which adjustments are required to one or multiple variables in order to achieve the desired output. Data can be real time extracted from and inserted in the operating machine through the PLC. This can be achieved with a method similar to the method used within this project to transfer the measurement data from the second Arduino to the operating machine within this project.
8 References


9 Appendices
<table>
<thead>
<tr>
<th>Measurement system</th>
<th>Measurement device</th>
<th>Advantage</th>
<th>Disadvantage</th>
<th>Price</th>
<th>Precision</th>
<th>Range</th>
<th>3D pointcloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOF (Time of Flight)</td>
<td>multiple cameras</td>
<td>3D point clouds acquired at video frame rates. Suitable for real-time measuring</td>
<td>accuracy can be high, but very expensive</td>
<td>10,000-15,000€ (1 camera = +5,000€)</td>
<td>Conditioned fixed setting (4-6mm) Similar project (cm)</td>
<td>&lt;7.5m</td>
<td>yes</td>
</tr>
<tr>
<td>Structured light</td>
<td>Projector + Camera</td>
<td>high accuracy</td>
<td>real-time capability is difficult; measurements only taken at illuminated pattern points, not usable outdoor (very controlled light environment needed), high software complexity results in slow response time</td>
<td>1000€ but increases when more accuracy is required</td>
<td>μm - cm</td>
<td>&lt;1m</td>
<td>yes</td>
</tr>
<tr>
<td>Stereo</td>
<td>2 cameras</td>
<td>low costs</td>
<td>accuracy drops when used as real-time measurement device. Computationally intensive algorithms. Requires sufficient intensity and color variation</td>
<td>400€ (low costs, Normal camera’s can be used)</td>
<td>cm</td>
<td>4-6m</td>
<td>yes</td>
</tr>
<tr>
<td>1D/2D Distance Laser</td>
<td>distance laser(s)</td>
<td>accuracy, easy data retrieval</td>
<td>no 3D model generated, cannot be used for more data generation</td>
<td>100-200€ and higher</td>
<td>μm - mm</td>
<td>40m</td>
<td>no, but possibilities for top-profile data</td>
</tr>
<tr>
<td>RGB-D Camera</td>
<td>RGB-D camera(s)</td>
<td>Very fast data processing so usable for real-time measuring</td>
<td>Not very accurate, especially at longer distances. Accuracy drops when used outside</td>
<td>minimal 400€ (200€ per camera)</td>
<td>Kinect 4m: 27.5mm</td>
<td>5m</td>
<td>yes</td>
</tr>
<tr>
<td>Photogrammetry</td>
<td>multiple cameras</td>
<td>Normal cameras can be used</td>
<td>High accuracy can be achieved with sophisticated algorithms, but not for real-time applications. Difference in texture is required</td>
<td>2000€ Cheap in a simple form, but when high accuracy is required many cameras are needed</td>
<td>When many cameras are used it can result in high precision, but not as good as 3D laser scanner</td>
<td>-</td>
<td>yes</td>
</tr>
<tr>
<td>3D Laser Scanner</td>
<td>3D scanning device</td>
<td>High accuracy</td>
<td>expensive, real-time operation is not possible</td>
<td>High costs (45,000€)</td>
<td>2mm</td>
<td>0.6-330m</td>
<td>yes</td>
</tr>
</tbody>
</table>

Appendix A2  Arduino 1 script – Transmitter Arduino

This appendix shows the Arduino script of the first Arduino, which performs the measurements, displays the measurements on the LCD display and transmits the data to the second Arduino. Comments which explain what the lines of the script do are added in green. The script starts with the used libraries. Links to find more information how the libraries work and how you can use them are added behind the libraries. Then the settings are made for all the different components of the first part of the measurement system. A setup of the measurement system is performed, Finally the main loop starts in which measurements are performed, transmitted to the second Arduino and displayed on the LCD display. The main loop transfers 50 measurements per second to the second Arduino.

//*******************************************************************************
// USED LIBRARIES
//*******************************************************************************
//MORE INFORMATION ABOUT HOW TO USE THE LIBRARIES CAN BE FOUND IN THE ADDED LINKS
#include <RF24Network.h>  //https://github.com/maniacbug/RF24Network
#include <RF24.h>  //https://github.com/maniacbug/RF24
#include <SPI.h>  //https://www.arduino.cc/en/reference/SPI
#include <Wire.h>  //https://www.arduino.cc/en/Reference/Wire
#include <LiquidCrystal.h>  //https://www.arduino.cc/en/Reference/LiquidCrystal
#include <SparkFun_VL6180X.h>  //https://github.com/sparkfun/ToF_Range_Finder_Sensor-VL6180/blob/master/Libraries/Arduino/src/SparkFun_VL6180X.h

//*******************************************************************************
// SETTINGS
//*******************************************************************************

// Define structure of packaged message
struct sensor_bericht{
    unsigned long ms;  // Current time in ms
    unsigned long sensor_waarde;  // The read sensor value
};

RF24 radio(9,10);  // Transceiver is connected using pins 9 and 10
RF24Network network(radio);  // Define the network

const uint16_t deze_node = 1;  // Adres of the sensor node
const uint16_t andere_node = 0;  // Adres of the receiver node
const unsigned long interval = 1;  // Interval for transmitting data (ms)
const unsigned long interval2 = 500;  // Interval for displaying data on display (ms)
unsigned long tijd_laatste_bericht;  // Time-stamp at which the last data was transmitted

//SENSOR CONNECTION AND DISPLAY/
#define VL6180X_ADDRESS 0x29  //Defining means that you give a name to a constant value. Value = 0x29
VL6180xIdentification identification;
VL6180x sensor(VL6180X_ADDRESS);
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

//*******************************************************************************
// SETUP OF THE MEASUREMENT SYSTEM
//*******************************************************************************

void setup(void){
    Serial.begin(9600);  //Start serial connection to sensor
    lcd.begin(16,2);  //Start LCD display
    lcd.print("Hello World!");  //Print starting message
    SPI.begin();  //Start serial peripheral interface data protocol
    radio.begin();  //Start Network
    network.begin(0,deze_node);  //Start wireless connection with the other Arduino
}
Serial.println("Sensor network transmitter (sensor node) start..."); //Print start network transmitter for first arduino (sensor)

//SENSOR PART OF THE SETUP/
Wire.begin(); //Start I2C library for communication with sensor
delay(100); // delay .1s
sensor.getIdentification(&identification); // Retrieve manufacture info from device memory
Serial.println("Sensor.VL6180xInit()=");
if(sensor.VL6180xInit() != 0){
    Serial.println("FAILED TO INITIALIZE"); //If failed to initialize gives an error
}
sensor.VL6180xDefaultSettings(); //Load default settings to get started.
    delay(1000); // delay is

//******************************************************************************
// MAIN LOOP: GET SENSOR DATA, DISPLAY DATA ON DISPLAY, TRANSMIT DATA TO SECOND ARDUINO
//******************************************************************************
void loop(void)
{
    //TRANSMIT SENSOR DATA/
    network.update(); // Update the network connection
    unsigned long tijd_nu = millis(); // update current time
    if( tijd_nu - tijd_laatste_bericht >= interval ) //Calculate if the interval time for transmitting data is achieved
    {
        RF24NetworkHeader header(andere_node); //Insert the adress of the receiver node
        unsigned long sensor_waarde = sensor.getDistance(); //Update the sensor_waarde with the newest measured distance
        sensor_bericht bericht = { millis(), sensor_waarde }; //Create a message with the current time in ms and the sensor value
        bool ok = network.write(header,&bericht,sizeof(bericht)); //Transmit the message to the second arduino
        tijd_laatste_bericht = millis(); //Update the time when the last message was send
    }

    //UPDATE LCD DISPLAY WITH SENSOR VALUE/
    if( tijd_nu - tijd_laatste_bericht >= interval2 )//Calculate if the interval time for displaying sensor value is achieved
    {
        lcd.clear(); //clear the display
        lcd.println(sensor.getDistance()); //Update display with latest sensor value
    }
}
Appendix A3  Arduino 2 script - Receiver Arduino
This appendix shows the Arduino script of the second Arduino which receives the measurement data, converts the data format and transfers the data to the PLC. Again, comments that explain what the lines of the script do are added in green. The data format in which the SIEMENS PLC requires the measurement data is Little Endian. The format of the raw measurement data in the Arduino is Big Endian. Therefore, the Settimino.h library is used to transform the format of the data from Big Endian to Little Endian. More information about how this library works and how the functions work used in this script can be found on (Nardella, settimino.sourceforge.net, 2017). The script starts with the used libraries. Then the settings for the wireless transfer need to be determined. In the next part, the Settings are determined for writing data to the PLC through an Ethernet connection. When adding an extra measurement device these settings are important, since the local address of the Arduino will need to be changed in order to successfully transfer data.

