Using machine learning to optimise Laser Powder Bed Fusion process parameters for dimensional tolerance

By
Thomas Astley
Student ID: 2394936
Supervisor: Dr Leonardo Stella

A thesis submitted to the University of Birmingham
for the degree of MSC Computer Science

School of Computer Science
University of Birmingham
September 2023

ABSTRACT

Machine learning techniques are being utilised in almost every aspect of society. Specifically relevant to this project, it is being used more and more in the optimisation of additive manufacturing parameters. New machines are built with different sets of adjustable parameters and new metal alloys are produced with different properties, resulting in different parameters being required to previous alloys and machines. These need to be optimised before usable components can be printed. In this dissertation, machine learning combined with image analysis is used to obtain predictions for optimum values for two parameters of the laser powder bed fusion process, beam compensation (BC) and contour distance (CD), when A20X powder is used. The main impact of these parameters is on the dimensional tolerance and accuracy of the smaller sections of the resulting product.

Canny edge detection was utilised in order to detect the edges of a section of a test product, and then an average width was determined from this. Then this was used to calculate a percentage error based on the target width for that section. This is then repeated for each section, for each product in the sample, and fed into a gradient decent algorithm, along with the BC and CD values to produce an equation that relates them to each other. Simulated annealing is then used to find the minimum of the equation, in the bounds of the minimum and maximum allowed BC and CD values. The minimum is the optimum values, as it is the lowest percent error.

The method successfully identified a number of candidate solutions which were tested, with preliminary measurements indicating a success of the method. It also identified a correlation between the two variables and the percentage error, specifically that, when compared with the CD, the BC had little impact on the dimensional tolerance for target dimensions greater than 0.2mm.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Dr Leonardo Stella, for his guidance and support throughout the project. Secondly, I would like to thank Francesco Careri for carrying out the experiments and providing the initial data and photographs used for this project, as well as carrying out further experiments based on the results.

Lastly, I would like to thank my friends and family for providing encouragement and motivation throughout the year.

Contents

Introduction

Machine learning is having an ever-increasing impact on our day to day lives. From applications on our phone knowing what we want to buy before we have thought of it, to aiding with the detection of cancers [1] it is making a difference to each of our lives. Most relevant to this dissertation, machine learning has been on the rise in the field of materials science, where it has been, and is being used to optimise many variables for a variety of additive manufacturing processes using an array of different metal alloys [2]. Because of the high costs associated with running many tests, it is of particular use in this field to run fewer tests, using machine learning to analyse the data, and then run more tests based on the predictions obtained.

In this dissertation, there were two main objectives. The first objective was to use machine learning in order to optimise the parameters of beam compensation (BC) and contour distance (CD) of the laser powder bed fusion (LPBF) process. These parameters affect the dimensions of products produced by this process, having the biggest impact on smaller features. To achieve this goal, we used prior knowledge of machine learning to formulate the problem and implemented algorithms to solve it. Real data was obtained beforehand of test products with six beams of different widths, with a known target width for each and known BC and CD values, which were used in the algorithms. These algorithms were then built upon in order to improve efficiency and accuracy, bringing their run time down to less than 1% of the original.

The second objective was to develop a way to analyse images of the test products that were used to obtain the data for the first objective, to obtain new measurements for the widths of the beams. The previous data was obtained by manually measuring each width, which is not very precise and can lead to human error. Prior to undertaking this project, I had little knowledge of computer vision techniques, so a large portion of time was dedicated to gaining an understanding of different useful techniques and deciding upon the best for the objective. Before the research, I had three rough ideas of how to complete the objective, however two turned out not to be viable. A full explanation into this will be detailed in the method section below.

My experience with python was quite limited before undertaking this project. However, due to knowledge in java programming, not much time was needed to become sufficiently proficient in Python. As both languages are object oriented, my knowledge was easily transferable. Due to pythons' popularity and ease of access and use, there was a huge amount of information available if I had any issues.

The dissertation will start by going into detail about the background regarding the project, then reviewing the material relevant to the project, followed by a formal specification of the problem and solution. Next we will go on to an explanation of the design of the solution developed, followed by a reflection of the project as a whole. Finally, we will present and evaluate the results obtained, and draw conclusions, with suggestions for how to build upon them in the future.

Background

2.1 Laser Powder Bed Fusion Process

The laser powder bed fusion (LPBF) process is a metal additive manufacturing system, which themselves are a collection of systems that are each designed to make objects from a computer aided design by joining materials together, layer by layer. [3] For LPBF, a thin layer of powder is applied to the building platform. Then a laser beam is used to selectively melt and fuse the metal powder together in the locations specified by the design. A new layer of powder is then applied and the process is repeated until a product is formed. [4] The particular powder used in the experiments was A205, with is an aluminium based alloy, containing copper and titanium, among other elements in smaller quantities. The machine is also filled with argon gas due to its chemically inert nature. This prevents the powder for oxidising and prevents any other unwanted reactions occurring. A steady stream of argon is blown parallel to the building platform, which prevents any of the smoke produced by the melting process interfering with the laser.

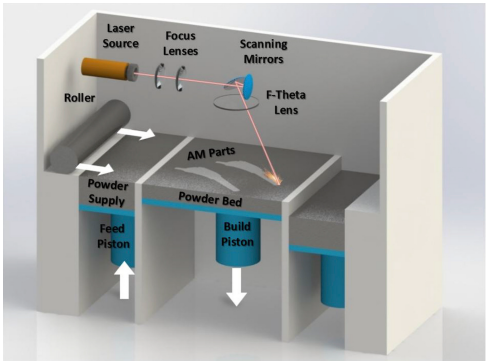

Figure 1: Example diagram of LPBF system [4].

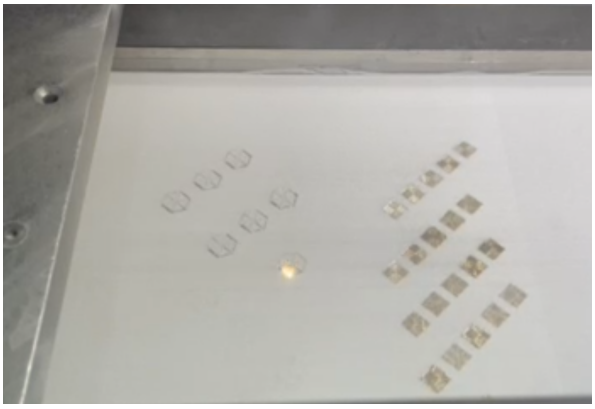

Figure 2: Photograph of LPBF machine in use (taken by ourselves)

2.2 Laser Powder Bed Fusion Parameters

Outlined in the diagram below are the main Parameters for LPBF. The parameters consist of process parameters and laser parameters, with the latter further divided into those affecting accuracy and those affecting surface properties of the product. We focused on the laser parameters that affect the accuracy of the resulting product for this project. These are the beam compensation (BC) and contour distance (CD).

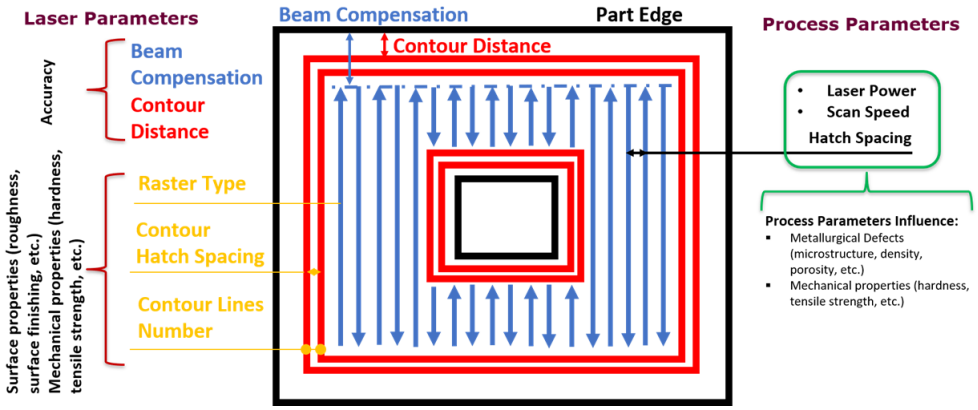

Figure 3: Example diagram of LPBF parameters [5].

The BC is the distance from the laser filling scan area, to the target edge of the part. The CD is the distance from the centre of the contour scan track, to the target edge of the part. The contour scan is usually used to improve the surface finish of the product.[6] However, for smaller features of a product, it also has a role in the accuracy of that feature.

