SCAN STRATEGY INTERPOLATION FOR LASER POWDER BED FUSION IN MANUFACTURING APPLICATIONS

A Thesis
Presented to
The Academic Faculty

By

Patrick V. Jung

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology

May 2021

COPYRIGHT © 2021 BY PATRICK V. JUNG

SCAN STRATEGY INTERPOLATION FOR LASER POWDER BED FUSION IN MANUFACTURING APPLICATIONS

Approved by:

Dr. Christopher Saldana, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Katherine Fu, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Thomas Kurfess
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: January 21, 2021

ACKNOWLEDGEMENTS

I would like to thank Dr. Christopher Saldana, for his advice and mentorship throughout this effort. My learning and understanding have accelerated due to his guidance driven by experience in this field of research. I would also like to thank Dr. Fu and Dr. Kurfess for their continuous support through this process and for the time and expertise they provided to assist this work. I would like to thank the other cohort members for their support and for creating a positive work environment. Finally, I would like to thank my girlfriend Lauren for the years of laughter and encouragement she has provided me, without which I could not have completed this work.

This work was supported in part by the Department of Energy DE-EE0008303.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

AM	Additive Manufacturing
AMMT	Additive Manufacturing Metrology Testbed
COV	Coefficient of Variance
DED	Directed Energy Deposition
EBM	Electron Beam Modeling
ED	Energy Density (J/mm^2)
G-Code	Preparatory Code Language for Automated Machines
h	Hatch Spacing
$\bar{I}$	Mean of Inputs I
IED	Input Energy Density (J/mm^3)
IN625	Inconel nickel-chromium alloy 625
IR	Infrared
ISO 50	Threshold at 50% Between Two Histogram Peaks
k-NN	k-Nearest Neighbors
L-PBF	Laser Powder Bed Fusion
ML	Machine Learning
P	Power (Watts)
PBF	Powder Bed Fusion
SLS	Selective Laser Sintering
STL	Standard Tessellation Language
T	Camera Trigger (Boolean)
V	Velocity (mm/s)
XYPT	Cartesian Coordinates, Power and Camera Trigger
σ	Standard Deviation
μm	Micrometers

SUMMARY

Laser Powder Bed Fusion (L-PBF) is a technique within additive manufacturing which uses a high power density laser to build parts from fused powdered metal alloy. This technology is well equipped to produce complex parts with otherwise impossible features such as hidden voids or lattice structures. Alongside capability, reliability and quality are key characteristics considered when choosing a manufacturing method, and these are gaining attention as this method becomes more prevalent in industry. One main indicator of a stable L-PBF process is consistent melt pool geometry, the properties of which are likely to determine the quality of the part produced. As computing power and sensing technologies become more advanced, this melt pool geometry could be studied in real time. Therefore, the loop could be closed on process monitoring in order to achieve optimal quality. This work addresses that challenge by capitalizing on machine learning techniques to reduce the latency between sending process commands and receiving process validation.

This thesis presents a novel application of a k-nearest neighbor (k-NN) model to identify key parameters within melt pool imagery and predict significant process parameters. The k-NN model was trained on data provided by the National Institute of Standards and Technology (NIST). This approach was used to accurately infer the energy density of unseen layers within the same part. The algorithm was subsequently tested with unique scan strategies and found to reasonably estimate the process parameters of different parts. A 5-fold cross validation found the algorithm to be consistently predicting the class of 95.42% of the in-situ melt pool images.

CHAPTER 1: INTRODUCTION

Research in automating the process level of manufacturing operations has been conducted, in both academia and industry, over the past few decades. This work is motivated by a strong belief that research in this area will provide increased productivity, improved part quality, reduced costs, and relaxed machine design constraints. The basis for this belief is two-fold. First, manufacturing process automation can be applied to both large batch production environments and small batch jobs. Second, process automation can autonomously tune machine parameters (velocity, power, scan strategy, etc.) on-line and off-line to substantially increase the machine's performance in terms of part tolerances and surface finish, operation cycle time, etc.

Process automation holds the promise of bridging the gap between product design and process planning, while reaching beyond the capability of a human operator. The success of manufacturing process automation hinges primarily on the effectiveness of the process monitoring and control systems. Particularly for metal additive manufacturing (AM), successful process control with precision measurement science will not only extend the capabilities for this technology, but also introduce reliability to advanced manufacturing methods which have not yet been widely incorporated into production environments. The most widely used metal AM technique in the commercial industries is Laser-Powder Bed Fusion (L-PBF), a method for melting thin layers of metal powder with a high-powered laser, recoating the build area with powder, and repeating this process until a dense part is completed.

A promising future is in store for L-PBF AM. However, widespread adoption of L-PBF with metallic parts hinges on solving a main challenge: the requirement that the final product should meet engineering quality standards. This includes increasing the predictability of melt pool geometry, since the melt pool has a strong impact on the final mechanical properties. Modeling advances on this front typically rely on physics-based simulation, which are available commercially, but most require a deep understanding of the process involved, as well as full access to the input parameters and machine specifications [16]. An attractive alternative to answering this challenge is through machine learning and predictive simulation. This alternative solves the problem as a "black box," and only follows the patterns of the inputs and outputs, without focusing on the underlying physics. Such an approach requires a high volume of data to study from and subsequently validate its performance, which is prohibitive in niche areas where extensive research is not being published. The physics-based models however only require enough data to evaluate accuracy, but the aforementioned basic research becomes the largest hurdle that requires time and resources to surpass [38-40]. This work focuses on improving accessibility of quality assurance to lower budget commercial applications, and the cost for physics based modelling may be prohibitive for the end user in that situation. Therefore, the inexpensive machine learning approach which utilizes open source algorithms is the path explored in this thesis to better understand its novel application to predicting machine parameters using in-situ monitoring.

1.1 Motivation

The shape of a melt pool is significantly affected by laser power and scan velocity, and melt pool formation has been shown in recent studies to determine material properties. An understanding of the fundamental science behind these relationships is currently being developed in the literature. However, an intermediate process such as melt pool formation is challenging to model because it includes several complex physical phenomena including thermal conduction, fluid dynamics, and phase changes of materials [9,16]. Machine Learning (ML) provides a different path to correlate the process parameters and results without the need for advanced modeling, and it has been shown to be effective in additive manufacturing applications [19].

1.2 Problem Statement

The objective of this work is to identify key features in monitoring the melt pool of L-PBF AM systems, and to use those features in a supervised machine learning algorithm to predict key input process parameters. An understanding of how melt pool images can predict AM machine commands or scan strategy to explore near closed loop control or anomaly detection is not well presented in the literature. To complement the current work being done to predict melt pool geometry from velocity and/or power, k-nearest neighbor (k-NN) models were trained and evaluated for interlayer prediction and transferability to other parts. Models were trained on data and images from the National Institute of Standards and Technology (NIST) and their Additive Manufacturing Metrology Testbed (AMMT) [1].

1.3 Structure

The following chapters are structured as follows. Chapter 2 will begin with a background of additive manufacturing while specifically focusing on laser powder bed fusion. A brief introduction to machine learning will also be presented, and the intersection of these two subjects will be discussed as it currently is addressed in the literature. Chapter 3 will include the methodology used in developing the training and testing matrices from the experimental dataset presented in [1]. The pre-processing of the in-situ monitoring images will be reviewed as well as the flow of information into the k-NN classification algorithm. In Chapter 4, the results of interlayer and transferability analysis will be presented. The k-fold validation will also be included with some inferences about the results. A further discussion is held in Chapter 5 regarding the challenges faced along with the assumptions made. Chapter 6 delivers the conclusions made and postulates future possibilities for improvement in this area.

CHAPTER 2: BACKGROUND

Additive manufacturing (AM), instead of the traditional subtractive technologies, is a process of adding material to make near-net shape parts directly from 3D models [8]. This technology has been rapidly developed for many applications requiring advanced manufacturing in sectors such as aerospace. AM is comprised of techniques involving a range of materials and capabilities spanning a wide distribution of machine cost.

2.1 Laser Powder Bed Fusion

Powder bed fusion, one of the seven AM techniques defined by ASTM F2792, uses a thermal energy source such as a high-powered laser to melt metal powder to the base plate or previous layers [36]. A standard machine set-up is displayed in Figure 1 below.

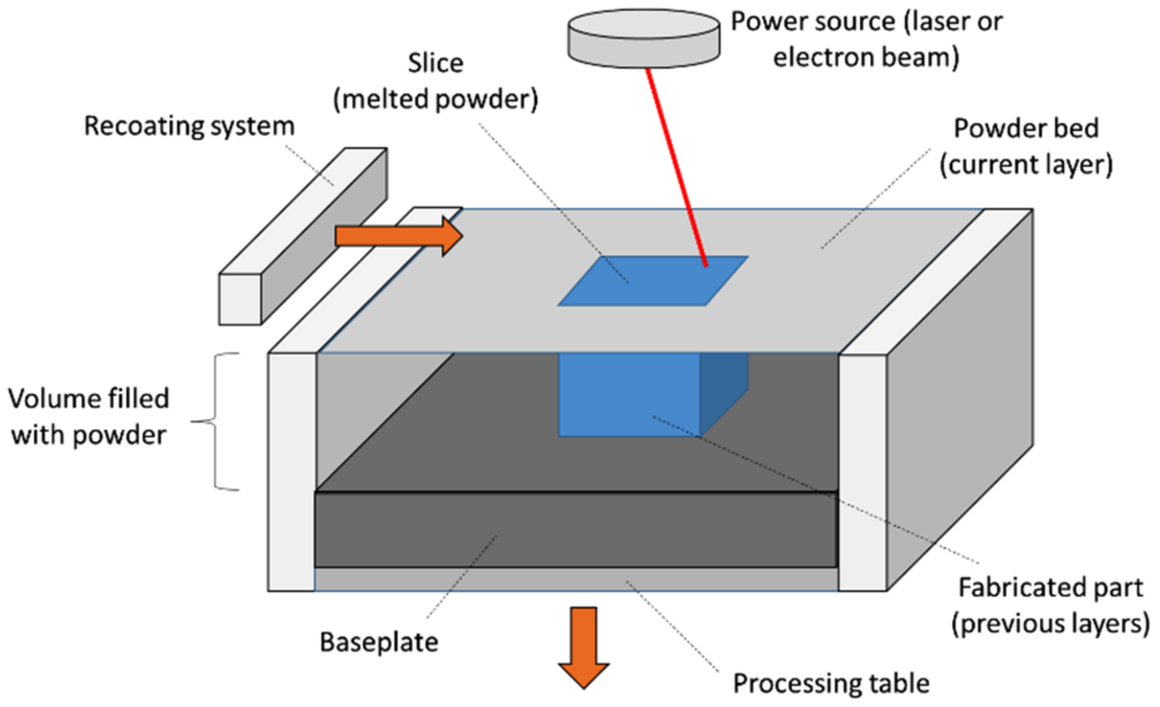
Power source (laser or electron beam)
Slice (melted powder)
Recoating system
Powder bed (current layer)
Volume filled with powder
Fabricated part (previous layers)
Baseplate
Processing table

Figure 1: The powder bed fusion process [35]

Most metal additive manufacturing in commercial applications today falls under powder bed fusion. This technique is particularly well suited for applications which require better precision and accuracy but can spare more time per build. Powder bed fusion is the parent technology of electron-beam melting (EBM) and laser-powder bed fusion (L-PBF). The main difference between the two technologies is the thermal energy source used to heat the metal powder to melting temperature. This work focuses on L-PBF, which is typically the more cost-effective of the two, and is the most widely used metal additive process in the field [37].

The process of L-PBF is executed as follows: First, the build area is filled with a layer of metal powder and the material is evenly distributed with a recoating system. A galvanometer scanner orients the energy source, such as a laser, and transfers thermal energy to the top layer of material through a predetermined scan strategy to add a layer typically 30-50 microns in thickness to the part. The build platform is then lowered by this thickness without removing the excess metal powder, and more powder is dispensed to recoat the build area for the next layer. The laser is oriented with great precision in the XY plane, has a focus diameter of less than five thousandths, and with a layer thickness less than half of a sheet of paper's, features can be built with high fidelity [35]. This process typically takes place in an environment filled with nitrogen, argon, or another inert gas to keep the solidified layers from oxidizing after they have been melted. Once the final layer has been fused to the part, the part will need to be removed from the build area and from its encapsulation of packed metal powder.

Numerous layers are fused together to create a part with near 100% density through a three-step process: thermal energy absorption, melt pool formation, and solidification

[10]. This work focuses on the second step of this process, melt pool evolution, which is significantly influenced by process parameters such as energy density and laser scan strategy [5, 26]. Several physical phenomena are at play as well during this process, including fluid mechanics, heat transfer and phase transformations. Figure 2 below illustrates a physics-based modelling approach to understanding the complex interactions happening throughout this fusion process.