IPAddress Local(192, 168, 214, 202); // Local Address of the Arduino

After this is done, the setup for the second Arduino is executed. When this is done, the main loop starts in which the measurement data is received, transformed and transferred to the PLC. When extra measurement data is inserted into the PLC, it is very important to insert this data to another DBB-number, which will then be used for receiving the height measurement data. The line in which the DBB-number is defined is the following line:

Result = Client.WriteArea(S7AreaDB, 0, 0, Size, bytes); //Request access to DB, DB number = 0, Start from byte number 0, required size, measurement value

The second item between the brackets is the DBB-number. This number needs to be changed when this script is used to insert other data.

/*********************************************************
* USED LIBRARIES
*************************************************************************
//MORE INFORMATION ABOUT HOW TO USE THE LIBRARIES CAN BE FOUND IN THE ADDED LINKS
#include <SPI.h>  //https://www.arduino.cc/en/reference/SPI
#include <Ethernet.h> //https://www.arduino.cc/en/Reference/Ethernet
#include "Settimino.h" //http://settimino.sourceforge.net/
#include <LiquidCrystal.h> //https://www.arduino.cc/en/Reference/LiquidCrystal
#include <RF24Network.h> //https://github.com/maniacbug/RF24Network
#include <RF24.h> //https://github.com/maniacbug/RF24

/*********************************************************
* SETTINGS FOR WIRELESS TRANSFER
*************************************************************************

// Define structure of packaged message//
struct sensor_bericht
{
    unsigned long ms;        // Current time in ms
    unsigned long sensor_waarde; // The read sensor value
};

RF24 radio(9,10);  // Transceiver is connected using pins 9 and 10
RF24Network network(radio);  // Define the network

const uint16_t deze_node = 1;  // Adres of the sensor node
const uint16_t andere_node = 0; // Adres of the receiver node
// SETTINGS FOR WRITING DATA TO PLC THROUGH ETHERNET CONNECTION
//******************************************************************************
#define DO_IT_SMALL  //Use small and fast data access
byte mac[] = {
  0x90, 0xA2, 0xDA, 0x0F, 0x08, 0xE11 };  //Create byte for writing data through the Ethernet connection
IPAddress Local(192, 168, 214, 202);  // Local Address of the Arduino
IPAddress PLC(192, 168, 214, 1);  // PLC Address of the SINUMERIK
byte Buffer[1024];
S7Client Client;
unsigned long Elapsed;  // To calculate the execution time
//******************************************************************************
// SETUP FOR MEASUREMENT SYSTEM PART 2
//******************************************************************************
void setup()
{
  //SETUP WIRELESS TRANSMITTING AND DATA TRANSFER TO PLC/
  Serial.begin(9600);  //Start serial connection
  while (!Serial)   //wait for serial port to connect
    ;
  Ethernet.begin(mac, Local);   // Start the Ethernet Library with byte and local address of Arduino
  delay(2000);  //Delay required for setup time
  SPI.begin();  //Start serial peripheral interface data protocol
  radio.begin();  //Start Network
  network.begin(90, deze_node);  //Start wireless connection with the other Arduino
  Serial.println("Sensor netwerk ontvanger start...");
}
// MAIN LOOP: RECEIVE MEASUREMENT DATA FROM FIRST ARDUINO, TRANSFORM FORMAT OF DATA, TRANSFER TO PLC
//******************************************************************************
void loop()
{
  //RECEIVE SENSOR DATA/
  network.update();  // Update the network connection
  while (network.available())  //Check if data is available on the network
  {
    RF24NetworkHeader header;  //The header of the message has a 'RF24NetworkHeader' structure
    sensor_bericht bericht;  //The message has a 'sensor_bericht' structure
    network.read(header, &bericht, sizeof(bericht));  //Read the message from the network
    //TRANSFORM FORMAT SENSOR DATA FROM BIG ENDIAN to LITTLE ENDIAN //
    int Data = bericht.sensor_waarde;  //Asign sensor value to Data integer
    int Size, Result;  //Create Size and Result integer
    void *Target;
    void* ptr = (void*) Data;
    printf("%d", (int)ptr);
    int VAR1 = bericht.sensor_waarde;
    byte* bytes = (byte*) &VAR1;
    //BUFFER SETTINGS FOR DATA TRANSFER TO PLC//
    #ifdef DO_IT_SMALL  //DO_IT_SMALL is used since this is defined in the settings which has a small buffer and fast data access
    Size=1;
    Target = ptr;
    #else
    Size=1024;
    Target = &Buffer;
    #endif
    //WRITE TRANSFORMED MEASUREMENT DATA TO PLC//
    while (!Client.Connected)  //As long as there is a connection to the PLC write data
    {
      if (!Connect())  //Uses connect function to connect to the PLC
delay(10); // Small delay in ms

MarkTime(); // Get the delayed time
Result=Client.WriteArea(S7AreaDB, 0, 0, Size, bytes); // Request access to DB, DB number = 0, Start from byte number 0, required size, measurement value

if (Result==0) // When result = 0 a connection is achieved and the showtime is given and the buffer is dumped
{
    ShowTime();
    Dump(Target, Size);
}
else // When result is not 0 no connection is achieved and the error is displayed
    CheckError(Result);

//*******************************************************
// FUNCTIONS USED IN THE SCRIPT ABOVE
//******************************************************************************

// CONNECT TO PLC/
bool Connect() // To connect to PLC
{
    int Result=Client.ConnectTo(PLC, 0, // Rack (see the doc.)
          2); // Slot (see the doc.)
    Serial.print("Connecting to ");Serial.println(PLC);
    if (Result==1)
    {
        Serial.print("Connected ! PDU Length = ");Serial.println(Client.GetPDULength());
    }
    else
        Serial.println("Connection error");
    return Result==0;
}

// DUMPS A BUFFER/
void Dump(void *Buffer, int Length) // DUMP A BUFFER
{
    int i, cnt=0;
    pbyte buf;
    if (Buffer!=NULL)
        buf = pbyte(Buffer);
    else
        buf = pbyte(&PDU.DATA[0]);
    for (i=0; i<Length; i++)
    {
        cnt++;
        if (buf[i]<0x10)
            if (cnt==16)
                cnt=0;
    }
}

// PRINTS THE ERROR NUMBER/
void CheckError(int ErrNo)
{
    Serial.print("Error No. 0x");
    Serial.println(ErrNo, HEX);

    // Checks if it’s a Severe Error -> we need to disconnect
    if (ErrNo & 0x00FF)
    {
        Serial.println("SEVERE ERROR, disconnecting.");
        Client.Disconnect();
    }
}

// PROFILING ROUTINES//
void MarkTime()
{
    Elapsed=millis();
}
//---------------------------------------------------------------------
void ShowTime()
{
    // Calculates the time
    Elapsed=millis()-Elapsed;
}
void Reverse2(void *ptr)  //TRANSFORMS BIG ENDIAN TO LITTLE ENDIAN
{
    byte *pb;
    byte tmp;
    pb=(byte*)ptr;
    // Swap byte 2 with byte 1
    tmp=*pb;
    *(pb+1)=*pb;
    *pb=tmp;
}
Appendix A4  Connection schematics measurement system part 1
Appendix A5  Connection schematics measurement system part 2

SIEMENS PLC - Ethernet cable - Ethernet Shield - NRF Wireless transceiver

3.3v - 3.3v
Gnd - Gnd
SCK - 13
MISO - 12
MOSI - 11
CE - 9
CSN - 8

Ethernet shield fits exactly on top of the Arduino

Arduino 2 - USB cable - Desktop
Appendix A6  How to attach and use the measurement system
This appendix explains stepwise how to attach part 1 and part 2 of the measurement system developed in this research on the 3DC-printer.