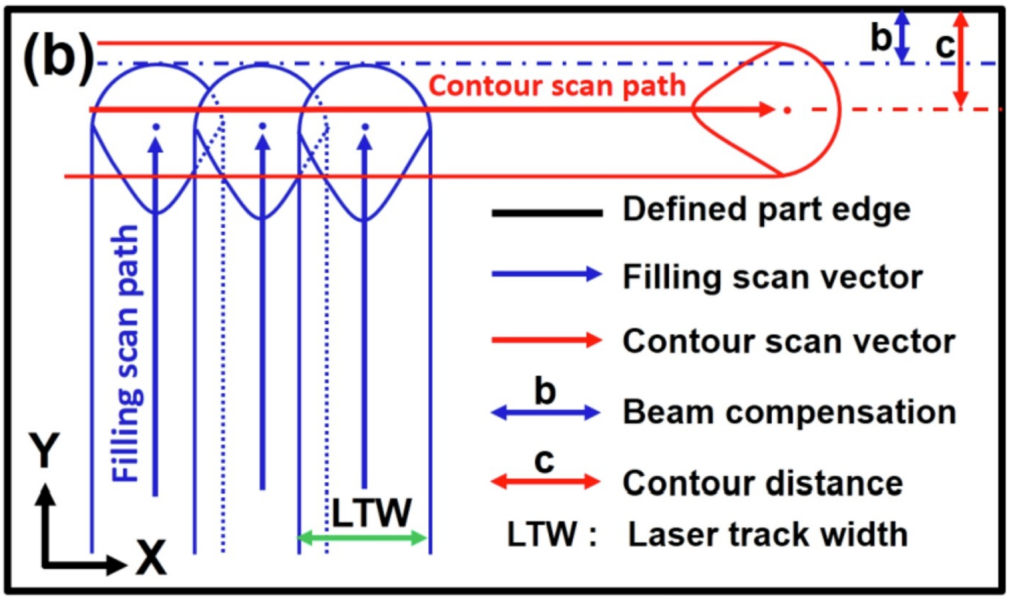

Figure 4: Example visual representation of BC and CD [6].

2.3 Data

PHD student Francesco Careri has conducted the preliminary experiments, creating twenty five test products based on the same design, varying the CD and BC values between 0 and 100 µm for each one and keeping the other parameters constant, while making a record of which products corresponded with which parameters. He then took closeup images of components of these products and manually measuring the width of each, writing both the width in the design and this measurement in a table. This width was calculating using open source software to manually drag a line over the portion of the image being measured, which would provide a measurement in pixels. This was then repeated with the scale bar, producing measurements in pixels for both. As the width that the scale bar represents was known, this could be used to calculate a width of the beam in millimeters. This could be repeated a few times per image in order to obtain an average.

Literature Review

To get a better idea of the work related to this project, a literature review was undertaken.

3.1 Machine Learning

Firstly, we go over the literature relating machine learning to additive manufacturing, and then papers focusing on gradient descent. A very recent publication [7] used a reinforcement learning approach. They focused on optimising the power and velocity of the laser in order to maintain a constant melt pool depth or 1mm. This research had a comparable goal to mine, although with different parameters to be optimised for a different purpose. The paper concludes by suggesting a few alternative methods, along with a couple of adjustments to their own method. While this paper shows that machine learning can be applied to similar problems, it differs in the fact that we had a labeled data-set to work with, whereas they did not. Therefore, reinforcement learning was less optimal for our specific problem, due to us having a labeled data-set. Another paper [2] analyses a multitude of machine learning techniques applied throughout the additive manufacturing process. They conclude that the current machine learning techniques could very well be applied to many more areas of additive manufacturing that those they looked at. However, they warn that in practice, additive manufacturing can be error prone, so they suggest large data sets to be used in order to easily spot and eliminate any errors. Therefore, a way of handling these errors had to be taken into account in this project. The computer vision aspect reduces some of these errors by eliminating human error from the measurement process. The next paper [8] uses an open-source data set and aims to predict 3D chord length distributions via machine learning regression algorithms. They managed to achieve a 90.6% accuracy with their results using polynomial regression. Although being used for a classification problem, this demonstrates the effectiveness of polynomial regression for problems in additive manufacturing. The next paper [9] looks at a problem similar to the one in the proposal, but for fused filament fabrication. It highlights unsupervised learning methods, due to having an unlabeled data-set. The conclusion mentions that its analysis should be extended to finer features in the future, which is what my project will look at, but for metal additive manufacturing. A couple of papers [6, 10] highlight the importance of optimising the parameters of beam compensation and contour distance to improve the dimensional tolerance of the resulting product. The paper [6] attempts to do this, however, they use a different powder and also use experimental methods rather than computational.

This paper [11] provides an overview of a simple gradient descent algorithm applied to a linear multivariate problem. Although this looked at a linear regression problem, this could be relatively easily adapted to work for polynomial regression, which was needed due to the shape of our data. Following on from this, another paper [12] provides an overview of multivariate polynomial regression. It provides an equation for the regression function in this case. It highlights some issues with using this, however these can be overcome with more modern techniques. Another paper [13] details a number of methods to optimise gradient descent. They also provide an overview of the strengths of each, along with a comparison of how well they worked. This provided us with a good basis for methods that could be used to optimise our own algorithm. This paper [14] expands on two of the methods highlighted in the previous paper, highlighting how momentum can be leveraged in dramatically accelerating convergence speeds. Next, these papers [15, 16] go into detail comparing different feature scaling techniques and their impact on accuracy and convergence rate respectively. The first paper concluded that there was little change in accuracy between the methods, however, the second saw a large difference in convergence time between the two methods analysed. Also, the second paper used multivariate gradient descent to test the performance, making the results highly relevant to our own project. Using this information, we

implemented the feature scaling with the quickest convergence identified from [16].

To find the minimum point on the graph produced by gradient descent, simulated annealing was used. This paper [17] details several optimisations and considerations to be made for this. For our project, the base simulated annealing was sufficient, but if the graph was of a higher order, with more peaks and troughs, the optimisations would be more necessary. The performance of the standard simulated annealing approach was also comparable to the optimisations proposed, in some cases performing better.

3.2 Computer Vision

Now, we focus on literature related to the computer vision aspect of the project. Research in measuring things using computer vision has existed for quite a number of years. This paper [18] from 1999 shows techniques for measuring the length of roots by processing digital images. The techniques used are quite basic, but show that even so long ago, measurement using computer vision was sort after, although took significantly longer than modern methods. We tried a variation of this method first, however due to our specific images, it did not work. While not related to the field of additive manufacturing, this paper [19] shows how, by using an object of a known size, we can calculate the size of another object by the number of pixels. With our images, there is a scale bar instead of using an object. With their setup, they only managed to achieve an accuracy of 80.4%. This accuracy, however, could be down to them developing the system as an android phone app (using a lower quality camera to what we have available) to be used in less controlled environment (such as the market or at home). As we did not need to detect the actual object in our project, as it was isolated, we could implement an edge detection algorithm, then calculate the area between the edges to find the average width instead. The next paper [20] provides a comparison of three edge detection methods. It concluded that Canny edge detection yielded both the highest accuracy and the lowest execution time. This was especially true for images with high noise, similar to some of the images that we are working with. This paper [21] goes into detail with improvements that can be made to Canny edge detection. These improvements yielded superior accuracy to other methods, however resulted in relatively slow run-time, so this was needed to be taken into consideration. Next, this paper [22] goes into a number of methods implemented in the openCV2 (open source computer vision 2) library. It highlights that the library is open source with many of its methods yielding better results than the commercial image processing packages. Canny edge detection is mentioned as being a good tool for edge detection.

Problem Specification

The project was split into two main sections, a machine learning section and a computer vision section. Overall, the goal was to optimise the BC and CD of the LPBF process for dimensional tolerance. Specifically, we needed to deduce an optimum set of parameters to be used in general for any width required, to produce a product with actual dimensions as close as possible to what is needed.

4.1 Machine Learning

Firstly, we were provided with the measurements of twenty five test products. Each had six metal beams of different sizes joining a central column to an outer ring. The sizes of these beams were 1mm, 0.7mm, 0.5mm, 0.4mm, 0.3mm and 0.2mm.

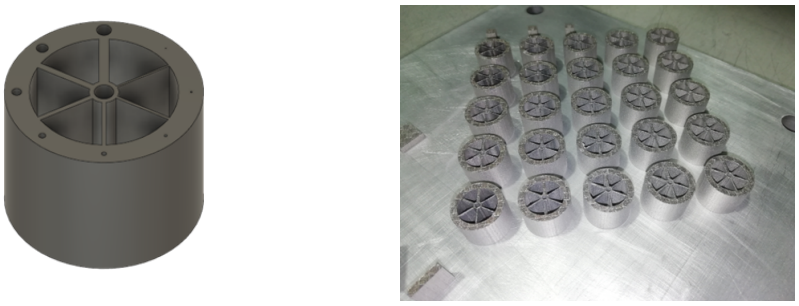

Figure 5: Example graphic and photograph of test products [5].

The BC and CD values were both in the range of zero to one hundred. They increased in steps of twenty five, providing twenty five different combinations of values. Each one was photographed by a high resolution microscope with a scale for the image provided. These photographs were then, using an open source program, manually measured by selecting and dragging over the scale bar to get how many pixels wide it was, taking a few measurements of the beams and dividing to get an average, dividing this by the number of pixels of the scale bar, then multiplying by the scale. This then gives us an initial data-set to work from with a BC and CD value along with six measurements and their targets for each test part.

Our first objective for this section was therefor to take these values and measurements provided to us, determine a relationship between them through a machine learning algorithm, and use another to calculate optimum values. The goal was that these values would yield widths within 10% of the target value. Our second objective of this section was to optimise the algorithm its self to perform the calculations quickly. We set a goal of reducing the time to run for this section to at least under a second. This objective was important to enable the algorithm to analyse new data and produce results faster than a human could.

4.2 Computer Vision

With the initial measurements being taken by hand, there arises the issue of human error. Due to the repetitive nature of individually measuring 150 different widths, human error was bound to occur. A 2010 study into human errors in aviation maintenance found that the joint highest cause were repetitive

tasks [23]. To overcome this, we will use the second section of the project. This objective was to develop a way, using computer vision, to automate the process of measuring each beam, and get more precise measurements. We would then store these values in a new file, and pass them through the machine learning algorithms to output a new set of optimum values. Another objective of this section is to be able to analyse the images quickly. While increasing the accuracy of the measurements themselves is a benefit on its own, there are additional benefits to also increasing the speed.