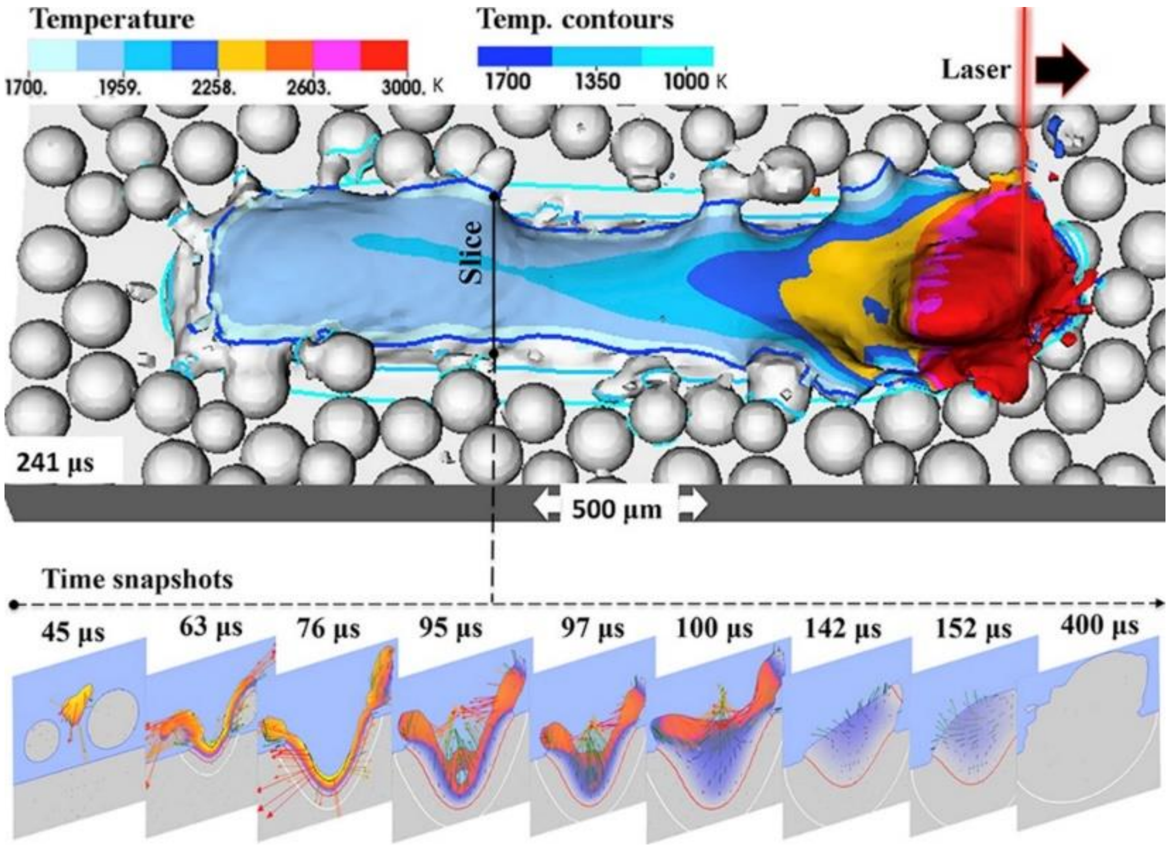

Figure 2: Example path of a laser in powder bed fusion [16]

2.2 Influence of Melt Pool Geometry

Within the process of Laser-Powder Bed Fusion, the characteristics of the melt pool have been shown to influence the part's microstructure and the resultant material properties [30]. The melt pool geometry, as is directly influenced by energy density, can be controlled to improve part quality as well [2, 4]. This could be partly due to the fact that energy density (J/mm^3) has been shown to be correlated to part density, as seen below in Figure 3. The effects of energy density parameters, scan speed and laser power, on material properties have been studied by many in the literature [42-46].

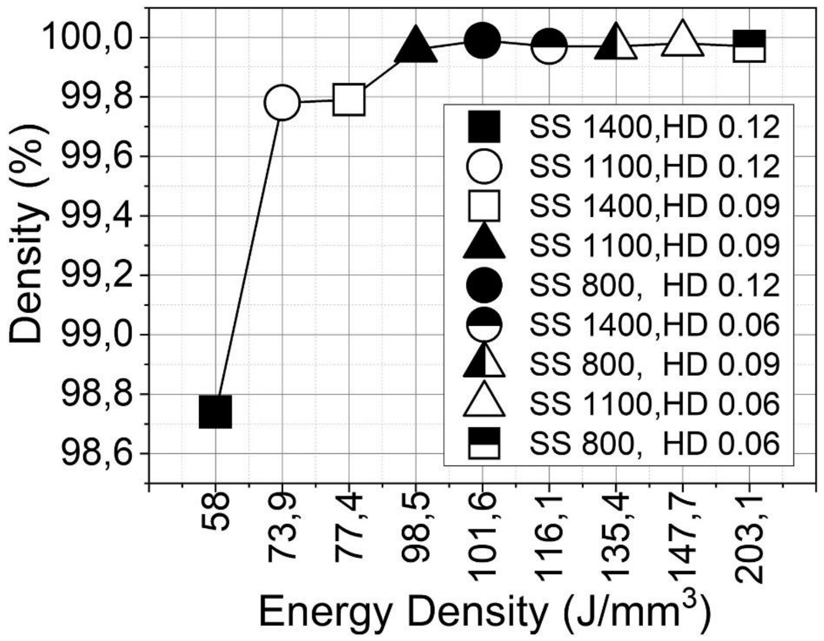

Figure 3: Energy density vs. part density in selective laser sintering (SLS) [45]

The underlying effects from reduced energy density manifest primarily as an increased level of porosity in the part, which in turn weakens the mechanical properties of the affected area by introducing stress concentrators during manufacturing in the form of external pores, voids, and micro-cracks [57]. Parts with 100% relative density on the other hand have been experimentally proven to mimic the material properties of wrought materials, if not surpass them. It has been shown that poor energy density may cause this because of incomplete fusion or trapped gases within the part [16]. At higher energy densities, the pores identified were typically smaller in size and were found to be more regular and even in their spherical shape. Porosity increases are essentially being linked to improperly formed melt pools, which emphasizes the importance of this aspect for PBF [6]. With energy density controlling melt pool geometry, and melt pool geometry heavily influencing porosity and subsequently the strength of the material, the melt pool acts as an intermediate step in the efforts to predetermine the properties of the resultant part.

In order to intelligently control melt pool geometry, a well-defined relationship between process parameters and melt pool dimensions will need to be established. Figure 4 below illustrates the typical geometric boundaries that a melt pool shape is defined by, including length, width, depth, and hatch space *h*.

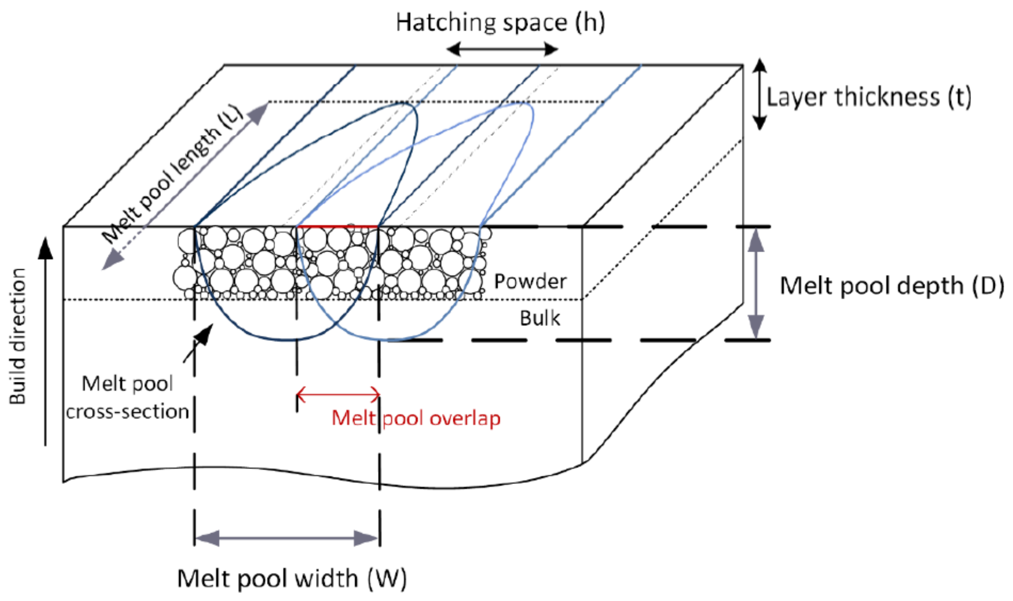

Figure 4: Melt pool geometry in L-PBF [31,32]

Figure 5 below displays some experimental data presented by Guo where energy densities were kept constant for three different sets of experiments focused on the relationship between melt pool dimensions and the individual parameters which make up energy density. The increasing range on the vertical axes indicate the correlation between energy density and melt pool volume overall, but within each energy density tier there is another linear relationship between melt pool dimension and the process parameters. These two correlations compound upon one another, forcing the models of this problem into the nonlinear optimization space.

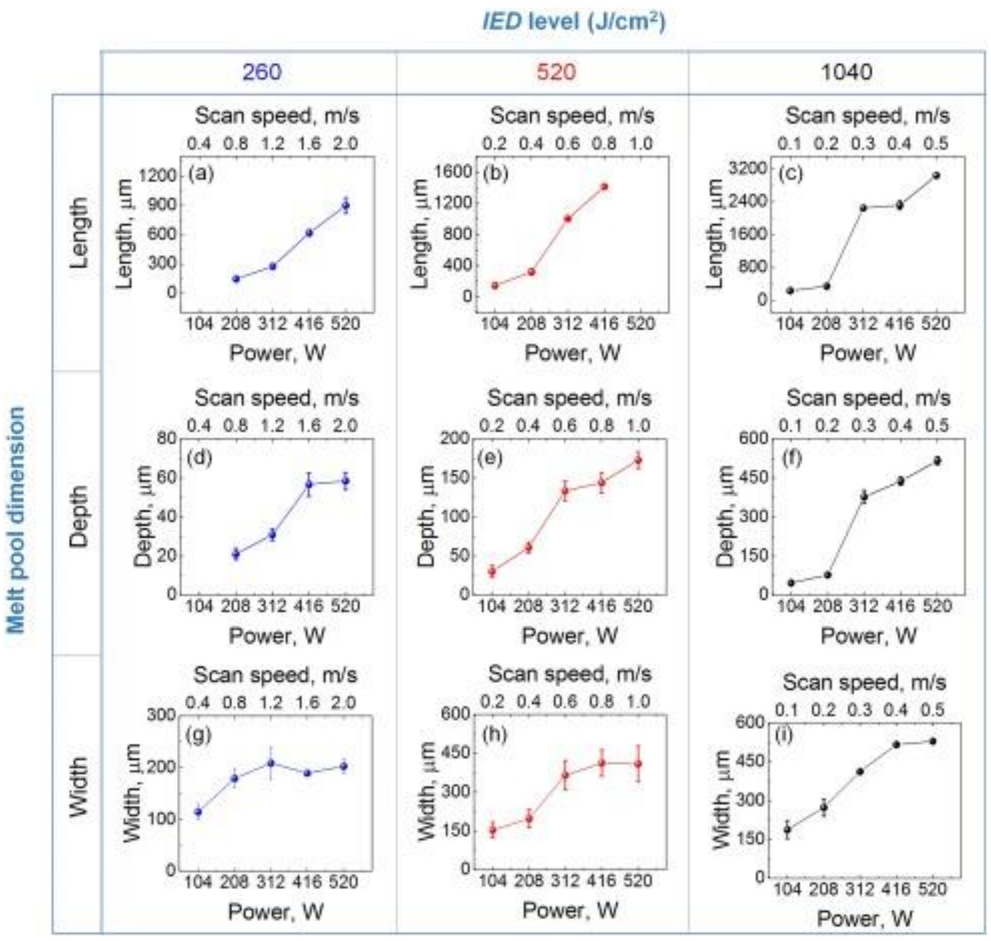

Figure 5: Change in melt pool size within constant input energy densities (IED) [58]

Despite increases among energy density and melt pool volume being correlated, it is difficult to linearly predict when the process is open to influence from uncontrollable or unexpected factors [33]. Even if the surrounding environment is controlled, there is a possibility of variation in melt pool size as a build progresses, and defects can grow to cause build failure if the signs of a defect are not caught early. Open loop control would likely be inadequate when quality and/or complexity is highly valued. As computing power becomes cheaper, including closed loop control in additive processes is becoming more accessible, and algorithms need to be developed to address this issue.

2.3 *In-situ* Process Monitoring

The necessity of high feature resolution has encouraged more interest in the development of sensor technologies to monitor laser-powder bed fusion, and defect avoidance has become critically important. The National Institute for Standards and Technology (NIST) has outlined measurement science needs for metal additive manufacturing, and these include qualification and certification of the parts and processes that make up powder bed fusion, the part's mechanical properties, and the observation of machine performance [34]. Closed-loop control in metal AM has subsequently been identified as an important focus for the scope of measurement science work, as robust process control *in-situ* can reduce outcome variability and increase the quality of parts produced by these means.