Part 1
1. Connect the measurement box (part 1) underneath the concrete input hose.
2. Attach the nozzle, with the measurement sensor.
3. Connect the 4-pins detachable connection so the colors match
4. Turn the measurement device on with the On/Off button
Part 2
5. Attach the USB cable to the Arduino and to the PC next to the printer.
6. Attach the Ethernet cable to the Arduino
7. Attach the Ethernet cable to either of the free connections of the hub which is in the top right of the Sinumerik cabinet. In the example the red marker connection port is used.
Appendix B  Logging and analyzing of variable data

Appendix B1  How to use the Trace function on the SINUMERIK

Step 3: Perform the trace function and export data, which consists of a few sub steps. These steps are further elaborated upon in this appendix. The sub steps are: The trace settings (step 3.1), performing and viewing the trace (step 3.2) and saving the trace (step 3.3).

Step 3.1: Trace settings

This step starts with the trace settings. These settings only need to be set once unless changes to the settings are required. This again helps to keep the process of logging simple and not adding unnecessary proceedings. These settings include several options:

- The variable settings: The selection of the variables to trace with multiple attributes
- The NC/PLC settings: The start and stop trigger settings, the sample rate settings

Before the variable and the NC/PLC settings can be set the Trace function needs to be opened on the SINUMERIK. This is done by pressing the “Menu Select” button and then pressing the “Diagnostics button”.

Open the Diagnostics window
After selecting the diagnostics function the “arrow” button needs to be pressed, which takes you to the location where the software key for the Trace-function is. By pressing the “Trace” button the Trace-function will be opened.

Open the Trace function
The next step is to choose the variables of which data needs to be traced. More variables can be added by using the “Choose variable” button if desired. An important attribute of a trace variable is the Event attribute. This attribute allows you to set when measurements on a variable should be performed. Measurements can be performed at cyclic events or at non-cyclic events (measurements are only performed at specific events). In this project, we want to monitor and log data during an entire printing session. Therefore, in this project only measurements are performed at cyclic events during the entire runtime of the print path. The Event OB1 cycle PLC results in data intervals of 8ms. Another important attribute of a trace variable is the Comment. The Comment allows you to give a name to a specific variable. This name is used further on in the logging process, from step 4, to import the data and to visualize it and should be unique for each of the variables.

Aside from the variable settings the NC/PLC settings need to be set. To get to these settings the button on the right side “Settings” needs to be pressed. The trace can be started by using a softkey. This softkey allows you to manually start the trace from the NC. But the trace can also be started based on a value of a variable. In this research the variable $AC_ACT_PROG_NET_TIME has been used, which is the net program runtime of the print path script. The trace will start when this value increases which means that the cycle of the print path program is started or continued. By using this variable for the start trigger, the trace is started simultaneously with the print path and the variable values are logged from the start. Similar settings are available for the stopping of the trace. The trace can also be stopped with a softkey, but also at a certain elapsed time, when the storage limit of the SINUMERIK reached or again at a specific condition of a variable. In this case, similar to the start trigger, the net program runtime variable is used. When the value of this variable is falling this means that the entire print path program is completed and the printing process is completed. By using this as a stop trigger for the trace, only data is generated during the actual printing and not before or after

<table>
<thead>
<tr>
<th>Variable</th>
<th>Comment</th>
<th>Color</th>
<th>Pen</th>
<th>Show</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AX_i1(X1)</td>
<td>X-axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AX_i2(Y1)</td>
<td>Y-axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AN_i1(Z1)</td>
<td>Z-axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AN_i2(C1)</td>
<td>C-position</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AC_ACT_PROG_NET_TIME</td>
<td>Program runtime</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SR[4]</td>
<td>AHU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$OB1 cycle PLC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Variable settings
printing. Other variables can also be used for the start and stop trigger. For example a specific X-coordinate has been used for the start and stop trigger in the tests performed in chapter 4,5 and 6. The location to where the data should be collected needs to be defined.

### Step 3.2: Performing and viewing the trace

When all the settings are correct, the actual trace and the execution of the print path program can start. First, click the “View Trace” button which opens the view trace window. Then press the “Start Trace” button. This does not actually start the trace, but this creates a process in which the trace-function awaits the trigger. As shown in the settings, the trigger for the start of the trace is an increase of the Net Program Runtime variable. Before the print path program is started by pressing the cycle start button, the SINUMERIK NC screen should be set to the “Program” screen. This step is essential, since the SINUMERIK machine does not have enough computing power to real time visualize the generated trace data and will otherwise crash/get stuck due to the large amount of generated data. It is therefore not possible to view the trace-graph during printing on the SINUMERIK machine. When the print path program is started by pressing the cycle start button, the trace will automatically start and when the program is finished the trace function will be automatically stop.
Open the View Trace window

Start the Trace
Step 3.3: Saving the trace
The SINUMERIK is only able to store one trace-file at the time. Therefore, after every trace the trace-file should be stored. This can be done by opening the Trace function by again clicking “Menu Select” → “Diagnostics”. This will automatically open the View Trace window of the Trace function. Due to the large amount of gathered data the opening of this screen takes some time and therefore patience is required. If any buttons are pressed during opening of this window the SINUMERIK machine will crash/get stuck. When the “View Trace” window is fully opened you can return to the Trace Settings window by pressing the “Back” button. The trace file can then be stored in an XML format on a USB-stick by pressing the “Save Trace” button. Choose the option: “Variables, settings and traced values” and select the location of the USB stick.

Store the Trace file
Appendix B2  How to use the SINUCOM offline trace tool
This appendix explains how the SINUCOM offline trace tool is used and how the PC is connected to the SINUMERIK. Since the content of the steps are very similar to the Trace function on the SINUMERIK, only a stepwise workflow is shown.

1. Connect the Ethernet cable to the PC and connect it to the Ethernet port in the middle left side of the SINUMERIK cabinet
2. Open the SinuCom folder on the desktop of the PC and start the “NC Connect Wizard”. This is used to create a connection between the PC and the SINUMERIK.

3. Select “840D solutionline” and click “Next”
4. Select “Engineering Tools” and click “Next”

5. Click “Next”
6. Click “Finish”

7. Open the SinuCom folder on the desktop of the PC and start the “SinuComNc”
7. When the starting screen opened click “OK”

8. Open the folders on the left panel and click “Machine data”. Then double click “MD block1”
9. Go to the “Diagnosis” tab and click “Trace”. This will open the Trace function.

10. The Trace function is opened. Click “Add” to add variables for logging.
11. Similar to the Trace function on the SINUMERIK, variables can be searched, selected and added to the Trace function. In this example the Path Velocity variable is added. Similar to the trace function on the SINUMERIK the Event should be OB1 cycle PLC.

12. Go the “Setup” tab and select “Waveforms...”
13. When the settings window is opened, the added variable is visible. The description can be changed to give the variable a name.