Design

We will present an overview of the code, along with the equations used.

5.1 Machine Learning

5.1.1 Problem formulation

Since we wanted to minimise the error of the width of the beams, we needed to minimise a function $f(x)$ that takes the BC and CD as input variables, with percentage error being the output. We also have the limits set by the machine, so the BC and CD could not be less than zero or greater than one hundred. So, the problem formulation is as follows:

$$\begin{aligned} \text{Minimise:} \quad & w_0 + w_1 x_1 + w_2 x_2 + w_3 {x_1}^2 + w_4 x_1 x_2 + w_5 {x_2}^2 \\ \text{Subject to:} \quad & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_1 \leq 100 \\ & x_2 \leq 100 \end{aligned}$$

From these, we can easily derive the objective function and constraints in the canonical formulation:

$$\begin{aligned} \text{Objective function:} \quad & f_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 {x_1}^2 + w_4 x_1 x_2 + w_5 {x_2}^2 \\ \text{Constraints:} \quad & g_1(x) = -x_1 \\ & g_2(x) = -x_2 \\ & g_3(x) = x_1 - 100 \\ & g_4(x) = x_2 - 100 \end{aligned}$$

Using these, we can write them as the canonical problem:

$$\begin{aligned} \text{Minimise:} \quad & f(x) \\ \text{Subject to:} \quad & g_i(x) \leq 0, i = 1, 2, 3, 4 \end{aligned}$$

An alternative method of obtaining results can be to obtain a different equation for each target width, making y equal the actual width instead of percentage error. Then we could obtain a graph of x_1 against x_2 by setting the value of y to be the value of the target width. This would give a range of viable results instead of the singular best, and is useful for visually representing the relationship between BC and CD. Graphs produced by this method will be found in the results and evaluation section, as the only differences in code are the exclusion of simulated annealing, and inclusion of a function to take a slice of the gradient descent graph at a specified value of y. The method produces the line formed when the plane equal to the target width crosses the 3d graph produced by the gradient descent. A visualisation of this method is shown below:

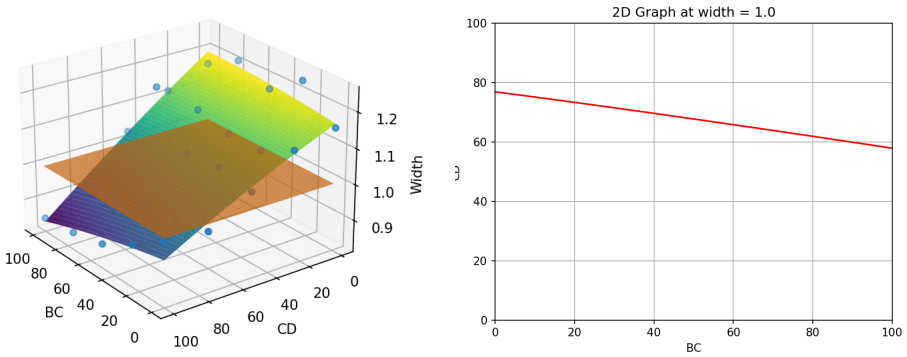

Figure 6: Visualisation of alternative method.

5.1.2 Gradient Descent

We will discuss the initial code developed first, then go into optimisations made afterwards.

Initial code

The first thing we did was import the relevant data from a csv file. The data was stored in a way that the heading for the measurements columns was that of the target measurement. Then we stored this data in three separate lists, x_1 for the BC values, x_2 for the CD values, and y for the measurements. For the measurements values, we divided each value by the value in position 0, which is the target, then multiplied by 100 to obtain a percentage error. For all 3 lists, we then remove the first value to remove the heading. To use gradient decent with the percentage error from every measurement, we repeated this for each target column, then added them together. We also added six x_1 and x_2 lists together to get the same length lists in the correct order. This data is known as the training set.

The next part of the code involves a while loop containing the gradient descent algorithm. The goal of the gradient descent algorithm is to minimise a function known as the cost function. We chose this cost function to be the equation for mean square error. This is the sum of the squared difference between the known values from the test products and the graph generated by the current iteration of the algorithm, divided by the number of examples in the training set. The equation is below, we divided through by $2N$ here instead of N due to it simplifying the differentiation later.

$$g(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^{N}(f_w(x^{(n)}) - y^{(n)})^2$$

Where:

- $g(\mathbf{w})$ is the cost function
- $\mathbf{w}$ is the vector containing the parameters of the target equation, ($\mathbf{w} = w_0...w_5$)
- N is the number of elements in the training set
- $f_w(x^{(n)})$ is the estimated output of the nth element of the training set
- $y^{(n)}$ is the y value at the nth element of the training set [11]

To minimise this cost function, we needed to find an optimum $\mathbf{w}$, such that $g(\mathbf{w})$ is minimised. This is because, the closer the to 0 the cost is, the closer the equation $f(x)$ is to fitting the data. This is achieved by repeatedly taking small steps in the direction of the steepest descent. The equation below describes how this is achieved:

$$\mathbf{w} := \mathbf{w} - \alpha \nabla g(\mathbf{w})$$

Where:

- α is the learning rate or step size, initially set to a small value greater than 0 but less than 1, usually set to around 0.01
- $\nabla g(\mathbf{w})$ is the vector of the partial derivatives of $g(\mathbf{w})$ with respect to $w_0, ..., w_5$

$\nabla g(\mathbf{w})$ gave us the steepest gradient at each $\mathbf{w}$ value, so we multiply by alpha and take this away from the previous iterations value to take a small descending step. This was repeated until the algorithm sufficiently converged. Below you can find pseudo-code of the initial algorithm.

$x1 \leftarrow [x_1]$
$x2 \leftarrow [x_2]$
$y \leftarrow [y]$
$w_0, w_1, w_2, w_3, w_4, w_5 \leftarrow 0$
$a \leftarrow 0.01$
$c1, c2 \leftarrow 0$
while $(c1 - c2) \geq 0.0001$ **do**
    $c \leftarrow 0$
    for $i \in length(y)$ **do**
        $f = w_0 + w_1 x1[i] + w_2 x2[i] + w_3 x1[i]^2 + w_4 x1[i] x2[i] + w_5 x2[i]^2$
        $c \leftarrow c + (f - y[i])^2$
        $w_0 \leftarrow a * (f - y[i])$
        $w_1 \leftarrow a * (f - y[i]) * x1[i]$
        $w_2 \leftarrow a * (f - y[i]) * x2[i]$
        $w_3 \leftarrow a * (f - y[i]) * x1[i]^2$
        $w_4 \leftarrow a * (f - y[i]) * x1[i] * x2[i]$
        $w_5 \leftarrow a * (f - y[i]) * x2[i]^2$
    end for
    $c = c/(2 * length(y)$
    $c2 = c1$
    $c1 = c$
end while

This algorithm required tuning of the alpha value for each $w_0, ..., w_5$ in order to not diverge. Appropriate initial values for $w_0, ..., w_5$ were also required to converge at the correct values. While a good starting point, this code took too long to converge to be viable, taking multiple minutes at best, and required manual specific changes to the learning rate and for the point at which to stop iterating. Therefor we concluded that optimisations to the algorithm would be necessary. Such optimisations are outlined in the following sections.

Feature standardisation

The first modification we made to the machine learning algorithm was to add feature standardisation to the y, x_1 and x_2 values. The method is commonly used in machine learning, as it assigns equal importance to each variable. Because of this, it reduces the effect of outliers and greatly reduces the number of iterations taken to converge.[24, 25]. For each item in the feature, the equation subtracted the mean of the feature, then divided by the standard deviation.

$$x_{\text{standardised}} = \frac{x_i - \mu}{s}, \text{ for } i = 1, ..., n$$

Where:

- x_i is the individual value in the feature being standardised
- n is the number of items in the feature
- μ is the mean of the feature ($\mu = \frac{\sum x_i}{n}$)
- s is the standard deviation of the feature ($s = \sqrt{\frac{\sum (x_i - \mu)^2}{n}}$)

The introduction of feature standardisation required us to "undo" it to convert the values obtained from the simulated annealing, and for $w_0, ..., w_5$ to give an equation that can be plotted using the original values of y, x_1 and x_2. To convert a value into one useful to us, we simply rearranged the standardisation equation for x_i.

$$x_i = x_{\text{standardised}} * s + \mu$$

Momentum

The next modification made to the initial design is the addition of classical momentum. This is a technique for accelerating the convergence of gradient descent by accumulating a velocity vector in directions of persistent reduction in the objective across iterations[14]. This means that the momentum term increases when moving in the same direction, and decreases when it changes direction. This results in increased convergence speed and reduced oscillation [13].The introduction of momentum added an additional equation and altered an existed one. The equation below is added in:

$$\mathbf{v} := \beta \mathbf{v} + \alpha \nabla g(\mathbf{w})$$

Where:

- $\mathbf{v}$ is the momentum vector containing the momentum for each $\mathbf{w}$
- β is the momentum coefficient, which is between 0 and 1, closer to 1. Usually this is at least initially set to 0.9.
- α is the learning rate
- $\nabla g(\mathbf{w})$ is the vector of the partial derivatives of $g(\mathbf{w})$ with respect to $w_0, ..., w_5$

The equation for $\mathbf{w}$ is then altered:

$$\mathbf{w} := \mathbf{w} - \mathbf{v}$$

AdaGrad

Adaptive gradient (shortened to AdaGrad) dynamically alters the learning rate for each parameter based on the previous gradients calculated. This calculated larger learning rate updates for infrequent parameters, and smaller updates for frequent parameters [26]. AdaGrad was a popular algorithm used in gradient descent algorithms, influencing the creation of multiple other popular algorithms such as Adam [27]. The combination of AdaGrad and momentum used here form the basis of this. Below is the combination of equations for AdaGrad:

$$\mathbf{d} := \mathbf{d} + (\nabla g(\mathbf{w}))^2$$
$$\mathbf{a} := \frac{\alpha}{\sqrt{\mathbf{d} + \epsilon}}$$

Where:

- **d** is the vector containing the sum of the squares of the gradients for each **w** up to the current iteration.
- **a** is the vector containing the new learning rate for each **w** at the current iteration. It is substituted into the momentum equation instead of α.
- ϵ is a very small value (usually around 10^-8 or smaller) that is added to **d** in order to prevent division by 0 [13].

