NTNU

Norwegian University of
Science and Technology

Visual Feedback for Large Scale Additive Manufacturing Process

Jørgen Jackwitz

Master of Science in Industrial Cybernetics
Submission date: June 2018
Supervisor: Jan Tommy Gravdahl, ITK
Co-supervisor: Linn Danielsen Evjemo, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Visual Feedback for Large-Scale Additive Manufacturing Process

Jørgen Jackwitz

Spring 2018, NTNU

Preface

This thesis is the final work of the two year master programme Industrial Cybernetics at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology. The work in this thesis has been carried out during the spring of 2018.

The purpose of the thesis is to look into visual feedback for an additive manufacturing process being developed at the Department of Engineering Cybernetics. The aim of the project is to use a Cold-Metal-Transfer welding method for additive manufacturing, and this thesis is supposed to highlight requirements and insight into what is needed for such a process.

For this purpose some equipment have been made available. The six degree-of-freedom industrial robot arm IRB-140 from ABB will be used for testing, together with a pneumatic caulking gun for extrusion of material. Two Structured light 3D-cameras, Intel SR-300 and the ASUS Xtion Pro, were also made available.

No software was provided by my supervisors, so most of the software have been found during the course of thesis. The project thesis this fall used Matlabs Computer Vision Toolbox, but was found lacking in features to deal with 3D point-cloud data. The open-source software suite Point-Cloud Library was proposed as an alternative because of its range of algorithms and datastructures created specifically to work with point clouds, but installation problems and compatibility with the 3D-cameras introduced plenty of frustration and wasted time. I was not very familiar with computer vision and the methods used in this field of study, lacking the required subjects, and thus some time was used in the wrong corners of the computer vision field.

During the experiment phase of the thesis there were problems calibrating the robot to be aligned with the working table and figuring out how to control the robot, since the aim of the thesis was to focus on the computer vision part some time was used outside of the scope of the thesis.

My supervisors on this project have been Jan Tommy Gravdahl with co-supervisor Linn Danielsen Evjemo. I have had regular meetings with them throughout the semester and Linns knowledge of the ABB robot have been invaluable.

Trondheim, June 11th 2018
Jørgen Jackwitz

Abstract

Large-scale Additive Manufacturing is blossoming and brings new challenges to the Additive Manufacturing scene. For larger constructions, material deformation from weight, environmental disturbances and extrusion may disturb the AM process. This thesis looks for a way to measure these errors. A proof of concept software workflow has been created to measure road-width and height of the extruded material directly after extrusion using a simple edge detection algorithm. Testing the workflow on a life-size robot arm and a practical extrusion test shows promise for future applications and improvements.

Sammendrag

Stor-skala additiv produksjon blomstrer og setter lys på nye utfordringer innen additiv produksjon. For større konstruksjoner spiller materialdeformasjon fra vekt, forstyrrelser fra miljø og ekstrudering større rolle for indusering av feil. Denne oppgaven ser på forskjellige metoder å måle disse feilene på. Et førsteutkast på en programvareflyt for måling av bredde og høyde på det ekstruderte materialet rett etter ekstrusjon ved bruk av en enkel kantdetekjsonsalgoritme er vist frem. Vi fikk se at metoden fungerte ved å teste denne programflyten på en virkelig robot og med en praktisk ekstrusjonstest, men metoden trenger forbedringer.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Additive Manufacturing (AM), also known as 3D-printing, has flourished in the home and hobby market in the last decade after the release of the open-source RepRap project [2]. Because of the open-source nature of the RepRap project, hobbyists and manufacturers were able to rapidly improve and contribute to better and cheaper parts and methods. While small-scale AM is interesting for rapid prototyping, large-scale 3D-printing has not flourished in the same way. As AM usually produces parts layer-for-layer, a larger object will require a much longer manufacturing period than a smaller one, if the same quality is wanted. To make large-scale AM feasible new materials and methods must be developed. A survey and proof of concept of large-scale AM were outlined in [3]. Large-scale AM can have a great impact on affordable and more environmental friendly housing, by reducing manufacturing waste and human labour. When new buildings can be built with reduced human labour, the cost is reduced proportionally.

The Department of Engineering Cybernetics at NTNU have recently established a project for large-scale Additive Manufacturing. The long-term goal is to use a sizeable

6-Degree-of-Freedom robot manipulator with a Cold Metal Transfer (CMT) welding method as the material extrusion. Traditional AM methods do not use any feedback or control-loops further than servo-control to control the printing process. These processes can control their position correctly, but deformation or unwanted imperfections in the deposited material are unknown. Large-scale AM by 6-DOF will have more freedom of movement and a more significant printing volume than traditional AM and effects such as deformation of hanging structures may occur.

Introducing a feedback system can go a long way in improving the results and detecting error states that a open loop system is not able to. I have thus conducted a study into how best to implement a feedback system The scope of this thesis is as follows:

1. Do a literature review on current research and solutions for live feedback in additive manufacturing (AM).
2. Discuss what kinds of sensors, software and algorithms that are necessary to get sufficient feedback from extrusion based AM with 6-DOF robot manipulator when trying to compare the actual build with the desired structure. Discuss if any existing solutions are suitable for this.
3. Test mapping of a simple geometry with an available sensor, for example, a 3D-camera. Discuss and evaluate how suitable this sensor is for live monitoring of an AM process, and what the greatest limitations and challenges are. Make suggestions for future work.

1.1 Abbreviations

Often used Abbreviations in the article.

- SDK = Software Development Kit

- TOF = Time Of Flight
- FPS = Frames Per Second
- RGB = Red Green Blue, color image
- AM = Additive Manufacturing
- DOF = Degrees of Freedom

Chapter 2

Background

In this chapter we will look at the most relevant related work regarding feedback for additive manufacturing as well as error modes and process parameters that have an effect on the finished manufactured part. An overview of technologies used for three dimensional data capture is also looked at.

2.1 Process Parameters and Quality Control in Additive Manufacturing

An overview of quality control in additive manufacturing is conducted in [4]. This article describes a range of process and quality parameters that are important for Field Deposition Modeling, but the parameters can be applied to any additive manufacturing technique.

Process parameters are classified as machine specific parameters, such as nozzle

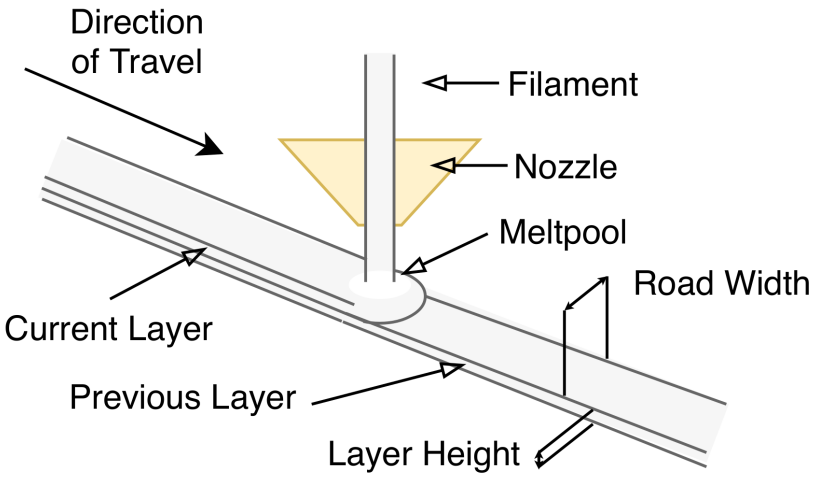


Figure 2.1: General principle of additive manufacturing using filament

diameter, filament diameter, top speed of actuators and so on. There is also operation specific parameters and these are controllable parameters used during manufacturing, these can be layer thickness, road width, fill pattern and so on. Finally there is material specific parameters, examples of these are viscosity, stiffness and thermal conductivity. All these process parameters are affecting the end quality of the manufactured part. figure 2.1 gives an overview of where we find certain process parameters.

The quality of the part can be described by a few quality parameters. These are profile specific: dimensional accuracy, surface roughness and mechanical property, and defect specific: underfill area and overfill area.

Under- and overfill are errors where the extruders material flow is too large or too small and the road width ends up either too wide or too narrow as illustrated in figure 2.2. With underfill you will get gaps between parallel paths reducing the surface quality. With overfilled parallel paths you will get ridges where the parallel paths connect.

Under- and overfill along parallel paths can be countered by tuning the feed-rate of the

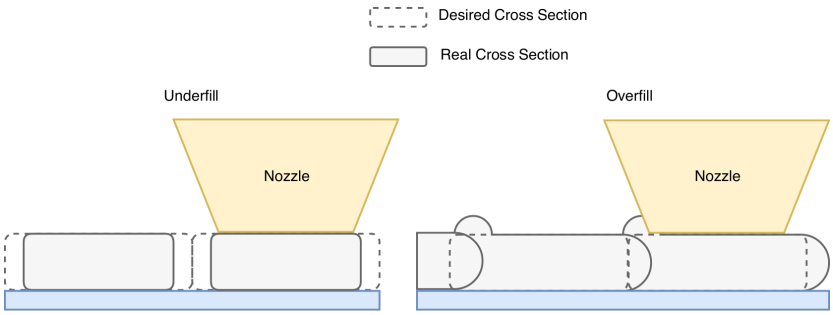


Figure 2.2: Under- and overfill errors along parallel paths

filament manually or by use of feedback. A method to counteract under and overfill error along parallel paths was proposed in [5]. They used color cameras with high magnification and a support vector machine to classify pictures of the surface into overfilled, underfilled and good layers and used a discretized PID-controller to change the filament feed rate between each layer.

This problem also arises in tight corners of the paths as explained in [6]. In the same article they try to reduce this problem by using a better path planning algorithm. Underfill areas are usually sharp edges and corners of 3d-models where the radius of the nozzle and road width are the limiting factor for the accuracy of the manufactured part.

Control of the layer height of a laser metal-wire deposition additive manufacturing technique was explored in [7]. They used a 2d laser triangulation sensor to scan each layer after its manufacture and then used an Iterative Learning Controller to control the feed rate of the wire for the next layer.

A wide range of sensors were used in [8], for a process using FDM and ABS plastic. Sensors used were accelerometers for vibrations in the tool head and frame, temperature sensors for the build table, extruder, ambient temperature and melt pool temperature, as well as a video camera. The video however was only used for manual observations.

These sensors were then combined using statistical sensor fusion methods to find a correlation between the process parameters feed/flow rate ratio, layer height and extruder temperature with the end goal of increasing the quality of the surface of the manufactured part. Their real-time system was also able to distinguish between if the print was behaving as normal or if an error state, such as nozzle clogging took place.

2.2 Additive Manufacturing Software-Toolchain

There are several different types of additive manufacturing methods[9] such as stereolithography, Fused Deposition Modeling (FDM), Selective Laser Sintering (SLS), Electron Beam Melting (EBM), and Direct Metal Deposition (DMD). These are very different in the way they manufacture the resulting part, but they have one thing in common and that is the software-toolchain. figure 2.3 gives and overview of the general flow from design to manufacturing. We start with a 3D-model created in CAD software then convert it to an STL file by reducing the highly detailed CAD-model into triangles. This lower detailed model is then fed into the preprocessing environment where we orient the part (design intent) based on the way the manufacturing method will realize the part and the desired mechanical properties this realization imbues into the part. Then the slicer, based on the process parameters such as layer height, nozzle diameter, actuator speed and material properties, generates a path for the AM-machine. This toolpath is then exported to the machine for manufacturing.

2.3 Consumer 3D-Printing

This section taken from project thesis [10]
The closest thing to feedback in the consumer additive manufacturing(3d-printer) market is the Unified Bed Leveling system of the Open Source 3D-printer firmware Marlin[11], which uses a preprocessing step to calibrate its coordinate axis to the

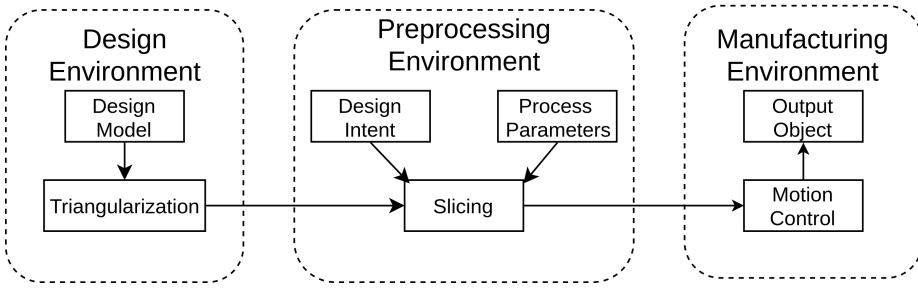


Figure 2.3: General Software-toolchain for Additive Manufacturing process

printing beds orientation mathematically. The firmware requires the 3D-printer to be fitted with a distance sensor so it can detect the build plate. The process mathematically projects a grid over the build area and moves the print head to each intersection in the grid and measures the height to the build plate. After this preprocessing is done, the 3D-printer will follow the trajectory for the 3D-model and offset the trajectory by the preprocessed grid. This only works for small deviations and is not necessary if you are able to calibrate the levelness of the build plate physically. This method is useful however if your build plate is warped or not wholly level by itself.

2.4 Capturing Model Data

This section taken from project thesis [10]

There are many different methods of capturing 3D measurements. Both contact measurements and non-contact measurements are available, but for our purpose, non-contact measurements are most suited. For non-contact measurements, there are two sub-classes: Passive and Active technologies.

2.4.1 Passive Techniques

Passive techniques use the ambient radiation of the environment like light and usually use normal cameras and video-cameras to capture the environment[12]. The main categories of passive techniques are stereoscopy and photogrammetry. The basic principle of both of these methods is that we have multiple pictures of the object or scene to be measured from different locations and that these locations are known.

2.4.1.1 Stereoscopy

Stereoscopy uses two parallel cameras with a known separation distance which capture the object or scene simultaneously, and we use software to triangulate pixels found in both pictures[13]. The cameras are also usually angled slightly to increase the overlap of their respective field of views. Stereoscopy does not work if the two cameras do not have any overlapping.

2.4.1.2 Photogrammetry

Photogrammetry uses the same principle as stereoscopy, but here the pictures are captured by a single camera at different locations often separated in time.

2.4.2 Active Techniques

Active techniques use radiation from the sensor to the object and process the reflected signal returned from the object or environment. Laser scanning is an active technique widely used to capture 3D data. Laser scanning is based on emitting some light wave to the object and recording the delay before the reflected wave is received. [1] describes the three primary methods of laser scanning. These are time-of-flight, triangulation and phase shift measurement.

2.4.2.1 Time-of-flight

Time-of-flight measurement uses a highly accurate timer to measure the distance to the object to be measured. This is possible because the propagation velocity of light is known. By measuring the round-trip time of a light pulse from the source, the scanner, to the reflective surface of the object to be measured, and back to the source, we can calculate the distance d. Equation (2.1)[1] shows us the relationship.

$$d = \frac{c_m \tau}{2} \tag{2.1}$$

The constant c_m is the speed of light in the current medium, and τ is the round-trip time of the light wave. In a vacuum, light will travel one millimetre in 3.3 picoseconds so for high accuracy measurements a very accurate timing device is needed.

2.4.2.2 Triangulation

Triangulation uses the proportions of a triangle to measure the distance to the object[1]. Figure 2.4 shows the basic principle. A laser emits a beam of light on to the target surface. This light is reflected back to the scanner with the same, but opposite, angle as the emitting laser. A strategically placed position-sensitive detector can measure the deflection of the beam of light. The distance from the scanner to the object, Z, can be found by the following equation:

$$Z = \frac{Bf_0}{p + f_0 tan(\alpha)} \tag{2.2}$$

where B is the distance between the emitting laser and the optical centre of the scanner, f_0 is the distance from the optical centre, and the position-sensitive detector and p is the position on the detector. The angle α is the angle of the laser emitter to the normal

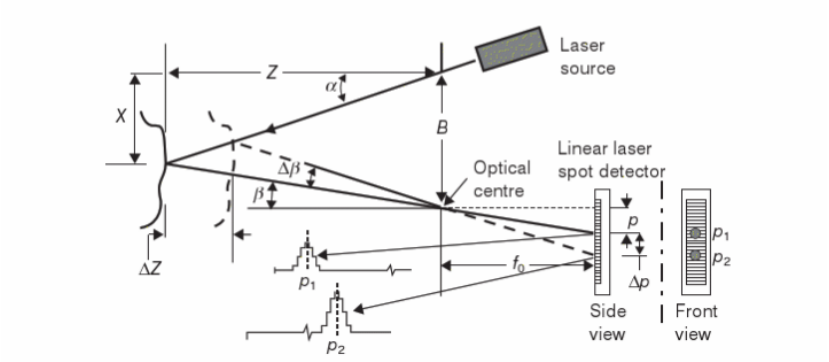


Figure 2.4: Basic principle of distance measurement using triangulation, from [1]

of the plane of the contact point between the light beam and object. See figure 2.4 for a visual representation.

2.4.2.3 Phase Shift

Phase shift measurement works by looking at the difference in phase between the emitted light and reflected light[1]. By inserting equation (2.3a) into equation (2.1) we get the distance to the surface. The underlying problem of phase shift measurement is that the measured $\delta\phi$ cannot be greater than one period of the light wave, assuming a sine wave. This is because a non-modulated light wave is identical for every period of the signal. This distance is the ambiguity interval and is given by equation (2.3c). Equation (2.3b) gives us the range uncertainty of the measured sine wave, where SNR

is the signal-to-noise ratio.

$$\tau = \frac{\Delta\phi}{2\pi}\frac{\lambda_m}{c_m} \tag{2.3a}$$

$$\delta_{r-AM} \approx \frac{1}{4\pi}\frac{\lambda_m}{\sqrt{SNR}} \tag{2.3b}$$

$$d_{max} = \frac{\lambda_m}{2} \tag{2.3c}$$

There are several solutions to this problem. One solution mentioned in [1] is to use a two-tone AM system, with, e.g. 10 MHz and 150MHz light waves. By employing equation (2.3c) on the lowest frequency signal and equation (2.3b), with $SNR = 1000$, on the high frequency signal we get an ambiguity of 15m and range uncertainty of 5mm.

Another method to remove this ambiguity is based on frequency modulation. The frequency of the laser beam is changed continuously over a specified period in either a triangular or saw-tooth pattern. This modulation period T_m can last up to several milliseconds. From [14] we have that the distance to the object is found by measuring the beat frequency and applying equation (2.4a). The expression $\alpha(\Delta\lambda)$ is the slope of the linear change in frequency of the saw-tooth pattern. The range uncertainty can be found from the Signal-to-noise ratio, SNR, and frequency range, Δf, as described in equation (2.4b). The ambiguity interval is a function of the modulation period, as

described in equation (2.4c).

$$d = \frac{f_b}{f_s}\frac{\lambda^2}{\Delta\lambda} = \frac{f_b}{\alpha(\Delta\lambda)} \tag{2.4a}$$

$$\delta_{r-FM} \approx \frac{\sqrt{3}}{2\pi}\frac{c_m}{\Delta f}\frac{1}{\sqrt{SNR}} \tag{2.4b}$$

$$d_{max-FM} = \frac{c_m}{4}T_m \tag{2.4c}$$

These are the standard principles for measuring single point distances to 3D objects using light. To capture whole 3D-scenes, we have to capture a series of points across the whole scene. This can be done by moving the sensor manually or use moving mirrors.

An example of moving the entire sensor is, e.g. spinning an array of ToF-sensors 360°capturing a cone-shaped cross-section of the area around the sensor. These are often called LiDaR, Light Detection and Ranging, and often come with an array of 16 to 32 ToF-sensors. An example is the Velodyne Lidar VLP-16 [15], offering 16 channels and a rotational frequency of 5-20Hz. These do however have a rather low accuracy in depth. This particular sensor has a range uncertainty of ±3centimetres. Most lidar systems are made for extended range mapping and surveying and do not come with any accuracy better than a few centimetres and are thus not suited for our purpose.

2.4.2.4 Structured Light 3D - Cameras

3D- or RGB-Depth-cameras are cameras able to output a continuous stream of both depth and colour data of a whole scene for each sample time. The result is a picture with pixels with both RGB data and depth information. The difference between measuring a single point as described earlier and a whole grid is that the RGB-d cameras project whats called a structured- or coded light pattern on the screen. The reflected pattern

will wrap around the object to be measured, and a distorted pattern will be reflected back.

Most 3D cameras like Microsoft Kinect[16], Asus Xtion Pro[17] and Intel SR300[18] uses infrared structured- and coded-light which is invisible to the human eye. The depth information is achieved by triangulation, but with a 2d grid of sensors instead of the one-dimensional solution as described in section 2.4.2.2. By combining the triangulated IR-image of the scene with the known projected pattern, a depth image is created.

3D - Cameras suffer from systematic depth errors and non-systematic depth errors[19]. Systematic errors can usually be mitigated by calibration, while filtering can mitigate non-systematic errors.

Systematic errors are depth distortion, integration time error, pixel-related errors, amplitude ambiguity, and temperature error.

Depth distortion error is how the infrared light emitter is not able to reproduce the infrared light precisely as wanted[19]. In our case, we have an infrared pattern, and the difference between the analytic pattern used for calculation and the actual projected pattern could be a detriment to our depth calculation. Integration time errors appear when the sensor is not measuring the reflected light for long enough. A more prolonged integration period increases the signal-to-noise ratio, reducing the depth errors[20]. A change in integration period directly affects the frame rate of the camera. A higher frame rate decreases the integration period and while the opposite increases the frame rate. Pixel-related errors deal with manufacturing errors between neighbouring pixels where two pixels outputs different values of depth for a similar real depth. Other pixel-related errors are time-delay in capacitors. These faults are usually fixed with a table with correction values for each pixel[19]. Amplitude ambiguity is how the amplitude of the reflected infrared light varies with distance to the object to be captured, the colour of the object and wherein the scene the object is situated. The infrared light will not reach the whole scene uniformly, and objects in the periphery of the scene will

therefore not be illuminated as much as the centre[19]. Temperature-related errors appear because the temperature of the depth sensor affects the accuracy of the depth processing[19].

Non-systematic Errors are low Signal-to-noise ratio, Multiple light reception errors, Light scattering and Motion blur[19].

Signal-to-noise ratio distortion is found in areas that are poorly lit by infrared light. In scenes with a substantial difference in depths, the areas furthest from the camera will be more prone to noise than other areas. Multiple light reception errors happen when pixels in the sensor is hit by more than one beam of light. This usually happens because of concavities or sharp edges on the geometry of the captured object. The light will reflect from multiple surfaces and will thus create errors[19]. Light scattering errors are light reflected between the lens and the pixel array of the sensor[19]. This is because the pixel array is reflective and the light hitting a pixel can bounce back up to the lens and down to its neighbouring pixels increasing the infrared intensity there. Motion blur is found in both 3D-cameras and conventional cameras. This effect appears when objects are captured in motion. If the object or scene moves during the integration time, motion blur will appear.

2.5 Point Cloud Manipulation

Point clouds are collections of points within a shared coordinate frame. Most distance-sensor as talked about in section 2.4 describe their measurements as discrete points in three dimensions and outputs a point cloud every sampling period.

2.5.1 Transformation of Point Clouds

When analysing data from point clouds it is useful to have a common reference point or use a coordinate system more intuitive than what the raw capturing device outputs. The SR-300 3d-camera outputs point clouds with the center of the camera as its origo (0, 0, 0). To transform the points into a desired coordinate frame we can use transformation and rotation matrices. [21]

A rotation matrix describes the rotation of a coordinate frame with respect to some other frame. A point in frame a, $\boldsymbol{p}^a$, described as a vector with Cartesian coordinates can be rotated to frame b by

$$\boldsymbol{p}^b = \boldsymbol{R}_a^b \boldsymbol{p}^a \tag{2.5}$$

where $\boldsymbol{R}_a^b$ is then the rotation by some angle between frame a and b, and $\boldsymbol{p}^b$ is the coordinates of $\boldsymbol{p}^a$ in frame b.