14. In the “Collection” tab the settings for the Trace function are set. Firstly, the “When To Start” box is set at “Immediately”. This causes the Trace function to start immediately when the start button is clicked. Similar to the Trace function on the SINUMERIK, a trigger can also be used. The “Trigger” tab will then become available. This is also the case case for the “When To Stop” box. Set the “Data Limit” box to the maximum (1e+007) KB to be able to perform a long lasting trace. Click the “Close” button when done.
15. Now the Trace function can be started and stopped using the “Start” and “Stop” button. Hitting the start button will directly start the Trace function and data will be gathered. The data is directly visualized in the Waveform screen.

![Waveform screenshot]

16. When the trace is stopped and thus completed, the data can be saved to an XML-file by going to the “File” tab → “Save As” → And then save it with the desired name on the desired location. Don’t forget to mark “With Data” so not only the settings will be stored, but also the gathered data. The XML-file is saved and can then be used for further analysis.

![Save Session dialog box]

![XML-file settings]

---

138
Appendix B3  How to import and XML-file into excel

In this appendix the workflow on how to import a logged XML file into the provided excel format created in this research is explained.

1. The empty excel format file needs to be opened and in the Developer tab the import button needs to be pressed from the XML component. When the Developer tab is not available, this tab needs to be activated first through the excel settings. The logged XML-file needs to be selected and will be automatically imported.

2. Each variable from the XML file has a f-number is mapped to their according column. On the right side, an extra column is created which looks for the description which belongs to the f-number variable. Each of these descriptions is displayed on top of the corresponding column.
3. In the second tab of the excel file a 2D graph, which displays all variable values on the Y-axis and the first column: Time on the X-axis. In order to display the variable names in the graph, the graph needs to be right clicked and the Select Data button needs to be selected.

4. The following window opens in which the selected 2 needs to be changed into a 1 to also incorporate the description name of the variables in the 2D graph. After this step the graph and the data can be used to analyze the variables of the 3DC-printing process.
Appendix B4  Grasshopper model
In this appendix a more detailed explanation is given about how the separate blocks of the Grasshopper model work and how they should be used.

Block 1 and 2

In Block 1: Data input, the user insert the XML data files. In the “Trace data file” box the name of the logged XML trace file, in this example TraceUseCase2.xml, is entered. In the second or third box, depending on whether the external data file has a smaller or bigger time interval, the external data file is entered. In this example that file is called TemperatureData.xml. All the used XML data files need to be in one predefined folder. The location of this folder is defined in the Python script of block 2. An example of how the external data file is created and how it needs to be formatted is added in Appendix B7.

The names of the XML data files serve as input for Block 2: Convert and combine all data. This block consists of one Python module. The entire script of the python module with explanation can be found in Appendix B5. Using the names the XML data files is imported and required logged data is extruded from the XML-files. The number_of_points, thus the amount of measurements in the XML-trace file are determined and exported. Aside from the number of points the names of the logged variables are determined and exported as all_variable_names. Both the number of points and the names of the logged variables serve as input for Block 3. All the variable data from the XML-trace file and the external data file are combined into one new
list with the name \textit{all\_variable\_information}. However, since the external data file often has a different time-interval compared to the XML-trace file the external data file first needs to be synchronized to the time stamps of the XML-data file. Two separate modules are created in the python script.

The first module is used when the interval of the external data file is bigger compared to the XML-trace file. This means that less measurements per time frame are available in the external data file. In order to synchronize the external data file to the XML-trace file, for each time stamp in the XML-trace file the correct external variable value is calculated and are combined to a new external data list. In order to calculate the correct external variable value for each time stamp, linear interpolation is used between the two external variable values whose time stamps are bigger and smaller compared to the time stamp of the XML-trace file. The bigger and smaller external variable time stamps are selected which are the closest to the time stamp of the XML-trace file.

The second module is used when the interval of the external data file is smaller compared to the XML-trace file. In order to synchronize the external data to the XML-trace file, for each time stamp in the XML-trace file the variable value is selected which has the time stamp which differs the least from the XML-trace file time stamp. Using the XML-trace file time stamp and the selected variable value a new external data list is created which the same time stamps as the XML-trace file.

When the external data file is synchronized both the data from the external data file and the XML-trace file are combined and exported to the \textit{all\_variable\_information} list.
In Block 3: Display settings, the user needs to define the display settings. One variable needs to be selected from the measured variable list which will be displayed on the 3D visualization in a black and white gradient. This measured variable list is created using the output of block two `all_variable_names`. The corresponding number of the desired variable needs to be entered in the yellow box “Select variable for display” Both the measured variable list and the number are exported and serve as input for Block 4. Secondly, a point needs to be selected with the Control Knob. On this point, a red marker is displayed in the 3D visualization and all variable data of that point is displayed. Before a point can be selected, the maximum value of the Control Knob needs to be manually adjusted by double clicking the Control Knob and change the maximum value of the knob to the number of points displayed below. The selected point value serves as input for Block 5, 6 and 7.

In Block 4, two components are present. The Item element selects a variable name from the measured variable list of Block 3 based on the number entered. The selected variable name serves as input for the second element of block 4 which is a Python-module. The entire script of the python module with explanation can be found in Appendix B6. Using the selected variable name and the `all_variable_information` list, a new list is created which only holds data of the selected variable for the black and white gradient. The list is exported as `gradient_variable` and serves as input for Block 5. The Python module extracts all X,Y and Z-coordinates from the data and combines each X,Y and Z-coordinate at the same time-stamp to a 3D point and exports a list with all the created 3D points with the name `point_list`. This list serves as input for Block 5 & 6.
To be able to display all variable values at the selected point in Block 7, the python module transforms the `all_variable_information` list into a Data Tree with multiple branches. The Data Tree is then exported with the name `Variable_data` and serves as input for Block 7.

**Block 5 and 6**

Block 5: Display 3D visualization and variable color gradient, creates a 3D visualization of the printed element using the 3D points in the `point_list`. This creates an cross (x) at the X-,Y-,Z-coordinate of each 3D point. Using `gradient_list` with the selected variable values, a black and white gradient is added to each of the crosses based on the value of the selected variable at that point. The lowest and highest values from the `gradient_list` serve as the range for the black and white gradient. For each 3D point, based on the value of the selected variable at that point, a color between black and white is calculated and assigned to the corresponding cross.

Block 6: Display Marker, uses the selected point number from Block 2 to select the corresponding 3D point from the `point_list`. At the location of this 3D point a Red Spherical Marker is displayed which shows the location of the selected point. This is also the location of which the variable values are displayed in Block 7.
In block 7: Display all variable values at marker, the value of each variable on the selected point of block 3 are displayed. For each displayed variable the same structure is used. The first structure, which displays the Time variable, is marked by the red rectangle. This structure is copied 12 times, which means that the grasshopper model is able to display up to 12 variables. This can amount can be increased if required. In the image above only 2 structures are shown.