5.1.3 Simulated Annealing

Simulated annealing was used to find the point on the graph (within the boundary conditions) with the lowest percentage error. The method is based on the annealing technique used in metallurgy used to heat up and then slowly cool down metals to change their physical properties [28]. This technique will always generate a value in the range used to find the probability of accepting a worse neighbour, to be explained in more detail below. The formula for this is the following:

$$p(x) = e^{\frac{\Delta E}{T}}$$

Where:

- $p(x)$ is the probability generated by the equation.
- ΔE is the quality of the random neighbour solution minus the quality of the current solution.
- T is the current temperature of the algorithm, which decreases with the iterations by $T := R * T$ where R is the cooling rate.

The algorithm proceeds as follows:

1. Generate an initial solution for the algorithm to start at. In our case, we generated a random number between 0 and 100 (our boundaries) for x_1 and x_2.
2. Set the temperature and cooling rate. These need to be configured for the problem, as if they are set too high, it takes longer to converge, but if set too low, the solution is more likely to be inaccurate. We chose an initial temperature of 1000 and cooling rate of 0.95 to be applied every 100 iterations.
3. Next, we selected values for the neighbour solution at random, using a Gaussian distribution of mean 0 and standard deviation of 1. This meant that the majority of neighbours would be between -1 and 1 away from the current values for x_1 and x_2, while having the possibility to generate a neighbour further away. This, along with the simulated annealing method as a whole, can help the algorithm get out of local minimums.
4. Then, we checked the value for the percentage error (y) found by the new and old values of x_1 and x_2. If the new y is less than the old, we replace the old values as the current values. If not, we accepted it with a probability calculated above by $p(x)$. This probability starts close to 1, but decreases as the algorithm runs. If the percentage error is the lowest found, this is stored in as a new variable.
5. Then we repeated steps 3 and 4, applying the cooling rate of 0.95 every 100 iterations, until the temperature is below a low threshold, that we set to 10. Then we stopped the algorithm and displayed the optimum values calculated.

Pseudo-code of the algorithm can be found below, with the values of w_0 to w_5 found by gradient descent outlined above:

function $f(x1, x2)$
  return $w_0 + w_1x1 + w_2x2 + w_3x1^2 + w_4x1x2 + w_5x2^2$
end function
$cur\_x1 \leftarrow random[0, 100]$
$cur\_x2 \leftarrow random[0, 100]$
$optimum\_x1 \leftarrow current\_x1$
$optimum\_x2 \leftarrow current\_x2$
$temperature \leftarrow 1000$
$cooling\_rate \leftarrow 0.95$
$iterations\_per\_temp \leftarrow 100$
$lowest\_err \leftarrow f(optimum\_x1, optimum\_x2)$
while $temperature > 10$ **do**
  for $iteration \in range(iterations\_per\_temp$ **do**
    $new_x1 \leftarrow clip(cur\_x1 + random.normal(0, 1), 0, 100)$ ▷ clip ensures that the algorithm will not go past the boundary conditions. random.normal is the Gaussian distribution in point 3 above
    $new_x1 \leftarrow clip(cur\_x2 + random.normal(0, 1), 0, 100)$
    $cur\_err \leftarrow f(cur\_x1, cur\_x2)$
    $new\_err \leftarrow f(new\_x1, new\_x2)$
    if $new\_err < cur\_err$ **or** $random[0, 1) < \exp(cur\_err - new\_err)/temperature$ **then**
      $cur\_x1 \leftarrow new\_x1$
      $cur\_x2 \leftarrow new\_x2$
      $cur\_err \leftarrow new\_err$
      if $new\_err < lowest\_err$ **then**
        $optimum\_x1 \leftarrow new\_x1$
        $optimum\_x1 \leftarrow new\_x1$
        $lowest\_err \leftarrow new\_err$
      end if
    end if
  end for
  $temperature \leftarrow temperature * cooling\_rate$
end while

5.1.4 Computer Vision

Overview

For this section of the project, we initially pursued a couple of different methods. The first was to simply count the pixels with values above a certain threshold, then divide by the height of the image. This would be done by first converting the image to greyscale. This means that each pixel has a value ranging from 0 to 255, with 0 being black and 255 being white. The initial idea had the presumption that the background of the image would always be black, or close to black, with the part of the product being measured being lighter. However, this was not always the case. As evidenced below, this varied from image to image, so could not be used as a consistent strategy to measure the width of the images. Another reason to not use this method was that, half way through the process of capturing the images, the microscope being used broke, so another had to be used instead. This produced images with significant portions of the part being measured to have black pixels.

Another method initially considered was to use a corner detection algorithm, draw lines between them and calculate the area enclosed. This method ran into issues, as it either wouldn't reliably detect every corner, or would detect too many corners due to the roughness of the edges of the products.

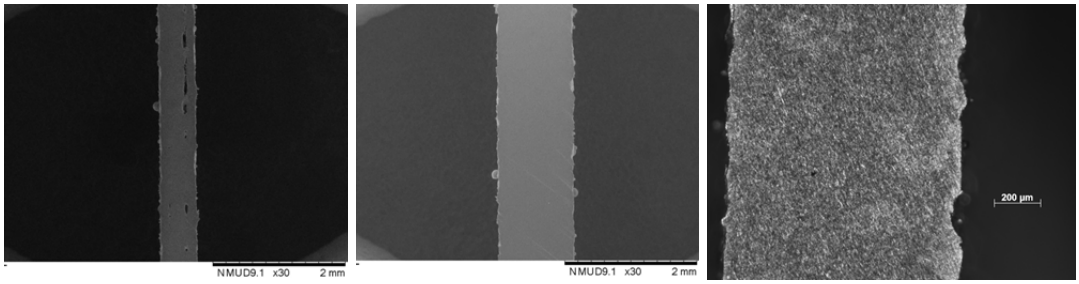

Figure 7: Example of the photographs provided to us by PHD student Francesco Careri.

The third method, and the method settled on, was to detect the edges of the part of the product being measured. Then count the pixels enclosed by the innermost and outermost edge pixels. This was due to edges being detected within the boundaries of the parts of the product being measured. To accomplish this, we used the Canny edge detection algorithm.

Canny Edge Detection

Canny edge detection was first developed by John Canny in 1986 [29], and has been widely used since to detect edges within images. The algorithm is composed of a five main steps:

1. The first step is to remove the noise from the image by applying a Gaussian filter to smooth the image. This essentially involves moving a mask of dimensions $2n-1$ x $2n-1$ where $n \in \mathbb{Z}^+$ over each pixel, to calculate a new value for the pixel. Each point on the mask is calculated by [30]

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

   Where:

   - $G(x,y)$ is the matrix produced
   - σ is the standard deviation of the Gaussian distribution, chosen by the user
   - x and y are the distances from the origin of the mask.

   An example of such a mask with dimensions of 5x5 and σ equal to 1 is: The centre of the mask is

0	.01	.02	.01	0
.01	.06	.11	.06	.01
.02	.11	.16	.11	.02
.01	.06	.11	.06	.01
0	.01	.02	.01	0

Figure 8: Example mask [31]

   placed on each pixel, then a new value for the pixel is calculated based on multiplying each pixel covered by the mask by its corresponding value in the mask, then adding them together. This technique attempts to smooth the image, removing random points of brightness or darkness.

2. The second step is to find the intensity gradient of the image. This was done by utilising the Sobel operator [20] to obtain an approximation of the first derivatives of the image in the horizontal and vertical directions, comparing each pixel to its neighbours. This works similar to the above,

by passing a 3x3 mask over the image, but two masks are used separately to get a value for the derivative in the horizontal and vertical directions. The equations for this are:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

Where:

- G_x and G_y are the approximations for the first derivatives in the x and y directions respectively
- I is the image
- $*$ is a convolution operation

Each of the G_x and G_y values are then passed into the following equations to calculate the overall edge gradient and its direction.

$$G = \sqrt{{G_x}^2 + {G_x}^2}$$

$$\Theta = \arctan(\frac{G_y}{G_x})$$

The value of Θ is then rounded to the nearest 45° to approximate its direction as vertical, horizontal, or either diagonal.

3. The third step in the algorithm is to apply non-maxima suppression. For each pixel, this checks the neighbour G values in both the positive and negative direction of Θ and compares to its own G value. If its own G value is the maximum of the three, it is kept, otherwise it is suppressed and set to 0. Therefore the only pixels kept are those which are on a local change in intensity [20].

4. Next, a double threshold is applied to the remaining edge pixels. Two thresholds are set by the user, an upper threshold and a lower threshold. These separate each pixel into three categories. All below the lower threshold are suppressed and set to 0. All above the upper threshold are considered sure-edges and set to 2. Everything in the middle is considered as a potential edge and deemed weak edges and set to 1.