Closed-loop control has been investigated before in a similar context for the process of directed energy deposition (DED) by monitoring the infrared (IR) radiation emitted by the melt-pool [56]. The results of this study were encouraging, as the microstructure became more homogeneous with the addition of a PID controller connected to the laser power responding to changes in temperature. Below in Figure 6 is an example of the visual effects closed loop control can have on metal additive manufacturing.

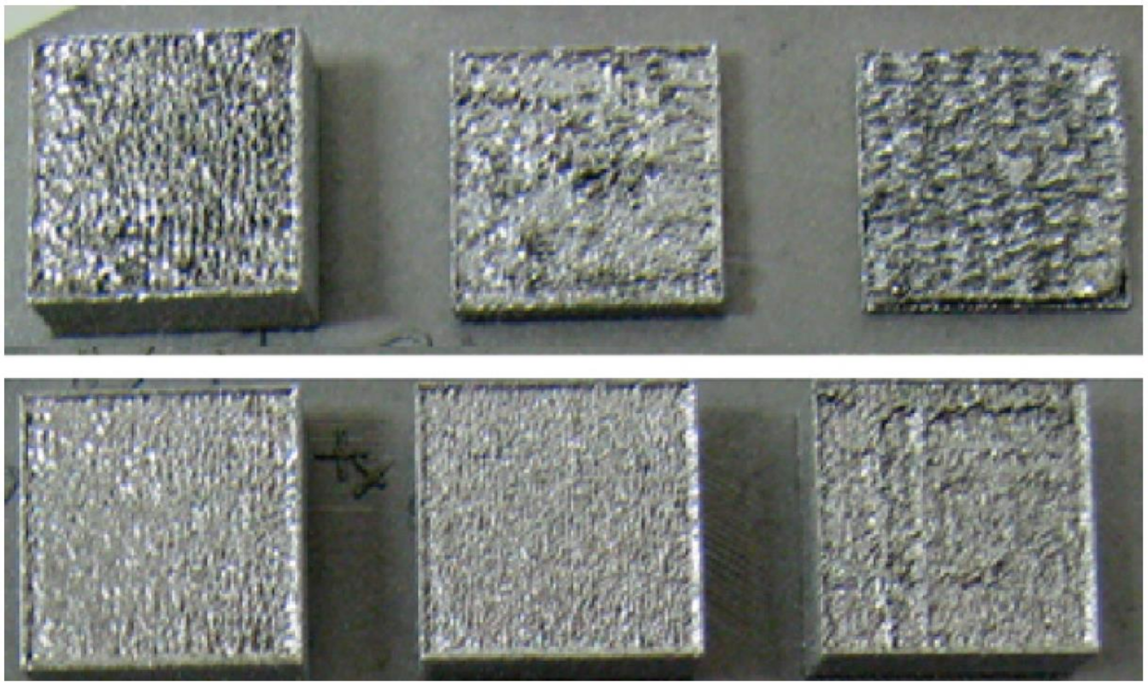

Figure 6: Built cubes without (Top) and with (Bottom) feedback control

2.4 Machine Learning

Machine learning algorithms have been introduced as a revolutionary addition to additive manufacturing, as they increase the probability of success in high-value and complex parts [10,14]. ML algorithms are oriented around prediction and interpolation within a large dataset where it is advantageous to discover patterns. Some algorithms, called unsupervised learning methods, do not even require labels for their data points as they simply search for natural subsets of data. If the expected data classification or response vector is available, then a supervised learning algorithm is preferred.

ML algorithms can be used in a monitoring and control context due to the low processing power needed to predict new values. As an additional benefit, ML models typically improve over time as the pattern boundaries automatically shift to optimally incorporate new data. Therefore, these algorithms are suitable for interpreting parameters in processes where the environment and part geometry may vary over time. With respect to AM, machine learning is already used to recognize issues and predict the quality of parts [20, 22].

2.4.1 k-Nearest Neighbor (k-NNs)

The k-nearest neighbor algorithm is a supervised classification algorithm that seeks to make predictions on a sample data point by determining the k nearest points by Euclidean distance. The most frequent class label among the k nearest neighbors is applied to the sample data point. In Figure 7, a k-NN algorithm of k equal to 5 is shown. The data point denoted by X_u is assigned the class label of the red circles due to majority vote.

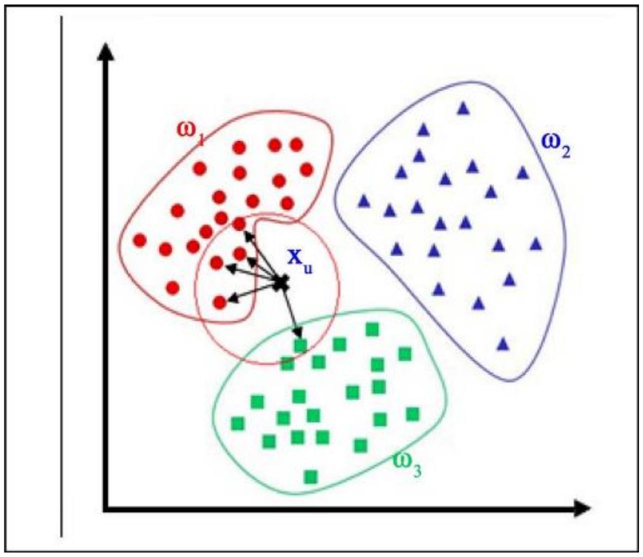

Figure 7: A k-NN algorithm with 3 class labels and k = 5

2.4.2 k-Fold Cross Validation

Cross validation of an ML algorithm is helpful to minimize bias and assess the effectiveness of a model such as k-NNs. For this process, a known dataset is split into several bins of data, and the number of bins is represented by the variable *k*. The number of folds (equal to the number of bins) indicates how many iterations the algorithm will be trained and tested on a unique dataset formed by iteratively choosing the test dataset *i*, which contains only one bin of data, across the number of folds. This process is used in practice to evaluate a model using *k* permutations of the same known dataset. The number of permutations can increase as the size of the dataset increases. Figure 8 below illustrates the typical k-fold validation process with 5 folds.

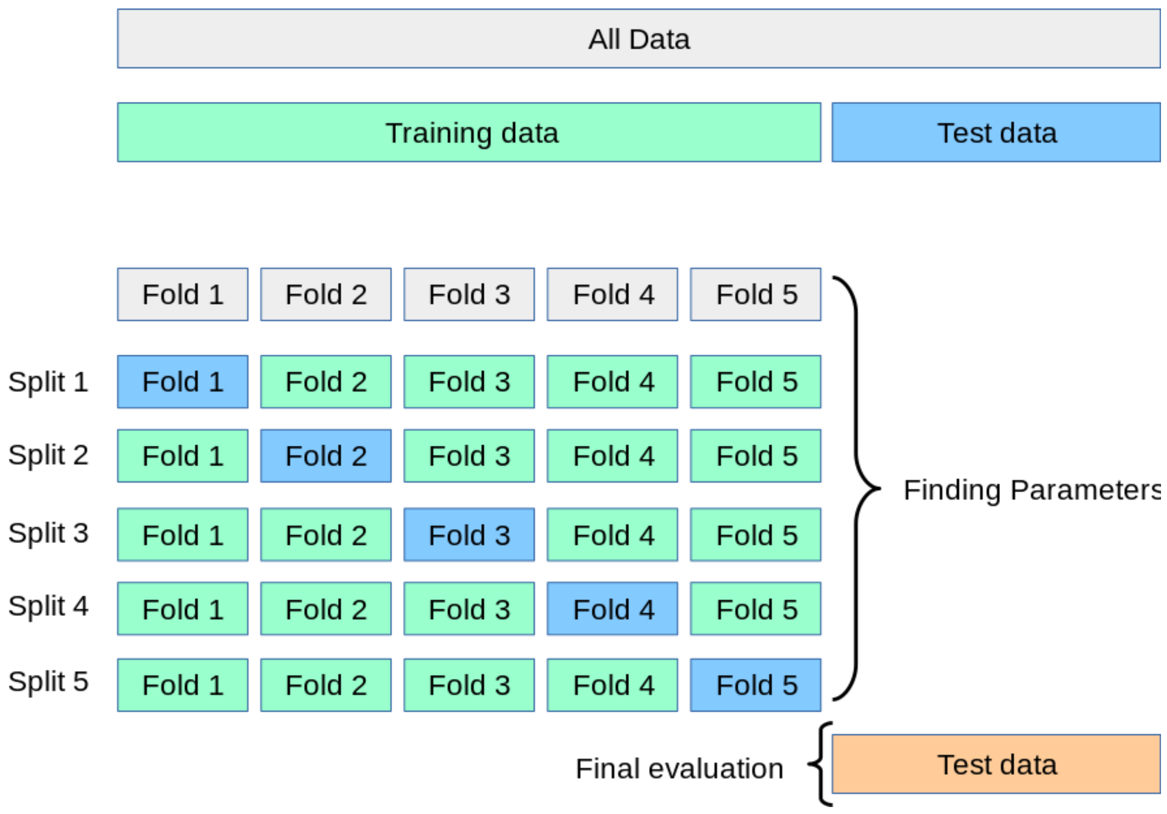

Figure 8: A standard k-fold validation procedure where k = 5. [59]

CHAPTER 3: METHODOLOGY

The goal of this work is to develop a prediction method for the process parameters of laser powder bed fusion systems in additive manufacturing. The significant process parameters that affect melt pool geometry, scan velocity and laser power, are more accurately represented in their effects by looking at energy density. It can approximated with the following:

Equation 1:

$$ED \sim \frac{P}{V}$$

Instead of predicting multiple dependent variables, this method rather delivers an estimation of the largest influence on melt pool size and requires less computing power. The faster method is desirable in this application for low-cost additive manufacturing prediction.

3.1 Data Interpretation

The data studied consists of input command files, in-situ process monitoring data, and metadata provided by the National Institute of Standards and Technology who performed a build with the Additive Manufacturing Metrology Testbed (AMMT) [1]. Figure 9 below illustrates the AMMT system schematic. Twelve parts, identical in shape, were built out of IN625 on the same build platform which consisted of hot rolled and annealed wrought nickel alloy. The standard tessellation language (STL) file, as illustrated below in Figure 10, represents the near rectangular prism that was built for this dataset.

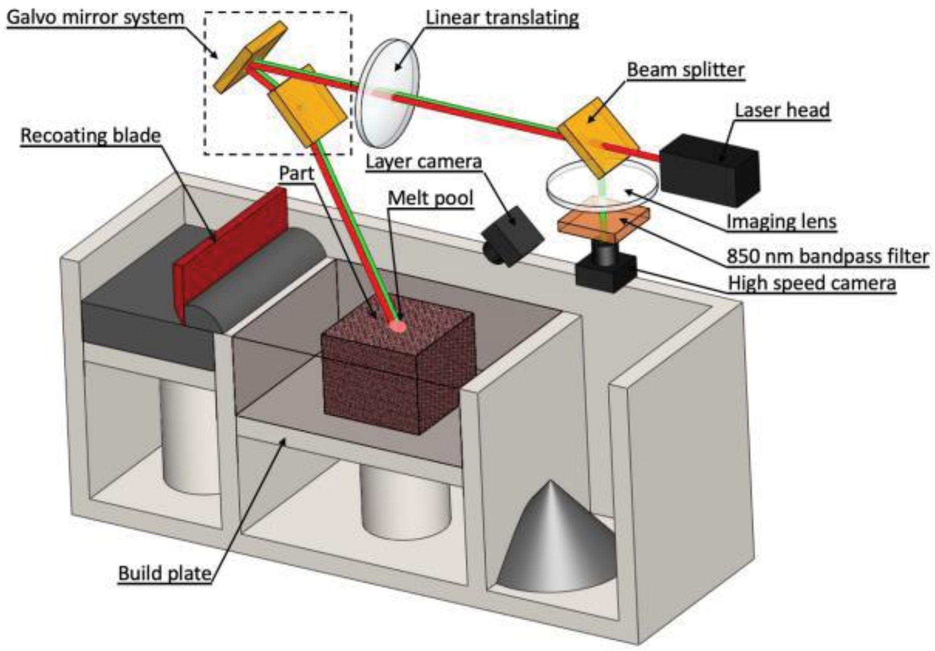

Figure 9: AMMT system setup [24]

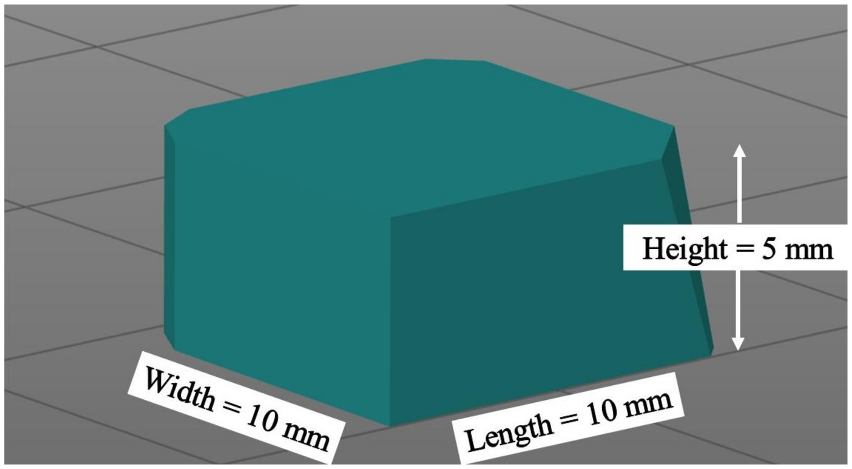

Figure 10: Graphical representation of studied part in STL format

None of the twelve parts varied in geometry, but rather they each had a unique scan strategy, and some of the differences can be observed in the snapshot below, Figure 11. The first category of information within the dataset provided by NIST, the build command data, can be used to highlight these differences, and there were two types of such data provided: AM G-Code and XYPT commands, where XY indicates the cartesian position (in mm) in the current layer, P represents power (in Watts) of the laser, and T represents the binary indication of capturing melt pool images. In this work, the XYPT commands were used because of the interest in directly comparing programmed position and power to the second category of information, the in-situ measurement data. This second category of information has two subcategories: co-axial melt pool images and grayscale layer images. From these, the melt pool images were more important to this work because their capture could be directly placed in the context of the location and power of the scanning laser.