In the first structure, which displays the Time variable, the 1st branch of the Data Tree from block 4 is selected. This tree holds the values of the Time variable. Based on the selected point number from Block 3, the corresponding Time variable value is selected from the branch. This value, combined with the variable name Time and an = mark are displayed at the absolute coordinate 0,0,0. Every time the structure is copied, the next branch is selected and the next variable is displayed. The location of display is shifted slightly in the Y-direction. This creates a displayed list with all variables.
import clr
import Rhino.Geometry as rg
import System.Xml
import Rhino
from Grasshopper import DataTree
import Grasshopper

#---------------IMPORTED LIBRARIES------------------
#---------------CREATE LIST FOR ALL VARIABLE INFORMATION-------------
all_variable_information_list = []
#---------------OPEN THE TRACE-FILE (SET THE LOCATION OF THE FOLDER)------------------
filename = r"D:\Data\Emiel\Dropbox\Graduation\Q2\Sinumerik\Tracing_and_logging\3D_visualization\Tracefiles\%s" % y
xmldoc = System.Xml.XmlDocument()
xmldoc.Load(filename)

#---------------GET ALL TRACED VARIABLE NAMES---------------------
variable_name_list = {}
for variable in xmldoc.SelectNodes("//traceData/dataFrame/dataSignal"):  
    variable_name_list[variable.GetAttribute("description")] = variable.GetAttribute("id")

#---------------GET AMOUNT OF POINTS------------------------
items = xmldoc.SelectNodes("//traceData/dataFrame/rec")
length_list = []
for item in items:
    length_list.append(item.GetAttribute("time"))
number_of_points = len(length_list)

#---------------ADDING TIME-STAMPS INFORMATION TO THE LIST----------------
items = xmldoc.SelectNodes("//traceData/dataFrame/rec")
for item in items:
    timestamp.append(item.GetAttribute("time"))

#---------------IMPORT EXTERNAL DATA BIGGER INTERVAL AND SYNCHRONIZE USING LINEAR INTERPOLATION------------------
if z <> "":
    print "There is an external data file with bigger interval available"
    filename_externaldata = r"D:\Data\Emiel\Dropbox\Graduation\Q2\Sinumerik\Tracing_and_logging\3D_visualization\Externaldata\%s" % z
    xmldoc2 = System.Xml.XmlDocument()
    xmldoc2.Load(filename_externaldata)
    external_variable_information_list = []
    #From each row in the XML file the Time stamp and the variable value are taken and added to the external_information_list
    for row in xmldoc2.GetElementsByTagName("Row"):
```python
cells = row.GetElementsByTagName("Cell")
sublist_externaldata = []
sublist_externaldata.append(cells[0].ChildNodes[0].InnerText)
sublist_externaldata.append(cells[1].ChildNodes[0].InnerText)
external_variable_information_list.append(sublist_externaldata)

new_external_data_list = []
# And a new_external_data list with the synchronized interval is created
for k in range(len(timestamp_list)-1):
    if k==0:
        print "First value is the timestamp so is skipped"
    else:
        for i in range(len(m)):
            if i==0:
                continue
            elif i==1:
                continue
            elif float(m[1][0])>float(timestamp[k]):
                small_list =[]
                j_value=i-1
                a_value=float(m[j_value][0])
                b_value=float(m[i][0])
                c_value=float(m[j_value][1])
                d_value=float(m[i][1])
                ab_value=b_value-a_value
                cd_value=d_value-c_value
                ratio=((float(timestamp[k])-a_value)/ab_value)
                answer=c_value+cd_value*ratio
                new_external_data_list.append(answer)
            break
        else:
            continue
    all_variable_information_list.append(new_external_data_list)
else:
    print "There is no external data file with a bigger interval available"

# Using linear interpolation the external data is synchronized to the interval of the XML-trace file
# And a new_external_data list with the synchronized interval is created
new_external_data_list.append(m[0][1])
for k in range(len(timestamp_list)-1):
    if k==0:
        print "First value is the timestamp so is skipped"
    else:
        for i in range(len(m)):
            if i==0:
                continue
            elif i==1:
                continue
            elif float(m[1][0])>float(timestamp[k]):
                small_list =[]
                j_value=i-1
                a_value=float(m[j_value][0])
                b_value=float(m[i][0])
                c_value=float(m[j_value][1])
                d_value=float(m[i][1])
                ab_value=b_value-a_value
                cd_value=d_value-c_value
                ratio=((float(timestamp[k])-a_value)/ab_value)
                answer=c_value+cd_value*ratio
                new_external_data_list.append(answer)
            break
        else:
            continue
    all_variable_information_list.append(new_external_data_list)
else:
    print "There is no external data file with a bigger interval available"

#------------------------IMPORT EXTERNAL DATA SMALLER INTERVAL AND SYNCHRONIZE BY SELECTING THE CLOSEST TIME-STAMP------------------------#
#This process is only executed when U is not empty, which means that there is an external data file present
if u <> "":
    print "There is an external data file with a smaller interval available"
    r"D:\Data\Emiel\Dropbox\Graduation\Q2\Sinumerik\Tracing_and_logging\3D_visualization\Externaldata \%s" % u
    xmldoc3 = System.Xml.XmlDocument()
    xmldoc3.Load(filename_externaldata)
    external_variable_information_list = []

    for row in xmldoc3.GetElementsByTagName("Row"):
        cells = row.GetElementsByTagName("Cell")
        sublist_externaldata = []
        sublist_externaldata.append(cells[0].ChildNodes[0].InnerText)
        sublist_externaldata.append(cells[1].ChildNodes[0].InnerText)
        external_variable_information_list.append(sublist_externaldata)

    new_external_data_list = []
    # And a new_external_data list with the synchronized interval is created
    new_external_data_list.append(m[0][1])
    for k in range(len(timestamp_list)-1):
        if k==0:
            continue
```

else:
    for i in range(len(m)):
        if i==0:
            continue
        elif float(m[i][0])>float(timestamp_list[k]):
            small_list = []
            j_value = i-1
            a_value = float(m[j_value][0])
            b_value = float(m[i][0])
            c_value = float(m[j_value][1])
            d_value = float(m[i][1])
            bk_value = abs(b_value-float(timestamp_list[k]))
            ak_value = abs(a_value-float(timestamp_list[k]))
            if bk_value<ak_value:
                new_external_data_list.append(d_value)
            else:
                new_external_data_list.append(c_value)
                break
        else:
            continue
    all_variable_information_list.append(new_external_data_list)
else:
    print "There is no external data file with a smaller interval available"

#------------------CREATE LIST FROM ALL VARIABLE LIST FOR EXPORT------------------#
all_variable_information = []
all_variable_information.append(all_variable_information_list)

#------------------DISPLAY ALL AVAILABLE VARIABLES------------------#
all_variable_names = []
for variable in all_variable_information_list:
    all_variable_names.append(str(variable[0]))
Appendix B6  Python script block 4

```python
#------------------IMPORTED LIBRARIES------------------
import clr
import Rhino.Geometry as rg
clr.AddReference("System.Xml")
import System.Xml
import Rhino
from Grasshopper import DataTree
import Grasshopper

#------------------CREATE LIST FOR THE SELECTED GRADIENT VARIABLE, THE TIMESTAMPS AND THE 3D POINTS------------------
gradient_variable = []
timestamp = []
point_list = []

#------------------DEFINITION USED TO EXTRACT SPECIFIC VARIABLE DATA FROM THE ALL_VARIABLE_INFORMATION(x)------------------
def GetSpecificList(variable_name):
    specific_variable_list = []
    for variable in x:
        if variable[0] == variable_name:
            for g in range(len(variable)-1):
                if g == 0:
                    continue
                else:
                    specific_variable_list.append(variable[g])
        else:
            continue
    return specific_variable_list

#------------------GET ALL X,Y,Z COORDINATES AND CREATE 3D POINTS THE COORDINATES------------------
x_out = GetSpecificList("X-axis")
y_out = GetSpecificList("Y-axis")
z_out = GetSpecificList("Z-axis")

i=0
for values in x_out:
    pt = rg.Point3d(float(values[0]),float(y_out[i]),float(z_out[i]))
    i+=1
    point_list.append(pt)

#------------------GET VARIABLE INFORMATION FOR GRADIENT DISPLAY USING THE SELECTED VARIABLE NAME p------------------
gradient_list = GetSpecificList(p)
for g in range(len(gradient_list)-1):
    if g == 0:
        continue
    else:
        gradient_variable.append(str(gradient_list[g]))
print gradient_variable

#------------------CREATE DATA TREE FROM DATA LIST------------------
i=0
Tree=DataTree(object())
while i < len(x):
    branch=Grasshopper.Kernel.Data.GH_Path(i)
    Tree.AddRange(x[i],branch)
    Variable_data=Tree
    i+=1
alldata=Tree.AllData()
```

149
Appendix B7  Create an external data file using excel

This appendix explains how to create an external data file using Excel that can be added to the grasshopper model as external data. The external data file needs to be formatted according to the following requirements. An example external data excel file is shown below.