The rotation about a principle axis, the x-axis, y-axis and z-axis, are called simple

rotations and can be done by

$$\boldsymbol{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix} \tag{2.6a}$$

$$\boldsymbol{R}_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \tag{2.6b}$$

$$\boldsymbol{R}_z(\psi) = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6c}$$

.

The rotation of any point can be done by a combination of the simple rotations, this is called *composite rotations*. Say we want to rotate from frame a to a new end frame c, and this rotation is not a simple rotation about any of a's principle axis, we can then combine i.e. a rotation about one of frame a's principle axis and a rotation of one of the principle axis of an intermediate coordinate system, frame b. An example

$$\boldsymbol{p}^c = \boldsymbol{R}_a^c \boldsymbol{p}^a \tag{2.7a}$$

$$\boldsymbol{R}_a^c = \boldsymbol{R}_b^c \boldsymbol{R}_a^b \tag{2.7b}$$

$$\boldsymbol{R}_b^c = \boldsymbol{R}_{z^b,\psi} \tag{2.7c}$$

$$\boldsymbol{R}_a^b = \boldsymbol{R}_{x^a,\phi} \tag{2.7d}$$

.

Rotation matrices are only able to describe rotations, with the frames cenceptually sharing an origo point. By combining the rotation matrix with the position distance between the frames we get a *homogeneous transformation matrix*. The position of the origin of frame b in frame a is described by

$$\boldsymbol{r}_{ab}^{a} = \begin{bmatrix} x^{a} & y^{a} & z^{a} \end{bmatrix}^{T} \tag{2.8a}$$

$$\boldsymbol{T}_{b}^{a} = \begin{bmatrix} \boldsymbol{R}_{b}^{a} & \boldsymbol{r}_{ab}^{a} \\ \boldsymbol{0}^{T} & 1 \end{bmatrix} \tag{2.8b}$$

,

with the *homogeneous transformation matrix* in Equation (2.8b) describing both the position and orientation of frame b in frame a.

2.5.2 Filters

2.5.2.1 Radius Outlier Removal

Outliers are points in a point cloud that are not part of the measured surface. They appear randomly or as a consequence of errors in the capturing device. They are usually unwanted as they make statistical analysis on the real points harder. They are often found farther away from other points than the real points and will often have fewer neighbours. One way to remove such points is to use a Radius Outlier Removal algorithm. First we have to specify a number of neighboring points that have to be within a certain radius of each point. The algorithm, as implemented by PCL, then iterates through all points in the point cloud once and removes the points that do not satisfy the specified number of neighboring points within a certain radius.

2.5.2.2 Range Filter

A range filter is a very simple filter that removes points that go outside some specified range in any combination of x, y or z direction.

2.5.3 Edge Detection

Edge detection is a range of mathematical methods to identify points in digital imagery or similarly structured data where, in images, the brightness or colour changes rapidly. Classic edge detection methods like the Canny edge detector [22] use the derivative of the brightness of the points in the image, as well as a thresholding scheme to find the points that are edges. The wanted output of an edge detection algorithm is a copy of the image with a binary value representing if the point is an edge or not.

Images are typically ordered in a two-dimensional n-by-m matrix where n is the horizontal resolution and m is the vertical resolution of the image. The fields at each coordinate are RGB data, for colour images, or brightness for gray scale images. Classical image edge detection algorithms can be used on point clouds that are *ordered*. This ordering relates to where the points are located in a datastructure related to its spatial information. Ordered point clouds have a relationship between two of its three variables for each point related to the data structure. For a point cloud captured by a structured light camera, like the SR-300, we get a "picture" of 640x480 points where the "brightness" of the pixels are z-distance from the camera to the object. The x and y values are also calculated to real world coordinates in meters, but the relationship between the points are kept.

2.5.3.1 Detection of Potential Edge Points

As mentioned earlier the derivative of the brightness can be used to find edges. Both first-order and second-order derivatives can be used for edge detection[23]. We can

see the general connection between the intensity of the image and the first and second derivatives in figure 2.5. What we see is that for an uniform increase in intensity from left to right we get a steady state positive first order response. If the intensity decreased the sign of the gradient would be negative. The second order gives a sharp response at the start of the increasing intensity and a sharp negative response when the intensity stops increasing. In the middle of the increase we get a zero-crossing point. This zero-crossing point can be used to find very sharp edges.

A tool for finding the edge *strength* and direction is the first order gradient ∇f which is defined as a vector

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.9}$$

this vector points in the direction of the greatest rate of change of f at the location (x, y). To find the absolute magnitude, M, of the rate of change at (x, y) we can use the pythagoras theorem

$$M(x, y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2} \tag{2.10}$$

the angle of the greatest rate of change is given by

$$\alpha(x, y) = tan^{-1} \begin{bmatrix} g_x \\ g_y \end{bmatrix} \tag{2.11}$$

To calculate the partial derivatives of every pixel in the image we can us a approximate with a discrete method. The following equations give us the edges in vertical and horizontal direction

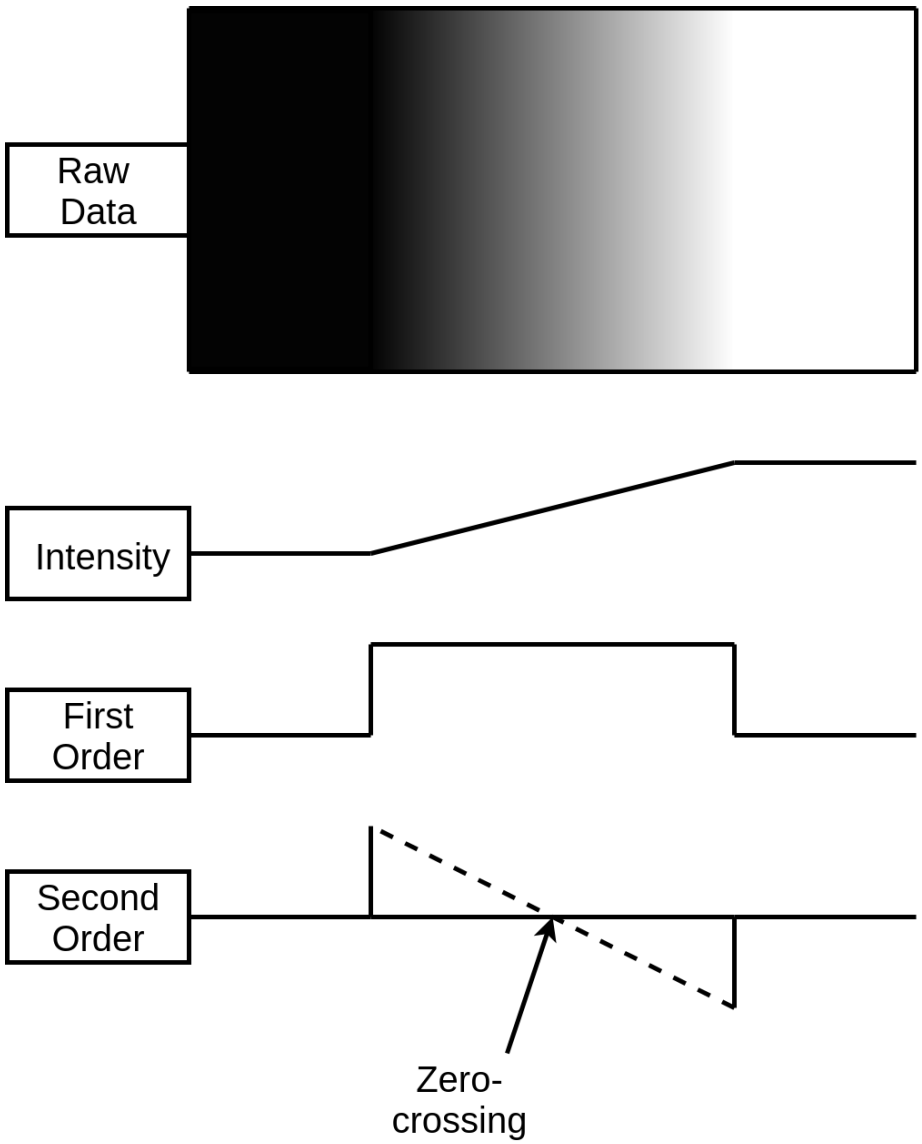


Figure 2.5: General connection between intensity, first order and second order derivatives

$$g_x = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y) \tag{2.12}$$

$$g_y = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y) \tag{2.13}$$

To capture vertical, diagonal and horizontal edges there are two famous masks, Prewitt and Sobel.

Prewitt:

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.14}$$

Sobel:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.15}$$

The main difference between Prewitt and Sobel is the 2 in the middle aisle. This gives a smoothing effect suppressing noise better. To apply these masks over the image or point cloud an operation called convolution is applied. The implementation of convolution is to let the current coordinate be the centre of the mask and then multiply the neighbouring points with the corresponding value in the mask and then add them together.

Consider the image Z with 3x3 pixels

$$Z = \begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix} \tag{2.16}$$

then we apply the Sobel mask on point z_5

$$g_x = \frac{\partial f(z_5)}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{2.17}$$

$$g_y = \frac{\partial f(z_5)}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{2.18}$$

2.5.3.2 RANSAC - Algorithm

The RANSAC (RANdom SAmple Consensus) - algorithm was outlined in [24]. The algorithm tries to fit a mathematical model of geometric shape, such as a plane, cone, sphere, line, circle or cylinder to a set of observed data. The algorithm assumes there are some "inliers", which fit the geometric shape, and "outliers" which are points that do not fit the model.

To find the best mathematical model the algorithm will choose a set number of points from the data set at random, then it will fit the mathematical model to these points. Then it will iterate over every point in the data set and if the point fits the model within a threshold value it will be added to the set of points that fit the model. After all the points have been tested the algorithm checks if the amount of points fitted to the model is over some threshold. If the threshold is reached it will check every point in the fitted points and get the error between the model and the points found. When the error is recorded it will do the whole algorithm again with a new set of randomised starting points. The algorithm will iterate as many times as you want and after each

iteration compare its current error with the smallest error found so far. When the iterations are over you can extract the best model the algorithm was able to find. The longer you let the algorithm run the more certain you can be that you have found the best fit for you model.

2.5.4 Registration of Point Clouds

This section taken from project thesis [10]
The problem of matching two 3D models represented by point clouds is called *registration*. This is a broad field with many algorithms. [25] presents an overview of the different algorithms and methods used to register 3D models. This article mentions two classes of registration, rigid and non-rigid. Rigid registration is where the two 3D models to be matched have the same dimensions, and the proportions of the features are equal. Non-rigid registration, on the other hand, deals with matching a deformed model with the original model. Deformed models could be soft objects like biological organs or trying to match a folded piece of cloth with an unfolded piece. Any object that does not have static proportions needs to use non-rigid registration techniques.

2.5.4.1 Algorithms

Registration algorithms have much in common with the field of data fitting. Thus Registration problems are often proposed as optimisation problems. The most widely used rigid registration algorithm is the ICP(Iterative Closest Point)-algorithm[26]. The ICP-algorithm tries to minimize the cost function equation (2.19), $\boldsymbol{q_i}$ and $\boldsymbol{p_i}$ are points in the two point clouds we want to register. $\boldsymbol{R}$ and $\boldsymbol{t}$ are Rotation and Translation matrices. The algorithm works by first finding the closest points in P, the static point cloud, to the points in Q, the point cloud to be matched. Then the algorithm finds the best transformation and rotation to bring the points in Q closer to its closest point in P. If This translation and rotation are smaller than a set threshold, the algorithm

will finish, if not it translates and rotate the points to the new position and do the algorithm all over again.

$$E(\boldsymbol{R}, \boldsymbol{t}) = min_{R,t} \sum_{i} ||\boldsymbol{q_i} - (\boldsymbol{R}\boldsymbol{p_i} + \boldsymbol{t}||^2 \tag{2.19}$$

2.5.4.2 Implementation

To implement the ICP algorithm, there are two reasonable possibilities. The Computer Vision Toolbox System in MATLAB[27] or the open source Point Cloud Library (PCL)[28]. There is only one registration algorithm in the MATLAB toolbox with some helper functions like denoising and downsampling of the point clouds. The PCL is widely used in more advanced academia and the computer vision industry[29] and offers many different registration algorithms and filtering options. MATLAB is also able to use PCL's point cloud file format ".pcd"[27].

Chapter 3

Post Processing

The main goal of the post-processing is to extract the relevant geometric data from the captured point clouds. The parameters we are trying to extract are the height, road width and center point of the extruded material.

3.1 Transforming the Point Cloud

The first step is to calibrate the captured point cloud and transform it so that the points are aligned with the world coordinates. Figure 3.1 gives an overview of the difference between the world aligned coordinate system t and the coordinate system of the capture points from the camera c. Using calipers the necessary measurements were measured on the camera and extruder and combined with a 180° rotation about the

x-axis and inserted into the transformation matrix $\boldsymbol{T}_c^t$ as described in equation (3.1).

$$\boldsymbol{T}_c^t = \begin{bmatrix} & & dX \\ \boldsymbol{R}_c^t & & dY \\ & & dZ + dH \\ \boldsymbol{0} & & 1 \end{bmatrix} \tag{3.1a}$$

$$\boldsymbol{R}_c^t = \boldsymbol{R}_x(180^\circ) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(180^\circ) & -sin(180^\circ) \\ 0 & sin(180^\circ) & cos(180^\circ) \end{bmatrix} \tag{3.1b}$$

$$\boldsymbol{T}_c^t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0.101 \\ 0 & 0 & -1 & 0.199 + dH \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1c}$$

This single translation would be enough if the camera mount was completely rigid, but because of slack in the mounting equipment a certain error was observed after transforming to world coordinates. The mechanical construction of the cameras built-in tilt hinge was also not rigid enough for perfect positioning, both figure 3.2 and figure 3.3 show the coordinate frame floating above the points. To remedy this problem a solution based on plane fitting using the RANSAC-algorithm, explained in section 2.5.3.2, was used. The RANSAC algorithm outputs a model of a plane described by its surface normal and distance from origin. The surface normal is described in relation to the fixed coordinate system by a vector, for experiment 2 we get the vector in equation (3.2a). From [30] we have that a vector can be described by two angles α and β. These angles can be inserted in rotations about the Z and Y-axis, equation (3.2d), to rotate the plane into our preferred coordinate system. By also translating the point

cloud by d we get a good registration with the world coordinate system, as seen in figure 3.4. After the transformation to world coordinates the z value of all points should be the height above the work table.

$$\boldsymbol{N}_p^t = \begin{bmatrix} k_x \\ k_y \\ k_z \\ d \end{bmatrix} = \begin{bmatrix} -0.0224 \\ 0.0149 \\ 0.9996 \\ 0.0030 \end{bmatrix} \tag{3.2a}$$

$$sin\alpha = \frac{k_y}{\sqrt{k_x^2 + k_y^2}} \quad cos\alpha = \frac{k_x}{\sqrt{k_x^2 + k_y^2}} \tag{3.2b}$$

$$sin\beta = \sqrt{k_x^2 + k_y^2} \quad cos\beta = k_z \tag{3.2c}$$

$$\boldsymbol{R}_p^t = \boldsymbol{R}_{z,\alpha}\boldsymbol{R}_{y,-\beta}\boldsymbol{R}_{z,-\alpha} \tag{3.2d}$$

$$\boldsymbol{T}_c^t = \begin{bmatrix} \boldsymbol{R}_p^t & \begin{matrix} 0 \\ 0 \\ d \end{matrix} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{3.2e}$$

3.2 Filtering

The depth frame output from the SR-300 camera has a resolution of 640*480 which results in 307200 points in the resulting point cloud. This covers a large area of the work table while we are mostly interested in what is directly behind the nozzle of the extruder for online, real-time, geometric information. This means that we have a lot of unnecessary information that we can remove to ease up calculation time.

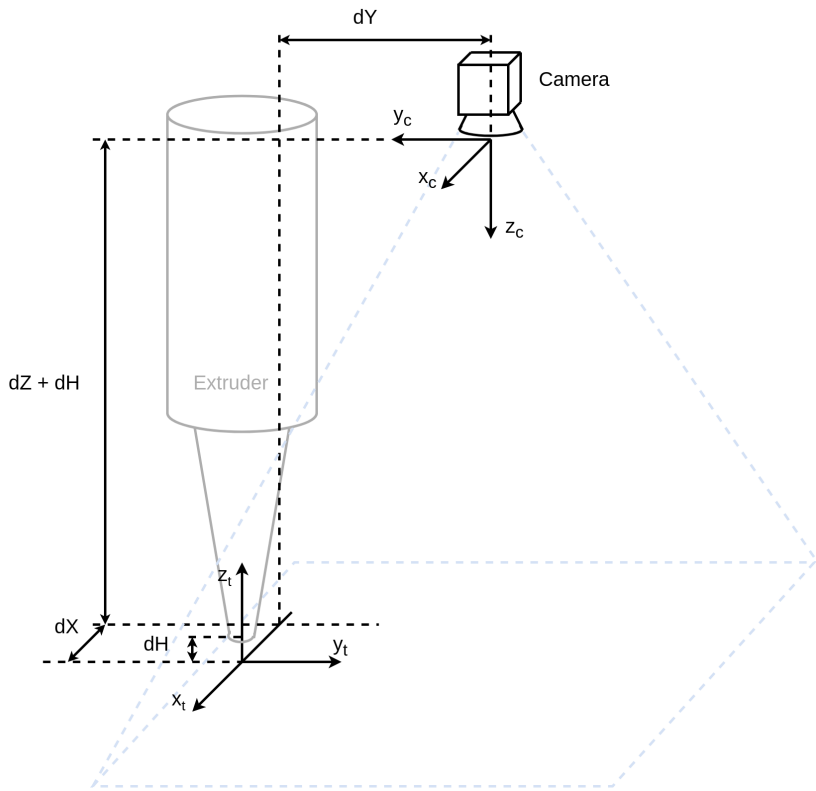


Figure 3.1: Relation between coordinate frame of the camera and the extruders coordinate frame

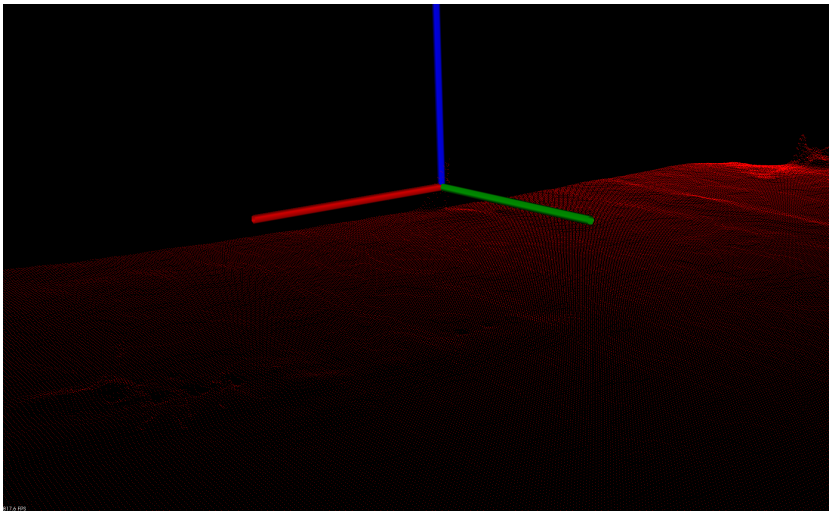


Figure 3.2: Point cloud of work surface before extrusion. The displayed coordinate frame is in origo (0,0,0) of the cloud.

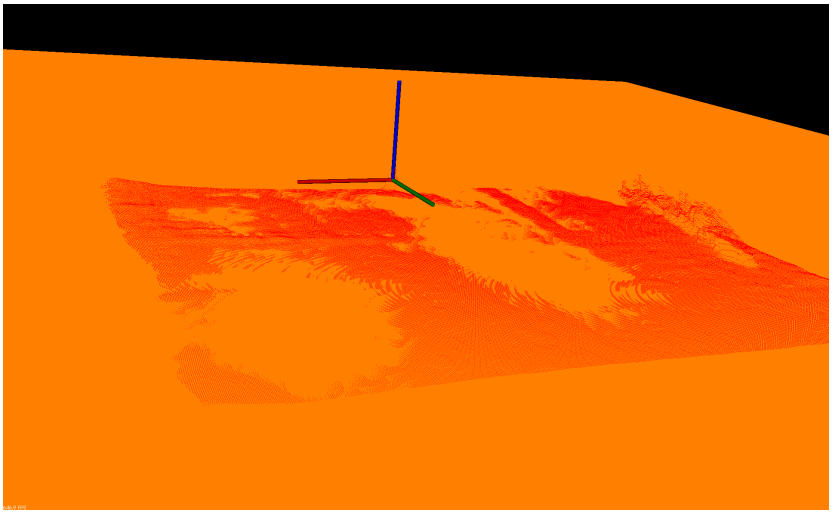


Figure 3.3: Plane fitted to the captured point cloud coloured in orange.

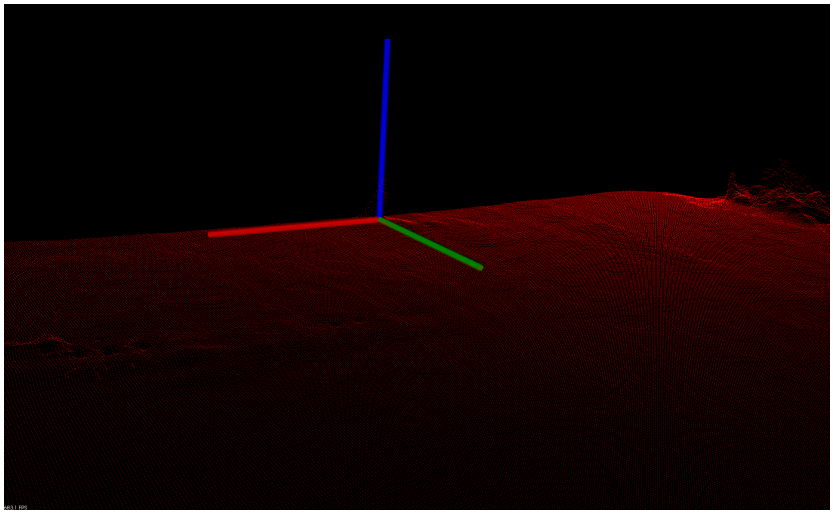


Figure 3.4: Transformed point cloud with the surface of the work table as zero in z-direction. We see that the surface of the point cloud fits the displayed coordinate frame.

Table 3.1: Comparison between full and filtered point clouds

Nr. of points	*Time to Range Filter (ms)*	*Time to Remove Outliers (ms)*	*Time to transform (ms)*	*Total time (ms)*
307200	0	5150.5	272.5	5445
2883	5.4	5.8	2.6	13.8
Percent reduction	-	~99.9%	~99%	~99.75%

There are two primary filters we use for our point clouds; a simple range filter and an outlier removal filter, as described in section 2.5.2. Table 3.1 shows a quick comparison between working with the full point cloud and a smaller one. If we only look at a small area around the extruder we can remove about 99% of our point cloud, which reduces the total filtering and transformation time by 99.75%.

3.3 Edge Detection for Extracting Geometric Data

Building on the principles from section 2.5.3 a simple edge detection algorithm was implemented. The method implemented uses a 3-by-3 Gaussian filter, the gradient is calculated using either the Prewitt or Sobel mask and a simple algorithm that looks for the largest gradients was created to choose the edges. Before testing the algorithm on real world data its viability was tested on ideal point clouds with known dimensions. The ideal point clouds were sampled with the same distance between the XY-points as observed from the experimental data. The distance observed from i.e. figure 4.3, was found to be approximately 0.5mm. The points are also generated as if they were seen from above, so only "visible" points are created. Three different cases were considered, a straight line, a straight line with incremental offset of the centre point and an incremental increase in width. The edge detection was tested first on a noiseless point cloud and then with added noise. The noise added was normally distributed with a standard deviation of 0.1mm. This noise was applied as an offset to the centre point and to the height value of each point in the cloud.