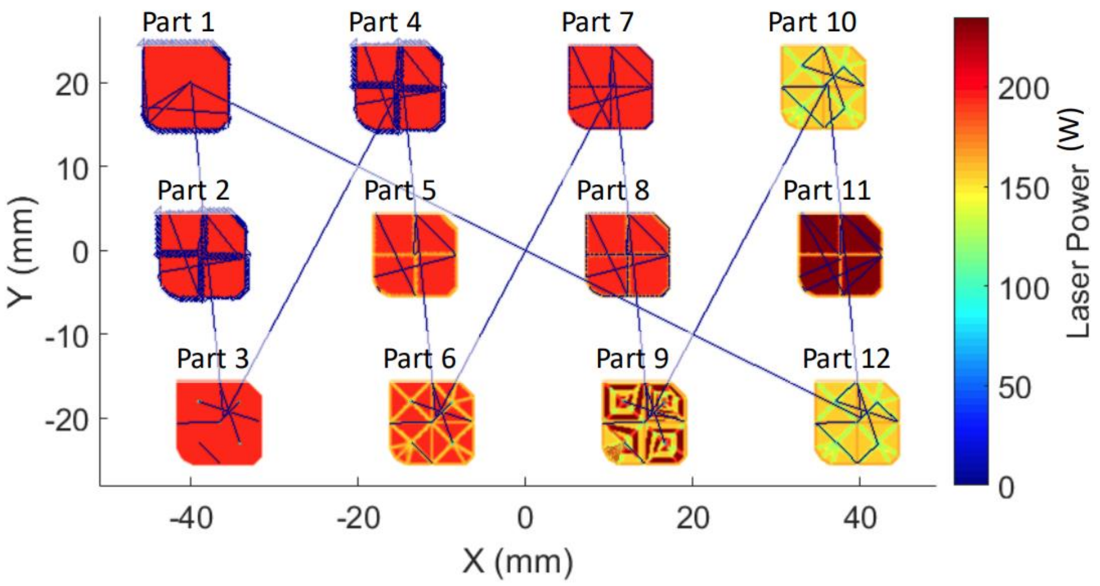

Figure 11: Unique scan strategies of the twelve parts [1]

The melt pool camera captured thousands of images per layer, and altogether this build required 250 layers each 20 µm thick, with a laser spot diameter of 85 µm. The images in this subcategory, all 120 pixels X 120 pixels, focused on a single part out of the twelve built during any layer. For example, during layer 2, the in-situ melt pool camera was only triggered when building part 2, and similarly, images from layers 1, 13, 25, 37, etc. (~20 equally distributed layers in total) were only attributed to part 1. Collecting a training set of images for studying a certain part was then approached by filtering out the other layers.

3.2 Data Preprocessing

The XYPT commands and melt pool images were studied layer by layer and, within each layer, frames 1-n were individually considered for viability in the training set. First, the grey image was binarized based on the ISO 50 method, where any pixel with greater intensity than 50% of the maximum intensity would be counted as a true value and all else were considered to be a Boolean 0 or false. Once binarized, there were several conditions which had to be met in order for an image to be included: The shape made up of Boolean 1 values must have one and only one centroid, and it must also be larger than 40 pixels squared in area. That said, if a secondary shape in the image is recognized and it is smaller than the required area, that shape was removed and the primary shape was not considered invalid. These cases are represented by Figure 12 below.

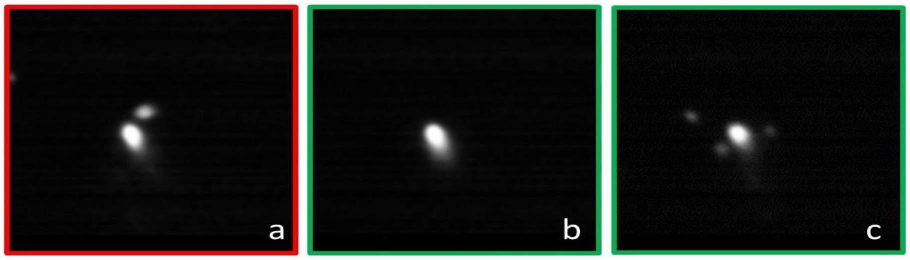

Figure 12: Image viability – (a) Images with two objects not considered (b) Typical melt pool image studied (c) Small, low intensity melt pools were removed

Once an image was classified as viable, features of the image were extracted to increase the speed and efficiency of a k-NN algorithm, including descriptive statistics of the melt pool area, dimensions of the melt pool shape, and centroid locations. Two centroid locations were found in each typical melt pool image, that of the shape greater than the ISO 50 threshold, and the centroid of the melt pool *tail*. The tail is defined in this work as the collection of pixels that lie between the ISO 50 and a similarly obtained 35% threshold. In order to extract this information, the original image was again binarized with these new limits, but no size requirements were imposed on the tail section. The process of binarization and isolation of the necessary features is illustrated below in Figure 13.

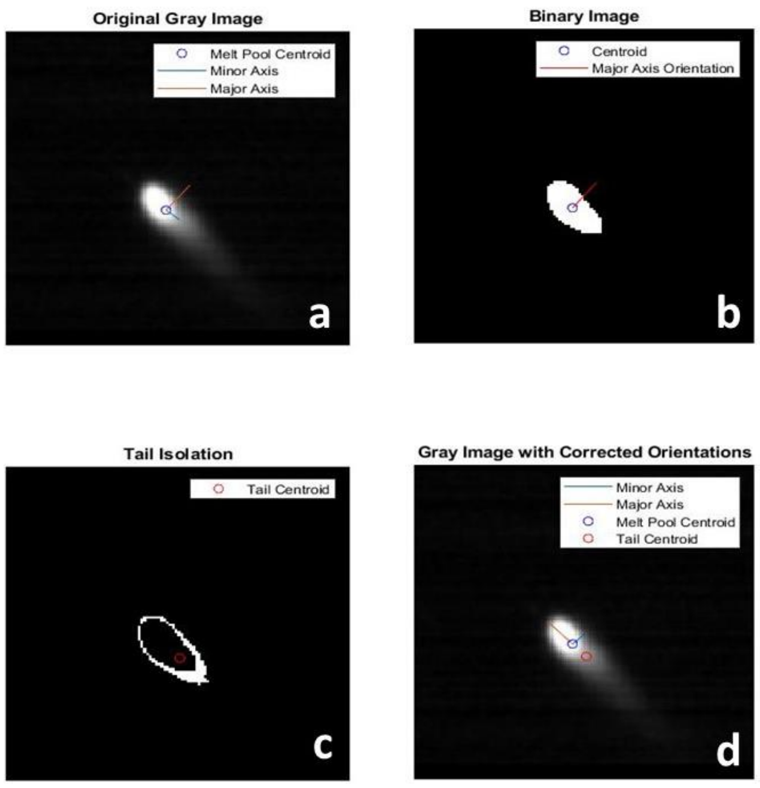

Figure 13: Process of image feature extraction – (a) original gray image with Matlab estimated orientation (b) binarized image of pixels > ISO 50 (c) binarized image to isolate melt pool *tail* (d) new estimated orientation superimposed upon original gray image

The original gray image was initially characterized by Matlab's *regionprops* function in the Image Processing Toolbox, because most of the melt pool images represented an elliptical shape. The centroid of the binarized shape and the *regionprops* estimation of the elliptical axes are superimposed on the original gray image in Figure 13(a). The same major axis estimation is carried over to the binary image in Figure 13(b). In Figure 13(c), the binary image of the isolated tail is shown, and a separate centroid is calculated from the new shape. In Figure 13(d), the culmination of these efforts is displayed, as the line of best fit through both centroids is taken to be the correct major axis orientation, while the original major and minor axis lengths are kept. These modified orientations give the k-NN algorithm better context and could allow for path planning prediction in future work. However, the in-situ monitoring camera was not aligned with the coordinate system of the substrate, and thus the orientations found using this method must once again be manipulated with a transformation matrix to be reconciled with the build command data.

Some of these independent variables can be combined or manipulated to give the model more relevant information, defined here as *derived variables*, such as Coefficient of Variance, which is calculated with the equation:

Equation 2:

$$COV = \sigma/\bar{I}$$

Figure 14 below illustrates the flow of information from the datasets to the k-NN classification algorithm to predict the process parameters velocity, power or energy density.

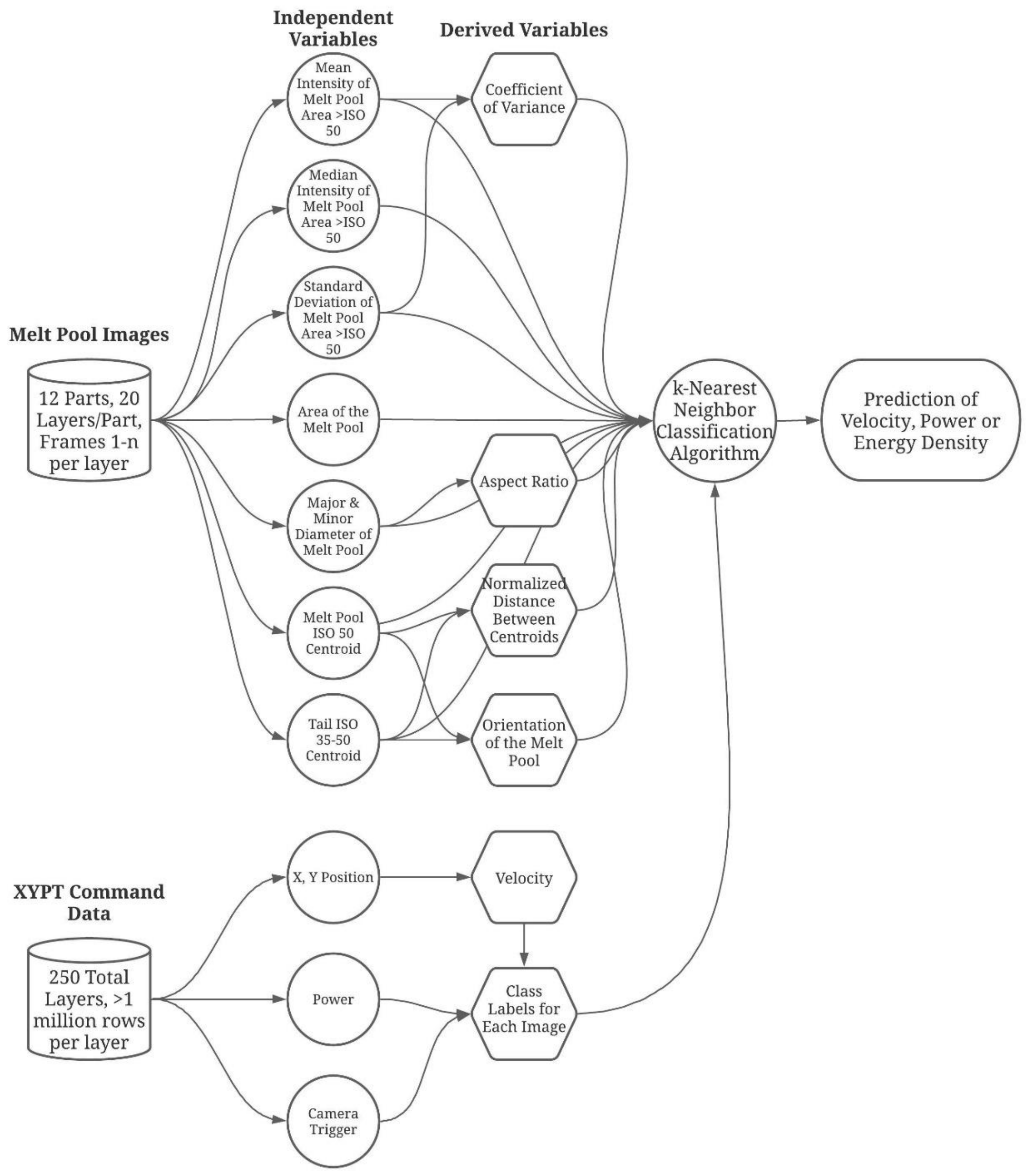
Independent Variables
Derived Variables
Melt Pool Images
12 Parts, 20 Layers/Part, Frames 1-n per layer
Mean Intensity of Melt Pool Area >ISO 50
Median Intensity of Melt Pool Area >ISO 50
Standard Deviation of Melt Pool Area >ISO 50
Area of the Melt Pool
Major & Minor Diameter of Melt Pool
Melt Pool ISO 50 Centroid
Tail ISO 35-50 Centroid
Coefficient of Variance
Aspect Ratio
Normalized Distance Between Centroids
Orientation of the Melt Pool
k-Nearest Neighbor Classification Algorithm
Prediction of Velocity, Power or Energy Density
XYPT Command Data
250 Total Layers, >1 million rows per layer
X, Y Position
Power
Camera Trigger
Velocity
Class Labels for Each Image

Figure 14: Flow chart of information throughout prediction process

The ground truth in relation to these images lies in the XYPT commands, as each image correlates to a trigger indicator in the “T” column of this build command data. These process parameters provided by this data give the algorithm its class names, represented by velocity, power or energy density around the time each image was captured. Even though this work mainly studied energy density prediction, the same algorithm and training matrix could be used to predict velocity or power as well.