5. Lastly, hysteresis edge tracking is applied. This checks to see whether each weak edge is connected to a sure-edge. If it is, it is changed to a sure-edge. These new sure-edges can also cause neighbouring weak edges to become sure-edges. If a weak edge is not connected to any sure-edges, it is discarded. An example of this is:

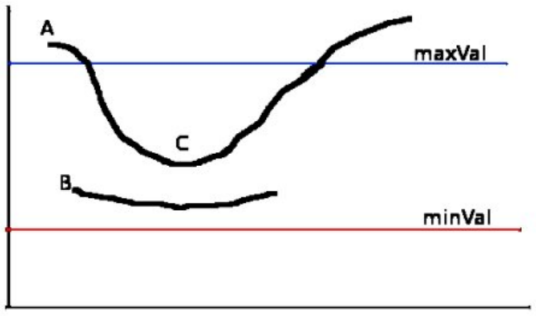

Figure 9: Example of hysteresis edge tracking [32]

In the above example, edge A would be accepted as a sure-edge. As C is between the thresholds, and connected to A, it would also be accepted. As B is between the two thresholds, but not connected to any sure-edges, it is discarded.

Calculating the average width

To calculate the average width, we first took a measurement, in pixels, of the scale. As this was the same for each image with each microscope, just one measurement was needed for each microscope. The scale from the scale bar was then divided by the number of pixels to obtain the width of an individual pixel in millimeters. The scale bar was then cut out of the image so that it would not interfere with the edge detection.

Canny edge detection was then used and the edges produced overlaid on the original image in blue.

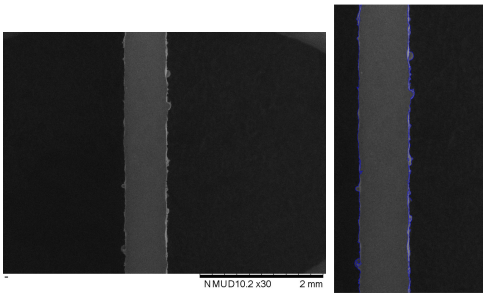

Figure 10: Example of the original image on the left, with the edges highlighted on the right

We had to check each image individually to ensure that the algorithm was working as intended. As there was variation in the intensities of the background and the test product, some were quite close in intensity. This necessitated the introduction of a while loop to check if each row of pixels in the image had at least two blue pixels. If not, the upper threshold for the double threshold step of the Canny edge detection, initially set relatively high, would be lowered, until each row did have at least two.

We then used the coordinates of these blue pixels to obtain the coordinates of the blue pixel furthest right and furthest left for each y value. This was necessary, as the edge detector regularly detected edges inside the beam. The difference between the two x coordinates was then calculated for each y value. This was then divided by the height of the image to get the average width in amount of pixels, then multiplied by the width of an individual pixel calculated above to get a value in millimeters.

Each of these values was then stored in a csv file to be used by the machine learning portion of the project.

Our own implementation of the canny edge detection algorithm was quite slow, taking around ten to twenty seconds per image, so due to time constraints in optimising this, we opted to use the predefined Canny function in the OpenCV import detailed below.

5.1.5 Python Imports

For the coding in the project, a few libraries were imported. These are:

- Math - This library is beneficial when dealing with simple calculations like calculating the square root of a number.
- NumPy - This library is useful for more complex calculations and functions, and had some particularly useful functions, such as clip, used in the simulated annealing section to prevent the generation of a point beyond the boundaries set. NumPy was also useful in assisting with the generation of 3d graphs.

- Matplotlib - This library enabled the creation of graphs in python.
- csv - This library allowed us to open and save csv files in python.
- time - This library simply let us time how long each algorithm took to run.
- OpenCV (cv2) - This library allowed for image manipulation in python, such as to open and save an image, to crop it and to modify properties. It also came with a number of functions that allowed for much faster processing times of the images.

5.2 Testing

Throughout the project, various means of testing have been used to ensure that the code was working correctly. The first step for the gradient descent was to get the algorithm to be able to produce a curve fitting the points. Initially this was simple, as we just plotted the curve and the points on the same axis. This would visually give us an idea if it was working, as when we were implementing the algorithm early on, the curve produced did not fit the points, giving us immediate feedback that something had not worked correctly. Tests on the performance of these functions were also carried out, firstly by simply outputting the amount of iterations taken, then later, when the algorithms were improved upon, timing them.

For the vision aspect, the edges detected by the algorithm were displayed overlaying the original image, allowing us to easily spot if edges were not being detected, or if too many edges were being detected. Each image was manually checked to ensure the algorithm was performing as expected. Once it was performing as expected, it was timed and optimised to improve the speed.

Project Management

6.1 Initial plan

The initial plan for the project was to follow a waterfall style approach. This meant that we would not move on until one section was finished. The timeline of events would roughly follow the diagram below:

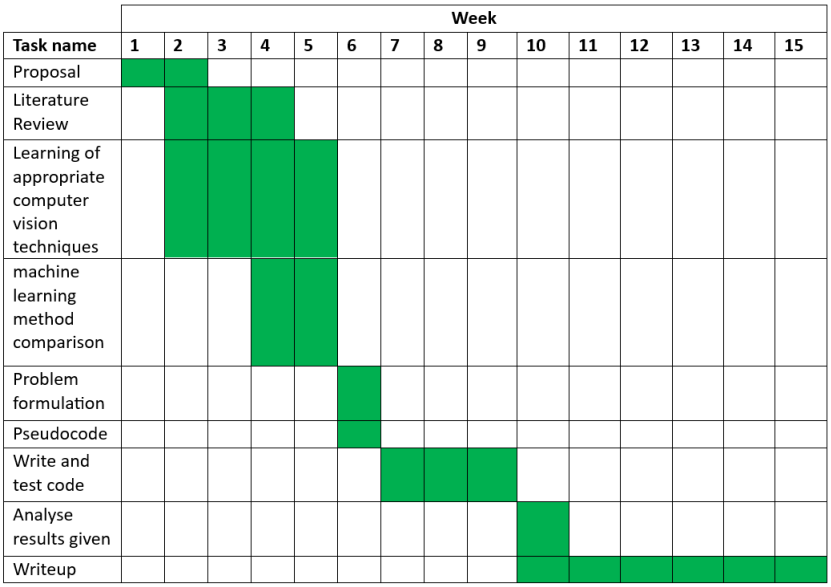

Figure 11: Initial Gantt chart developed for the project

The main stages could roughly be defined as:

- Research
- Coding
- analysis and write up

During the research phase, the plan was to gather as much information as I could about what I needed for the project. As I did not have much experience with computer vision, my project supervisor, Leonardo Stella organised for me to gain access to the computer vision module taught on a separate course. Firstly, I decided on the use of python over Matlab for the implementation. There were several reasons for this choice. Firstly, I had not used either language in a great capacity prior to this project, but python was more similar to java, which was used throughout the course. Python is used more in companies than Matlab, so I thought it would be a benefit for the future to be more familiar with it. In machine learning, python is much more widely used, so gathering information would be much easier if I ran into any issues. For the computer vision aspect, although the computer vision course that I had access to was taught in Matlab, I still decided on python, as similar libraries existed that served the same purpose. Another consideration was that Matlab has a cost associated with it if I wanted to use it after this degree, whereas python is free. This would allow me to carry out further projects, building on the knowledge gained from this one. Then I carried out research into each area of the project.

The next stage was to code the implementation of the methods found in the first stage. Each stage was coded and tested individually, then added together. This was to be able to more easily identify and correct errors when they occurred, instead of searching through the entire code. This also prevented early errors from having a knock on impact further down.

The last stage was to analyse the results produced by the code. I made several observations, discussed in the following sections. The results were also handed to Francesco Careri, so that further experiments could be carried out based on them. During this stage, Francesco provided me with the opportunity to see the machine in work, which provided a better understanding of how it worked.

Throughout the process, regular meetings, in person and using Zoom took place with the project supervisor Leonardo Stella.

6.2 Issues and future changes

Issues

This approach had a few issues. It was quite inflexible and didn't leave much room for changes if big changes were needed for the code. It also vastly underestimated the time that it would take me to be able to implement the code. I had failed to appreciate that, while similar to java in a lot of ways, python had many small variations in the syntax used between the two. The time taken to obtain the correct syntax added up to become a sizable increase in time taken over the initial prediction. Another presumption was that once I had done the research portion, I would not have to do more. This was an incorrect presumption, as several times methods either did not work for my specific case, or needed modification to speed them up. Another issue was that I had failed to take into account the time it would take to learn how to use Latex for typing up my report. This took me a lot longer than anticipated, due to the amount of different commands and customisation available. Another issue encountered was the use of git, I had not previously used git, although it was an extremely useful system to learn for the future. I encountered an issue that I did not know how to fix at the time, which ended up taking a lot longer than anticipated.

Future changes

Changes that I would make to the initial plan would be to do more things in parallel. What I mean by this is to do some initial research, attempt to implement those methods, then conduct more research on improvements (or alternatives if they did not work) to those methods. This would have allowed for more flexibility, meaning that changes in direction to be less impactful on the overall time-frame of the project.

Results and Evaluation

7.1 Performance

The initial implementation of the gradient descent algorithm took into the tens of minutes to converge to a point that had a graph that fitted the data. Any attempt to bring this down through modification of hyper-parameters led to a graph that did not represent the data. After the modifications outlined in the design section, along with some tuning of the hyper-parameters, this now runs in between 0.02 and 0.4 seconds depending on the amount of data being used.