Points in the point clouds used in this section will be coloured by their gradient

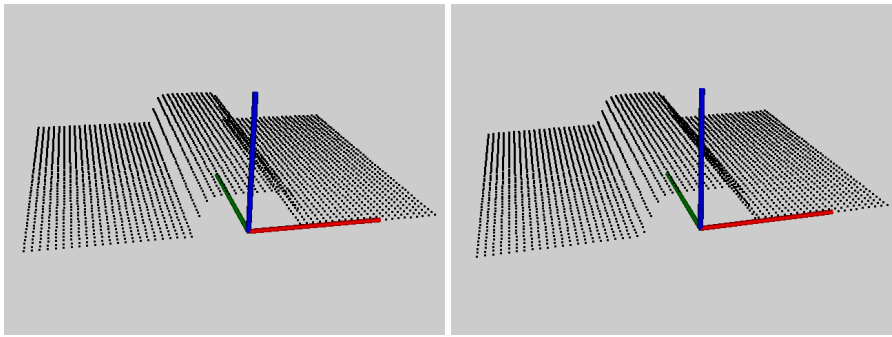


Figure 3.5: Straight line as a point cloud. Unfiltered to the left, Smoothed out by Gaussian kernel to the right. There is only a slight visible smoothing of the edges closest to the "floor".

intensity from black, zero intensity, through red, orange, yellow to white as the highest intensity.

3.3.1 Straight Line

The first test was a simple straight line with height of 3mm and 7mm width. These dimensions are quite similar to the real world experiments conducted earlier. With no noise all the gradient kernels gave the same result, giving the correct width as seen in table A.1. This first test with no noise is also perfectly centered and the outer edges of the line coincides with the resolution of the point cloud giving a point exactly on the edge. Figure 3.5 gives an overview of how the straight line with no noise looks, both smoothed out and raw data. figure 3.6 show the gradient of the points colour coded. When we add noise we can see from table A.5 and table A.4, that Sobel have an error of 7.14% on two measurements. Prewitt gives us the same result as well but only after first being filtered.

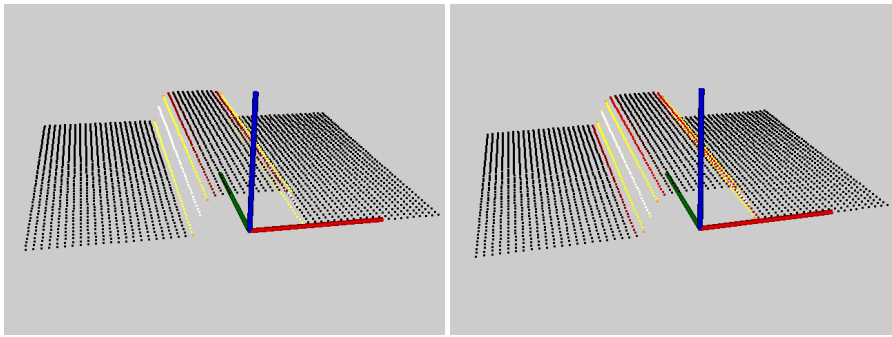


Figure 3.6: Calculated gradient with Sobel operator. Unfiltered to the left, Smoothed out by Gaussian kernel to the right.

3.3.2 Incremental Offset

The second test increments an offset of the center of the line by 0.125mm every 5mm. From table A.2 we see that we get the same error of 7.14% for all offset values that are not equal the sampling distance. When adding noise we see from table A.6 and table A.7, that we get the same error of 7.14% except for a few cases. The cases where we get 0.0% error is when the applied offset noise pushes the model so that it fits between the sampling distances.

3.3.3 Incremental Change in Width

In the third experiment the width of the line was increased from 6mm to 8.25mm with an increment of 0.25mm every 5mm. Looking at the data from the noiseless test, table A.3, we see that every width is rounded down, floored, to closest millimetre. The tests with added noise, table A.8 and table A.9, show actually a better results in width accuracy. This can be attributed to the noise induced as an offset in the x-direction and this added noise may push the profile of the line better aligned with the sampling

distances.

3.3.4 Summary of Edge Detector on Ideal Cloud

The proposed edge detection algorithm finds the edges in the tested ideal cloud, but we are inherently limited by the resolution of the capturing device. Figure 3.7 gives a visual representation of the quantization errors that have been found by this analysis. With the top example giving us the best possible result with the edges of the object aligning with the sampling distance of the capture device. The worst possible result gives us an error close to two times the sampling distance, as seen from lower example in figure 3.7.

3.3.5 Proposed Algorithm

Initialization:

- Use the transformation from camera coordinate system to transform the first point cloud into world coordinates.
- Use the RANSAC-algorithm to find the normal vector of the plane through the points in the point cloud transformed to world coordinates.
- Use the normal vector to find a calibration transformation to complement the camera transformation which will align the points with the world coordinates.

Online:

- Remove 99% of the points, only keeping the points in a box around the nozzle of the extruder.

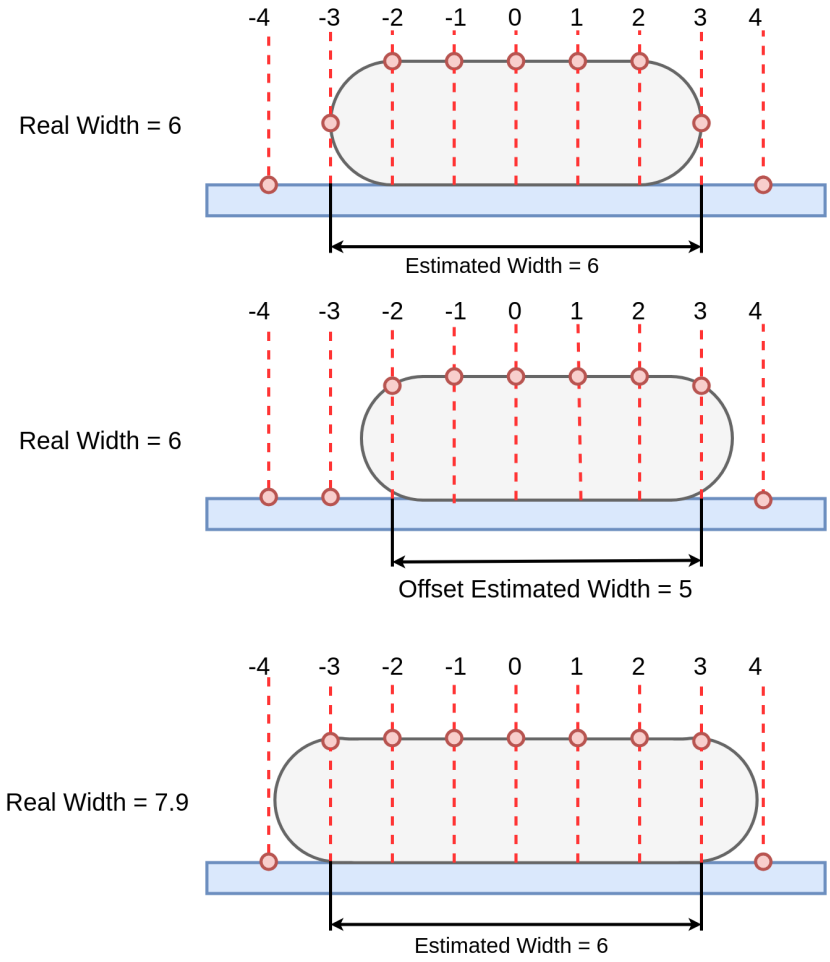


Figure 3.7: Quantization errors for a scanner with resolution of 1.

- Transform this reduced point cloud into world coordinates using the camera transform combined with the calibration transform.
- Remove every point closer to the camera than the nozzle.
- Transform the smaller cloud to world coordinates.
- Optionally do a Gaussian filter to remove noise.
- Calculate the gradient for every point in the smaller cloud.
- Calculate the average gradient for every point a certain distance from the y-axis back to the last sample. This distance along the y-axis is equal to the product of velocity and time since last sample.
- Choose the distance from y-axis with highest average gradient both in positive and negative x-direction. These are our positive and negative edges.
- The height is found by taking the average of the points between the two edges with an offset of 1/4th of the width between the edges

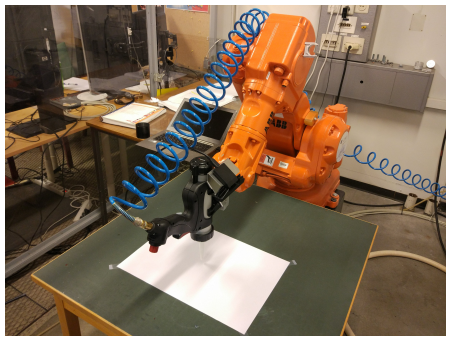
Chapter 4

Experiments

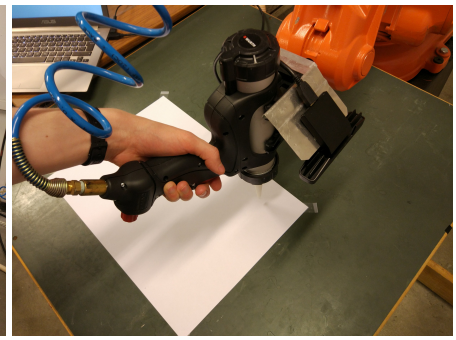
The aim of the experiments is to validate the use of the Intel SR-300 as a capture device of point clouds as well as test the proposed post-processing pipeline with the end goal of extracting an error between the desired trajectory and real world geometric data. The experiments will also highlight positional requirements of the capture device for best possible quality of the captured point clouds. We will also see if we are able to detect some of the common errors mentioned in the background chapter.

4.1 Experiment Setup

The experiment setup is based on the proof of concept experiment conducted in [3]. Here the robot arm was mounted with an pneumatic caulking gun using a type of fast-curing construction glue, STP Quickfast from Würth. This material and method was chosen over traditional 3D-printing methods because of the extrusion volume required by the larger scale this project focuses on and its ease of use. The material is extruded by compressed air controlled by a manual valve with a variable flow rate.



(a) IRB-140 robot by ABB



(b) Manual extrusion using the chaulking gun

For the testing an industrial robot made by ABB was available. The robot in question is the IRB-140 [31]. This is a six-degrees of freedom multipurpose robot arm able to lift 6kg with good repeatability and speed. The robot is controlled either manually via an ABB Flexpendant or by a pre written program either running on the FlexPendant or on a PC connected with a TCP/IP-connection. The programming language used by ABB is called RAPID.

4.1.1 Capturing Software

For capturing the point clouds a program written in C++ using the Intel SR-300's Software Development Kit [32] and the Point Cloud Library [29] was used. This program took ten pictures a second saving them to the harddrive with a timestamp. This enabled us to try different post-processing schemes after the experiments and not having to rely on the real-world robot at all times.

4.1.2 Camera Mount and Positioning

The camera was mounted as seen in figure 4.1b pointed vertically down towards the table. From this distance the mean distance between the points was 0.45mm.

4.2 Creating the Test Samples

To test the method of visual feedback we wanted to do a few straight lines with different height and size profiles, while capturing point clouds of the extruded material along the path. These would later be post-processed to extract the relevant geometric information. From some initial trials with the manual caulking gun and robot setup three different height profiles were chosen; 3mm, 4mm and 5mm. . The first experiment was just one layer while the second experiment was two layers stacked on top of each other. A final test was conducted to see if the camera was able to capture the error states of over- and underfill. After some initial trials with the caulking gun and different velocities of the tool-head a velocity of 25mm/s was chosen for best possible quality of the extruded material.

4.3 Measuring the Test Samples

The width and height of the samples were measured by hand using a pair of digital calipers. The calipers were accurate down to a hundredth of a millimetre, but the measurements are only certain to ±0.2mm. This is because the extruded material was very soft after extrusion and the measurement jaws of the calipers were very sharp. This meant that the jaws of the calipers very easily dug into the material. It was up to human dexterity and eyesight to determine when the calipers were in direct contact and as such the full accuracy of the measuring tool were not possible to utilise. We use the method proposed in section 3.3.5 to record the width and height of the extruded material. With a velocity of 25mm/s and 10Hz the tool-head moves 2.5mm per sampled point cloud. The algorithm will look at the edges 2.5mm along the trajectory and average the gradients and heights and combine to one measurement for each sample.

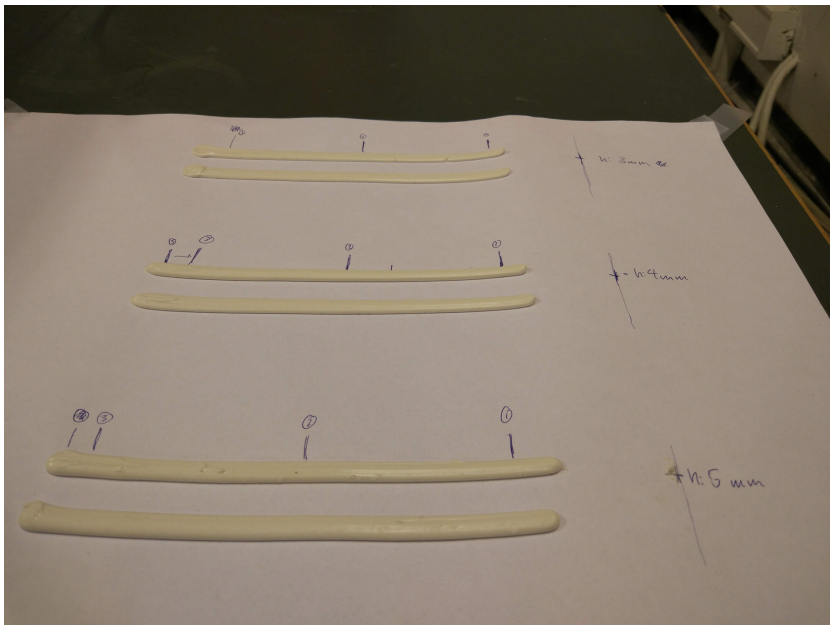


Figure 4.2: View of the three pairs of different single layer tests, with 3mm at the top and 5mm at the bottom.

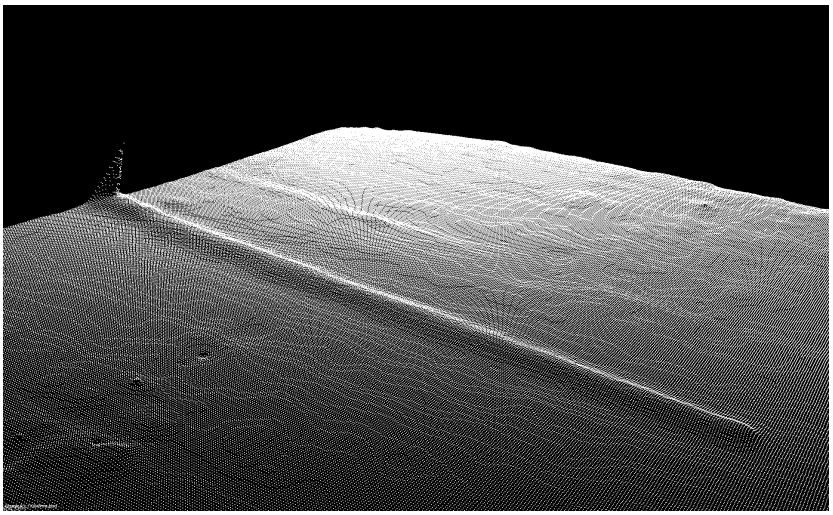


Figure 4.3: View of one extruded line with the the extruder tool tip set to 3mm above the table. Extruding from right to left. Point cloud captured at end of line extrusion.

4.4 Quality of Captured Point Cloud

The quality of the captured clouds varied quite a bit with the height of the extruded layers. For the very low height at 3mm we can see from experiment 2, figure 4.4, that the edges are very smooth and averages out. Increasing the height to 5mm we see that in experiment 5, figure 4.5, the edges become more distinct and we begin to see a bias to the left edge of the extruded lines. This bias is even more prevalent in experiment 14 set to 8mm, as seen in figure 4.6.

4.5 Straight Line

For each height profile two extrusions were completed to increase the likelihood of a good result, resulting in six experiments. Figure 4.2 show the three pairs of test lines

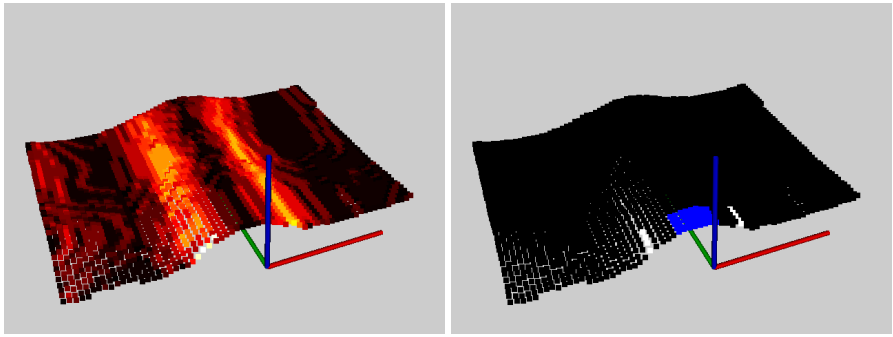


Figure 4.4: Experiment 2 at 120mm, gradient to the left and the edges and area used for height shown to the right.

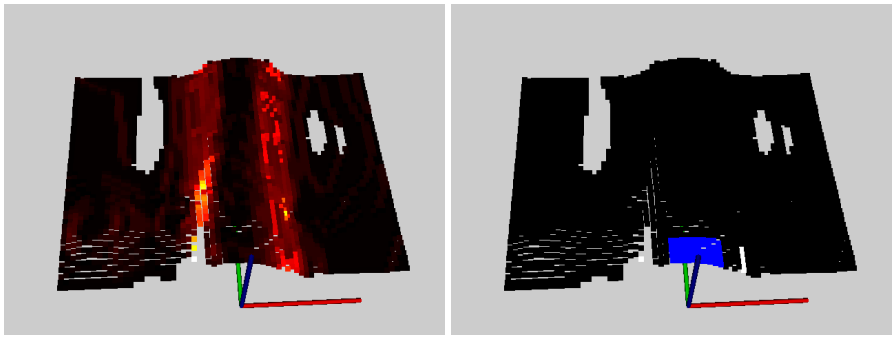


Figure 4.5: Experiment 5 at 120mm, gradient to the left and the edges and area used for height shown to the right.

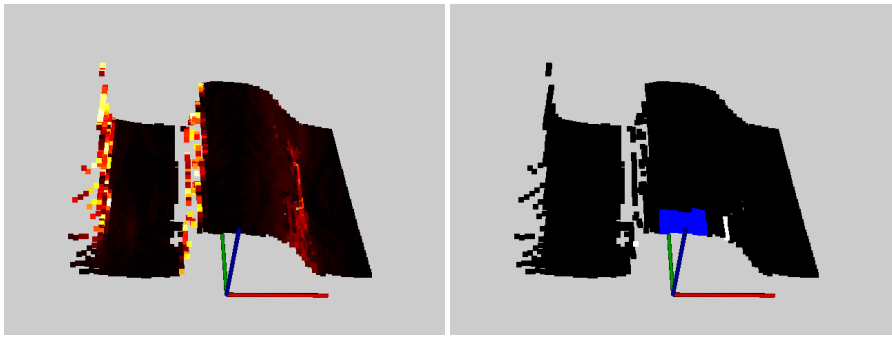


Figure 4.6: Experiment 14 at 120mm, gradient to the left and the edges and area used for height shown to the right.

at 3, 4 and 5mm height. Figure 4.3 gives an overview of how a fully extruded line looks like in point cloud form. A

4.5.1 Height

First of all we can see from table 4.1 that the real height of the extruded material is lower than desired. The height from the nozzle to the table was checked before every test, but the extruder was actually a bit loose in its mounting and this may have caused the problem. Since the extruder had to be operated manually the error could have been induced by the weight of the operators arm. The error is however quite stable and can be seen to be approximately the same for each pair of experiments.

The raw height data can be seen in the height-plots in the appendix, figure B.1 to figure B.11. They look erratic, but they seem to be quantization errors. They are not perfect quantization errors as they are an average of many measurements, but if we look at the averages in table B.1, we see that most of the measurements have a standard deviation close to 0.1mm indicating. The height plots for the different experiments also show a sawtooth pattern around its average value, indicative of quantization errors.

4.5.2 Width

We see quite a lot of the same behaviour with the width measurement. The values in figure B.2 to figure B.12 are also fluctuating the same way as the height measurement, but with a larger error. If we compare the average values with the quantization errors visualized in figure 3.7, we see that with a mean distance between the points of 0.45mm and a maximum error of just shy of $2 * 0.45 = 0.9$mm is quite close to what we see in table B.2.

4.6 Straight Line, 2 Layers

The straight line with two layers was conducted the same as the single layer experiment, but the height of the extruder was incremented by the desired layer height for a second pass. A total of 10 experiments were done, with the same layer heights as the first set of experiments. We did run out of material for the last 5mm height experiment and thus theres only one two-layer 5mm experiment. One other problem with the manual measurements was that they were conducted after the two layers had been extruded, and we see that the width of the first layer for each two layer experiment is much wider than its corresponding experiment in the single layer experiments. This is because the first layer was still soft and sticky when the second layer was applied, this pushed the first layer down into the table and out to the sides.

When looking at the resulting plots for width and height, figure B.13 to figure B.32, we see much of the same behaviour as in the first series of experiments. The errors are in the same ballpark as the single line experiments. The exception being the first layers of the two layer test (odd numbered experiments. e7, e9 and so on), since they were measured after the second layer was extruded.

Table 4.1: Results from single layer experiment

Experiment	*Target Height (mm)*	*Measuring Position (mm)*	*Measured Height (mm)*	*Measured Width (mm)*
		50	1.8	5.7
1	3.0	120	1.8	7.8
		180	1.8	8.7
		50	2.0	7.7
2	3.0	120	1.8	8.5
		180	1.9	8.4
		50	2.4	5.5
3	4.0	120	2.5	6.7
		180	2.5	7.6
		50	2.5	6.7
4	4.0	120	2.5	7.8
		180	2.5	8.4
		50	3.1	6.4
5	5.0	120	3.2	7.0
		180	3.2	7.8
		50	3.2	7.2
6	5.0	120	3.3	7.6
		180	3.3	8.0

4.7 Under-, Over- and Normally Filled

The under, over and normally filled test was done by extruding five parallel lines with constant pressure in the caulking gun for each test case. They were separated by 5mm and the length was the same as earlier experiments, for underfilled the pressured was lowered to reduce the flow rate of the material, it was then increased for the normally filled test and then increased even further for the overfilled test. We can see the gaps between the extruded lines in figure 4.8 for the underfilled test. The normally filled test, figure 4.7, show a quite smooth surface with small grooves between the lines. The overfilled experiment, figure 4.9, show a very rough surface.