3.3 Training/Testing Split

This algorithm was validated through the evaluation of interlayer prediction within a certain part, as well as through the use of k-fold cross validation. The prediction of process parameters within a certain part required a testing set and a training set split which gives the model enough data to train with, while adequately representing the error of the result with a large enough testing set. A typical split has a maximum of 90% of the dataset reserved for training, if the dataset is small. As there were typically 50,000 melt pool images available for each part, a slightly larger testing set of 25% was used, leaving 75% of the data to train the k-NN.

The use of k-fold cross validation can reduce bias and evaluate the stability of an ML algorithm. The k parameter in this context specifies the number of bins into which the dataset is divided. Once the dataset is divided, one bin is chosen to be the testing set, while the other bins are combined to become the training set. Once the model has been tested and evaluated, the performance is recorded and the model is replaced with the next. Bin *i* out of parameter *k* is chosen as the new testing set and this process is repeated until *k* models have been trained, tested and evaluated. The model was also tested for transferability across the different parts, as the future work in this area would ideally use transfer learning to predict process parameters of any part printed using L-PBF. A different path was taken for each part, and their general descriptions can be seen below in Table 1.

Table 1: Scan strategy descriptions

Part #	Scan Strategy
1	Standard
2	Island
3	Concentric island
4	Island with alternate starting quadrant
5	Island with higher vertical shell laser power
6	Concentric island with higher vertical shell laser power
7	Island with fewer vertical shells
8	Island with fewer vertical shells and alternate starting quadrant
9	Concentric island with alternating power
10	Concentric island with alternating island and concentric island interleaved
11	Interleaved island
12	Interleaved concentric island

CHAPTER 4: RESULTS AND DISCUSSION

The results of the trained k-NN on the process command and monitoring data are presented in this section. The dataset was divided up based on part number (1-12), and then further divided for the training/test split. With an established data set per part of ~30,000 images, the relevant features were derived and saved into the training matrix with 15 columns of independent variables and 1 column of class labels. The matrix was then normalized to reduce bias which might drive the algorithm towards the prediction of an infill class label.

After reduction, a typical training matrix had a length of 4,000 rows, which was a significant reduction (87%) from the identified training data set. This training matrix was input into the testing program, and each new image found in the applicable bounds was assigned an estimate of energy density. A testing matrix could be then saved from those estimates, and error was generated by finding the percent difference between the predicted class label and actual energy density value.

4.1 Feature Extraction Results

The first step in this process, extracting features from the images, played a crucial role in building this low-bandwidth training matrix. The histograms of the training matrix variables, including the features of the image data and their respective class names, are displayed below in Figure 15. The distribution of independent variables, including normal, bimodal, or skewed distributions, provided context for the relationship between these variables and their respective class names.

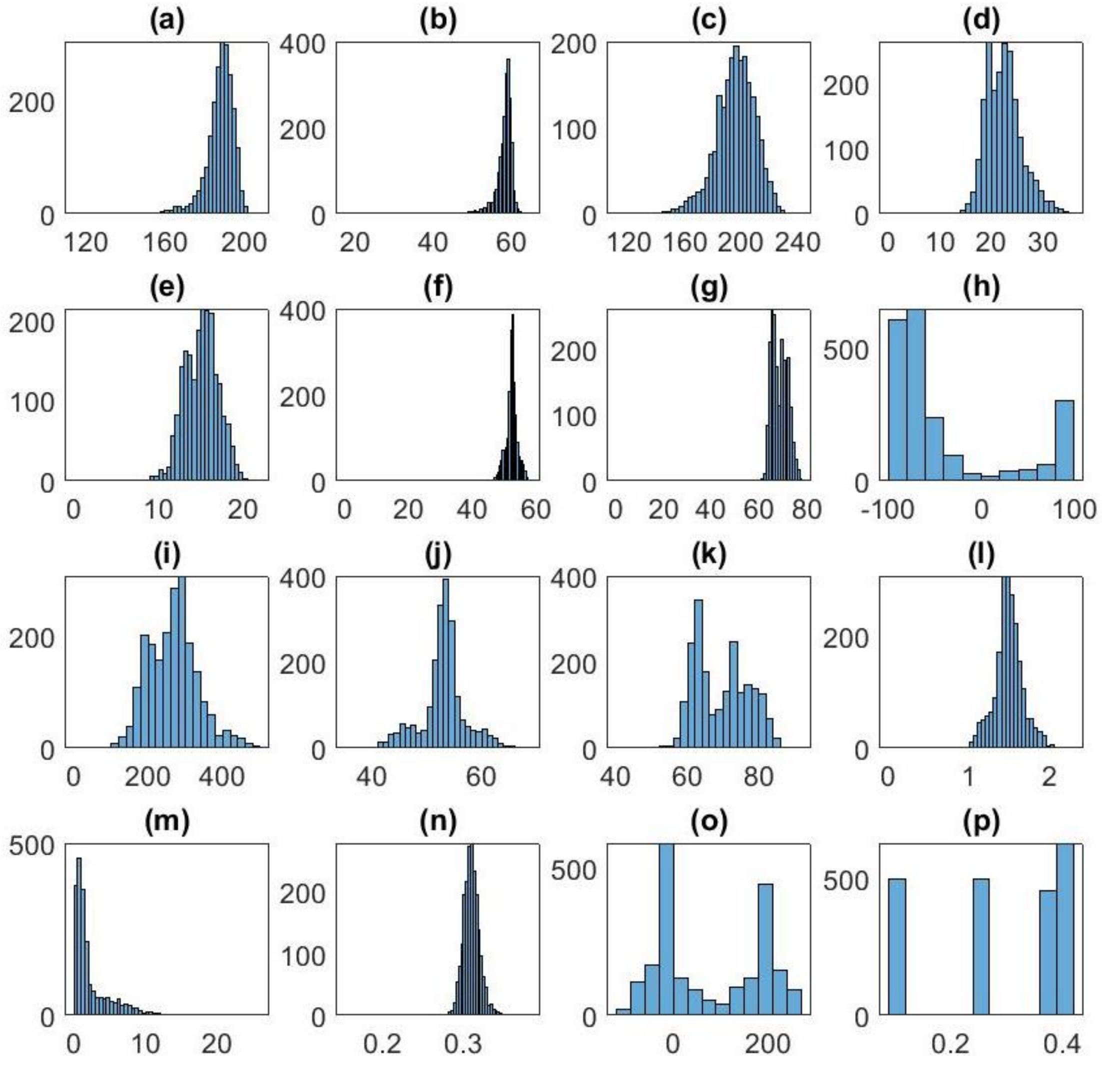

Figure 15: Elements of the training matrix: (a) Average image intensity (b) Standard deviation of image intensity (c) Median image intensity (d) Major axis length (e) Minor axis length (f) Centroid X position (g) Centroid Y position (h) Image processing toolbox's *regionprops* estimate of orientation (i) Melt pool area (j) Tail centroid X position (k) Tail centroid Y position (l) Aspect ratio (m) Distance between centroid and tail (n) Coefficient of variance (o) Estimated orientation (p) Energy density class labels

4.2 Inflection Points and Transitions

The transition points between scan strategies in the various parts were studied to understand the viability of this dataset for the interpolation of process parameters, and to also understand the potential sources of error in the results. Below in Figure 16, the ideal response for the prediction algorithm to study is represented over a 0.1 second period in the build. This transition from 100 to 195 Watts instigated an immediate directly proportional response from the resultant melt pool area, as its average area in square pixels increased from 112 to 238. It is also important to note that scan velocity did not significantly vary over these periods during the build, and therefore the energy density scaled with laser power.

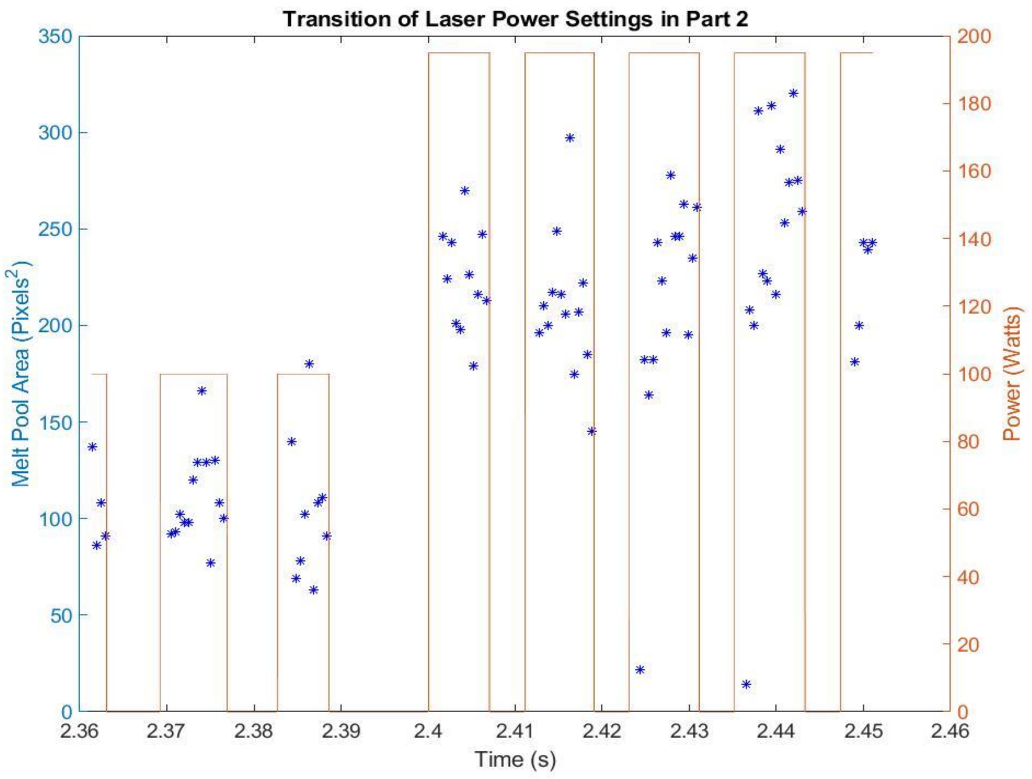

Figure 16: Typical Delineation Between Power Settings and Melt Pool Areas over Time for Part 2

Part #2 however employs a simple serpentine island scan strategy which is more predictable than others such as the alternating power concentric island strategy used by Part #9. Shown in Figure 17 below, the standard laser power transition for this part is more difficult to observe from studying the melt pool area. The difference in bias between the two sections is only 50 pixels squared, a 60% decrease from the typical shift in part 2. This increases the difficulty of prediction, but melt pool area is not the only indication of a change in energy density. This emphasizes the advantage of using a machine learning algorithm, as the relationship between the process parameters and aspects of the images are complicated by additional physical factors. For example, the physics of thermal energy retention could be influencing the lack of melt pool area response, as the powder bed absorbs the extra thermal energy when the laser does not cycle on and off, such as in part #2.