The Excel file needs to consist of two columns. The first column consists of the time stamps of the measured values. The time stamps need to be displayed in seconds. The second column consists of the measured values. In this example, the temperature is the measured variable. In the first cell of the second column the name of the variable needs to be placed. The duration of the external measurement needs to have the same or a longer duration compared to the XML-trace file. Otherwise, the external measurement cannot be fully synchronized with the XML-trace file. The starting time stamp of the external data file needs to be manually synced with the XML-trace file when the external measurement is not started simultaneously with the Trace.

<table>
<thead>
<tr>
<th>Time stamp</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>23.5</td>
</tr>
<tr>
<td>1</td>
<td>23.4</td>
</tr>
<tr>
<td>2</td>
<td>23.3</td>
</tr>
<tr>
<td>3</td>
<td>23.2</td>
</tr>
<tr>
<td>4</td>
<td>23.1</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>22.9</td>
</tr>
<tr>
<td>7</td>
<td>22.8</td>
</tr>
<tr>
<td>8</td>
<td>22.7</td>
</tr>
<tr>
<td>9</td>
<td>22.6</td>
</tr>
<tr>
<td>10</td>
<td>22.5</td>
</tr>
<tr>
<td>11</td>
<td>22.4</td>
</tr>
<tr>
<td>12</td>
<td>22.3</td>
</tr>
</tbody>
</table>

When the data is formatted in the correct way, the excel file needs to be transformed into an XML-file. This is done by saving the excel file as an “XML spreadsheet 2003”. The resulting XML-file can be inserted in the Grasshopper model as external data.
Appendix C  Optimizing using real time adjustments

All the G-code scripts of the response modules in appendix C and D have comments which explain in detail how the scripts work. The comments are displayed in red and the Synchronized actions identification is displayed in purple.

Appendix C1  Basic optimization module (SPF)

;-------- DESIRED DISTANCE SETTINGS --------
R25 = 20 ;Height difference between nozzle and measurement device (mm)
R26 = 80 ;Desired distance to previous layer (mm) (from nozzle to previous layer)
R22 = R25 + R26 ;Total desired distance (from measurement device to previous layer)

;-------- INITIALIZING VALUES --------
R16 = 0 ;Required offset, initialized at 0
R23 = 999 ;Previous measurement value, initialized at 999

;-------- SYNCHRONIZED ACTIONS --------
ID=1  Do $R20=$A_DBB[0] ;Measurement value
ID=2  Do $R21=$R20-$R22 ;Difference between measurement value and total distance
ID=3  Whenever ($R20<>$R23) Do $R16=-$R21,$R23=$R20 ;Calculate required offset and update previous measurement value
ID=4  Do $AA_OFF[Z]=$R16 ;Perform calculated offset to the Z-axis offset
M17 ;End
Appendix C2  Start and stop trigger settings test setup

In this appendix, the trigger settings used to perform the tests and to be able to directly compare the test results of multiple tests are shown. The printer moves in the X direction from X5054 to X3245 during the test. When the X-value becomes smaller than 4900 the trace is started and when the printer passed X3500 the trace is automatically stopped. By not directly starting the trace from the beginning, the printer has reached the desired speed of 5000mm/m when the trace is started. By stopping the trace at X3500, the printer is not yet slowing down, thus there is a constant speed during the entire trace and each trace has a similar duration.
Appendix C3  Print path script of the test setup (MPF)

;======= PARAMETER SETTINGS =======
 R2 = 10000.0 ;PRINTER SPEED 200%
 R9 =5045 ;STARTPOINT X
 R10 = 926 ;STARTPOINT Y
 R11= -1579 ;STARTPOINT Z

;======= STARTING POINT AND SETTINGS ==========
 G90       ;ABSOLUTE COORDINATES
 G1        ;STARTING POSITION
            ;INTERPOLATE MOVEMENT
 X=R9 Y=R10 Z=R11 C0 F=R2

;======= SYNCHRONIZED ACTION MODULES =========
 BASICRESP ;ACTIVATED OPTIMIZATION MODULE

;----------------- PRINT PATH -----------------
 X=-1800    ;MOVE -1800mm IN X-COORDINATES
 M30

Appendix C4  Delayed response module (SPF)

;------- DESIRED DISTANCE SETTINGS -------
 R25=20 ;Height difference between nozzle and measurement device (mm)
 R26=80 ;Desired distance to previous layer (mm) (from nozzle to previous layer)
 R22=R25+R26 ;Total desired distance (from measurement device to previous layer)

;------- DELAYED RESPONSE SETTINGS -------
 R196=35.0 ;D = Distance from measurement device to nozzle (delayed distance)
 R197=0.3 ;Setting Sensor Collecting Distance

;------- VARIABLES AND INITIALIZING VALUES -------
 R16=0 ;Required offset, initialized at 0
 R23=999 ;Previous measurement value, initialized at 999
 R190=0 ;DT (Distance travelled) initialized at 0
 R195=$A_DBB[0] ;Delayed measurement value
 R198=1 ;Counter Buffer for filling ring-buffer
 R200=$AA_IM[X] R201=$AA_IM[Y] R202=$A_DBB[0] ;updating the first elements in the ring-buffer with the X- and Y-coordinates and the measurement value

;------- MARKERS FOR FILLING AND READING DATA IN THE RING-BUFFER -------
 $AC_MARKER[0] = 0 ;Pointer for filling Buffer
 $AC_MARKER[1] = 0 ;Pointer for reading buffer

;------- STORING DATA IN THE RING-BUFFER -------
 ID=1 Do $R190= SQRT(POT($AA_IM[X]-$R[$AC_Marker[0]+200]) + POT($AA_IM[Y]-$R[$AC_Marker[0]+201]))
 ;Calculate DT for filling the buffer using pythagoras

 ID=2 Whenever ($R190 > $R197) Do $AC_MARKER[0]=($AC_MARKER[0]+3) MOD 795
 $R198=$R198+1 ;whenever (DT > Sensor Collecting Distance):Add +3 to the fill-marker, perform modulus function,
 R200=$AA_IM[X] R201=$AA_IM[Y] R202=$A_DBB[0] ;updating the first elements in the ring-buffer

;------- READING DATA FROM THE RING-BUFFER -------
 ;Calculate DT for reading

 ID=4 Whenever ($R190>$R196) And ($R198>0) Do $R195=$R[$AC_MARKER[1]+202]
 ;whenever (DT > D) update delayed measurement value, Add +3 to the read-marker, perform modulus function, -1 to the counter buffer

;------- RESPONDING TO THE DELAYED MEASUREMENT VARIABLE -------
 ID=5 Do $R20=$R195 ;Delayed measurement value
 ID=6 Do $R21=$R20-$R22 ;Difference between measurement value and total distance
 ID=7 Whenever ($R20>$R23) Do $R16=$R21 $R23=$R20 ;Calculate required offset and update
 previous measurement value
 ID=8 Do $AA_OFF[Z]=$R16 ;Perform calculated offset to the Z-axis offset

M17 ;End
Appendix C5  Filtered response module (SPF)

;-------- DESIRED DISTANCE SETTINGS --------
R25=20 ;Height difference between nozzle and measurement device (mm)
R26=80 ;Desired distance to previous layer (mm) (from nozzle to previous layer)
R22=R25+R26 ;Total desired distance (from measurement device to previous layer)

;-------- INITIALIZING VALUES --------
R16=0 ;Required offset, initialized at 0
R23=999 ;Previous measurement value, initialized at 999
R20=$A_DBB[0] ;Filtered Value (FV)
R24=25 ;Filter Factor (FF)