Table 4.2: Results from double layer experiment

Experiment	*Target Height (mm)*	*Measuring Position (mm)*	*Measured Height (mm)*	*Measured Width (mm)*
		50	2	10.2
7	3.0	120	1.8	10.7
		180	1.8	11
		50	4	7.8
8	6.0	120	3.9	8.8
		180	3.9	9.7
		50	1.8	10.7
9	3.0	120	1.8	12
		180	1.8	12.1
		50	3.9	7.8
10	6.0	120	4	9
		180	3.8	10
		50	3	9.6
11	4.0	120	3	11.3
		180	3.2	12
		50	6.8	7.8
12	8.0	120	6.8	8.5
		180	7	9.3
		50	3	11.5
13	4.0	120	3	12.7
		180	3	12.9
		50	6.8	8.5
14	8.0	120	6.7	9.4
		180	6.2	9.7
		50	3	7.8
15	5.0	120	3.7	10.5
		180	4	10.1
		50	7.9	6.7
16	10.0	120	7.8	6.5
		180	7.9	6

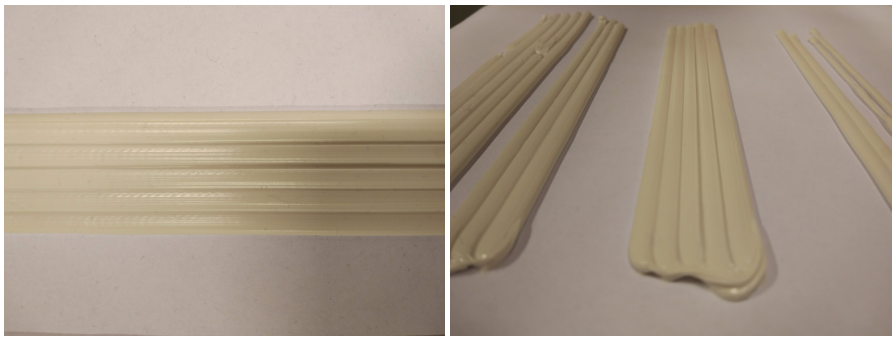


Figure 4.7: Normally filled extrusion.

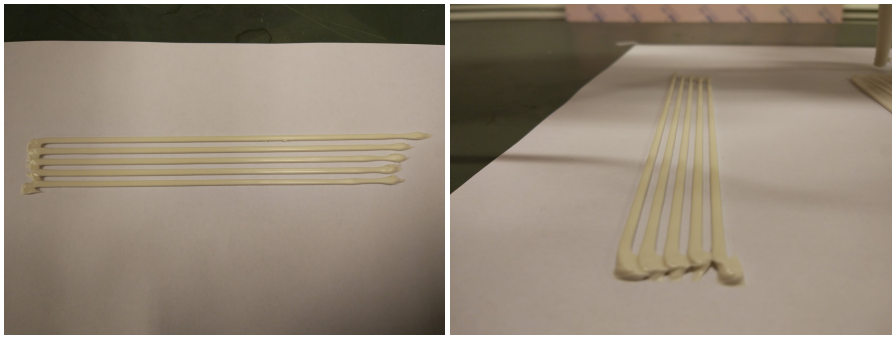


Figure 4.8: Underfilled extrusion.

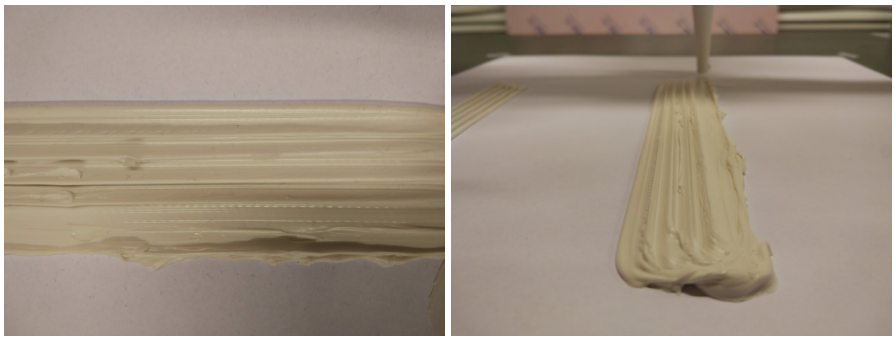


Figure 4.9: Overfilled extrusion.

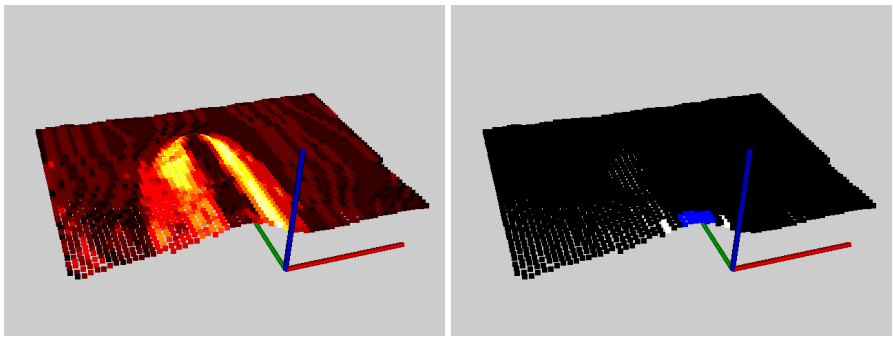


Figure 4.10: Normally filled extrusion at 50mm

4.7.1 Normally filled

If we look at figure 4.10 we see the same results as in the straight line experiments. If we then look at the subsequent lines in figure 4.11 we see that we are not able to recognise any features in the surface. We get the high gradient at the edges as we expect from the straight line experiments, but we are not able to distinguish any other lines.

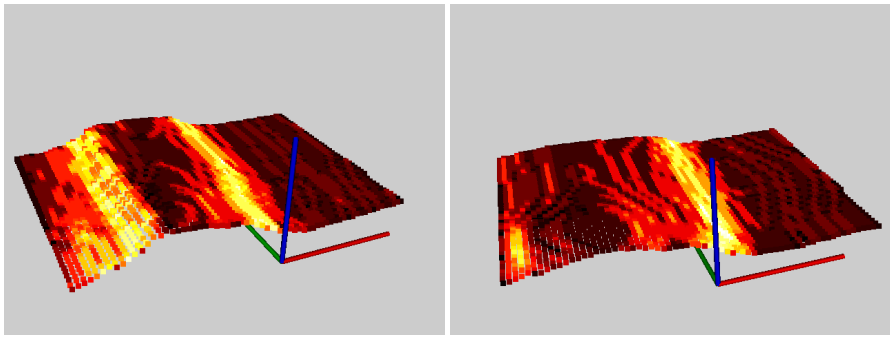


Figure 4.11: Normally filled extrusion at 120mm, line 2 to the left and line 4 to the right.

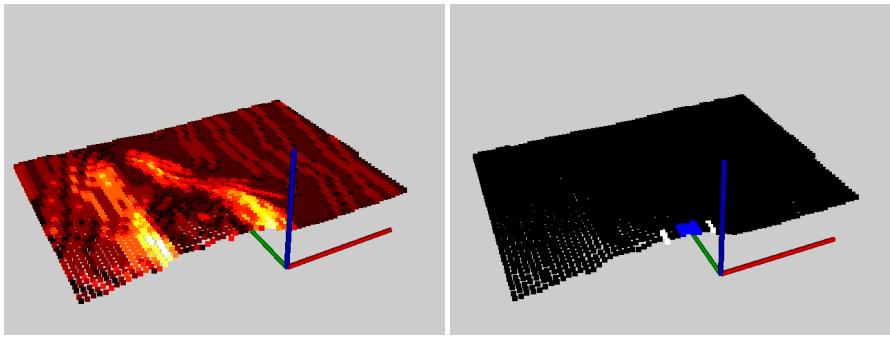


Figure 4.12: Underfilled straight line at 50mm, second line

4.7.2 Underfill

In the underfilled test we see much the same as in the normally filled. The capture device have trouble representing the fine details of the gaps seen in section 4.7. The surface of the extruded lines are however flatter than seen in figure 4.11. In line two seen in figure 4.12, we can also see a small dip between the two lines to the left, which is captured in the visualization of the edges to the right. There are indications of a dip in the right picture in figure 4.13 as well.

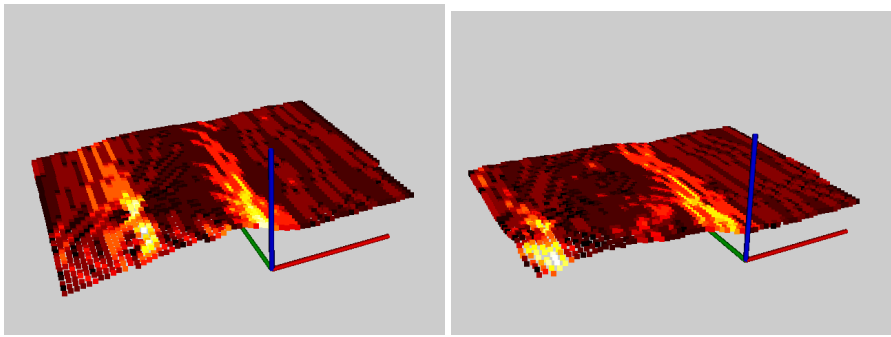


Figure 4.13: Underfilled extrusion at 120mm, line 2 to the left and line 4 to the right.

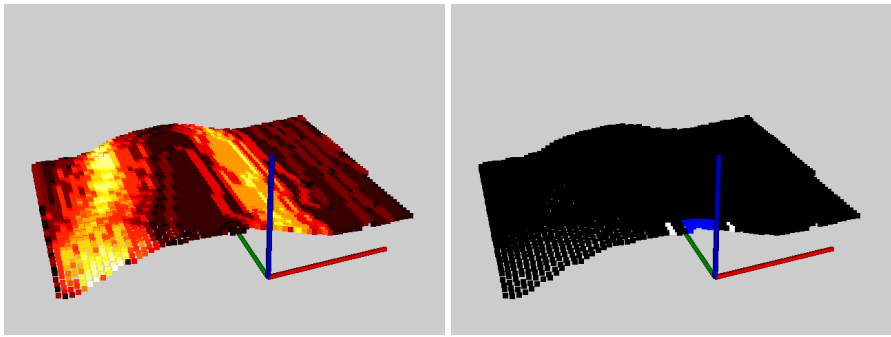


Figure 4.14: Overfilled straight line at 50mm, second line

4.7.3 Overfill

We see much the same qualitities as seen in the normally filled test. With no discernible or trustworthy edges in figure 4.14, we also see in figure 4.15 that the more lines we add the bigger the concavity of the surface increase.

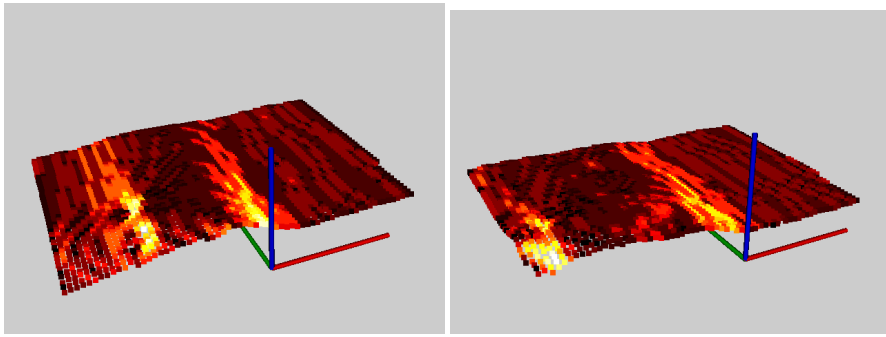


Figure 4.15: Overfilled extrusion at 120mm, line 2 to the left and line 4 to the right.

Chapter 5

Discussion

5.1 Capture Quality and Sensor

In the experiments the Intel SR-300 3D-camera was used. This camera outputs point clouds with a refresh rate of 30Hz and is quite suitable for real time depth perception, however we have found that it is not quite suitable for the purpose proposed in this thesis. The distance between the points in the xy-plane is too large to get a very accurate reading as seen through the results of the experiments. The resolution of 0.5mm is close to 10% of the width of the objects we measured and is dominating our errors as shown in figure 3.7. The depth-resolution however is quite good and as can be seen from many of the experiments seem to be close to 0.1mm. Another prevalent error is smoothing of the depth around objects, this reduces the accuracy of the edges at low changes in depth and reduces accuracy when dealing with error modes such as underfill and finding edges in normally filled extrusions. This smoothing also smooths out the overfill errors and makes them hard to capture.

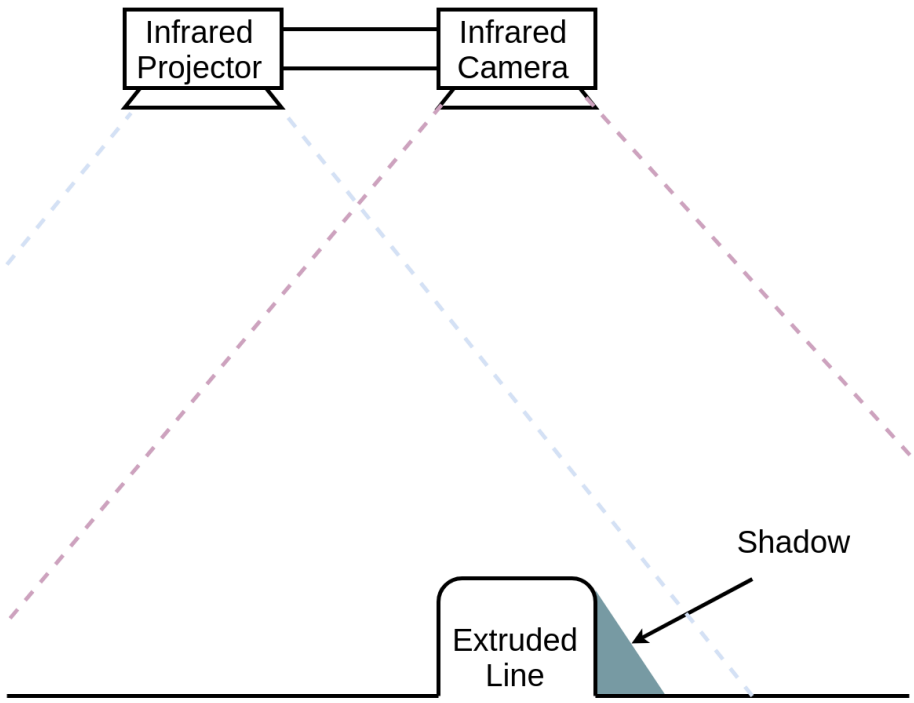


Figure 5.1: Working Principle of Structured Light Camera.

The smoothing error is a consequence of how the SR-300 camera functions, figure 5.1 gives an overview of how the infrared projector and camera is mounted relative to each other. The same orientation of projector and camera is the same in our experiment as in the figure and we see that the projector gets a small shadow on the right side of the extruded line. This problem could be alleviated by mounting the camera further to the right, but then the same error may appear in the camera.

For further testing with the main project and Cold-Metal-Transfer welding a high precision, high quality laser profile sensor from Micro-Epsilons scanControl product

line may be advisable [33]. The additive manufacturing process using a laser metal-wire deposition technique mentioned in the background chapter had some trouble with welding and getting good readings with their laser sensor from the very hot glowing metal. Micro-Epsilon has a version of their laser profile sensor that uses a blue shorter wave-length light that they suggest should be better for such a purpose. These sensor are also very accurate with down to 2µm accuracy.

Regardless of sensor, with the type of feedback strategy tested in this thesis, the sensor have to be directly tangential to the trajectory to be able to capture the width and height of the extruded material immediately after extrusion. This might become impractical, because of the solutions this requires. One solution requires multiple sensors around the tool-head to be able to capture the extruded material in all axis of movement. Another solution is to have the tool-head rotating with a fixed capturing device or having the capturing device on a rotating mount that keeps it always tangential to the trajectory. This is also why we only tested straight extrusions in the experiments, the way the tool-head was mounted on the robot arm blocked it from rotating with a curved trajectory in the xy-plane.

5.2 Edge Detection Algorithm

The edge detection algorithm used in the experiments used the first order gradient of the height, this seemed to be fine at least for the ideal point clouds in section 3.3. The problem arises when we want to find sharp valleys or tops in our point clouds which might be necessary for detecting under- and overfill. This can be seen in figure 5.2 where a perfect normally filled layer would look something like the one to the right. In the same plot the edges found by using our proposed algorithm are coloured in white. We can see that the right edge is found correctly as before but the edge where the two parallel lines connect have a slight offset in the narrow underfill example. We would like the left edge to be in the bottom of the valley.

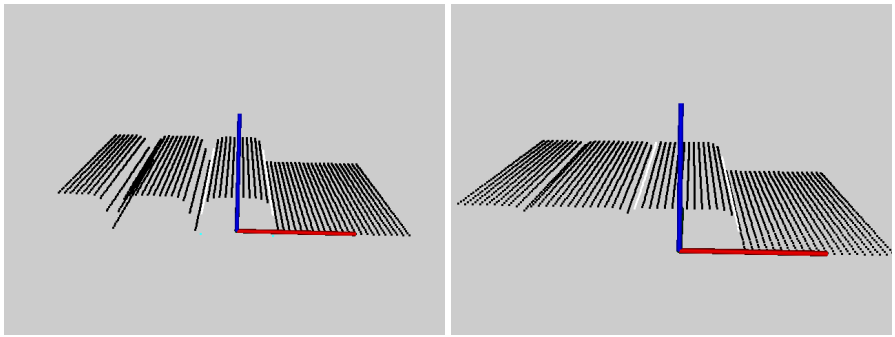


Figure 5.2: Wide underfill valley to the left, narrow underfill to the right.

In this thesis we have worked primarily with the Sobel operator which is a first order convolutional kernel. A second order method called the LaPlacian of Gaussian was proposed by [34], this method combines a Gaussian filter with the second derivative, a discrete 3x3 kernel can be approximated by

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5.1}$$

which looks at zero-crossings of the second derivative in a 3x3 neighbourhood. A test with the ideal underfilled point clouds in figure 5.2 using both the Sobel and LoG-kernel was conducted. A cross section of the resulting first-order and second-order values along the x-axis is plotted in figure 5.3 and figure 5.4. We can see that for a narrow gap both the Sobel and the LoG value are small in the valley of interest while for a wide gap the gradient is equal to the right hand side and the LoG value is quite large. A better algorithm for choosing edges might not only look at the largest gradient but also look at the size of the gradient compared to the largest gradient. If the ratio is rather small and we see that there is both a LoG peak and a gradient bottom close by we might choose the point where both of these are located as the edge point.

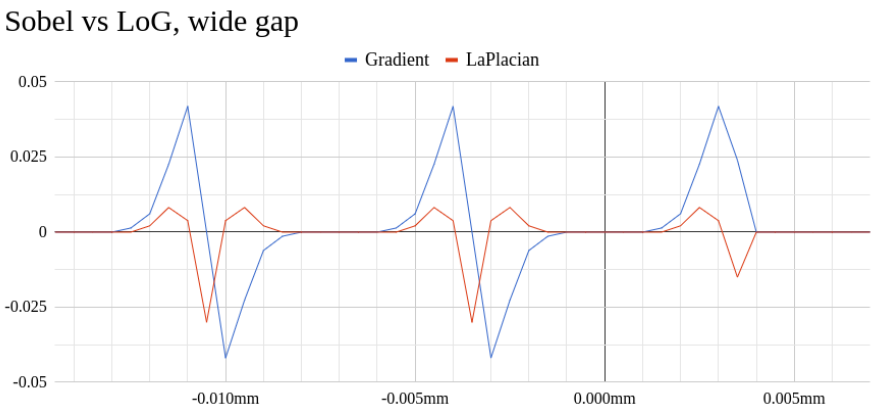


Figure 5.3: The first order Sobel operator versus the LaPlacian of Gaussian operator on the wide underfill valley.

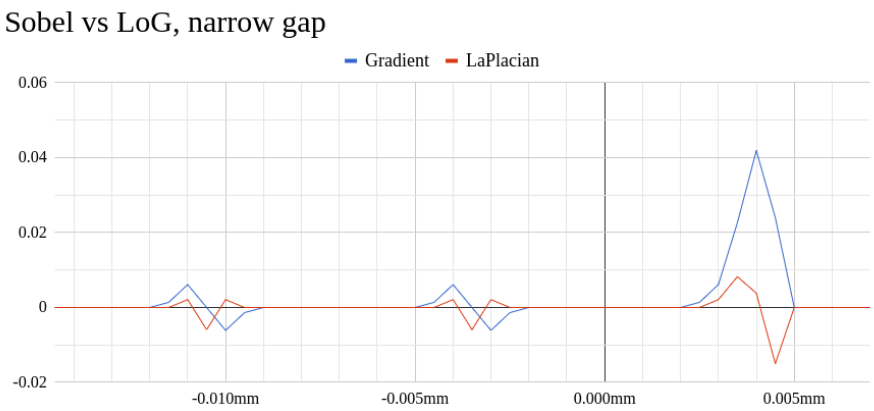


Figure 5.4: The first order Sobel operator versus the LaPlacian of Gaussian operator on the narrow underfill valley.

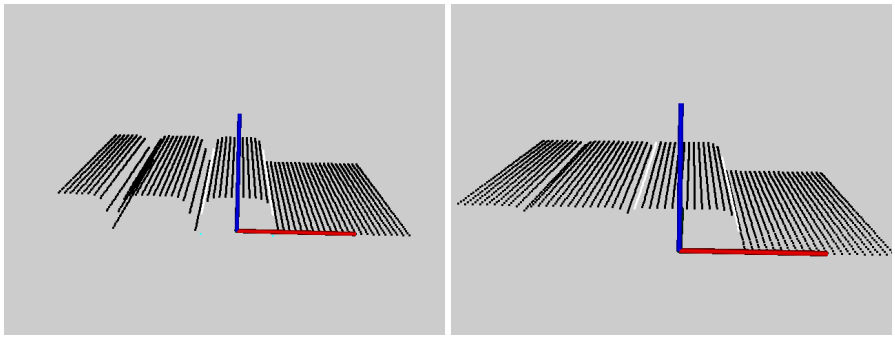


Figure 5.5: Wide underfill valley to the left, narrow underfill to the right.

The same consideration can be done for the overfill case. If we compare figure 5.6 and figure 5.7 with the underfill plots figure 5.3 and figure 5.4. We can see that the gradient increases when we see an overfilled peak and decreases when we reach a valley. This is direction dependant and the described case is only if we move from right to left, the signs are switched if we go from left to right. This observation can also be used to classify over and underfill from each other.

The edge detection algorithms we have looked at only classifies the points as edges, but there are algorithms that are able to extract edges with sub-pixel accuracy such as [35] which uses the LaPlacian of Gaussian convolution. Machine Learning and Neural Networks can also be used for edge detection[36].

5.3 Timing and Efficiency

Even though the point clouds for the experiment were capture first then post-processed later, some of the goal was to get the algorithm to run in real time. The point clouds in the experiment were captured at 10Hz and then saved. While applying the edge-detection algorithm the duration of all the operations were tracked. The average duration of the operations can be found in table 5.1. The major time sink is the

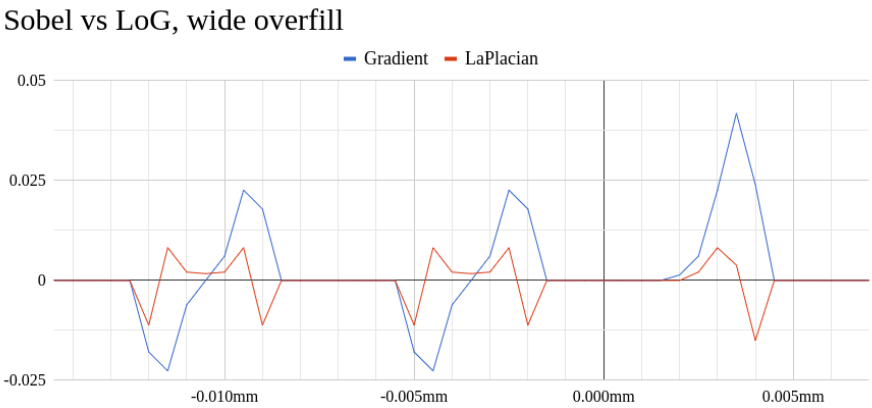


Figure 5.6: The first order Sobel operator versus the LaPlacian of Gaussian operator on the wide overfill peak.

Figure 5.7: The first order Sobel operator versus the LaPlacian of Gaussian operator on the narrow overfill peak.