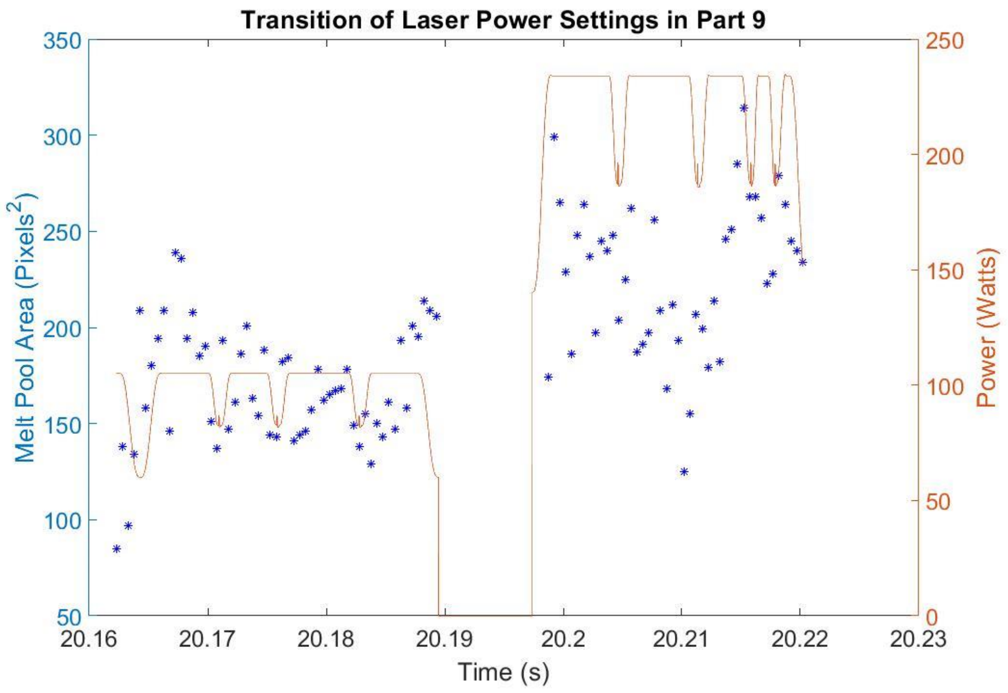

Figure 17: Typical Delineation Between Power Settings and Melt Pool Areas over Time for Part 9

Patterns such as changes in melt pool size with shifts in power can be noted for future prediction, and they will be the features which influence the weighting scheme in ML algorithms such as the k-NN approach. Since this work is focused on completing the control loop by estimating process parameters, the overlap in the spans in melt pool area make it tough for the algorithm to be perfectly accurate, but it does indicate a general trend which should match up with the programmed laser power and scan velocity. This lack of precision is also exacerbated by phenomena occurring at the edge cases where the parameters do not match up with the *in-situ* monitoring data at all.

In Figure 18 below, the unique pattern of drastic increases in melt pool area can be seen over the course of 0.07 seconds during the build of part #9. This pattern occurs regularly for parts with concentric island scan strategies as the laser moves in a spiral motion towards the center of each island. The cause for this spike in melt pool area without changing energy density could fall on the increase in thermal energy being retained when the Euclidean distance between the current melt pools and recently melted powder continues to be small. This concentration of thermal energy does indicate an increase in energy density within the part at this point, but it is no longer monotonically related based on process parameters. Therefore, this type of behavior has been classified in this work as atypical, and the algorithm does not attempt to predict such occurrences in the future.

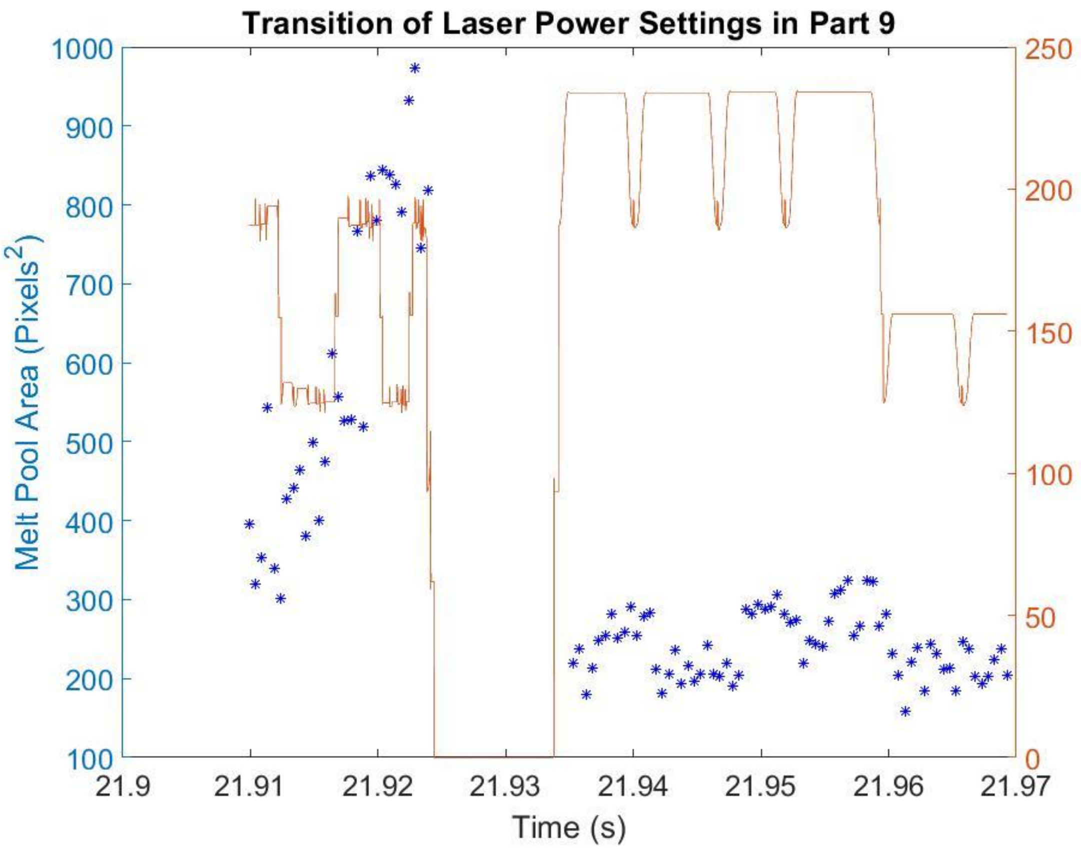

Figure 18: Atypical Delineation Between Power Settings and Melt Pool Areas over Time for Part 9

4.3 k-NN Layer Comparison Results

The simplest scan strategies represented by part numbers 1 and 2 were the first studied for validation of the model. Table 2 displays the results of a k-NN (k = 5) employed on parts with a split between training and test sets at around layer 188. From the table, it is clear that training on a subset data from a given part provides an effective model for classifying other data from that same part. This initial check evaluates the viability of predicting other layers' energy density values within the same part.

Table 2: Layer comparison

Training Part : Layer Count	Testing Part : Layer Count	Mean Error (%) ± σ
1 – Standard : 2-188	1 – Standard : 189-250	1.83 ± 19.9
2 - Island: 2-188	2 – Island : 189-250	5.60 ± 29.4

4.4 k-NN Part Comparison Results

Examples of the various scan strategies can be seen below in Figure 19. The standard scan strategy is a vertical serpentine pattern with a constant hatch spacing *h*, and the laser scans from one side of the part across the entire depth to the other side. In contrast, the island technique divides the part into smaller identical "islands," typically 4, and there the laser would only scan halfway across the depth and then move hatch spacing *h* orthogonal to the nominal scanning direction before scanning back to the edge. The serpentine strategy is replaced in part 3 by the "concentric" style which begins near an edge of the part and moves inward with a centering spiral pattern. Different variations of these basic strategies and their permutations make up the remaining 9 parts.

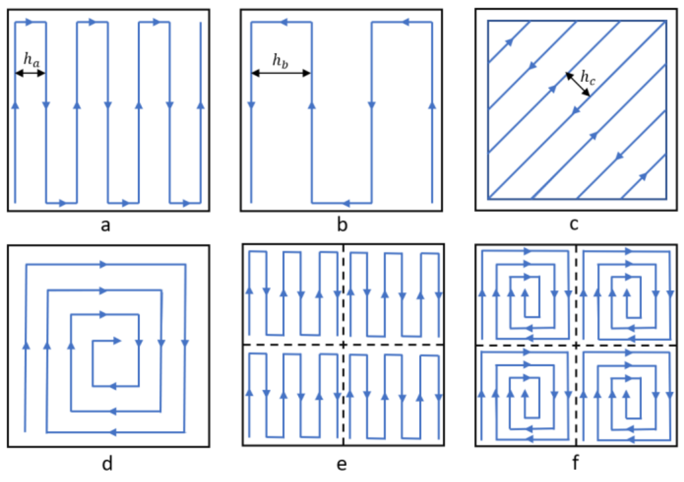

Figure 19: Examples of different scan strategies: (a) Standard/serpentine with hatching space h (b) Serpentine with larger hatching space (c) Off-axis serpentine (d) Concentric (e) Serpentine Island (f) Concentric Island [23]

Using parts 1-3 as the pool for training sets, the variations from similar parts were studied for transferability of the k-NN model. First, the interchangeability between the training set parts was tested, and the experiment was organized with enough permutations so that each subsequent part tested could be related back to any of the training parts. The scan strategies at the core of each pair of parts compared were built upon the same foundation, as parts 2 and 4 both use serpentine island techniques, but part 4 starts with a different island or quadrant.

The training/testing pairs and their respective errors are presented below in Table 3. For each row in the table, the k-NN model was trained on the training part number indicated in column 1 and all applicable layers in the total of 250. The strategy used for that parts manufacture is copied from Table 1 alongside part number for reference. In column 2, a similar format follows, as the k-NN model was tested on each applicable layer within that part's dataset. The model's predictions were evaluated against the XYPT command data, and the mean number of misclassifications as well as the standard deviation is presented in column 3.

Table 3: Part comparison

Training Part – Strategy Layers 1-250	Testing Part - Strategy Layers 1-250	Mean Error (%) ± σ
1 – Standard	2 – Island	1.51 ± 28.7
2 – Island	3 – Concentric Island	3.59 ± 38.9
1 – Standard	3 – Concentric Island	4.49 ± 39.8
2 – Island	4 – Alt. Start Island	3.18 ± 20.5
2 – Island	5 – High Power Island	17.7 ± 42.4
3 – Concentric Island	6 – High Power Con. Island	35.3 ± 46.4
3 – Concentric Island	9 – Alt. Power Con. Island	22.4 ± 48.6

The transferability of the model between various scan strategy differences was studied to produce the above data, as each part (1-12) has consistent geometry. The training part pool, parts 1-3, was initially investigated to establish a baseline set of results for the three simplest scan strategies that were used. For three unique paths to compare, three permutations of the train/test split were established and executed to achieve connections between each baseline part from another. An average misclassification rate of 3.2% was reached with the combinations in the first three rows of Table 3, and this reduced error was only matched in the future comparisons with parts 2 and 4. As more dissimilar parts or more complex scan strategies were compared, the error was observed to significantly increase in probability. As mentioned, training part 4 on a test matrix generated by part 2 produced similar error to that of the baseline part pool due to the only difference originating from the alternating starting quadrants. Otherwise, the features which the model could learn from were discovered in the testing part data as well, which resulted in fewer misclassifications. Inversely, the proportion of features possibly mapping from parts 3 to 6 decreases due to the exponential increase in thermal absorption and subsequently thermal energy density with a concentrically focused scan strategy. The inflection points mentioned in the above section are critical here in the part comparison, as the introduction of more variability in the strategy as in part 9 for example will reduce the possibility for the model to project trends from the simpler scan strategy. The overall higher error with pairs [2,5], [3,6], and [3,9] seems to be consistent with more of these fluctuations in laser power.

Below, Tables 4-10 illustrate the context behind the error presented in Table 3, as each predicted energy density is recorded in relation to the actual energy density. In order to better observe the general trend for the success of the algorithm, the range of energy density class labels was categorized into three main groups: Category 1 (Energy Density less than 0.25 J/mm^3), Category 2 (Energy Density greater than 0.25 J/mm^3 but less than 0.45 J/mm^3), and Category 3 (Energy Density greater than 0.45 J/mm^3).