;-------- SYNCHRONIZED ACTIONS --------
ID=1 Do $R20=($(R20*($R24-1))+$A_DBB[0])/$R24 ;Calculate filtered measurement value
ID=2 Do $R21=$R20-$R22 ;Difference between filtered value and total distance
ID=3 Whenever ($R20<=$R23) Do $R16=-$R21 $R23=$R20 ;Calculate required offset and update previous filtered value
ID=4 Do $AA_OFF[Z]=$R16 ;Perform calculated offset to the Z-axis offset
M17 ;End
Appendix C6  AHV response module (SPF)

;-----------------  DESIRED DISTANCE SETTINGS  -----------------
R25=20 ;Height difference between nozzle and measurement device (mm)
R26=80 ;Desired distance to previous layer (mm) (from nozzle to previous layer)
R22=R25+R26 ;Total desired distance (from measurement device to previous layer)

;-----------------  AHV SETTINGS  -----------------
R197=0.3 ;Sensor Collecting Distance
R189=25 ;Filter Factor (FF)

;-----------------  RESET ALL VALUES IN RING-BUFFER TO 0  -----------------
R120 = 200 ;Loop is initialized at variable R200 (the first element in the
Ring-Buffer)
Loop1:
;Start loop 1
R[R120] = 0 ;Reset R-variable to 0
R120 = R120+1 ;Continue to next R-variable
IF R120<995 GOTO Loop1 ;If variable R995 is not yet reached, continue resetting next R-
variable in ring-buffer

;-----------------  VARIABLES AND INITIALIZING VALUES  -----------------
R16=0 ;Required offset, initialized at 0
R23=999 ;Previous measurement value, initialized at 999
R190=0 ;DT (Distance travelled) initialized at 0
R187=0 ;AHV SUM
R192=0 ;AHV
R194=$A_DBB[0] ;Raw measurement value
R198=1 ;Counter Buffer for filling ring-buffer
R200=$AA_IM[X] R201=$AA_IM[Y] R202=$A_DBB[0] ;updating the first elements in the ring-
buffer with the X- and Y-coordinates and the measurement value

;-----------------  MARKERS FOR FILLING RING-BUFFER AND AHV CALCULATION  -----------------
$AC_MARKER[0] = 0 ;Pointer Fill Buffer
$AC_MARKER[2] = 0 ;Pointer for AHV value removal
$AC_MARKER[3] = 1000 ;Marker to check if new measurement value is available

;-----------------  STORING DATA IN THE RING-BUFFER  -----------------
ID=1 Do $R190= SQRT(POT($AA_IM[X]-$R[$AC_MARKER[0]+200]) + POT($AA_IM[Y]-$R[$AC_MARKER[0]+201])) ;Calculate DT for filling the buffer
Update next elements in buffer, +1 to the counter buffer

;-----------------  CALCULATING THE AHVSUM AND AHV  -----------------
element in buffer to be deleted from AHVSUM, Update AHVSUM by removing last measurement value and
adding new measurement value, Calculate AHV and update Marker[3]

;-----------------  RESPONDING TO THE AHV  -----------------
ID=4 Do $R20=$R122 ;AHV
ID=5 Do $R21=$R20+$R22 ;Difference between AHV and total distance
ID=6 Whenever ($R20<>$R23) Do $R16=+$R21 $R23=$R20 ;Calculate required offset and update
previous AHV value
ID=7 Do $AA_OFF[Z]=$R16 ;Perform calculated offset to the Z-axis offset

M17 ;End
Appendix C7  PID response module (SPF)

;--------- DESIRED DISTANCE SETTINGS ---------
R25=20 ;Height difference between nozzle and measurement device (mm)
R26=80 ;Desired distance to previous layer (mm) (from nozzle to previous layer)
R22=R25+R26 ;Total desired distance (from measurement device to previous layer)

;--------- PID-Coefficient settings ---------
R20 = 0.2 ;Kp = Coefficient for the Proportional term
R21 = 1.5 ;Ki = Coefficient for the Integral term
R22 = 0 ;Kd = Coefficient for the Derivative term

;--------- INITIALIZING VALUES AND CREATE VARIABLES ---------
R23 = $MN_IPO_CYCLE_TIME ;Cycle time of the Sinumerik
R24=0 ;Previous error value
R25=0 ;e(t) = Error value
R26=0 ;Integral term
R27=0 ;Derivative term
R28=R22 ;SP = Setpoint (mm)
R29=$A_DBB[0] ;PV(t) = Process variable (Measured Value)
R30=0 ;U(t) = Control variable

;--------- SYNCHRONIZED ACTIONS ---------
ID=1 Do R29=$A_DBB[0] $R25=$R28-$R29 $R26=$R26+($R25*$R23) $R27=($R25-$R24)/$R23 $R30=($R20*$R25)+($R21*$R26)+($R22*$R27) $R24=$R25 ;Update PV(t), calculate P,I and D term, then calculate U(t) and update previous measurement
ID=2 Do $AA_OFF[2]=$R30 ;Perform Z-offset with U(t)

M17 ;End
Appendix D  Use cases

Appendix D1  PIDFilter response module On (SPF)

;-------- DESIRED DISTANCE SETTINGS --------
R25=20 ;Height difference between nozzle and measurement device (mm)
R26=80 ;Desired distance to previous layer (mm) (from nozzle to previous layer)
R22=R25+R26 ;Total desired distance (from measurement device to previous layer)
R28=R22 ;SP = Setpoint (mm)

;-------- PID-Coefficient settings --------
R20 = 0.2 ;Kp = Coefficient for the Proportional term
R21 = 2 ;Ki = Coefficient for the Integral term
R22 = 0 ;Kd = Coefficient for the Derivative term

;-------- FILTER SETTINGS --------
R31=25 ;Filter Factor (FF)

;-------- INITIALIZING VALUES AND CREATE VARIABLES --------
R23 = $MN_IPO_CYCLE_TIME ;Cycle time of the Sinumerik
R24=0 ;Previous error value
R25=0 ;e(t) = Error value
R26=0 ;Integral term
R27=0 ;Derivative term
R29=$A_DBB[0] ;PV(t) = Process variable (Filtered measured value)
R30=0 ;U(t) = Control variable
R32=$A_DBB[0] ;Raw measurement value

;-------- SYNCHRONIZED ACTIONS --------
ID=1 Do $R29=((($R29*($R31-1))+$A_DBB[0])/$R31) $R25=$R28-$R29 $R26=$R26+($R25*$R23) $R27=(($R25- $R24)/$R23) $R28=$R22+$R26 $(R22*($R27)) $R24=$R25 ;Update PV(t) with filtered measurement value, calculate P, I and D term, then calculate U(t) and update previous measurement

ID=2 Do $AA_OFF[Z]=$R30 ;Perform Z-offset with U(t)
M17 ;End

Appendix D2  PIDFilter response module Off (SPF)

;-------- CANCEL SYNCHRONIZED ACTIONS --------
CANCEL(1) ;Synchronized action 1 is cancelled
CANCEL(2) ;Synchronized action 2 is cancelled
M17 ;End
Appendix D3  Print path script use case 1 – With preprogrammed Z-movement (MPF)

;------- DATE 2017-01-18 ------- ;by Rob

;------- PARAMETER SETTINGS -------
R1 = 98.304 ;PUMP FREQ 98304 is 30 Hz op 100%
R2 = 10000.0 ;PRINTER SPEED 200%
R3 = 10 ;LAYER TO LAYER Z-OFFSET

;------- INITIALIZE PRINT -------
H=100*R1 M50 ;PUMP FREQ CONVERT - PUMP OFF
TANG(C,X,Y,-1) ;TANG NOZZLE SETUP
TANGON(C,0) ;TANG NOZZLE ON

;------- STARTING POINT -------
G90 G1 X4000 Y3500 Z-1800 C0 F=R2 ;STARTING POSITION - ABSOLUTE COORDINATES
M51 ;PUMP ON
Y3125 Z-2011