Table 5.1: Average Duration of the main computations in the Algorithm

Operation	***Time (seconds)***	***Percent of total %***
Range Filter	0.0079	8.67%
Transform:	0.0058	6.36%
Trimming Filters:	0.0065	7.17%
Gaussian blur:	0.0073	7.97%
Calcultate gradient:	0.0631	69.29%
Find Edges:	0.0003	0.37%
Calculate Height:	0.0002	0.18%
Total	***0.0911***	
Frequency	***11.0***	

calculation of the gradient with 70% of the total time. This is no surprise since this algorithm had to be programmed from scratch and certain choices during programming was less than optimal. With these operations and setup we are able to get a frequency of 11Hz which is slightly faster than the 10Hz picture frequency.

The size of the cloud that the gradient calculation needed to work on was also quite a bit bigger than strictly needed, for visualization purposes. We are really only looking at a 2.5mm x 15mm slice of the cloud for width and height information compared to the 25mm x 30mm seen in all the figures. If we removed everything except this slice we would go from 3000 points to 150 which would reduce the calculation time of the gradient by 95%, which again would reduce the total time from 0.0911 seconds to 0.03 seconds, increasing the frequency up to 35Hz.

5.4 Software

The Point Cloud Library(PCL)[29] and librealsense SDK[32] was the primary software used during this thesis. The PCL-implements many algorithms for handling point clouds and 3d-data. PCL is very powerful, but is also quite complicated if you need to do things that are not already implemented. There are also no tutorials on the more advanced algorithms for manipulation of point clouds and you need to know what the algorithms do before trying to use PCL, since PCL only implements the algorithm.

Because of some interface-problems between librealsense and PCL some of the beneficial structure of the data from the camera was lost in converting from librealsense-data to PCL-data. This complicated the implementation of the edge detection algorithm.

5.5 Alternative Feedback Strategy

In this thesis a direct measurement of the control parameters road-width and height were attempted with partial success. In my project thesis[10] a strategy to do real time 3D-scanning was attempted, but was deemed too complicated. After further thought during this semester i believe for a 6-degree-of-freedom robot arm with, perhaps in the future, a more complicated path-planning-scheme might need the wider overview a 3D-camera is able to provide. This is if the more complicated path-planning moves away from extruding simple lines and layers. 3D-scanning is a complicated field and maybe a partnership with a commercial actor would be advisable.

5.6 Conclusion and Future Work

The literature is quite sparse in regards to practical feedback systems for additive manufacturing. The most direct examples are not in real time, but after each layer

of extrusion either using colour cameras or laser triangulation sensors. A proof of concept for feedback for an additive manufacturing process using a robot arm has been conducted on a simple test geometry. Using the structured light camera, Intel SR-300, we have extracted geometric information and determined the usefulness and quality of this camera. A simple edge-detection algorithm has been programmed and tested with the data from the camera. The algorithm proposed is possible to run in real time with a sampling frequency of 11Hz with the current setup, with up to 35Hz possible with some simple optimisation. The algorithm was able to extract edge information from the data, but with quite a large error. With the quality of the camera and sub-par algorithm we were also not able to extract the error cases under- and overfill. The camera has been deemed to be too inaccurate and unsuited for the particular solution and further work should use a better product.

In conclusion, the accuracy of equipment and efficiency of algorithms are crucial for an effective feedback system. Mechanical and practical considerations must also be considered and the requirements of the positioning of the capturing device may prohibit adoption of this particular feedback system.

5.6.1 Future Work

Based on my findings I would recommend the following topics for future study:

- For edge-detection a smarter algorithm should be tested together with a better capturing device. An algorithm looking at the second derivative or a survey into edge detection with machine learning could be interesting.
- A survey into closed loop control of the actual welding operation should be considered to find connection between quality parameters such as surface roughness, road-width and height, with the control parameters feedrate, temperature and actuator speed. I would recommend using a high precision laser profile sensor such as the ones from Micro-Epsilon[33] and perhaps a non-contact

temperature sensor, since temperature is important in CMT-welding quality[37].

- Customised path-planning algorithms and commercial use cases for the project may highlight new requirements for feedback. Perhaps real-time scanning and registration of point clouds for 3D-reconstruction should be investigated later.

References

[1] G. Vosselman and H.-G. Maas, *Airborne and terrestrial laser scanning*. CRC Press, 2010.

[2] R. Jones, P. Haufe, E. Sells, P. Iravani, V. Olliver, C. Palmer, and A. Bowyer, "Reprap–the replicating rapid prototyper," *Robotica*, vol. 29, no. 1, pp. 177–191, 2011.

[3] L. D. Evjemo, S. Moe, J. T. Gravdahl, O. Roulet-Dubonnet, L. T. Gellein, and V. Brøtan, "Additive manufacturing by robot manipulator: An overview of the state-of-the-art and proof-of-concept results," 2017.

[4] T. Huang, S. Wang, and K. He, "Quality control for fused deposition modeling based additive manufacturing: Current research and future trends," in *Reliability Systems Engineering (ICRSE), 2015 First International Conference on*. IEEE, 2015, pp. 1–6.

[5] C. Liu, D. Roberson, and Z. Kong, "Textural analysis-based online closed-loop quality control for additive manufacturing processes," in *IIE Annual Conference. Proceedings*. Institute of Industrial and Systems Engineers (IISE), 2017, pp. 1127–1132.

[6] Y.-a. Jin, Y. He, G.-h. Xue, and J.-z. Fu, "A parallel-based path generation method for fused deposition modeling," *The International Journal of Advanced Manufacturing Technology*, vol. 77, no. 5-8, pp. 927–937, 2015.

[7] A. Herali, A.-K. Christiansson, and B. Lennartson, "Height control of laser metal-wire deposition based on iterative learning control and 3d scanning," *Optics and lasers in engineering*, vol. 50, no. 9, pp. 1230–1241, 2012.

[8] P. K. Rao, J. P. Liu, D. Roberson, Z. J. Kong, and C. Williams, "Online real-time quality monitoring in additive manufacturing processes using heterogeneous sensors," *Journal of Manufacturing Science and Engineering*, vol. 137, no. 6, p. 061007, 2015.

[9] I. Gibson, D. Rosen, and B. Stucker, *Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*. New York, NY: Springer New York, 2015. [Online]. Available: https://doi.org/10.1007/978-1-4939-2113-3_15

[10] J. Jackwitz, "Visual feedback for large scale additive manufacturing," 2017.

[11] Marlin. (2017, dec) Marlin open source 3d printer firmware: Unified bed leveling. [Online]. Available: http://marlinfw.org/docs/features/unified_bed_leveling.html

[12] S. Se and N. Pears, "Passive 3d imaging," in *3D Imaging, Analysis and Applications*. Springer, 2012, pp. 35–94.

[13] N. Pears, Y. Liu, and P. Bunting, *3D imaging, analysis and applications*. Springer, 2012, vol. 3.

[14] Y. Ishii, "Phase-conjugate distance measurement interferometer with frequency modulation of a laser diode," *Proc.SPIE*, vol. 4459, pp. 4459 – 4459 – 8, 2002. [Online]. Available: http://dx.doi.org/10.1117/12.454008

[15] V. Lidar. (2017, nov) Velodyne lidar product vlp-16. [Online]. Available: http://velodynelidar.com

[16] OpenKinect. (2017, nov) Openkinect, hardware information for the microsoft kinect. [Online]. Available: https://openkinect.org/wiki/Hardware_info

[17] Asus. (2017, nov) Asus xtion pro live. [Online]. Available: https://www.asus.com/3D-Sensor/Xtion_PRO_live/overview/

[18] Intel. (2017, nov) Intel sr300 specifications. [Online]. Available: https://ark.intel.com/products/92329/Intel-RealSense-Camera-SR300

[19] S. Foix, G. Alenya, and C. Torras, "Lock-in time-of-flight (tof) cameras: A survey," *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917–1926, 2011.

[20] M. Hansard, S. Lee, O. Choi, and R. Horaud, *Time-of-Flight Cameras Principles, Methods and Applications*. Springer, 2013.

[21] O. Egeland and J. T. Gravdahl, *Modeling and simulation for automatic control*. Marine Cybernetics Trondheim, Norway, 2002, vol. 76.

[22] J. Canny, "A computational approach to edge detection," in *Readings in Computer Vision*. Elsevier, 1987, pp. 184–203.

[23] R. C. Gonzalez, *Digital image processing*. Prentice hall, 2016.

[24] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," in *Readings in computer vision*. Elsevier, 1987, pp. 726–740.

[25] G. K. Tam, Z.-Q. Cheng, Y.-K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X.-F. Sun, and P. L. Rosin, "Registration of 3d point clouds and meshes: a survey from rigid to nonrigid," *IEEE transactions on visualization and computer graphics*, vol. 19, no. 7, pp. 1199–1217, 2013.

[26] P. J. Besl, N. D. McKay *et al.*, "A method for registration of 3-d shapes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[27] MathWorks. (2017, nov) Matlab student r2017b computer vision system toolbox. [Online]. Available: https://se.mathworks.com/help/pdf_doc/vision/vision_ref.pdf

[28] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[29] P. C. Library. (2017, dec) Point cloud library main website. [Online]. Available: http://www.pointclouds.org/about/

[30] M. W. Spong, S. Hutchinson, M. Vidyasagar *et al.*, *Robot modeling and control*. Wiley New York, 2006, vol. 3.

[31] ABB. (2018, May) Irn 140 data sheet. [Online]. Available: http://new.abb.com/products/robotics/industrial-robots/irb-140

[32] IntelRealSense. (2017, nov) Intel realsense open source sdk. [Online]. Available: https://github.com/IntelRealSense/librealsense

[33] Micro-Epsilon. (2018, jun) scancontrol 2d/3d laser scanner (laser profile sensors). [Online]. Available: https://www.micro-epsilon.com/download/products/cat--scanCONTROL--en.pdf

[34] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. R. Soc. Lond. B*, vol. 207, no. 1167, pp. 187–217, 1980.

[35] A. Huertas and G. Medioni, "Detection of intensity changes with subpixel accuracy using laplacian-gaussian masks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, pp. 651–664, 1986.

[36] A. Baştürk and E. Günay, "Efficient edge detection in digital images using a cellular neural network optimized by differential evolution algorithm," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2645–2650, 2009.

[37] B. Cong, R. Ouyang, B. Qi, and J. Ding, "Influence of cold metal transfer process and its heat input on weld bead geometry and porosity of aluminum-copper alloy welds," *Rare Metal Materials and Engineering*, vol. 45, no. 3, pp. 606–611, 2016.

Appendices

Appendix A

Edge Detection Data

Table A.1: Result from Edge Detection: Straight Line, unfiltered and filtered, 0.5mm distance between points. Identical result for Prewit and Sobel gradients and same for both filtered and unfiltered

Y:	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.00250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%
0.00500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%
0.00750	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%
0.01000	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%
0.01250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%
0.01500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%
0.01750	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%
0.02000	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%
0.02250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%
0.02500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%

Table A.2: Result from Edge Detection: Straight Line with incremental offset in x direction, unfiltered and filtered, 0.5mm distance between points. Identical result for Prewitt and Sobel gradients and same for both filtered and unfiltered

X:	*Y:*	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.000000	0.00250	0.00700	0.00700	0.00353	0.00350	-0.00348	-0.00350	0.00300	0.00300	0.00%	0.00%	7.14%
0.000125	0.00500	0.00700	0.00650	0.00365	0.00300	-0.00335	-0.00350	0.00300	0.00300	7.14%	0.00%	7.14%
0.000250	0.00750	0.00700	0.00650	0.00378	0.00300	-0.00323	-0.00350	0.00300	0.00300	7.14%	0.00%	7.14%
0.000375	0.01000	0.00700	0.00650	0.00390	0.00300	-0.00310	-0.00350	0.00300	0.00300	7.14%	0.00%	7.14%
0.000500	0.01250	0.00700	0.00650	0.00403	0.00250	-0.00298	-0.00400	0.00300	0.00300	7.14%	0.00%	7.14%
0.000625	0.01500	0.00700	0.00650	0.00415	0.00250	-0.00285	-0.00400	0.00300	0.00300	7.14%	0.00%	7.14%
0.000750	0.01750	0.00700	0.00650	0.00428	0.00250	-0.00273	-0.00400	0.00300	0.00300	7.14%	0.00%	7.14%
0.000875	0.02000	0.00700	0.00650	0.00440	0.00250	-0.00260	-0.00400	0.00300	0.00300	7.14%	0.00%	7.14%
0.001000	0.02250	0.00700	0.00700	0.00453	0.00250	-0.00248	-0.00450	0.00300	0.00300	0.00%	0.00%	7.14%
0.001125	0.02500	0.00700	0.00650	0.00465	0.00200	-0.00235	-0.00450	0.00300	0.00300	7.14%	0.00%	7.14%

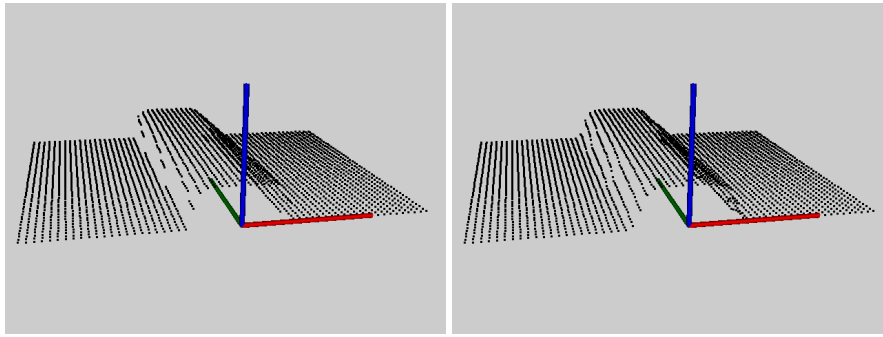


Figure A.1: Incremental offset as a point cloud. Unfiltered to the left, Smoothed out by gaussian kernel to the right.

Table A.3: Result from Edge Detection: Straight Line with incremental increase in width, unfiltered and filtered, 0.5mm distance between points. Identical result for Prewitt and Sobel gradients and same for both filtered and unfiltered

Y:	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.00250	0.00604	0.00600	0.003020	0.003000	-0.003020	-0.003000	0.00300	0.00300	0.66%	0.00%	8.28%
0.00500	0.00625	0.00600	0.003125	0.003000	-0.003125	-0.003000	0.00300	0.00300	4.00%	0.00%	8.00%
0.00750	0.00650	0.00600	0.003250	0.003000	-0.003250	-0.003000	0.00300	0.00300	7.69%	0.00%	7.69%
0.01000	0.00675	0.00600	0.003375	0.003000	-0.003375	-0.003000	0.00300	0.00300	11.11%	0.00%	7.41%
0.01250	0.00700	0.00700	0.003500	0.003500	-0.003500	-0.003500	0.00300	0.00300	0.00%	0.00%	7.14%
0.01500	0.00725	0.00700	0.003625	0.003500	-0.003625	-0.003500	0.00300	0.00300	3.45%	0.00%	6.90%
0.01750	0.00750	0.00700	0.003750	0.003500	-0.003750	-0.003500	0.00300	0.00300	6.67%	0.00%	6.67%
0.02000	0.00775	0.00700	0.003875	0.003500	-0.003875	-0.003500	0.00300	0.00300	9.68%	0.00%	6.45%
0.02250	0.00800	0.00800	0.004000	0.004000	-0.004000	-0.004000	0.00300	0.00300	0.00%	0.00%	6.25%
0.02500	0.00825	0.00800	0.004125	0.004000	-0.004125	-0.004000	0.00300	0.00300	3.03%	0.00%	6.06%

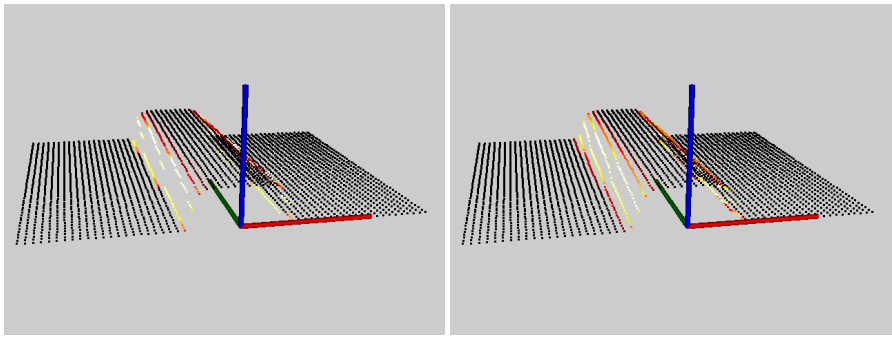


Figure A.2: Calculated gradient with Sobel operator. Unfiltered to the left, Smoothed out by gaussian kernel to the right.

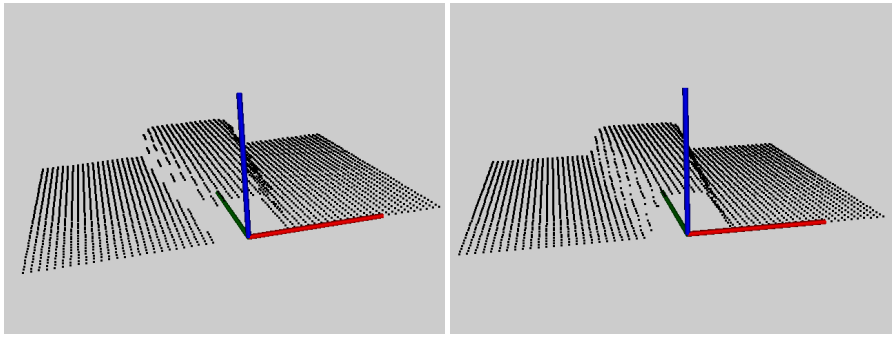


Figure A.3: Incremental width as a point cloud. Unfiltered to the left, Smoothed out by gaussian kernel to the right.

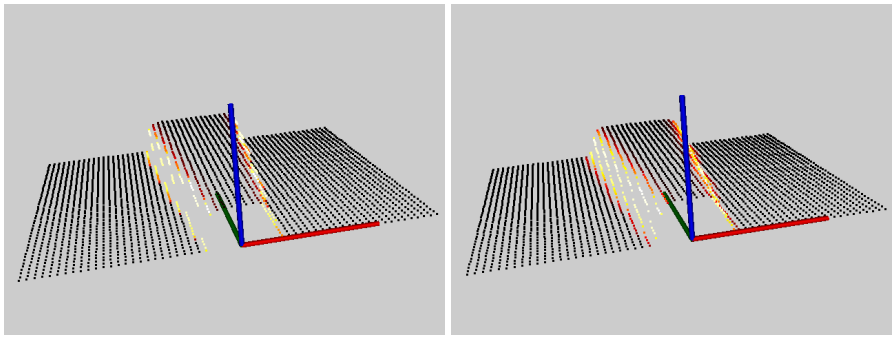


Figure A.4: Calculated gradient with Sobel operator. Unfiltered to the left, Smoothed out by gaussian kernel to the right.

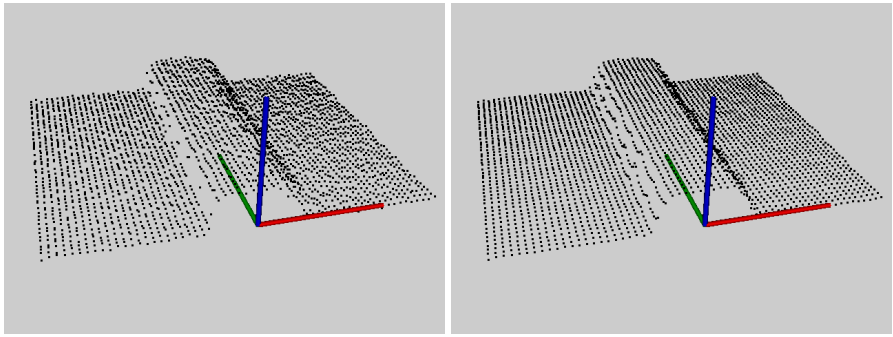


Figure A.5: Straight line with noisy data. Unfiltered to the left, Smoothed out by gaussian kernel to the right.

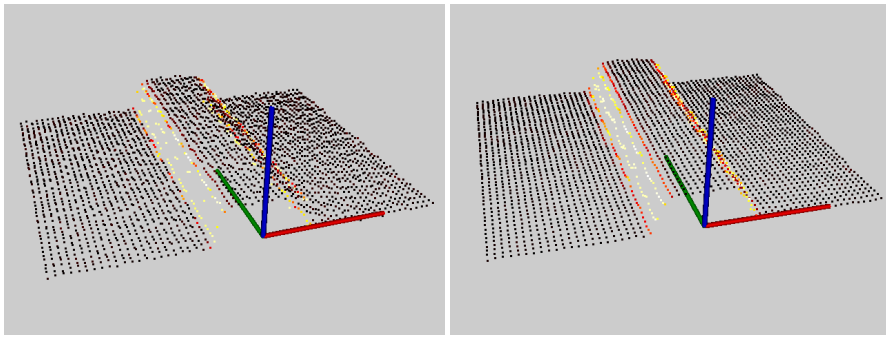


Figure A.6: Straight line with noisy data. Calculated gradient with Sobel operator. Unfiltered to the left, Smoothed out by gaussian kernel to the right.

Table A.4: Result from Edge Detection: Straight line using the Sobel Gradient. Unfiltered first then filtered, 0.5mm distance between points with noisy data.

Y:	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.00250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00298	0.00%	0.63%	7.14%
0.00500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00297	0.00%	0.84%	7.14%
0.00750	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00297	0.00%	0.98%	7.14%
0.01000	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00301	0.00%	-0.37%	7.14%
0.01250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00303	0.00%	-0.94%	7.14%
0.01500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00301	0.00%	-0.40%	7.14%
0.01750	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00305	0.00%	-1.69%	7.14%
0.02000	0.00700	0.00650	0.00350	0.00350	-0.00350	-0.00300	0.00300	0.00300	7.14%	-0.07%	7.14%
0.02250	0.00700	0.00650	0.00350	0.00300	-0.00350	-0.00350	0.00300	0.00304	7.14%	-1.29%	7.14%
0.02500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00298	0.00%	0.81%	7.14%
0.00250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00297	0.00%	1.06%	7.14%
0.00500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00299	0.00%	0.49%	7.14%
0.00750	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00298	0.00%	0.64%	7.14%
0.01000	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	-0.05%	7.14%
0.01250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00302	0.00%	-0.73%	7.14%
0.01500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00302	0.00%	-0.68%	7.14%
0.01750	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00305	0.00%	-1.72%	7.14%
0.02000	0.00700	0.00650	0.00350	0.00350	-0.00350	-0.00300	0.00300	0.00301	7.14%	-0.42%	7.14%
0.02250	0.00700	0.00650	0.00350	0.00300	-0.00350	-0.00350	0.00300	0.00303	7.14%	-1.03%	7.14%
0.02500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00298	0.00%	0.56%	7.14%

Table A.5: Result from Edge Detection: Straight line using the Prewitt Gradient. Unfiltered first then filtered, 0.5mm distance between points with noisy data.

Y:	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.00250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00298	0.00%	0.63%	7.14%
0.00500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00297	0.00%	0.84%	7.14%
0.00750	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00297	0.00%	0.98%	7.14%
0.01000	0.00700	0.00650	0.00350	0.00350	-0.00350	-0.00300	0.00300	0.00301	7.14%	-0.30%	7.14%
0.01250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00303	0.00%	-0.94%	7.14%
0.01500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00301	0.00%	-0.40%	7.14%
0.01750	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00305	0.00%	-1.69%	7.14%
0.02000	0.00700	0.00650	0.00350	0.00350	-0.00350	-0.00300	0.00300	0.00300	7.14%	-0.07%	7.14%
0.02250	0.00700	0.00650	0.00350	0.00300	-0.00350	-0.00350	0.00300	0.00304	7.14%	-1.29%	7.14%
0.02500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00298	0.00%	0.81%	7.14%
0.00250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00297	0.00%	1.06%	7.14%
0.00500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00299	0.00%	0.49%	7.14%
0.00750	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00298	0.00%	0.64%	7.14%
0.01000	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00300	0.00%	-0.05%	7.14%
0.01250	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00302	0.00%	-0.73%	7.14%
0.01500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00302	0.00%	-0.68%	7.14%
0.01750	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00305	0.00%	-1.72%	7.14%
0.02000	0.00700	0.00650	0.00350	0.00350	-0.00350	-0.00300	0.00300	0.00301	7.14%	-0.42%	7.14%
0.02250	0.00700	0.00650	0.00350	0.00300	-0.00350	-0.00350	0.00300	0.00303	7.14%	-1.03%	7.14%
0.02500	0.00700	0.00700	0.00350	0.00350	-0.00350	-0.00350	0.00300	0.00298	0.00%	0.56%	7.14%

Table A.6: Result from Edge Detection: Straight line with incremental offset in x direction using the Sobel Gradient. Unfiltered first then filtered, 0.5mm distance between points with noisy data.