Table 4: Results of training of part 1 and testing on part 2

		Predicted Category		
		1	2	3
Actual Category	1	1833	289	141
	2	769	27538	3073
	3	141	2138	2995

Table 5: Results of training on part 2 and testing on part 3

		Predicted Category		
		1	2	3
Actual Category	1	2723	36	102
	2	2726	15961	5192
	3	746	2259	45620

Table 6: Results of training on part 1 and testing on part 3

		Predicted Category		
		1	2	3
Actual Category	1	2385	179	270
	2	2494	27735	9636
	3	1762	8284	33436

Table 7: Results of training on part 2 and testing on part 4

		Predicted Category		
		1	2	3
Actual Category	1	2104	84	50
	2	723	27229	3733
	3	178	1354	4001

Table 8: Results of training on part 2 and testing on part 5

		Predicted Category		
		1	2	3
Actual Category	1	2289	166	238
	2	1728	27754	11541
	3	722	2686	24141

Table 9: Results of training on part 3 and testing on part 6

		Predicted Category		
		1	2	3
Actual Category	1	2523	154	219
	2	1574	33857	19711
	3	825	7095	35678

Table 10: Results of training on part 3 and testing on part 9

		Predicted Category		
		1	2	3
Actual Category	1	2641	102	152
	2	2146	26074	24233
	3	944	5037	34628

From the above results, it was observed that the algorithm performed consistently well when the XYPT commands indicated category 1 energy density, as the algorithm predominately estimated the first category with 91% ± 3.2% accuracy. In contrast, actual category 2 resulted in an average ratio of correct predictions near 68% with a standard deviation of 27%. The majority of melt pools fell into this category, and it is reasonable to assume this category would have the largest error with the edge cases introduced and large variation in melt pools which resulted in similar average features distributed normally. Finally, category 3 was estimated correctly 75% of the time while having a 16% standard deviation. For the purposes of quick validation in a closed loop AM system, perfect prediction of values is not necessary as this algorithm serves more as a sanity check to the system. More time-consuming analysis can be done after the build is completed if necessary, and future work could even reconcile ML predictions with physics-based simulations. This comparison demonstrates a capability for the k-NN algorithm to accurately transfer between similar parts with a mean error ~3% and a standard deviation similar to that of same part comparison.

4.4 Time Usage Analysis

The average time needed to complete each image prediction and layer analysis is shown below in Tables 11 & 12. Mean time per image or layer represents the average time across just the testing matrices, including the time required to parse through the data and construct those matrices. Sources of error were explored, and the relationship between error and average time required was studied.

Table 11: Image processing time comparison

Training Part – Strategy Layers 1-250	Testing Part - Strategy Layers 1-250	Mean Time/Image (ms) ± σ
1 – Standard	2 – Island	11.9 ± 3.0
2 – Island	3 – Concentric Island	13.4 ± 4.0
1 – Standard	3 – Concentric Island	12.1 ± 2.5
2 – Island	4 – Alt. Start Island	12.6 ± 3.2
2 – Island	5 – High Power Island	13.2 ± 3.0
3 – Concentric Island	6 – High Power Con. Island	19.6 ± 3.0
3 – Concentric Island	9 – Alt. Power Con. Island	19.5 ± 3.0

Table 12: Layer time comparison

Training Part – Strategy Layers 1-250	Testing Part - Strategy Layers 1-250	Mean Time/Layer (s) ± σ
1 – Standard	2 – Island	39.8 ± 1.80
2 – Island	3 – Concentric Island	84.3 ± 13.7
1 – Standard	3 – Concentric Island	76.7 ± 11.9
2 – Island	4 – Alt. Start Island	41.0 ± 2.09
2 – Island	5 – High Power Island	58.0 ± 13.6
3 – Concentric Island	6 – High Power Con. Island	121.7 ± 18.5
3 – Concentric Island	9 – Alt. Power Con. Island	116.1 ± 15.0

From these results, it was observed that testing the algorithm which was trained on serpentine scan strategies generally required less time, an average of 64 seconds per layer and 12.5 seconds per image. In contrast, the concentric island scan strategies encourage higher computation times, as training on part 3 resulted in an average increase of 6.3 milliseconds per image, which translated to a jump in processing time of 54 seconds to 118 seconds needed to evaluate each layer. These trends held true in the layer wise comparison as training on a serpentine island part (1 or 2) and testing on part 3 resulted in a mean increase of 16 seconds from the standard training results. The effect concentric scan strategies had on the computation time could be attributed to the irregularity that these techniques introduce with respect to the energy density build up surrounding the center of the inward spiral path. If the algorithm failed to correctly predict the energy density classification, no change in the analysis time per image could be observed on average. The computing time was also observed to be higher during trials where scan strategy greatly differed, which is expected if the model is having difficulty placing new values which are far from the established class labels. The kurtosis, defined in the equation below, of the distribution from each time study was also calculated to uncover possible context behind these variations, and those results can be seen below in Table 13.

Equation 3:

$$Kurt[X] = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2} = \frac{\mu_4}{\sigma^4}$$

Table 13: Time distribution kurtosis comparison

Training Part – Strategy Layers 1-250	Testing Part - Strategy Layers 1-250	Layer Kurtosis	Image Kurtosis
1 – Standard	2 – Island	2.083	80.06
2 – Island	3 – Concentric Island	2.120	465.2
1 – Standard	3 – Concentric Island	2.170	216.8
2 – Island	4 – Alt. Start Island	2.04	188.5
2 – Island	5 – High Power Island	12.35	320.8
3 – Concentric Island	6 – High Power Con. Island	1.735	65.66
3 – Concentric Island	9 – Alt. Power Con. Island	1.877	58.32

The kurtosis results presented above describe a different relationship between scan strategy and the shape of the time distribution. Compared to a standard normal distribution which has a kurtosis of 3, an average kurtosis of 198 for the required time to process each image indicates a heavily tailed distribution. The kurtosis across both the layer and image time distributions was significantly lower for the last two transferability studies where the algorithm was trained on a concentric island scan strategy, which surprisingly shows that these studies more closely resembled a normal distribution despite requiring much more time to compute on average. Also, the second row of Table 13 below the header presents the highest kurtosis for image time distributions which might be interpreted as the reason the mean time per image for that transferability study was calculated to be the highest out of all studies not trained on concentric island strategies.

4.5 k-NN k-Fold Cross Validation Results

The cross validation of the k-NN algorithm was computed to evaluate the interlayer comparison within part 2, which was chosen for its relevance to the most parts within the set while being simple enough to represent a baseline for other comparisons to follow. The computation time required for this result was significant, and as such prohibited exhaustive testing across the training set.

5-fold Cross Validation Error of k-NN interlayer comparison with 90/10 split = 4.58 ± 29.7%

The scatter plot of the error averaged across the 5 folds is shown below in Figure 20. The descriptive statistics presented above describe the average error and standard deviation of this scatter plot. No clear source of error has been interpreted from this plot, as the error does not follow the contours of the vertical shells, the intersection between the islands or any other feature of the part’s geometry.

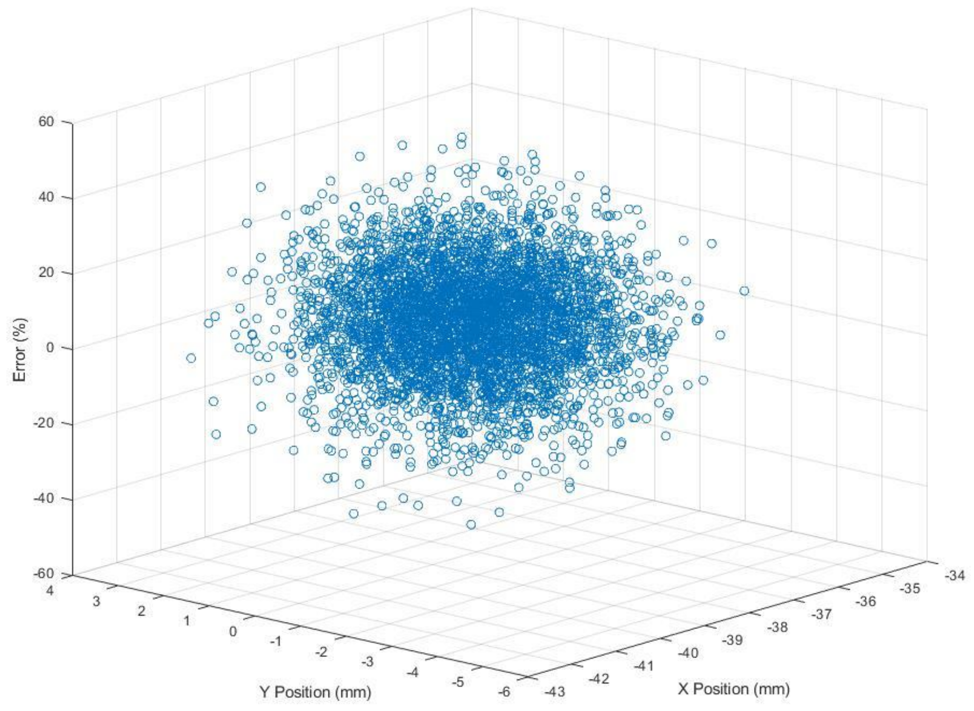

Figure 20: Error scatter plot for prediction of part 2 with k-fold validation

4.6 Assumptions

A critical assumption was made to estimate velocity around the point of the image capture. It was assumed that the XYPT data accurately represented the discrete position of the laser, and travelled distance could be interpreted from the surrounding cluster of points by plotting a line of best fit through those XY positions, and assuming the length of the best fit line segment to be average distance through the path of the laser. That average distance was divided by the average time represented by the number of passed time intervals multiplied by the time step of 10 μs. This provided the energy density equation with the average velocity component, and the power of the laser was evaluated instantaneously by using the index of the image capture.

4.7 Challenges Faced

A large portion of the training matrix initially skewed the weighting of the k-NN algorithm, as the infill section of the scan strategy represented the majority of points in the matrix. The resultant error was calculated to be ~10% for interlayer prediction, but the bias presented in k-fold cross validation was high, as a typical standard deviation rose above 100% from the mean. The training matrix was normalized by removing the excess infill data points, and the matrix subsequently shrank by an order of magnitude. With most of the melt pool images unusable for the purposes of this work, the misclassifications made by the k-NN remained to be significant. Due to the low number of class labels, any mistake resulted automatically in high error and the standard deviation reflects the effects of this type of error.

CHAPTER 5: CONCLUSION

In this thesis, a novel application of k-NN modeling to predict process parameters in laser powder bed fusion has been presented. The k-NN was trained on in-situ monitoring images and build command data from the Additive Manufacturing Metrology Testbed at NIST which contains 12 parts of identical geometry but unique scan strategies. The model was initially validated by studying interlayer prediction within the same part, with average error ranging from 1.83% to 5.60%. This performance is an improvement upon other image classification models using k-NN algorithms [27, 28].

This approach was modified to also study transferability between parts of differing scan strategies, as training matrices were used to predict the energy density of images taken of a different part. A higher mean error was expected with this experiment, and as such this deviation ranged from 1.51% to 35.3%. This average error increased as the parts became more dissimilar, and especially as the process parameters of the test parts began to fall outside of the bounds of the identified classes.

Finally, a k-fold cross validation was performed to evaluate the consistency of this modelling application. An average error of 4.58% across the 5 folds was calculated, and a scatter plot of the error with respect to position showed no clear evidence of bias. This work demonstrated the capability of ML algorithms in predicting the process parameters within additive manufacturing applications, and that these approaches are especially robust when changes to scan strategy are minimal. It also plays an important role in demonstrating the possibility of using ML to predict these relationships instead of physics-based modeling, which is a common approach to solving this problem. Opening up the possibilities for closed-loop control in the metal additive manufacturing field allows for more choice and competition which encourages more widespread adoption of laser-powder bed fusion, satisfying an overarching goal of this work. Without more basic research needed, the ML approach can improve greatly over time as more data becomes available from the application of this advanced manufacturing technique, and that combined with the open-source nature of these algorithms implies a promising future in store for low-cost quality assurance of powder bed fusion.

5.1 Contributions

The prediction of process parameters which significantly contribute to melt pool geometry had not been previously studied in this application, and the robustness of prediction across scan strategies has performed just as well if not better than other k-NN image classifiers. Data preprocessing was used to give the training set context, and to reduce the amount of information required for the k-NN to model. Fifteen parameters described each image, which decreased computing time and allowed for more images to be included in the training/testing matrix split.

5.2 Future Work

An ML algorithm especially lends itself to in-situ process validation when the learning of the model can be transferred to new scan strategies, because of the inherent variation while building AM parts. This would require a similar experiment to the one presented by NIST [1], with an addition of training the k-NN in real time during the manufacturing process. The presented low-density approach to training the model would decrease the computing time required, but may still need improvements in efficiency to function properly under those conditions. Next steps should include the introduction of variable geometries to increase the robustness of the ML and subsequently make this more applicable to closing the loop.