;------- STARTING PATH -------
G642 G91 ;INTERPOLATE MOVEMENT - INCREMENTAL COORDINATES
Y=300
BSPLINE
X=0.37 Y=-7.84 ;STARTING PATH
X=0.37 Y=-7.84 ;STARTING PATH

R99=0
LoopCircle:
X=0.37 Y=-7.84
X=0.62 Y=-7.83
X=0.86 Y=-7.81
X=1.11 Y=-7.78
X=1.35 Y=-7.74
X=1.59 Y=-7.69
X=1.83 Y=-7.64
X=2.07 Y=-7.58
X=2.31 Y=-7.51
X=2.54 Y=-7.49
X=2.78 Y=-7.35
X=3.01 Y=-7.26
X=3.23 Y=-7.16
X=3.46 Y=-7.05
X=3.67 Y=-6.94
X=3.89 Y=-6.82
X=4.1 Y=-6.7
X=4.31 Y=-6.56
X=4.52 Y=-6.43
X=4.72 Y=-6.28
X=4.91 Y=-6.13
X=5.1 Y=-5.97
X=5.29 Y=-5.81
X=5.47 Y=-5.64
X=5.64 Y=-5.47
X=5.81 Y=-5.29
X=5.97 Y=-5.1
X=6.13 Y=-4.91
X=6.28 Y=-4.72
X=6.43 Y=-4.52
X=6.56 Y=-4.31
X=6.7 Y=-4.1
X=6.82 Y=-3.89
X=6.94 Y=-3.67
X=7.05 Y=-3.46
X=7.16 Y=-3.23
X=7.26 Y=-3.01
X=7.35 Y=-2.78
X=7.43 Y=-2.54
X=7.51 Y=-2.31
X=7.58 Y=-2.07
X=7.64 Y=-1.83
X=7.69 Y=-1.59
X3.23 Y-7.16
X3.01 Y-7.26
X2.78 Y-7.35
X2.54 Y-7.43
X2.31 Y-7.51
X2.07 Y-7.58 Z=(R3/10)
X1.83 Y-7.64 Z=(R3/10)
X1.59 Y-7.69 Z=(R3/10)
X1.35 Y-7.74 Z=(R3/10)
X1.11 Y-7.78 Z=(R3/10)
X0.86 Y-7.81 Z=(R3/10)
X0.62 Y-7.83 Z=(R3/10)
X0.37 Y-7.84 Z=(R3/10)
X0.12 Y-7.85 Z=(R3/10)
X-0.12 Y-7.85 Z=(R3/10)
R99 = R99+1
IF R99<50 GOTOB LoopCircle

;--------------------- END ---------------------
G90 G1 X4000 Y3500 Z-1800 C0 F=R2
M30
Appendix D4  Print path script use case 1 – With PID-filter response module (MPF)

;---------- DATE 2017-01-18 --------------- ;by Rob

;---------- PARAMETER SETTINGS ---------
;R1 = 163.84  ;PUMP FREQ 16384 is 50 Hz op 100%  NOT ACTIVATED!!!
;R2 = 10000.0  ;PRINTER SPEED 200%
;R3 = 9.2  ;LAYER TO LAYER Z-OFFSET

;---------- INITIALIZE PRINT ---------
H=100*R1  ;PUMP FREQ CONVERT - PUMP OFF
TANGL(C,0)  ;TANG NOZZLE SETUP

;---------- STARTING POINT ------------
G90 G1 X4000 Y3500 Z-1800 C0 F=R2  ;STARTING POSITION - ABSOLUTE COORDINATES
M51  ;PUMP ON
Y3125 Z=2011

;---------- STARTING PATH -------------
G642 G91  ;INTERPOLATE MOVEMENT - INCREMENTAL COORDINATES
Y=100
PIDFILTER2ON  ;THE PIDFILTER RESPONSE MODULE IS ACTIVATED
Y=200
BSPLINE
X=0.37 Y=-7.84  ;STARTING PATH
X=0.37 Y=-7.84  ;STARTING PATH

R99=0
LoopCircle:
X=0.37 Y=7.84
X=0.62 Y=7.83
X=0.86 Y=7.81
X=1.11 Y=7.78
X=1.35 Y=7.74
X=1.59 Y=7.69
X=1.83 Y=7.64
X=2.07 Y=7.58
X=2.31 Y=7.51
X=2.54 Y=7.43
X=2.78 Y=7.35
X=3.01 Y=7.26
X=3.23 Y=7.16
X=3.46 Y=7.05
X=3.67 Y=6.94
X=3.89 Y=6.82
X=4.1 Y=6.7
X=4.31 Y=6.56
X=4.52 Y=6.43
X=4.72 Y=6.28
X=4.91 Y=6.13
X=5.1 Y=5.97
X=5.29 Y=5.81
X=5.47 Y=5.64
X=5.64 Y=5.47
X=5.81 Y=5.29
X=5.97 Y=5.1
X=6.13 Y=4.91
X=6.28 Y=4.72
X=6.43 Y=4.52
X=6.56 Y=4.31
X=6.7 Y=4.1
X=6.82 Y=3.89
X=6.94 Y=3.67
X=7.05 Y=3.46
X=7.16 Y=3.23
X=7.26 Y=3.01
X=7.35 Y=2.78
X=7.43 Y=2.54
X=7.51 Y=2.31

162
X3.09 Y-6.82
X3.67 Y-6.94
X3.46 Y-7.05
X3.23 Y-7.16
X3.01 Y-7.26
X2.78 Y-7.35
X2.54 Y-7.43
X2.31 Y-7.51
X2.07 Y-7.58
X1.83 Y-7.64
X1.59 Y-7.69
X1.35 Y-7.74
X1.11 Y-7.78
X0.86 Y-7.81
X0.62 Y-7.83
X0.37 Y-7.84
X0.12 Y-7.85
X-0.12 Y-7.85
R99 = R99+1
IF R99<50 GOTOB LoopCircle

PIDFILTER2OFF ;THE PIDFILTER RESPONSE MODULE IS DEACTIVATED WHEN PATH IS COMPLETED

;---------------- END ----------------
G90 G1 X4000 Y3500 Z-180 C0 F=R2
M30
Appendix D5  Print path script use case 2 (MPF)

;------- DATE 2017-01-18----------- ;by Rob

;------- PARAMETER SETTINGS -------
R1 = 163.84 ;PUMP FREQ 16384 is 50 Hz op 100% NOT ACTIVATED!!!
R2 = 10000.0 ;PRINTER SPEED 200%
R3 = 9.2 ;LAYER TO LAYER Z-OFFSET

;------- INITIALIZE PRINT -------
H=100*R1 M50 ;PUMP FREQ CONVERT - PUMP OFF
TANG(C,X,Y,-1) ;TANG NOZZLE SETUP
TANSN(C,0) ;TANG NOZZLE ON

;------- STARTING POINT -------
G90 G1 X4000 Y3500 Z-1800 C0 F=R2 ;STARTING POSITION - ABSOLUTE COORDINATES
M51 ;PUMP ON

G91 X-800

G90 Y3125 Z-2000

;------- STARTING PATH -------
G64S G91 ;INTERPOLATE MOVEMENT -_INCREMENTAL COORDINATES
Y=200
PIDFILTER1ON ;THE PIDFILTER RESPONSE MODULE IS ACTIVATED
x=200
RNDM=50

R99=0
LoopRoute:
X=2000
ty=300
X2000
Y300
R99 = R99+1
IF R99<5 GOTOB LoopRoute

PIDFILTER1OFF ;THE PIDFILTER RESPONSE MODULE IS DEACTIVATED WHEN PATH IS COMPLETED

;------- END -------
G90 G1 X4000 Y0 Z-1800 C0
M30