X:	*Y:*	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.000000	0.00250	0.00700	0.00700	0.003525	0.003500	-0.003475	-0.003500	0.00300	0.00296	0.00%	1.21%	7.14%
0.000125	0.00500	0.00700	0.00700	0.003650	0.003500	-0.003350	-0.003500	0.00300	0.00299	0.00%	0.37%	7.14%
0.000250	0.00750	0.00700	0.00650	0.003775	0.003000	-0.003225	-0.003500	0.00300	0.00300	7.14%	0.11%	7.14%
0.000375	0.01000	0.00700	0.00650	0.003900	0.003000	-0.003100	-0.003500	0.00300	0.00299	7.14%	0.36%	7.14%
0.000500	0.01250	0.00700	0.00700	0.004025	0.003000	-0.002975	-0.004000	0.00300	0.00303	0.00%	-0.94%	7.14%
0.000625	0.01500	0.00700	0.00700	0.004150	0.003000	-0.002850	-0.004000	0.00300	0.00300	0.00%	0.06%	7.14%
0.000750	0.01750	0.00700	0.00650	0.004275	0.002500	-0.002725	-0.004000	0.00300	0.00298	7.14%	0.76%	7.14%
0.000875	0.02000	0.00700	0.00650	0.004400	0.002500	-0.002600	-0.004000	0.00300	0.00299	7.14%	0.33%	7.14%
0.001000	0.02250	0.00700	0.00700	0.004525	0.002500	-0.002475	-0.004500	0.00300	0.00298	0.00%	0.55%	7.14%
0.001125	0.02500	0.00700	0.00650	0.004650	0.002000	-0.002350	-0.004500	0.00300	0.00299	7.14%	0.48%	7.14%
0.000000	0.00250	0.00700	0.00700	0.003525	0.003500	-0.003475	-0.003500	0.00300	0.00296	0.00%	1.49%	7.14%
0.000125	0.00500	0.00700	0.00700	0.003650	0.003500	-0.003350	-0.003500	0.00300	0.00299	0.00%	0.38%	7.14%
0.000250	0.00750	0.00700	0.00650	0.003775	0.003000	-0.003225	-0.003500	0.00300	0.00300	7.14%	-0.14%	7.14%
0.000375	0.01000	0.00700	0.00650	0.003900	0.003000	-0.003100	-0.003500	0.00300	0.00299	7.14%	0.22%	7.14%
0.000500	0.01250	0.00700	0.00700	0.004025	0.003000	-0.002975	-0.004000	0.00300	0.00302	0.00%	-0.72%	7.14%
0.000625	0.01500	0.00700	0.00700	0.004150	0.003000	-0.002850	-0.004000	0.00300	0.00300	0.00%	-0.06%	7.14%
0.000750	0.01750	0.00700	0.00650	0.004275	0.002500	-0.002725	-0.004000	0.00300	0.00297	7.14%	0.97%	7.14%
0.000875	0.02000	0.00700	0.00650	0.004400	0.002500	-0.002600	-0.004000	0.00300	0.00301	7.14%	-0.17%	7.14%
0.001000	0.02250	0.00700	0.00700	0.004525	0.002500	-0.002475	-0.004500	0.00300	0.00299	0.00%	0.34%	7.14%
0.001125	0.02500	0.00700	0.00650	0.004650	0.002000	-0.002350	-0.004500	0.00300	0.00297	7.14%	0.90%	7.14%

Table A.7: Result from Edge Detection: Straight line with incremental offset in x direction using the Prewitt Gradient. Unfiltered first then filtered, 0.5mm distance between points with noisy data.

X:	*Y:*	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.000000	0.00250	0.00700	0.00700	0.003525	0.003500	-0.003475	-0.003500	0.00300	0.00296	0.00%	1.21%	7.14%
0.000125	0.00500	0.00700	0.00700	0.003650	0.003500	-0.003350	-0.003500	0.00300	0.00299	0.00%	0.37%	7.14%
0.000250	0.00750	0.00700	0.00650	0.003775	0.003000	-0.003225	-0.003500	0.00300	0.00300	7.14%	0.11%	7.14%
0.000375	0.01000	0.00700	0.00650	0.003900	0.003000	-0.003100	-0.003500	0.00300	0.00299	7.14%	0.36%	7.14%
0.000500	0.01250	0.00700	0.00700	0.004025	0.003000	-0.002975	-0.004000	0.00300	0.00303	0.00%	-0.94%	7.14%
0.000625	0.01500	0.00700	0.00700	0.004150	0.003000	-0.002850	-0.004000	0.00300	0.00300	0.00%	0.06%	7.14%
0.000750	0.01750	0.00700	0.00650	0.004275	0.002500	-0.002725	-0.004000	0.00300	0.00298	7.14%	0.76%	7.14%
0.000875	0.02000	0.00700	0.00650	0.004400	0.002500	-0.002600	-0.004000	0.00300	0.00299	7.14%	0.33%	7.14%
0.001000	0.02250	0.00700	0.00700	0.004525	0.002500	-0.002475	-0.004500	0.00300	0.00298	0.00%	0.55%	7.14%
0.001125	0.02500	0.00700	0.00650	0.004650	0.002000	-0.002350	-0.004500	0.00300	0.00299	7.14%	0.48%	7.14%
0.000000	0.00250	0.00700	0.00700	0.003525	0.003500	-0.003475	-0.003500	0.00300	0.00296	0.00%	1.49%	7.14%
0.000125	0.00500	0.00700	0.00700	0.003650	0.003500	-0.003350	-0.003500	0.00300	0.00299	0.00%	0.38%	7.14%
0.000250	0.00750	0.00700	0.00650	0.003775	0.003000	-0.003225	-0.003500	0.00300	0.00300	7.14%	-0.14%	7.14%
0.000375	0.01000	0.00700	0.00650	0.003900	0.003000	-0.003100	-0.003500	0.00300	0.00299	7.14%	0.22%	7.14%
0.000500	0.01250	0.00700	0.00700	0.004025	0.003000	-0.002975	-0.004000	0.00300	0.00302	0.00%	-0.72%	7.14%
0.000625	0.01500	0.00700	0.00700	0.004150	0.003000	-0.002850	-0.004000	0.00300	0.00300	0.00%	-0.06%	7.14%
0.000750	0.01750	0.00700	0.00650	0.004275	0.002500	-0.002725	-0.004000	0.00300	0.00297	7.14%	0.97%	7.14%
0.000875	0.02000	0.00700	0.00650	0.004400	0.002500	-0.002600	-0.004000	0.00300	0.00301	7.14%	-0.17%	7.14%
0.001000	0.02250	0.00700	0.00700	0.004525	0.002500	-0.002475	-0.004500	0.00300	0.00299	0.00%	0.34%	7.14%
0.001125	0.02500	0.00700	0.00650	0.004650	0.002000	-0.002350	-0.004500	0.00300	0.00297	7.14%	0.90%	7.14%

Table A.8: Result from Edge Detection: Straight line with incremental increase in width using the Sobel Gradient. Unfiltered first then filtered, 0.5mm distance between points with noisy data.

Y:	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.00250	0.00600	0.00600	0.003020	0.003000	-0.003020	-0.003000	0.00300	0.00298	0.00%	0.61%	8.33%
0.00500	0.00625	0.00600	0.003125	0.003000	-0.003125	-0.003000	0.00300	0.00298	4.00%	0.73%	8.00%
0.00750	0.00650	0.00600	0.003250	0.003000	-0.003250	-0.003000	0.00300	0.00298	7.69%	0.71%	7.69%
0.01000	0.00675	0.00700	0.003375	0.003500	-0.003375	-0.003500	0.00300	0.00303	-3.70%	-1.12%	7.41%
0.01250	0.00700	0.00700	0.003500	0.003500	-0.003500	-0.003500	0.00300	0.00296	0.00%	1.20%	7.14%
0.01500	0.00725	0.00700	0.003625	0.003500	-0.003625	-0.003500	0.00300	0.00295	3.45%	1.70%	6.90%
0.01750	0.00750	0.00700	0.003750	0.003500	-0.003750	-0.003500	0.00300	0.00298	6.67%	0.67%	6.67%
0.02000	0.00775	0.00750	0.003875	0.003500	-0.003875	-0.004000	0.00300	0.00302	3.23%	-0.66%	6.45%
0.02250	0.00800	0.00800	0.004000	0.004000	-0.004000	-0.004000	0.00300	0.00298	0.00%	0.70%	6.25%
0.02500	0.00825	0.00800	0.004125	0.004000	-0.004125	-0.004000	0.00300	0.00303	3.03%	-1.12%	6.06%
0.00250	0.00600	0.00600	0.003020	0.003000	-0.003020	-0.003000	0.00300	0.00298	0.00%	0.63%	8.33%
0.00500	0.00625	0.00600	0.003125	0.003000	-0.003125	-0.003000	0.00300	0.00298	4.00%	0.68%	8.00%
0.00750	0.00650	0.00600	0.003250	0.003000	-0.003250	-0.003000	0.00300	0.00298	7.69%	0.70%	7.69%
0.01000	0.00675	0.00650	0.003375	0.003000	-0.003375	-0.003500	0.00300	0.00302	3.70%	-0.76%	7.41%
0.01250	0.00700	0.00700	0.003500	0.003500	-0.003500	-0.003500	0.00300	0.00297	0.00%	1.12%	7.14%
0.01500	0.00725	0.00700	0.003625	0.003500	-0.003625	-0.003500	0.00300	0.00296	3.45%	1.38%	6.90%
0.01750	0.00750	0.00700	0.003750	0.003500	-0.003750	-0.003500	0.00300	0.00297	6.67%	0.90%	6.67%
0.02000	0.00775	0.00700	0.003875	0.003500	-0.003875	-0.003500	0.00300	0.00302	9.68%	-0.76%	6.45%
0.02250	0.00800	0.00800	0.004000	0.004000	-0.004000	-0.004000	0.00300	0.00298	0.00%	0.65%	6.25%
0.02500	0.00825	0.00800	0.004125	0.004000	-0.004125	-0.004000	0.00300	0.00303	3.03%	-0.86%	6.06%

Table A.9: Result from Edge Detection: Straight line with incremental increase in width using the Prewitt Gradient. Unfiltered first then filtered, 0.5mm distance between points with noisy data.

Y:	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.00250	0.00600	0.00600	0.003020	0.003000	-0.003020	-0.003000	0.00300	0.00298	0.00%	0.61%	8.33%
0.00500	0.00625	0.00600	0.003125	0.003000	-0.003125	-0.003000	0.00300	0.00298	4.00%	0.73%	8.00%
0.00750	0.00650	0.00600	0.003250	0.003000	-0.003250	-0.003000	0.00300	0.00298	7.69%	0.71%	7.69%
0.01000	0.00675	0.00700	0.003375	0.003500	-0.003375	-0.003500	0.00300	0.00303	-3.70%	-1.12%	7.41%
0.01250	0.00700	0.00700	0.003500	0.003500	-0.003500	-0.003500	0.00300	0.00296	0.00%	1.20%	7.14%
0.01500	0.00725	0.00700	0.003625	0.003500	-0.003625	-0.003500	0.00300	0.00295	3.45%	1.70%	6.90%
0.01750	0.00750	0.00700	0.003750	0.003500	-0.003750	-0.003500	0.00300	0.00298	6.67%	0.67%	6.67%
0.02000	0.00775	0.00750	0.003875	0.003500	-0.003875	-0.004000	0.00300	0.00302	3.23%	-0.66%	6.45%
0.02250	0.00800	0.00800	0.004000	0.004000	-0.004000	-0.004000	0.00300	0.00298	0.00%	0.70%	6.25%
0.02500	0.00825	0.00800	0.004125	0.004000	-0.004125	-0.004000	0.00300	0.00303	3.03%	-1.12%	6.06%
0.00250	0.00600	0.00600	0.003020	0.003000	-0.003020	-0.003000	0.00300	0.00298	0.00%	0.63%	8.33%
0.00500	0.00625	0.00600	0.003125	0.003000	-0.003125	-0.003000	0.00300	0.00298	4.00%	0.68%	8.00%
0.00750	0.00650	0.00600	0.003250	0.003000	-0.003250	-0.003000	0.00300	0.00298	7.69%	0.70%	7.69%
0.01000	0.00675	0.00650	0.003375	0.003000	-0.003375	-0.003500	0.00300	0.00302	3.70%	-0.76%	7.41%
0.01250	0.00700	0.00700	0.003500	0.003500	-0.003500	-0.003500	0.00300	0.00297	0.00%	1.12%	7.14%
0.01500	0.00725	0.00700	0.003625	0.003500	-0.003625	-0.003500	0.00300	0.00296	3.45%	1.38%	6.90%
0.01750	0.00750	0.00700	0.003750	0.003500	-0.003750	-0.003500	0.00300	0.00297	6.67%	0.90%	6.67%
0.02000	0.00775	0.00700	0.003875	0.003500	-0.003875	-0.003500	0.00300	0.00302	9.68%	-0.76%	6.45%
0.02250	0.00800	0.00800	0.004000	0.004000	-0.004000	-0.004000	0.00300	0.00298	0.00%	0.65%	6.25%
0.02500	0.00825	0.00800	0.004125	0.004000	-0.004125	-0.004000	0.00300	0.00303	3.03%	-0.86%	6.06%

Table A.10: Result from Edge Detection: Straight Line with incremental offset in x direction using the Sobel Gradient. Unfiltered first then filtered, 0.25mm distance between points.

X:	*Y:*	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.000000	0.00250	0.00700	0.00700	0.00351	0.00350	-0.00349	-0.00350	0.00300	0.00301	0.00%	-0.34%	3.57%
0.000125	0.00500	0.00700	0.00675	0.00364	0.00325	-0.00336	-0.00350	0.00300	0.00300	3.57%	-0.02%	3.57%
0.000250	0.00750	0.00700	0.00700	0.00376	0.00325	-0.00324	-0.00375	0.00300	0.00299	0.00%	0.26%	3.57%
0.000375	0.01000	0.00700	0.00700	0.00389	0.00325	-0.00311	-0.00375	0.00300	0.00300	0.00%	-0.06%	3.57%
0.000500	0.01250	0.00700	0.00700	0.00401	0.00300	-0.00299	-0.00400	0.00300	0.00301	0.00%	-0.24%	3.57%
0.000625	0.01500	0.00700	0.00675	0.00414	0.00275	-0.00286	-0.00400	0.00300	0.00299	3.57%	0.33%	3.57%
0.000750	0.01750	0.00700	0.00675	0.00426	0.00250	-0.00274	-0.00425	0.00300	0.00301	3.57%	-0.25%	3.57%
0.000875	0.02000	0.00700	0.00725	0.00439	0.00275	-0.00261	-0.00450	0.00300	0.00300	-3.57%	-0.09%	3.57%
0.001000	0.02250	0.00700	0.00700	0.00451	0.00250	-0.00249	-0.00450	0.00300	0.00301	0.00%	-0.35%	3.57%
0.001125	0.02500	0.00700	0.00675	0.00464	0.00225	-0.00236	-0.00450	0.00300	0.00301	3.57%	-0.23%	3.57%
0.00000	0.00250	0.00700	0.00700	0.00351	0.00350	-0.00349	-0.00350	0.00300	0.00301	0.00%	-0.23%	3.57%
0.00013	0.00500	0.00700	0.00675	0.00364	0.00325	-0.00336	-0.00350	0.00300	0.00300	3.57%	0.09%	3.57%
0.00025	0.00750	0.00700	0.00700	0.00376	0.00325	-0.00324	-0.00375	0.00300	0.00299	0.00%	0.33%	3.57%
0.00038	0.01000	0.00700	0.00700	0.00389	0.00325	-0.00311	-0.00375	0.00300	0.00300	0.00%	-0.17%	3.57%
0.00050	0.01250	0.00700	0.00700	0.00401	0.00300	-0.00299	-0.00400	0.00300	0.00300	0.00%	-0.14%	3.57%
0.00063	0.01500	0.00700	0.00675	0.00414	0.00275	-0.00286	-0.00400	0.00300	0.00299	3.57%	0.32%	3.57%
0.00075	0.01750	0.00700	0.00675	0.00426	0.00250	-0.00274	-0.00425	0.00300	0.00301	3.57%	-0.21%	3.57%
0.00088	0.02000	0.00700	0.00700	0.00439	0.00275	-0.00261	-0.00425	0.00300	0.00301	0.00%	-0.23%	3.57%
0.00100	0.02250	0.00700	0.00700	0.00451	0.00250	-0.00249	-0.00450	0.00300	0.00302	0.00%	-0.53%	3.57%
0.00113	0.02500	0.00700	0.00675	0.00464	0.00225	-0.00236	-0.00450	0.00300	0.00300	3.57%	-0.14%	3.57%

Table A.11: Result from Edge Detection: Straight Line with incremental offset in x direction using the Prewitt Gradient. Unfiltered first then filtered, 0.25mm distance between points.

X:	*Y:*	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.00000	0.00250	0.00700	0.00700	0.00351	0.00350	-0.00349	-0.00350	0.00300	0.00301	0.00%	-0.34%	3.57%
0.00013	0.00500	0.00700	0.00675	0.00364	0.00325	-0.00336	-0.00350	0.00300	0.00300	3.57%	-0.02%	3.57%
0.00025	0.00750	0.00700	0.00700	0.00376	0.00325	-0.00324	-0.00375	0.00300	0.00299	0.00%	0.26%	3.57%
0.00038	0.01000	0.00700	0.00700	0.00389	0.00325	-0.00311	-0.00375	0.00300	0.00300	0.00%	-0.06%	3.57%
0.00050	0.01250	0.00700	0.00700	0.00401	0.00300	-0.00299	-0.00400	0.00300	0.00301	0.00%	-0.24%	3.57%
0.00063	0.01500	0.00700	0.00675	0.00414	0.00275	-0.00286	-0.00400	0.00300	0.00299	3.57%	0.33%	3.57%
0.00075	0.01750	0.00700	0.00700	0.00426	0.00275	-0.00274	-0.00425	0.00300	0.00301	0.00%	-0.21%	3.57%
0.00088	0.02000	0.00700	0.00700	0.00439	0.00275	-0.00261	-0.00425	0.00300	0.00301	0.00%	-0.18%	3.57%
0.00100	0.02250	0.00700	0.00700	0.00451	0.00250	-0.00249	-0.00450	0.00300	0.00301	0.00%	-0.35%	3.57%
0.00113	0.02500	0.00700	0.00675	0.00464	0.00225	-0.00236	-0.00450	0.00300	0.00301	3.57%	-0.23%	3.57%
0.00000	0.00250	0.00700	0.00700	0.00351	0.00350	-0.00349	-0.00350	0.00300	0.00301	0.00%	-0.23%	3.57%
0.00013	0.00500	0.00700	0.00675	0.00364	0.00325	-0.00336	-0.00350	0.00300	0.00300	3.57%	0.09%	3.57%
0.00025	0.00750	0.00700	0.00700	0.00376	0.00325	-0.00324	-0.00375	0.00300	0.00299	0.00%	0.33%	3.57%
0.00038	0.01000	0.00700	0.00700	0.00389	0.00325	-0.00311	-0.00375	0.00300	0.00300	0.00%	-0.17%	3.57%
0.00050	0.01250	0.00700	0.00700	0.00401	0.00300	-0.00299	-0.00400	0.00300	0.00300	0.00%	-0.14%	3.57%
0.00063	0.01500	0.00700	0.00675	0.00414	0.00275	-0.00286	-0.00400	0.00300	0.00299	3.57%	0.32%	3.57%
0.00075	0.01750	0.00700	0.00675	0.00426	0.00250	-0.00274	-0.00425	0.00300	0.00301	3.57%	-0.21%	3.57%
0.00088	0.02000	0.00700	0.00700	0.00439	0.00275	-0.00261	-0.00425	0.00300	0.00301	0.00%	-0.23%	3.57%
0.00100	0.02250	0.00700	0.00700	0.00451	0.00250	-0.00249	-0.00450	0.00300	0.00302	0.00%	-0.53%	3.57%
0.00113	0.02500	0.00700	0.00675	0.00464	0.00225	-0.00236	-0.00450	0.00300	0.00300	3.57%	-0.14%	3.57%

Table A.12: Result from Edge Detection: Straight Line with incremental increase in width using the Sobel Gradient. Unfiltered first then filtered, 0.25mm distance between points.

Y:	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.00250	0.00705	0.00700	0.00352	0.00350	-0.00352	-0.00350	0.00300	0.00299	0.64%	0.21%	3.55%
0.00500	0.00750	0.00725	0.00375	0.00375	-0.00375	-0.00350	0.00300	0.00300	3.33%	-0.05%	3.33%
0.00750	0.00800	0.00775	0.00400	0.00375	-0.00400	-0.00400	0.00300	0.00301	3.13%	-0.46%	3.13%
0.01000	0.00850	0.00850	0.00425	0.00425	-0.00425	-0.00425	0.00300	0.00301	0.00%	-0.48%	2.94%
0.01250	0.00900	0.00900	0.00450	0.00450	-0.00450	-0.00450	0.00300	0.00299	0.00%	0.30%	2.78%
0.01500	0.00950	0.00950	0.00475	0.00475	-0.00475	-0.00475	0.00300	0.00299	0.00%	0.38%	2.63%
0.01750	0.01000	0.01000	0.00500	0.00500	-0.00500	-0.00500	0.00300	0.00300	0.00%	0.05%	2.50%
0.02000	0.01050	0.01025	0.00525	0.00525	-0.00525	-0.00500	0.00300	0.00302	2.38%	-0.60%	2.38%
0.02250	0.01100	0.01100	0.00550	0.00550	-0.00550	-0.00550	0.00300	0.00300	0.00%	-0.14%	2.27%
0.02500	0.01150	0.01150	0.00575	0.00575	-0.00575	-0.00575	0.00300	0.00300	0.00%	0.15%	2.17%
0.00250	0.00705	0.00700	0.00352	0.00350	-0.00352	-0.00350	0.00300	0.00299	0.64%	0.17%	3.55%
0.00500	0.00750	0.00725	0.00375	0.00375	-0.00375	-0.00350	0.00300	0.00300	3.33%	-0.04%	3.33%
0.00750	0.00800	0.00775	0.00400	0.00375	-0.00400	-0.00400	0.00300	0.00301	3.13%	-0.37%	3.13%
0.01000	0.00850	0.00850	0.00425	0.00425	-0.00425	-0.00425	0.00300	0.00301	0.00%	-0.39%	2.94%
0.01250	0.00900	0.00900	0.00450	0.00450	-0.00450	-0.00450	0.00300	0.00299	0.00%	0.32%	2.78%
0.01500	0.00950	0.00950	0.00475	0.00475	-0.00475	-0.00475	0.00300	0.00299	0.00%	0.33%	2.63%
0.01750	0.01000	0.01000	0.00500	0.00500	-0.00500	-0.00500	0.00300	0.00300	0.00%	-0.01%	2.50%
0.02000	0.01050	0.01025	0.00525	0.00525	-0.00525	-0.00500	0.00300	0.00302	2.38%	-0.60%	2.38%
0.02250	0.01100	0.01100	0.00550	0.00550	-0.00550	-0.00550	0.00300	0.00300	0.00%	-0.03%	2.27%
0.02500	0.01150	0.01150	0.00575	0.00575	-0.00575	-0.00575	0.00300	0.00300	0.00%	0.10%	2.17%

Table A.13: Result from Edge Detection: Straight Line with incremental increase in width using the Prewitt Gradient. Unfiltered first then filtered, 0.25mm distance between points.