The initial implementation of the simulated annealing part fared much better. This achieved a run-time of 1.8 to 2.5 seconds. Just through modifying the parameters of the algorithm, we managed to bring this down to 0.09 to 0.13 seconds. This brought the overall run-time of the machine learning section to between 0.1 and 0.5 seconds.

For the initial implementation of the computer vision section, the run-time was quite slow. As mentioned in the design section, it took ten to twenty seconds per image, so with 125 images this would average at over thirty minutes. After switching to the imported Canny edge detection function, this was brought down to between 0.02 and 0.1 seconds per image, totaling to between 9 and 10 seconds for all 125 images.

7.2 Results

Our first results were calculated using the measurements obtained from Francesco Careri. The algorithm was run for each target width, and then run once more to try and obtain an overall value for the beam compensation and contour distance that would lead to an accurate product for all dimensions. It first outputs a 3d graph, plotting BC, CD and the percentage error against one another. This graph contains a scatter-plot of the values in the input, along with a curve that is a plot of the equation calculated by the gradient descent.

Examples of which are:

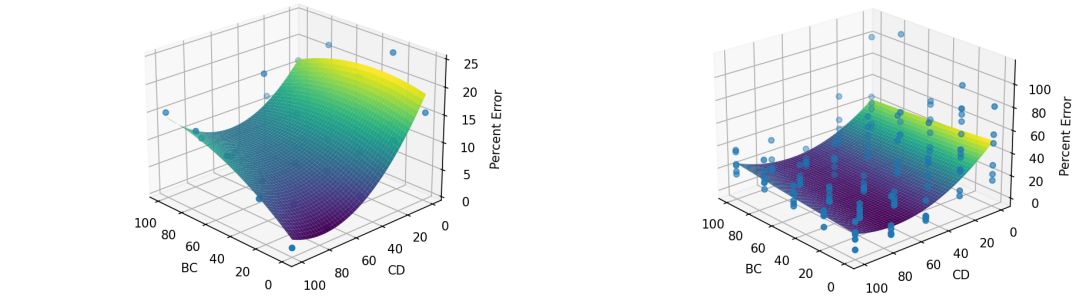

Figure 12: Example graphs produced by gradient descent using initial values. Target is 1mm on the left and an overall value on the right

The simulated annealing then produced optimum values for the BC and CD, based on the lowest point on the graph. These values are below:

Nominal [mm]	BC (µm)	CD (µm)
1	0	77.8215
0.7	0	59.6035
0.5	100	57.3237
0.4	0	71.7166
0.3	0	85.4639
0.2	48.3565	71.5002
Overall	0	72.8877

Figure 13: Table of optimum BC and CD found by gradient descent and simulated annealing using initial values

Next, the computer vision code was run to obtain new measurements for each product. The output of this can be found in appendix A. These results were then used in the machine learning algorithms, producing the below graphs and optimum values:

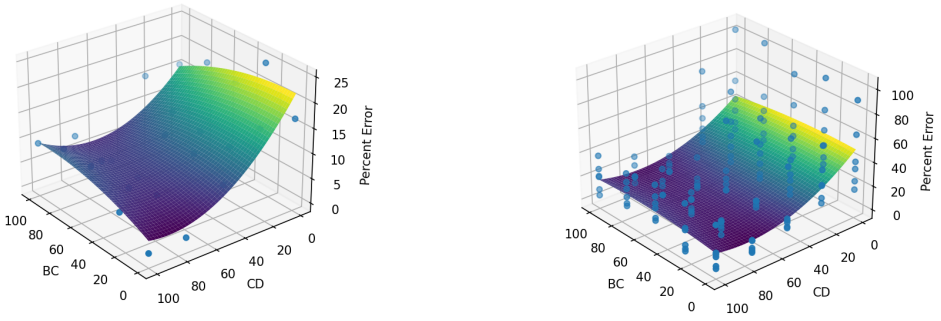

Figure 14: Example graphs produced by gradient descent using computer vision values. Target is 1mm on the left and an overall value on the right

Nominal [mm]	BC	CD
1	0	87.0293
0.7	100	58.6119
0.5	100	61.8684
0.4	0	89.5822
0.3	0	87.3401
0.2	100	81.8075
Overall	0	79.5866

Figure 15: Table of optimum BC and CD found by gradient descent and simulated annealing using computer vision values

The values from both the above tables were then handed to Francesco Careri, where further experiments were carried out. Unfortunately, due to scheduling and availability, these experiments were only able to

be carried out in the week before the deadline, not leaving enough time for them to be analysed by the computer vision code. The results of these can be found below:

Based on initial measurements

Single Parameters

	Nominal [mm]	BC	CD	Measured [mm]
Wall 1	1	0	77.8215	1.012
Wall2	0.7	0	59.6035	0.694
Wall 3	0.5	100	57.3237	0.508
Wall 4	0.4	0	71.7166	0.394
Wall 5	0.3	0	85.4639	0.314
Wall 6	0.2	48.3565	71.5002	0.171

Overall Parameter

	Nominal [mm]	BC	CD	Measured [mm]
Wall 1	1	0	72.8877	1.037
Wall2	0.7	0	72.8877	0.657
Wall 3	0.5	0	72.8877	0.468
Wall 4	0.4	0	72.8877	0.408
Wall 5	0.3	0	72.8877	0.312
Wall 6	0.2	0	72.8877	0.251

Based on computer vision measurements

Single Parameters

	Nominal [mm]	BC	CD	Measured [mm]
Wall 1	1	0	87.0293	0.993
Wall2	0.7	100	58.6119	0.704
Wall 3	0.5	100	61.8684	0.517
Wall 4	0.4	0	89.5822	0.407
Wall 5	0.3	0	87.3401	0.313
Wall 6	0.2	100	81.8075	0.19

Overall Parameter

	Nominal [mm]	BC	CD	Measured [mm]
Wall 1	1	0	79.5866	1.029
Wall2	0.7	0	79.5866	0.687
Wall 3	0.5	0	79.5866	0.486
Wall 4	0.4	0	79.5866	0.401
Wall 5	0.3	0	79.5866	0.325
Wall 6	0.2	0	79.5866	0.237

Figure 16: Tables of results from further experiments

As mentioned in the design section, an additional graph was produced for each set of values. This was a graph showing the range of values for BC and CD that would potentially yield the target width for each beam. This can be seen below:

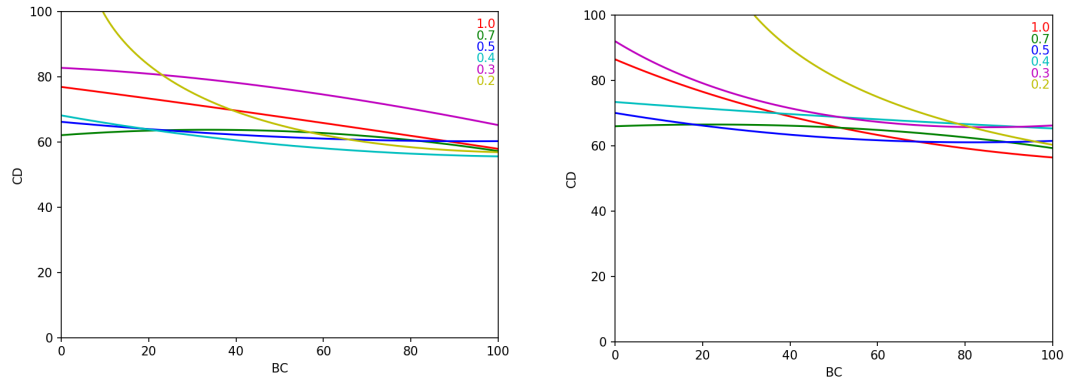

Figure 17: 2D graphs produced plotting BC against CD for target values of width. The graph on the left was produced with the initial values as input, the graph on the right used the new computer vision inputs

7.3 Evaluation

The performance of the algorithms drastically improved over the course of the project. Initially, the run-time would have been so long that it would be impossible to make changes and receive results in a reasonable amount of time. However, after the modifications made to the code, it now takes just seconds if images need to be analysed, or is near instant if the values are already ready to be analysed. This increase in performance allows for much bigger data sets to be reasonably analysed, if available.

The machine learning section of the project was able to achieve the objective of returning optimum values for BC and CD. The algorithm can also take results from additional experiments to build upon these initial predictions to provide improved predictions. The more accurate data that can be fed into the algorithm, the better the predictions will be.

For the most part, the measurements produced by the computer vision component of the project were similar to those measured prior. There were a couple of values that changed significantly, most notably those of test #14. The values for test #14 seemed to be outliers in the initial set of results, but are inline with what would be expected in the computer vision results, which seems to point to an element of human error involved. The computer vision algorithm can be shown to save time, while also minimising the potential for error in the experimental process.

Although there was unfortunately no time for a detailed analysis of the results from the followup experiment, we can observe that the optimum values for BC and CD provided from the computer vision values, on average, lead to more accurate predictions for the parameters.

The graphs produced by the alternative method show that, generally, the BC has much less of an impact on the measurements than the CD. The exception to this is at 0.2mm. Observing the 0.2mm graph, we see that the BC plays a much bigger role. The BC seems to need to be above 25 µm to yield BC and CD values that produce a beam width of 0.2mm.

Discussion

In this section, we will summarise the achievements of the project, while also detailing deficiencies and areas that can be improved.