Y:	*Real Width:*	*Est Width:*	*Real P edge:*	*Est P edge:*	*Real N edge:*	*Est N edge:*	*Real Height:*	*Est Height:*	*Width Error %*	*Height Error %*	*Stepsize Error %*
0.00250	0.00705	0.00700	0.00352	0.00350	-0.00352	-0.00350	0.00300	0.00299	0.64%	0.21%	3.55%
0.00500	0.00750	0.00725	0.00375	0.00375	-0.00375	-0.00350	0.00300	0.00300	3.33%	-0.05%	3.33%
0.00750	0.00800	0.00775	0.00400	0.00375	-0.00400	-0.00400	0.00300	0.00301	3.13%	-0.46%	3.13%
0.01000	0.00850	0.00850	0.00425	0.00425	-0.00425	-0.00425	0.00300	0.00301	0.00%	-0.48%	2.94%
0.01250	0.00900	0.00900	0.00450	0.00450	-0.00450	-0.00450	0.00300	0.00299	0.00%	0.30%	2.78%
0.01500	0.00950	0.00950	0.00475	0.00475	-0.00475	-0.00475	0.00300	0.00299	0.00%	0.38%	2.63%
0.01750	0.01000	0.01000	0.00500	0.00500	-0.00500	-0.00500	0.00300	0.00300	0.00%	0.05%	2.50%
0.02000	0.01050	0.01025	0.00525	0.00525	-0.00525	-0.00500	0.00300	0.00302	2.38%	-0.60%	2.38%
0.02250	0.01100	0.01100	0.00550	0.00550	-0.00550	-0.00550	0.00300	0.00300	0.00%	-0.14%	2.27%
0.02500	0.01150	0.01150	0.00575	0.00575	-0.00575	-0.00575	0.00300	0.00300	0.00%	0.15%	2.17%
0.00250	0.00705	0.00700	0.00352	0.00350	-0.00352	-0.00350	0.00300	0.00299	0.64%	0.17%	3.55%
0.00500	0.00750	0.00725	0.00375	0.00375	-0.00375	-0.00350	0.00300	0.00300	3.33%	-0.04%	3.33%
0.00750	0.00800	0.00775	0.00400	0.00375	-0.00400	-0.00400	0.00300	0.00301	3.13%	-0.37%	3.13%
0.01000	0.00850	0.00850	0.00425	0.00425	-0.00425	-0.00425	0.00300	0.00301	0.00%	-0.39%	2.94%
0.01250	0.00900	0.00900	0.00450	0.00450	-0.00450	-0.00450	0.00300	0.00299	0.00%	0.32%	2.78%
0.01500	0.00950	0.00950	0.00475	0.00475	-0.00475	-0.00475	0.00300	0.00299	0.00%	0.33%	2.63%
0.01750	0.01000	0.01000	0.00500	0.00500	-0.00500	-0.00500	0.00300	0.00300	0.00%	-0.01%	2.50%
0.02000	0.01050	0.01025	0.00525	0.00525	-0.00525	-0.00500	0.00300	0.00302	2.38%	-0.60%	2.38%
0.02250	0.01100	0.01100	0.00550	0.00550	-0.00550	-0.00550	0.00300	0.00300	0.00%	-0.03%	2.27%
0.02500	0.01150	0.01150	0.00575	0.00575	-0.00575	-0.00575	0.00300	0.00300	0.00%	0.10%	2.17%

Appendix B

Experiment 1 - Straight Line

Table B.1: Average values for the height measurements of the experiments

Exp:	*Avg Meas.*	*Unf. Avg.*	*Unf. Err.*	*Unf. St.Dev.*	*Gaus. 3x3 Avg.*	*Gaus. 3x3 Err.*	*Gaus. 3x3 St.Dev.*	*Gaus. 5x5 Avg.*	*Gaus. 5x5 Err.*	*Gaus. 5x5 St.Dev.*
e1	1.80	1.72	-0.08	0.14	1.73	-0.07	0.14	1.73	-0.07	0.14
e2	1.90	1.88	-0.02	0.07	1.89	-0.01	0.07	1.89	-0.01	0.06
e3	2.47	2.25	-0.22	0.11	2.25	-0.22	0.11	2.25	-0.22	0.11
e4	2.50	1.99	-0.51	0.12	1.98	-0.52	0.12	1.98	-0.52	0.12
e5	3.19	2.61	-0.58	0.10	2.61	-0.58	0.10	2.63	-0.56	0.10
e6	3.27	2.87	-0.40	0.11	2.87	-0.40	0.11	2.87	-0.40	0.11
e7	1.87	1.45	-0.42	0.09	1.45	-0.42	0.09	1.46	-0.41	0.09
e8	3.93	4.32	0.39	0.06	4.32	0.39	0.06	4.32	0.39	0.07
e9	1.80	1.77	-0.03	0.13	1.77	-0.03	0.12	1.78	-0.02	0.13
e10	3.90	4.12	0.22	0.10	4.12	0.22	0.10	4.11	0.21	0.10
e11	3.10	2.82	-0.28	0.07	2.82	-0.28	0.07	2.82	-0.28	0.08
e12	6.90	6.59	-0.31	0.10	6.57	-0.33	0.08	6.57	-0.33	0.08
e13	3.00	2.77	-0.23	0.08	2.77	-0.23	0.08	2.77	-0.23	0.08
e14	6.45	6.51	0.06	0.07	6.51	0.06	0.07	6.51	0.06	0.07
e15	3.85	3.20	-0.65	0.08	3.20	-0.65	0.08	3.20	-0.65	0.08
e16	7.85	7.62	-0.23	0.12	7.63	-0.22	0.12	7.63	-0.22	0.12

Table B.2: Average values for the width measurements of the experiments

Exp:	*Avg Meas.*	*Unf. Avg.*	*Unf. Err.*	*Unf. St.Dev.*	*Gaus. 3x3 Avg.*	*Gaus. 3x3 Err.*	*Gaus. 3x3 St.Dev.*	*Gaus. 5x5 Avg.*	*Gaus. 5x5 Err.*	*Gaus. 5x5 St.Dev.*
1	7.40	7.18	-0.22	0.94	7.13	-0.27	0.92	7.23	-0.17	0.92
2	8.20	7.70	-0.50	0.70	7.85	-0.35	0.70	7.73	-0.47	0.72
3	6.60	6.90	0.30	0.95	7.06	0.46	0.91	7.11	0.51	0.91
4	7.63	7.18	-0.45	0.55	7.05	-0.58	0.60	6.97	-0.66	0.84
5	7.07	6.83	-0.24	0.47	7.15	0.08	0.68	6.85	-0.22	0.92
6	7.60	7.07	-0.53	0.57	7.05	-0.55	0.60	7.09	-0.51	0.80
7	7.80	8.54	0.74	1.12	8.84	1.04	0.92	8.99	1.19	0.80
8	8.77	8.51	-0.26	0.49	8.55	-0.22	0.51	8.34	-0.43	0.65
9	7.12	8.44	1.33	0.99	8.56	1.45	0.95	8.58	1.47	0.76
10	9.50	8.52	-0.98	0.89	8.63	-0.87	0.63	8.62	-0.88	0.75
11	7.34	8.19	0.86	0.51	8.19	0.86	0.51	8.51	1.18	0.56
12	8.90	8.10	-0.80	0.72	8.28	-0.62	0.35	8.27	-0.63	0.34
13	8.29	8.38	0.10	0.37	8.51	0.23	0.45	8.58	0.30	0.54
14	9.55	7.99	-1.56	0.37	7.94	-1.61	0.34	7.93	-1.62	0.36
15	8.31	8.24	-0.07	0.57	8.33	0.02	0.56	8.26	-0.05	0.64
16	6.25	6.93	0.68	1.04	6.78	0.53	0.99	6.80	0.55	0.89

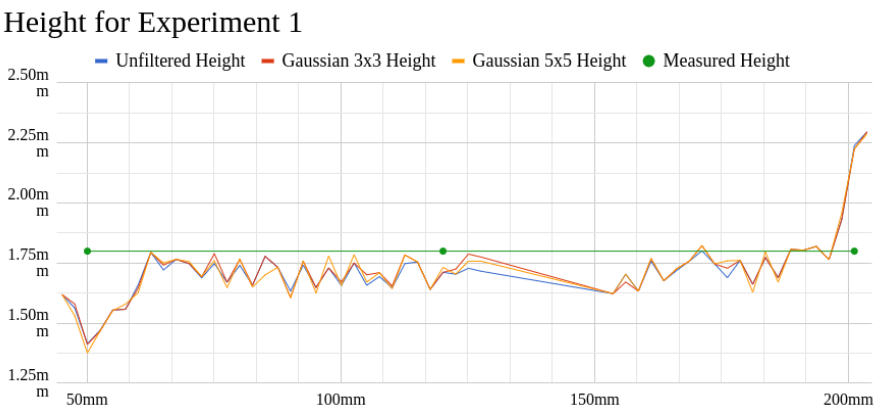


Figure B.1: Experiment 1 height plot using the Sobel gradient.

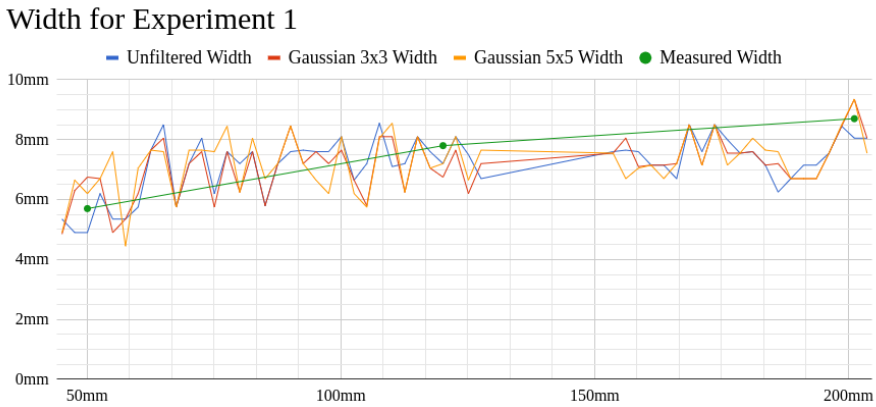


Figure B.2: Experiment 1 width plot using the Sobel gradient.

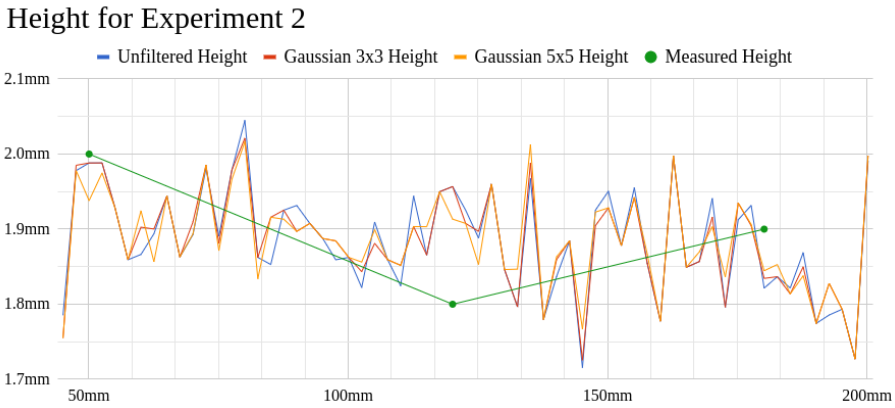


Figure B.3: Experiment 2 height plot using the Sobel gradient.

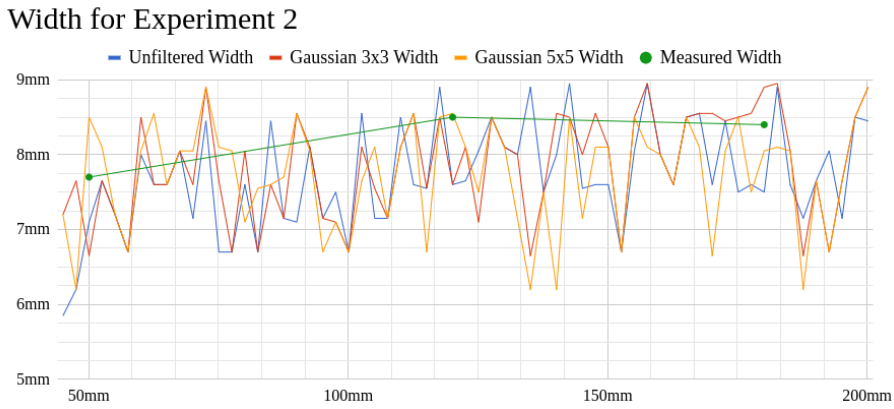


Figure B.4: Experiment 2 width plot using the Sobel gradient.

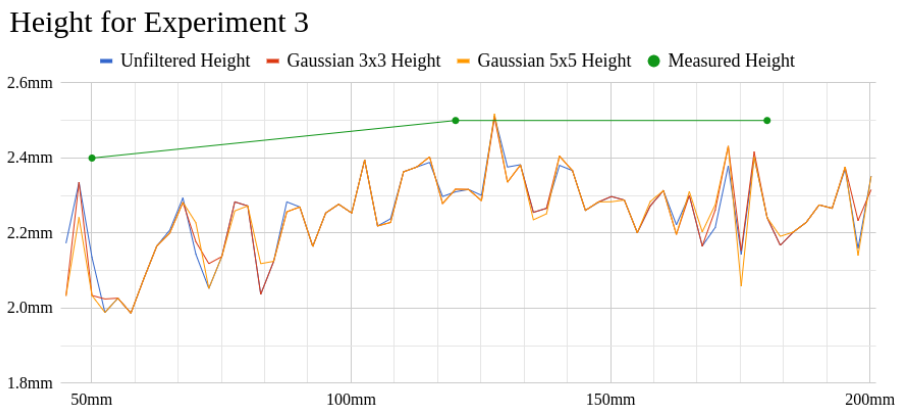


Figure B.5: Experiment 3 height plot using the Sobel gradient.

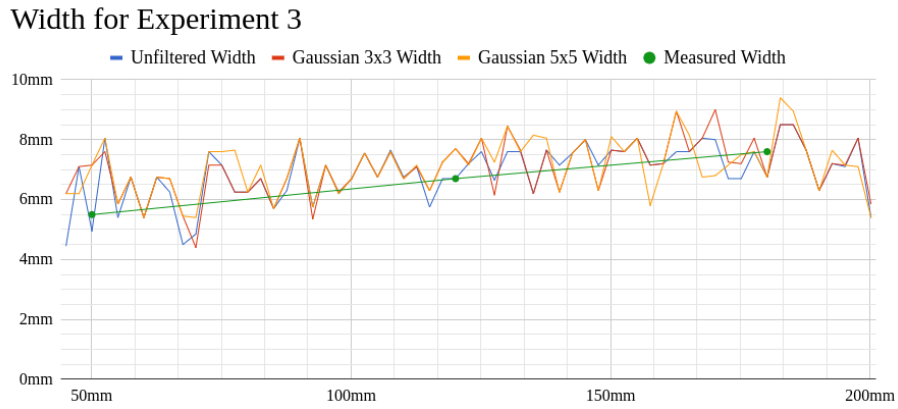


Figure B.6: Experiment 3 width plot using the Sobel gradient.

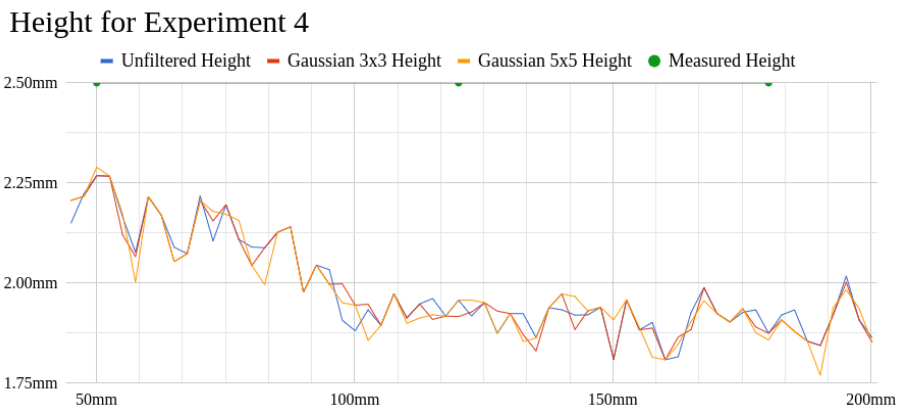


Figure B.7: Experiment 4 height plot using the Sobel gradient.

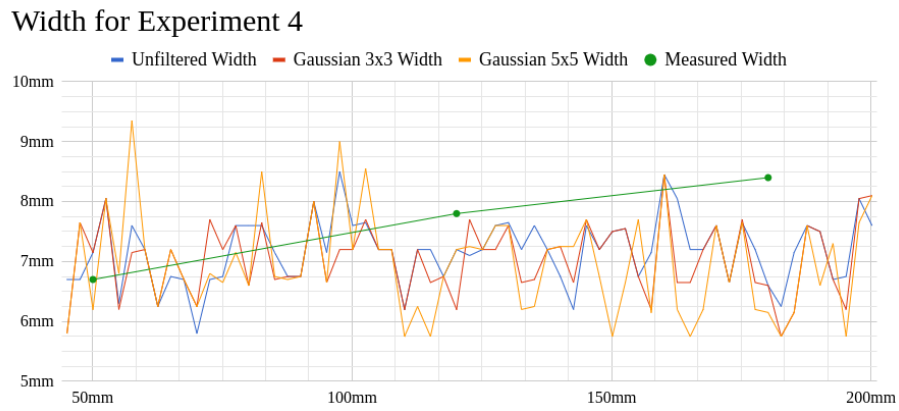


Figure B.8: Experiment 4 width plot using the Sobel gradient.

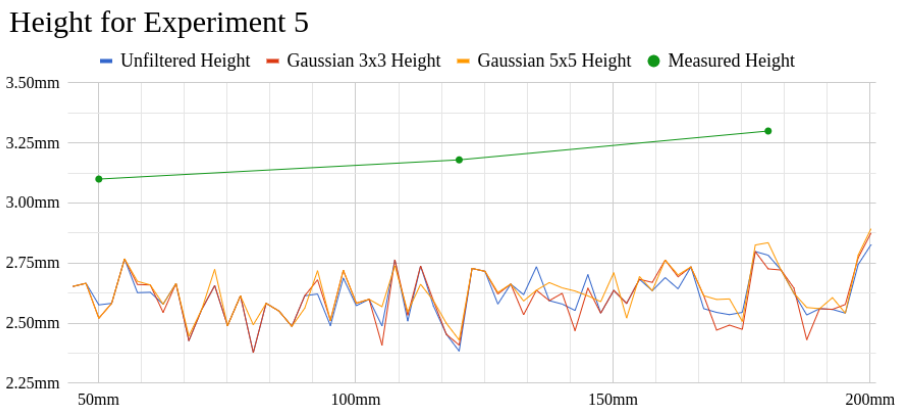


Figure B.9: Experiment 5 height plot using the Sobel gradient.

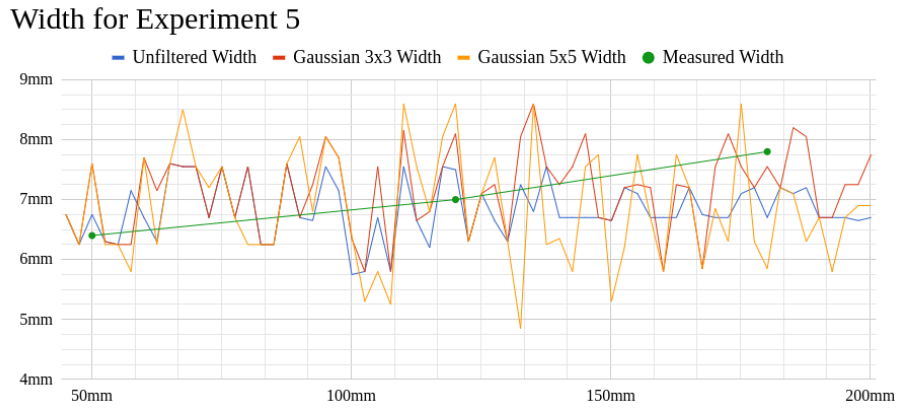


Figure B.10: Experiment 5 width plot using the Sobel gradient.

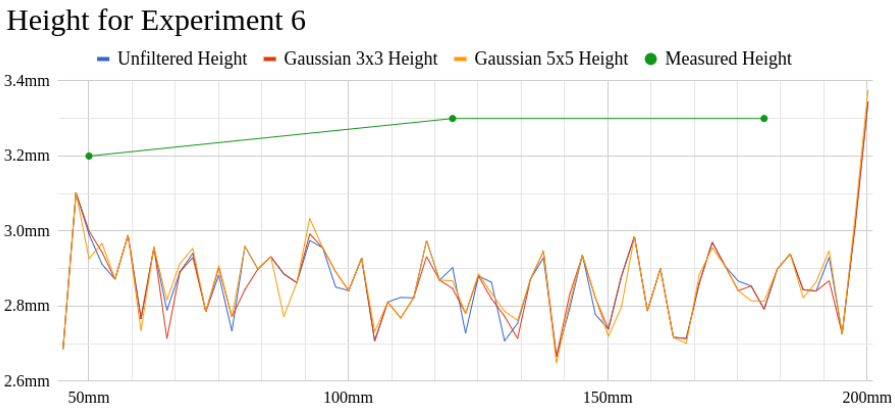


Figure B.11: Experiment 6 height plot using the Sobel gradient.

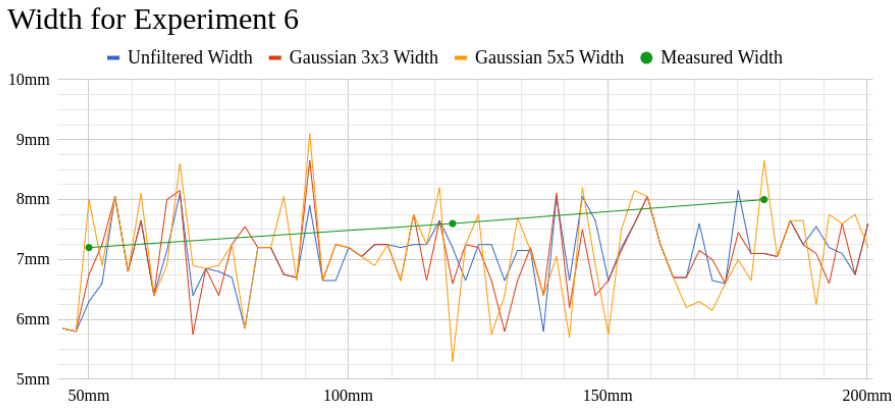


Figure B.12: Experiment 6 width plot using the Sobel gradient.

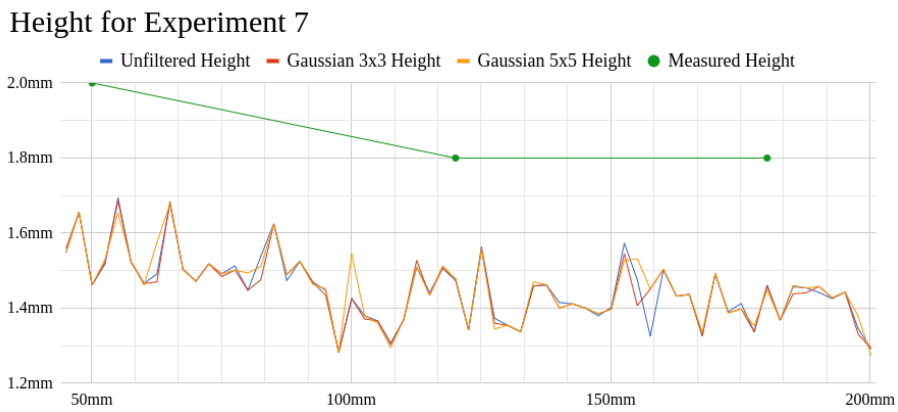


Figure B.13: Experiment 7 height plot using the Sobel gradient.