8.1 Achievements

Overall, the project was successful. It obtained parameters that can be used to optimise the BC and CD of the LPBF process. These values, based on initial analysis of data, proved to be fairly reliable. The parameters produced for the specific target widths, when produced with values obtained via the vision analysis, proved to yield results well within the target range of 10% tolerance. The overall optimum parameters produced worked well for each target value except for 0.2mm. The alternative method used also provided some interesting observations on the relationship between the BC and CD. In general, the BC has little to no effect on the accuracy of the beam width. An exception can be made at 0.2mm however, where it plays a significantly larger role. This may explain why the predicted overall optimal BC and CD did not work as well for the 0.2mm target. The computer vision image analysis is a significant improvement over human image analysis, allowing for more accurate measurements of the widths of the beams to be taken place over a much smaller amounts of time.

8.2 Deficiencies

There are multiple aspects of the project that can be improved. Using the mean squared error for the cost function can allow for outliers to affect the final result more than other methods [33]. However, this cost function was the one most familiar to us, so was selected. Given more time, we could have learned more about other cost functions and how to implement those instead.

Another aspect that can be improved is the overall adaptability of the algorithms for new data. Because of the way the code reads data from the csv file for the machine learning algorithms, it requires all columns to be filled in for each row. Therefor, if new data is obtained for just one target width, the code will break. A solution for this would be to remove the data at a certain position in the x1, x2 and y lists if a value in the y list is null. When finding optimum overall parameters, a unique x1 and x2 list would also be required before combining them together. For the computer vision section, the values for the BC and CD used, along with the target width were written into the code, so these values will be incorrect if new images are used. To get around this, the files may need to be renamed in a specific way, with that data in the filename. Then, for each image, it could obtain this from the filename and save the data accordingly. The code produced is also specific to images in a specific orientation, with the beam running either vertically, or with the entirety of it crossing the top and bottom of the image. If images are produced in different orientations, such as horizontally, the code will not work. This can be partially fixed by allowing the code to also check if each x coordinate contained at least two y coordinates and classify the image as either horizontal or vertical. This would not work if the beams were angled in such a way that they cross the corners of the image. The scale bar for each image must also be in the same place.

A small issue encountered with the images provided was that half were taken with a different microscope than the other half, also containing a difference in the position of the scale bar. The code had to be constructed with this in mind, so there should not have been any issues with the actual results because of this. An example of what the visual differences are is on the following page.

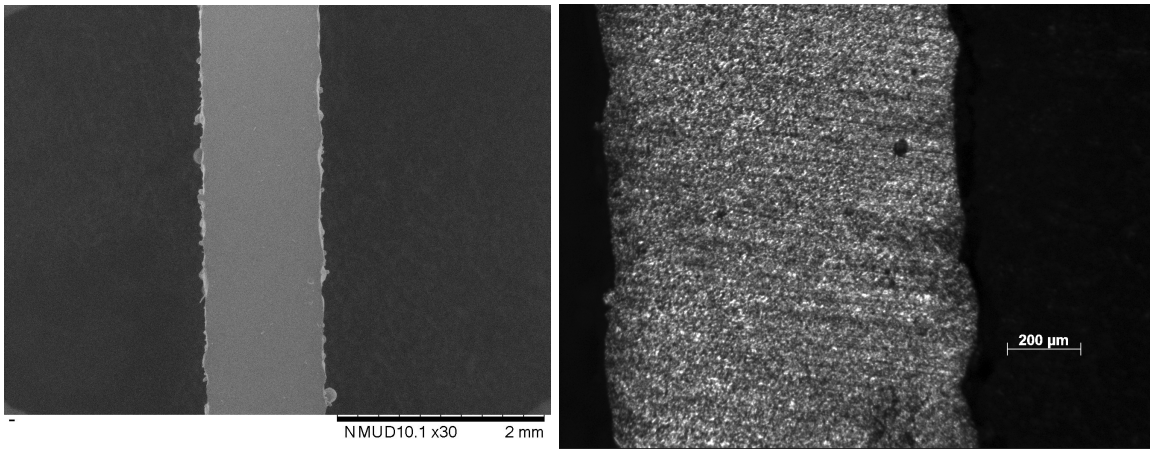

Figure 18: Example images from both types of microscope

8.3 Improvements

An improvement that could be made to the overall code would potentially be to combine both the methods together. The alternative method used could be modified slightly to produce graphs for the target width +/- 10% instead of just the target width. Then the area that is contained within all pairs of curves could be isolated, with the highest and lowest BC and CD in the area found and used for the upper and lower boundaries for the simulated annealing. This would then be able to produce optimum values that would work for each target measurement, instead of leaving not working as well for the 0.2mm value.

A quick visual representation of this can be found below, using graphs produced by my alternative method and Microsoft Paint:

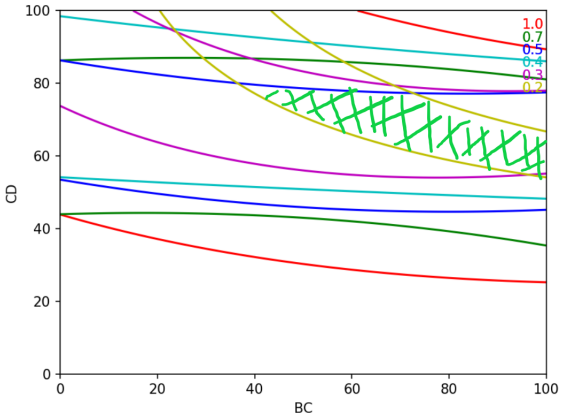

Figure 19: Quick visual representation of suggested method

For example, in the above image, the area shaded in green is contained between the set of curves produced for the target +/- 10% for each target. This area could then be used by the simulated annealing algorithm in the code for the boundaries, instead of zero to one hundred.

Further improvements could also be made to the gradient descent algorithm, more complex algorithms

and equations could be used to speed up the run-time and accuracy further. These changes would not have a big impact with the current sample size, however, if the sample size was to be extended, these may be necessary to reduce the run-time. A further change to the algorithm could be to change the equation to be cubic to see what impact that has on the output. This could lead to over-fitting however.

Validation methods would also potentially improve the algorithms hyper-parameters, however this could again lead to over-fitting. Validation could be carried out by leaving either one or a percentage of the data set out of the gradient descent algorithm, then comparing their values to the values that was predicted. This could be repeated a few times with various sets of hyper-parameters, with the hyper-parameters providing the best prediction to the validation set chosen.

Conclusion

In this project, we aimed to use machine learning to optimise laser powder bed fusion process parameters for dimensional tolerance. We have managed to achieve this goal, achieving optimum values for the appropriate parameters of beam compensation and contour distance. These values were obtained by utilising gradient descent combined with simulated annealing. The computer vision technique of canny edge detection was also successfully leveraged in order to quickly and accurately measure the widths of the different beams from test products produced with various combinations of values of parameters. Using computer vision methods in conjunction with machine learning is a novel approach to optimising the accuracy of an additive manufacturing product.

References

[1] Dilber Uzun Ozsahin et al. "The Systematic Review of Artificial Intelligence Applications in Breast Cancer Diagnosis". In: *Diagnostics* 13.1 (2022), p. 45.

[2] Chengcheng Wang et al. "Machine learning in additive manufacturing: State-of-the-art and perspectives". In: *Additive Manufacturing* 36 (2020), p. 101538.

[3] Wayne E King et al. "Laser powder bed fusion additive manufacturing of metals; physics, computational, and materials challenges". In: *Applied Physics Reviews* 2.4 (2015).

[4] H. Shipley et al. "Optimisation of process parameters to address fundamental challenges during selective laser melting of Ti-6Al-4V: A review". In: *International Journal of Machine Tools and Manufacture* 128 (2018), pp. 1–20. ISSN: 0890-6955. DOI: https://doi.org/10.1016/j.ijmachtools.2018.01.003. URL: https://www.sciencedirect.com/science/article/pii/S0890695518300233.

[5] Francesco Careri. "Collaboration Work Proposal". Presentation given by Francesco Careri for collaboration project.

[6] Chaolin Tan et al. "Laser Powder Bed Fusion of Ti-rich TiNi lattice structures: Process optimisation, geometrical integrity, and phase transformations". In: *International Journal of Machine Tools and Manufacture* 141 (2019), pp. 19–29. ISSN: 0890-6955. DOI: https://doi.org/10.1016/j.ijmachtools.2019.04.002. URL: https://www.sciencedirect.com/science/article/pii/S0890695519300392.

[7] Susheel Dharmadhikari, Nandana Menon, and Amrita Basak. "A reinforcement learning approach for process parameter optimization in additive manufacturing". In: *Additive Manufacturing* 71 (2023), p. 103556.

[8] Marc Ackermann and Christian Haase. "Machine learning-based identification of interpretable process-structure linkages in metal additive manufacturing". In: *Additive Manufacturing* 71 (2023), p. 103585.

[9] Mojtaba Khanzadeh et al. "Quantifying geometric accuracy with unsupervised machine learning: Using self-organizing map on fused filament fabrication additive manufacturing parts". In: *Journal of Manufacturing Science and Engineering* 140.3 (2018), p. 031011.

[10] Francesco Careri et al. "Additive Manufacturing of Heat Exchangers in Aerospace Applications: A Review". In: *Applied Thermal Engineering* (2023), p. 121387.

[11] Mourad Nasri and Mohamed Hamdi. "LTE QoS parameters prediction using multivariate linear regression algorithm". In: *2019 22nd conference on innovation in clouds, internet and networks and workshops (ICIN)*. IEEE. 2019, pp. 145–150.

[12] Priyanka Sinha. "Multivariate polynomial regression in data mining: methodology, problems and solutions". In: *Int. J. Sci. Eng. Res* 4.12 (2013), pp. 962–965.

[13] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[14] Ilya Sutskever et al. "On the importance of initialization and momentum in deep learning". In: *International conference on machine learning*. PMLR. 2013, pp. 1139–1147.

[15] Dilber Uzun Ozsahin et al. "Impact of feature scaling on machine learning models for the diagnosis of diabetes". In: *2022 International Conference on Artificial Intelligence in Everything (AIE)*. IEEE. 2022, pp. 87–94.

[16] Xing Wan. "Influence of feature scaling on convergence of gradient iterative algorithm". In: *Journal of physics: Conference series*. Vol. 1213. 3. IOP Publishing. 2019, p. 032021.

[17] Thomas Guilmeau, Emilie Chouzenoux, and Victor Elvira. "Simulated annealing: A review and a new scheme". In: *2021 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE. 2021, pp. 101–105.

[18] Kazuhiko Kimura, Seiji Kikuchi, and Shin-ichi Yamasaki. "Accurate root length measurement by image analysis". In: *Plant and Soil* 216 (1999), pp. 117–127.

[19] Rattapoom Waranusast, Pongsakorn Intayod, and Donlaya Makhod. "Egg size classification on Android mobile devices using image processing and machine learning". In: *2016 Fifth ICT International Student Project Conference (ICT-ISPC)*. IEEE. 2016, pp. 170–173.

[20] Ahmed Shihab Ahmed. "Comparative study among Sobel, Prewitt and Canny edge detection operators used in image processing". In: *J. Theor. Appl. Inf. Technol* 96.19 (2018), pp. 6517–6525.

[21] Weibin Rong et al. "An improved CANNY edge detection algorithm". In: *2014 IEEE international conference on mechatronics and automation*. IEEE. 2014, pp. 577–582.

[22] Ivan Culjak et al. "A brief introduction to OpenCV". In: *2012 proceedings of the 35th international convention MIPRO*. IEEE. 2012, pp. 1725–1730.

[23] Guo-Feng Liang et al. "Preventing human errors in aviation maintenance using an on-line maintenance assistance platform". In: *International Journal of Industrial Ergonomics* 40.3 (2010), pp. 356–367.

[24] Caitlin A Owen, Grant Dick, and Peter A Whigham. "Feature standardisation in symbolic regression". In: *Australasian Joint Conference on Artificial Intelligence*. Springer. 2018, pp. 565–576.

[25] Grant Dick, Caitlin A Owen, and Peter A Whigham. "Feature standardisation and coefficient optimisation for effective symbolic regression". In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 2020, pp. 306–314.

[26] Agnes Lydia and Sagayaraj Francis. "Adagrad—an optimizer for stochastic gradient descent". In: *Int. J. Inf. Comput. Sci* 6.5 (2019), pp. 566–568.

[27] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[28] Leandro L. Minku. "Simulated Annealing". Lecture given by Leandro Minku on simulated annealing.

[29] John Canny. "A Computational Approach to Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[30] Zhao Xu, Xu Baojie, and Wu Guoxin. "Canny edge detection based on Open CV". In: *2017 13th IEEE international conference on electronic measurement & instruments (ICEMI)*. IEEE. 2017, pp. 53–56.

[31] Hamid Dehghani. "Computational Vision, Lecture 2.2: Noise Filtering". Lecture given by Hamid Dehghani on Noise Filtering.

[32] *Open CV Canny edge detection*. https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html. Accessed: 09-09-2023.

[33] Vidushani Dhanawansa et al. "Comparative Study of Deep Learning Parameter Selection for Multi-Output Regression on Head Pose Estimation". In: *2022 IEEE International Conference on Industrial Technology (ICIT)*. IEEE. 2022, pp. 1–6.

[34] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. "Design of an image edge detection filter using the Sobel operator". In: *IEEE Journal of solid-state circuits* 23.2 (1988), pp. 358–367.

Appendix

Data

			Wall 1	Wall2	Wall 3	Wall 4	Wall 5	Wall 6
BC	CD	Nominal [mm]	1	0.7	0.5	0.4	0.3	0.2
0	0	test #1	1.15	0.85	0.64	0.59	0.47	0.36
0	25	test #6	1.13	0.82	0.68	0.54	0.46	0.38
0	50	test #11	1.05	0.73	0.53	0.39	0.32	0.24
0	75	test #16	0.98	0.65	0.45	0.41	0.29	0.27
0	100	test #21	0.99	0.60	0.42	0.34	0.27	0.15
25	0	test #2	1.24	0.89	0.72	0.62	0.49	0.38
25	25	test #7	1.08	0.83	0.65	0.52	0.40	0.33
25	50	test #12	1.07	0.72	0.55	0.42	0.31	0.23
25	75	test #17	0.97	0.62	0.42	0.43	0.31	0.22
25	100	test #22	0.93	0.55	0.40	0.33	0.35	0.24
50	0	test #3	1.17	0.85	0.67	0.59	0.49	0.32
50	25	test #8	1.08	0.77	0.60	0.46	0.41	0.32
50	50	test #13	1.06	0.73	0.53	0.39	0.35	0.22
50	75	test #18	0.95	0.67	0.41	0.36	0.28	0.17
50	100	test #23	0.88	0.60	0.38	0.31	0.25	0.14
75	0	test #4	1.21	0.85	0.66	0.57	0.43	0.43
75	25	test #9	1.10	0.80	0.61	0.52	0.42	0.30
75	50	test #14	1.20	0.83	0.60	0.45	0.40	0.21
75	75	test #19	0.93	0.65	0.44	0.34	0.27	0.15
75	100	test #24	0.86	0.56	0.38	0.32	0.21	0.12
100	0	test #5	1.16	0.84	0.67	0.56	0.44	0.40
100	25	test #10	1.12	0.80	0.60	0.50	0.40	0.31
100	50	test #15	1.03	0.70	0.52	0.43	0.35	0.21
100	75	test #20	0.92	0.64	0.46	0.33	0.33	0.13
100	100	test #25	0.85	0.54	0.38	0.33	0.19	0.13

Figure 20: Measurements provided by Francesco Careri

			Wall 1	Wall2	Wall 3	Wall 4	Wall 5	Wall 6
BC	CD	Nominal [mm]	1	0.7	0.5	0.4	0.3	0.2
0	0	test #1	1.17	0.87	0.67	0.56	0.49	0.38
0	25	test #6	1.17	0.85	0.66	0.53	0.44	0.33
0	50	test #11	1.09	0.77	0.57	0.46	0.35	0.32
0	75	test #16	0.99	0.62	0.44	0.39	0.31	0.21
0	100	test #21	1.01	0.60	0.44	0.36	0.31	0.27
25	0	test #2	1.25	0.91	0.70	0.61	0.51	0.39
25	25	test #7	1.10	0.81	0.61	0.51	0.42	0.32
25	50	test #12	1.12	0.80	0.59	0.47	0.35	0.33
25	75	test #17	0.94	0.64	0.43	0.35	0.30	0.22
25	100	test #22	0.94	0.57	0.40	0.40	0.31	0.28
50	0	test #3	1.18	0.87	0.68	0.58	0.49	0.38
50	25	test #8	1.10	0.80	0.62	0.53	0.42	0.32
50	50	test #13	1.12	0.79	0.59	0.49	0.39	0.34
50	75	test #18	0.96	0.66	0.42	0.36	0.25	0.15
50	100	test #23	0.89	0.61	0.39	0.30	0.21	0.12
75	0	test #4	1.19	0.87	0.68	0.58	0.47	0.40
75	25	test #9	1.11	0.81	0.62	0.53	0.44	0.34
75	50	test #14	1.00	0.71	0.52	0.39	0.32	0.25
75	75	test #19	0.94	0.66	0.41	0.38	0.26	0.14
75	100	test #24	0.89	0.58	0.38	0.36	0.21	0.13
100	0	test #5	1.15	0.85	0.66	0.57	0.48	0.40
100	25	test #10	1.15	0.84	0.64	0.55	0.44	0.35
100	50	test #15	1.00	0.71	0.53	0.43	0.34	0.20
100	75	test #20	0.92	0.65	0.46	0.33	0.31	0.15
100	100	test #25	0.91	0.56	0.40	0.38	0.22	0.12

Figure 21: Measurements obtained from computer vision algorithm

GitLab Repository

The code developed in this project can be found using the link https://git.cs.bham.ac.uk/projects-2022-23/txa236.

The repository includes:

- Images: A folder containing all the images used.
- Resources: A folder containing book1.csv, which is the original data obtained from human measurements, and book2.csv, containing the data obtained from the running the vision algorithm.
- GradientDescentAnnealing.py: This contains the code for the gradient descent and simulated annealing portions of the project
- 2DsliceGradientDescent.py: This contains the code for the alternative method of obtaining 2D slices from the 3D graphs generated by gradient descent
- Vision.py: This contains the code for the vision part of the project.

The main code to be ran is found in the "GradientDescentAnnealing.py" file. When ran, the code will ask if you want to run the vision code or not, this means that it does not need to be ran each time if its already been ran once. Next, it will ask whether you want to use the initial human measurements, or the measurements obtained by the vision analysis. Lastly, it will ask what the target width that you want to optimise for is. Then the code will run and generate optimum values and a graph of the results.