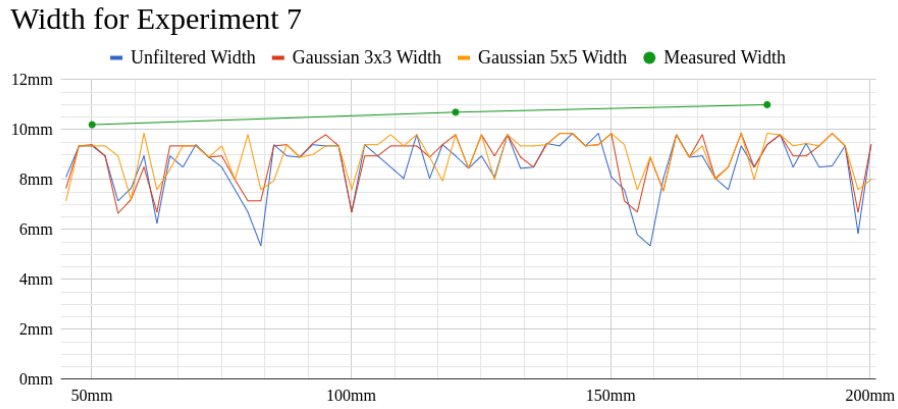


Figure B.14: Experiment 7 width plot using the Sobel gradient.

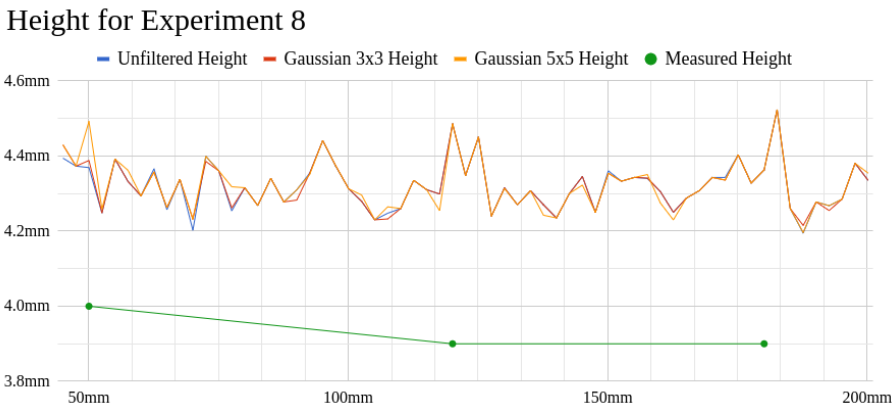


Figure B.15: Experiment 8 height plot using the Sobel gradient.

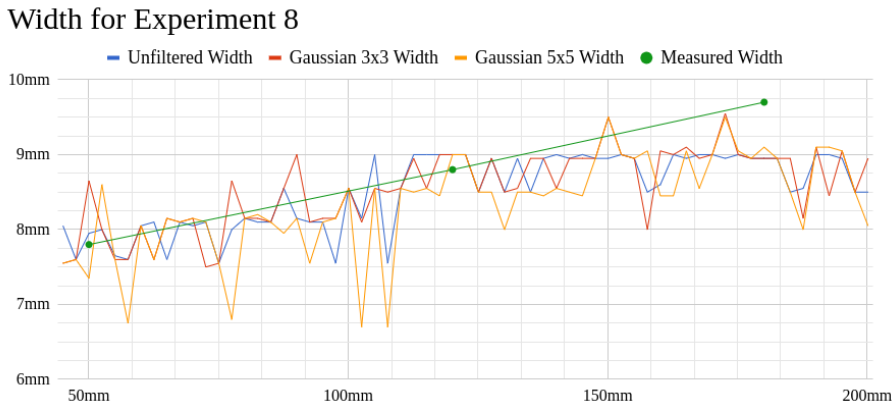


Figure B.16: Experiment 8 width plot using the Sobel gradient.

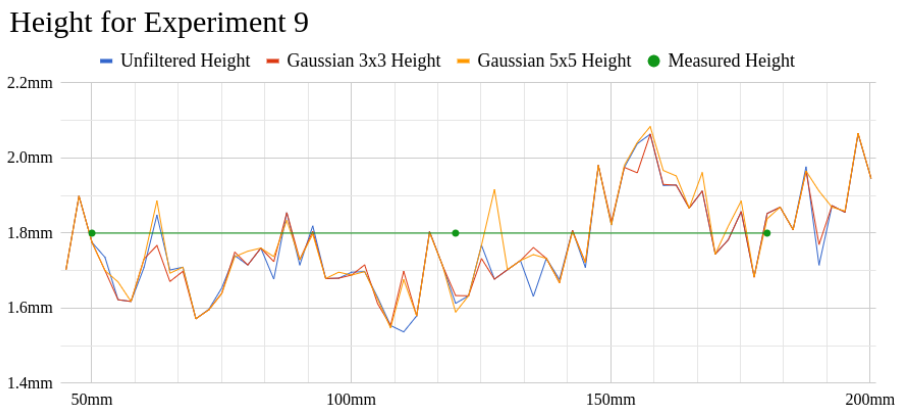


Figure B.17: Experiment 9 height plot using the Sobel gradient.

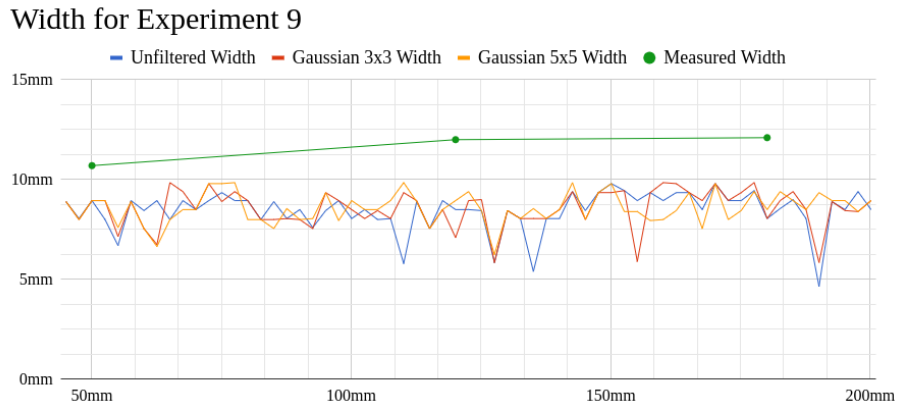


Figure B.18: Experiment 9 width plot using the Sobel gradient.

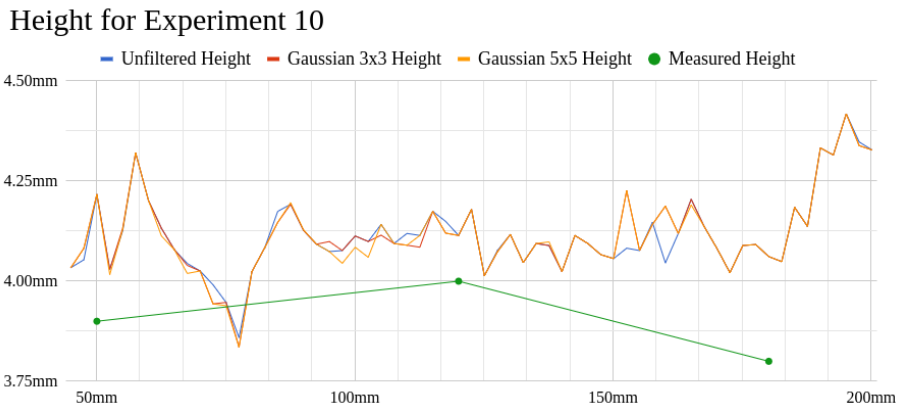


Figure B.19: Experiment 10 height plot using the Sobel gradient.

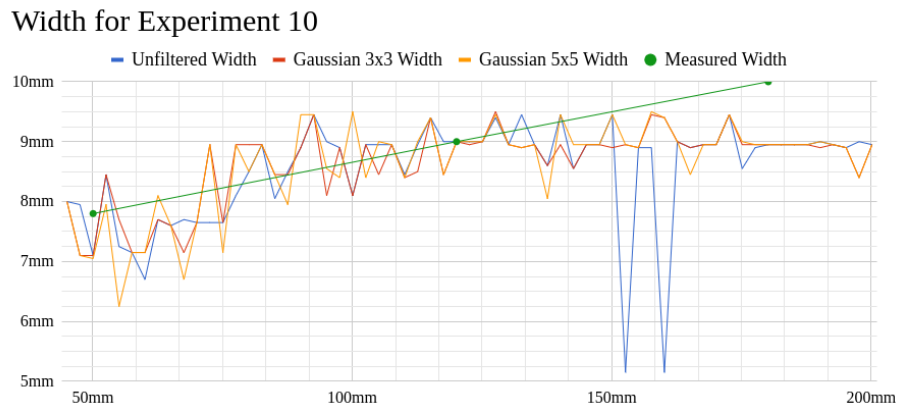


Figure B.20: Experiment 10 width plot using the Sobel gradient.

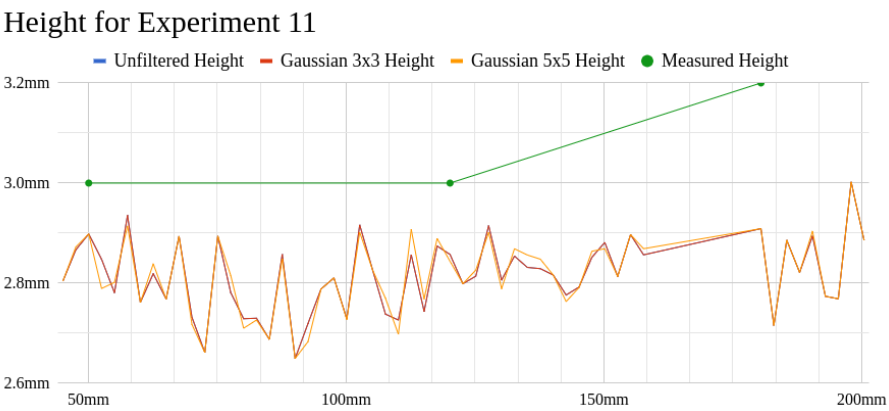


Figure B.21: Experiment 11 height plot using the Sobel gradient.

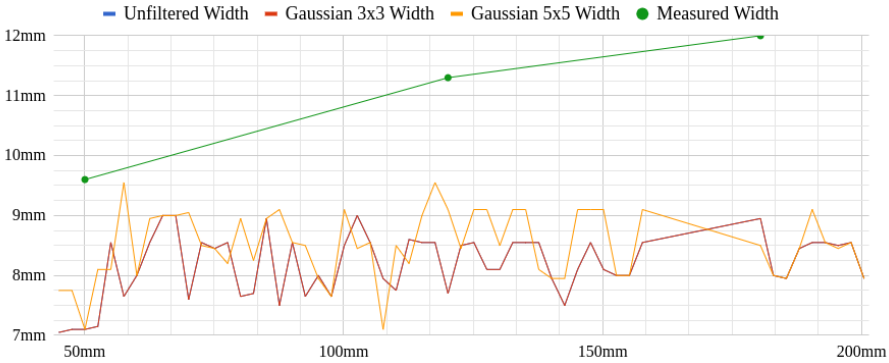


Figure B.22: Experiment 11 width plot using the Sobel gradient.

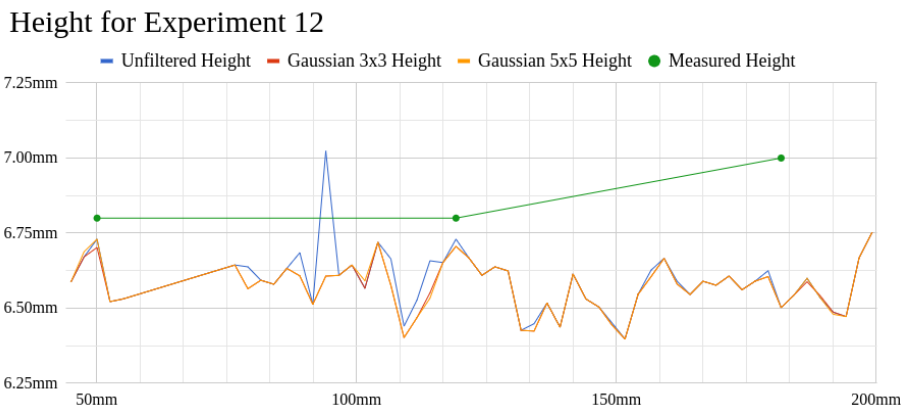


Figure B.23: Experiment 12 height plot using the Sobel gradient.

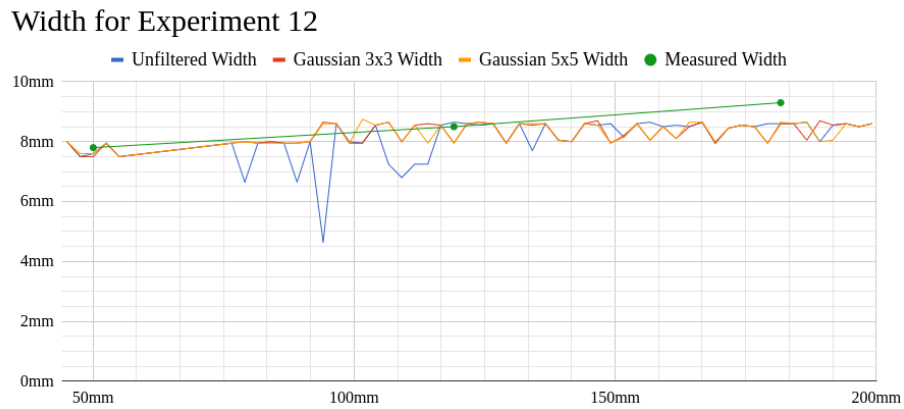


Figure B.24: Experiment 12 width plot using the Sobel gradient.

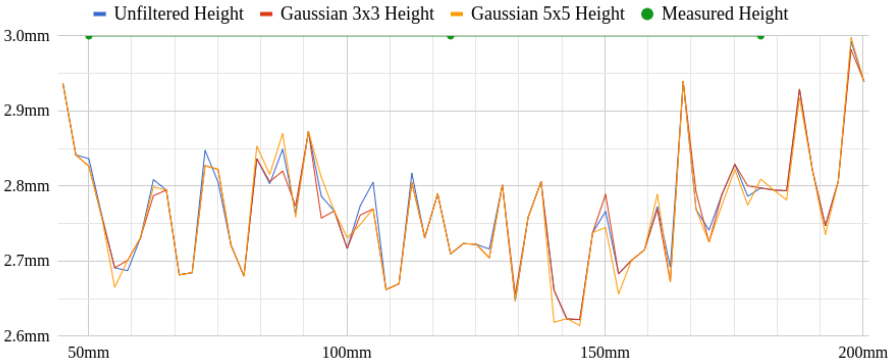


Figure B.25: Experiment 13 height plot using the Sobel gradient.

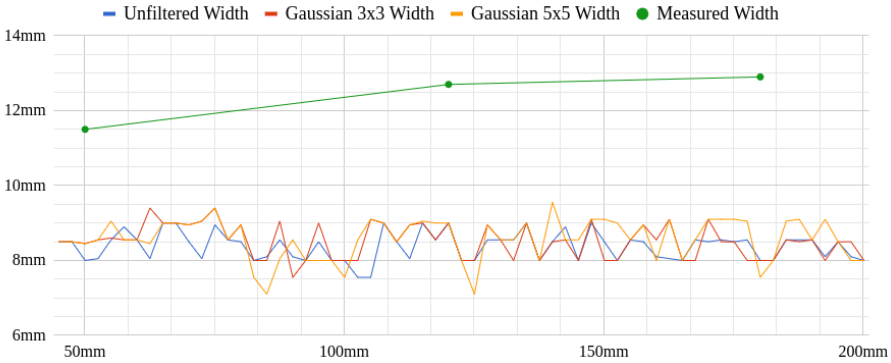


Figure B.26: Experiment 13 width plot using the Sobel gradient.

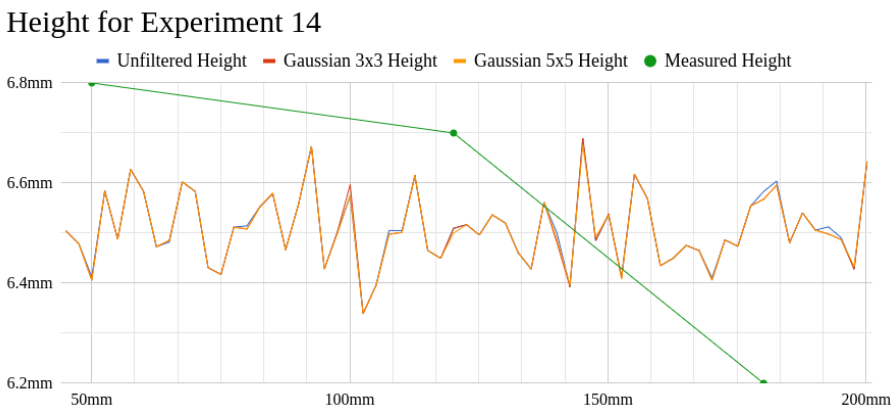


Figure B.27: Experiment 14 height plot using the Sobel gradient.

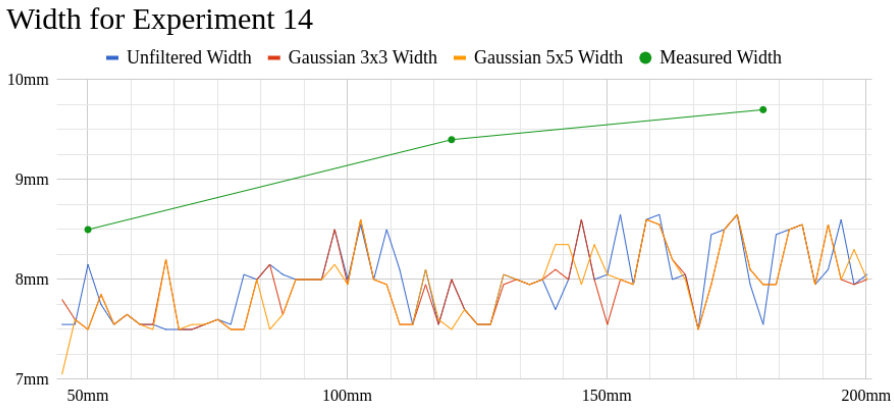


Figure B.28: Experiment 14 width plot using the Sobel gradient.

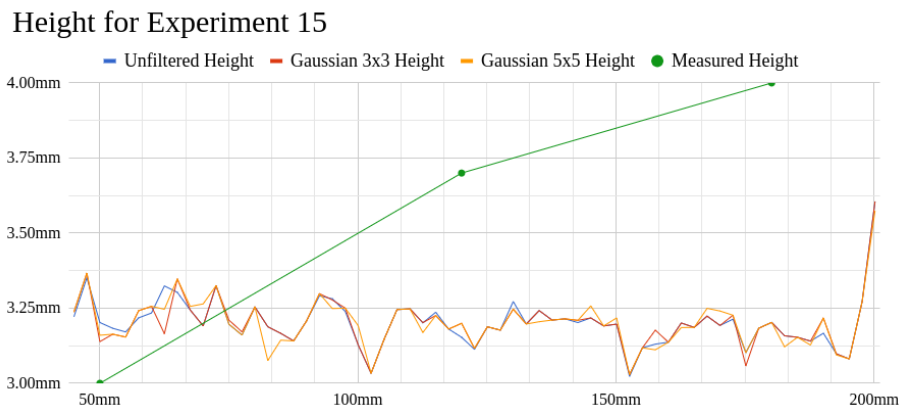


Figure B.29: Experiment 15 height plot using the Sobel gradient.

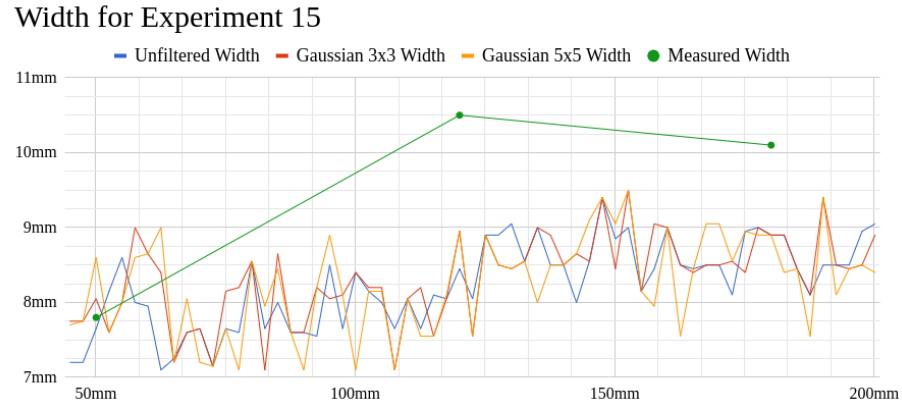


Figure B.30: Experiment 15 width plot using the Sobel gradient.

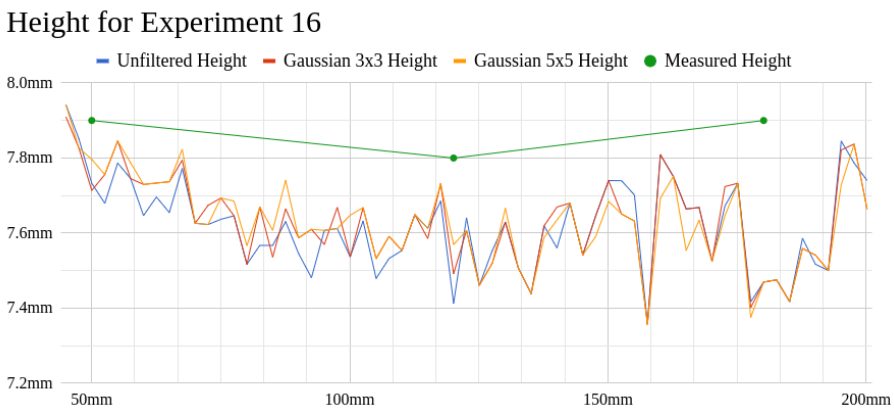


Figure B.31: Experiment 16 height plot using the Sobel gradient.

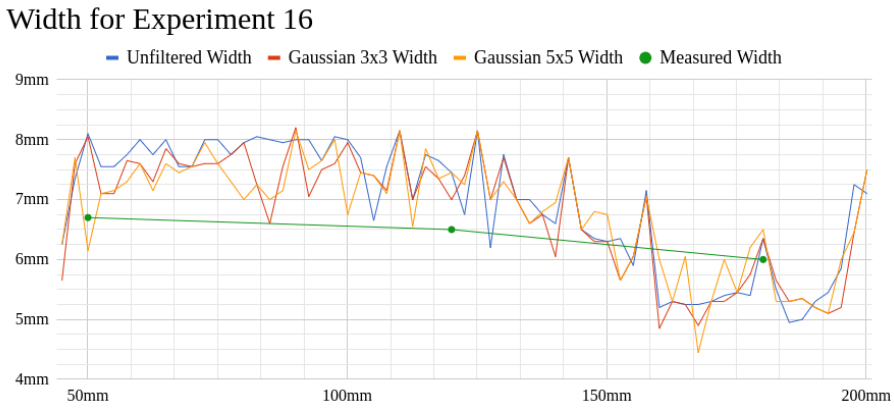


Figure B.32: Experiment 16 width plot using the Sobel gradient.