APPENDIX: k-NN Train/Test Code

```
%Train
%for fold = 1:5
%    fprintf('----------    Fold = %u    ----------\n',fold)
    trainMatrix = [];
%    final_layer = 100+fold*25;
    final_layer = 188;
    layer = 2:final_layer;
    for layer = layer
        fprintf('----------    Layer = %u    ----------\n',layer)
        imgfolder = 'E:\NIST Data\In-situ Meas Data\In-situ Meas Data\Melt Pool Camera';
        imgbaseFileName = sprintf('MIA_L%04u',layer);
        imgfullFileName = fullfile(imgfolder, imgbaseFileName);
        a=dir([imgfullFileName '/*.bmp']);
        k = size(a,1);

        commandfolder = 'E:\NIST Data\Build Command Data\Build Command Data\XYPT 
Commands';
        commandbaseFileName=sprintf('T500_3D_Scan_Strategies_fused_layer%04u.csv',layer);
        commandfullFileName = fullfile(commandfolder, commandbaseFileName);
        if ~exist(commandfullFileName, 'file')
          % Didn't find it there.  Check the search path for it.
          commandfullFileName = commandbaseFileName; % No path this time.
          if ~exist(commandfullFileName, 'file')
            % Still didn't find it.  Alert user.
            errorMessage = sprintf('Error: %s does not exist.', commandfullFileName);
            uiwait(warndlg(errorMessage));
            return;
          end
        end

        A = dlmread(commandfullFileName,',');
        X = A(:,1);
        Y = A(:,2);
        P = A(:,3);
        T = A(:,4);
        a = find(T==2);
        %Initialize Variables
        data = zeros(k,7);
        tail = zeros(k,3);

        for k = 1:k
            if X(a(k)-121)>-30 || Y(a(k)-121)<10
                continue
            end
            imagefullFileName = fullfile(imgfullFileName, sprintf('frame0%04u.bmp',k));
            grayImage = imread(imagefullFileName);
            [rows, columns, numberOfColorBands] = size(grayImage);
            if numberOfColorBands > 1
                fprintf('This image is RGB.  I will change it to gray scale.\n');
                grayImage = grayImage(:, :, 2);
            end
            binaryImage = grayImage > 90;
            binaryImage = bwareaopen(binaryImage, 40);
            binaryImage = imfill(binaryImage, 'holes');
            props = regionprops(binaryImage, 'Area', 'MajorAxisLength',
'MinorAxisLength', 'Centroid', 'Orientation');
            allAreas = [props.Area];
            majorAxisLength = cat(1,props.MajorAxisLength);
            minorAxisLength = cat(1,props.MinorAxisLength);
            centroid = [props.Centroid];
            orientation = [props.Orientation];
            if isempty(centroid)
                data(k,:) = [k, 0, 0, 0, 0, 0, 0];
                aspect_ratio =[];
            else
                try
                    data(k,:) = [k , majorAxisLength , minorAxisLength , centroid ,
orientation, allAreas];
                    aspect_ratio = majorAxisLength/minorAxisLength;
                catch
                    data(k,:) = [k, 0, 0, 0, 0, 0, 0];
                    aspect_ratio = 0;
                end
            end % End if statement

            IntenseImage = grayImage(find(binaryImage==1));
            IntenseImage = IntenseImage(:);
            try
                meanImg = mean(IntenseImage);
                stdevImg = std(double(IntenseImage));
                medImg = median(IntenseImage);
                COV = stdevImg/meanImg;
            catch
                meanImg = [];
                stdevImg = [];
                medImg = [];
                COV = 0;
            end

            binaryImage = grayImage > 50 & grayImage < 90;

            if any(any(binaryImage))==0
                tail(k,:) = [k, 0, 0];
                tail_centroid = [];
            else
                [r, c] = find(binaryImage == 1);
                tail_centroid = [mean(c), mean(r)];
                tail(k,:) = [k ,tail_centroid];
            end
            if isempty(tail_centroid) || isempty(centroid) || size(centroid,2)>2 ||
size(tail_centroid,2)>2
                orientation = 0;
            else
```

```
                dist = sqrt((tail_centroid(2)-centroid(2))^2+(tail_centroid(1)-
centroid(1))^2);
                projected(1) = centroid(1)+dist*cosd(orientation);
                projected(2) = centroid(2)+dist*sind(orientation);
                dist2 = sqrt((projected(2)-tail_centroid(2))^2+(projected(1)-
tail_centroid(1))^2);
                projected2(1) = centroid(1)-dist*cosd(orientation);
                projected2(2) = centroid(2)-dist*sind(orientation);
                dist3 = sqrt((projected2(2)-tail_centroid(2))^2+(projected2(1)-
tail_centroid(1))^2);
                if dist2 > dist3
                    CorrectOrientation = atan2d(centroid(2)-tail_centroid(2),centroid(1)-
tail_centroid(1));
                    dist = dist3;
                else
                    CorrectOrientation = atan2d(centroid(2)-tail_centroid(2),centroid(1)-
tail_centroid(1));
                    dist = dist2;
                end

                orientation = CorrectOrientation;
                orientation = 83.4+orientation;
            end % End if statement

            newline = [];
            try
                %data(k,:) = [k , majorAxisLength , minorAxisLength , centroid ,
orientation, allAreas];
                %tail(k,:) = [k ,tail_centroid]; tail_centroid = [mean(c), mean(r)];
                newline = [meanImg,stdevImg,double(medImg),data(k,2:end), ...
                    tail(k,2:end),aspect_ratio,dist,COV,orientation];
    %
    %             label = predict(Mdl,newline);
    %             newline(end) = label;

                loc = a(k)-121;
                spread = -5:5;
                locs = loc + spread'.*ones(11,1);
                x = X(loc);
                x_smooth = X(locs);
                y = Y(loc);
                y_smooth = Y(locs);
                x_smooth = [ones(length(x_smooth),1) x_smooth];
                b = x_smooth\y_smooth;
                yCalc = b'*x_smooth';
                yCalc = yCalc';
                dist = sqrt((yCalc(end)-yCalc(1))^2+(x_smooth(end,2)-x_smooth(1,2))^2);
                t = 100*10^-6;
                v_avg = dist./t;
                p = P(loc);

                point = p./v_avg;
                if point > 1.5 || point < 0.01
                    continue
                end
                newline = [newline point];
                trainMatrix = [trainMatrix ; newline];
            catch
                continue
            end
        end
    end % End layer level for loop
```

```
     b = find(abs(trainMatrix(:,end)-0.25)<0.05);
     c = find(abs(trainMatrix(:,end)-0.1)<0.05);
     d = randperm(length(b),length(c));
     b(d) = [];
     trainMatrix(b,:)=[];
     e = find(trainMatrix(:,end)>1);
     trainMatrix(e,:) = [];
    %dlmwrite('E:\NIST Data\Results\TrainMatrix2.txt',trainMatrix)
```

```
    X = double(trainMatrix(:,1:end-1));
    Y = round(double(trainMatrix(:,end)),3);
    Mdl = fitcknn(X,Y,'NumNeighbors',5,'Standardize',1);
    %Mdl = fitcsvm(X,Y);
    %net = feedforwardnet(10)
    % net.numinputs = 12;
    % net.inputConnect = logical(int32(randi([0,1],[2,12])));
    % net = configure(net,X');
    %net = train(net,X',Y')
    %view(net)
    %newline = newline(1:end-1);
    %label = net(newline')
    %label = predict(Mdl,double(newline));
```

```
    subplot(4,4,1)
    histogram(trainMatrix(:,1))
    title('(a)')
    xticks([120 160 200])
    xticklabels({'120' '160' '200'})
    set(gca,'fontsize',16);
    subplot(4,4,2)
    histogram(trainMatrix(:,2))
    title('(b)')
    set(gca,'fontsize',16);
    subplot(4,4,3)
    histogram(trainMatrix(:,3))
    title('(c)')
    set(gca,'fontsize',16);
    subplot(4,4,4)
    histogram(trainMatrix(:,4))
    title('(d)')
    set(gca,'fontsize',16);
    subplot(4,4,5)
    histogram(trainMatrix(:,5))
    title('(e)')
    set(gca,'fontsize',16);
    subplot(4,4,6)
    histogram(trainMatrix(:,6))
    title('(f)')
    set(gca,'fontsize',16);
    subplot(4,4,7)
    histogram(trainMatrix(:,7))
    title('(g)')
    set(gca,'fontsize',16);
    subplot(4,4,8)
    histogram(trainMatrix(:,8))
    title('(h)')
    set(gca,'fontsize',16);
    subplot(4,4,9)
    histogram(trainMatrix(:,9))
    title('(i)')
    set(gca,'fontsize',16);
    subplot(4,4,10)
    histogram(trainMatrix(:,10))
    title('(j)')
    set(gca,'fontsize',16);
    subplot(4,4,11)
    histogram(trainMatrix(:,11))
    title('(k)')
    set(gca,'fontsize',16);
    subplot(4,4,12)
    histogram(trainMatrix(:,12))
    title('(l)')
    set(gca,'fontsize',16);
    subplot(4,4,13)
    histogram(trainMatrix(:,13))
    title('(m)')
    set(gca,'fontsize',16);
    subplot(4,4,14)
    histogram(trainMatrix(:,14))
    title('(n)')
    set(gca,'fontsize',16);
    subplot(4,4,15)
    histogram(trainMatrix(:,15))
    title('(o)')
    set(gca,'fontsize',16);
    subplot(4,4,16)
    histogram(trainMatrix(:,16))
    title('(p)')
    set(gca,'fontsize',16);

%     pause()
```

```
    error = [];
    xpos = [];
    ypos = [];
    vel = [];
    pow = [];
    start_layer = final_layer+1;
    layer = start_layer:250;
    for layer = layer
```

```
        fprintf('----------    Layer = %u    ----------\n',layer)
        imgfolder = 'E:\NIST Data\In-situ Meas Data\In-situ Meas Data\Melt Pool Camera';
        imgbaseFileName = sprintf('MIA_L%04u',layer);
        imgfullFileName = fullfile(imgfolder, imgbaseFileName);
        a=dir([imgfullFileName '/*.bmp']);
        k = size(a,1);

        commandfolder = 'E:\NIST Data\Build Command Data\Build Command Data\XYPT
Commands';
        commandbaseFileName=sprintf('T500_3D_Scan_Strategies_fused_layer%04u.csv',layer);
        commandfullFileName = fullfile(commandfolder, commandbaseFileName);
        if ~exist(commandfullFileName, 'file')
          % Didn't find it there.  Check the search path for it.
          commandfullFileName = commandbaseFileName; % No path this time.
          if ~exist(commandfullFileName, 'file')
            % Still didn't find it.  Alert user.
            errorMessage = sprintf('Error: %s does not exist.', commandfullFileName);
            uiwait(warndlg(errorMessage));
            return;
          end
        end

        A = dlmread(commandfullFileName,',');
        X = A(:,1);
        Y = A(:,2);
        P = A(:,3);
        T = A(:,4);
        a = find(T==2);
        %Initialize Variables
        data = zeros(k,7);
        tail = zeros(k,3);

        for k = 1:k
            if X(a(k)-121)>25 || X(a(k)-121)<0 || Y(a(k)-121)>-10
                continue
            end
            imagefullFileName = fullfile(imgfullFileName, sprintf('frame0%04u.bmp',k));
            grayImage = imread(imagefullFileName);
            [rows, columns, numberOfColorBands] = size(grayImage);
            if numberOfColorBands > 1
                fprintf('This image is RGB.  I will change it to gray scale.\n');
                grayImage = grayImage(:, :, 2);
            end
            binaryImage = grayImage > 90;
            binaryImage = bwareaopen(binaryImage, 40);
            binaryImage = imfill(binaryImage, 'holes');
            props = regionprops(binaryImage, 'Area', 'MajorAxisLength',
'MinorAxisLength', 'Centroid', 'Orientation');
            allAreas = [props.Area];
            majorAxisLength = cat(1,props.MajorAxisLength);
            minorAxisLength = cat(1,props.MinorAxisLength);
            centroid = [props.Centroid];
            orientation = [props.Orientation];
            if isempty(centroid)
```

```
                data(k,:) = [k, 0, 0, 0, 0, 0, 0];
                aspect_ratio =[];
            else
                try
                    data(k,:) = [k , majorAxisLength , minorAxisLength , centroid ,
orientation, allAreas];
                    aspect_ratio = majorAxisLength/minorAxisLength;
                catch
                    data(k,:) = [k, 0, 0, 0, 0, 0, 0];
                    aspect_ratio = 0;
                end
            end % End if statement

            IntenseImage = grayImage(find(binaryImage==1));
            IntenseImage = IntenseImage(:);
            try
                meanImg = mean(IntenseImage);
                stdevImg = std(double(IntenseImage));
                medImg = median(IntenseImage);
                COV = stdevImg/meanImg;
            catch
                meanImg = [];
                stdevImg = [];
                medImg = [];
                COV = 0;
            end

            binaryImage = grayImage > 50 & grayImage < 90;

            if any(any(binaryImage))==0
                tail(k,:) = [k, 0, 0];
                tail_centroid = [];
            else
                [r, c] = find(binaryImage == 1);
                tail_centroid = [mean(c), mean(r)];
                tail(k,:) = [k ,tail_centroid];
            end
            if isempty(tail_centroid) || isempty(centroid) || size(centroid,2)>2 ||
size(tail_centroid,2)>2
                orientation = 0;
            else
                dist = sqrt((tail_centroid(2)-centroid(2))^2+(tail_centroid(1)-
centroid(1))^2);
                projected(1) = centroid(1)+dist*cosd(orientation);
                projected(2) = centroid(2)+dist*sind(orientation);
                dist2 = sqrt((projected(2)-tail_centroid(2))^2+(projected(1)-
tail_centroid(1))^2);
                projected2(1) = centroid(1)-dist*cosd(orientation);
                projected2(2) = centroid(2)-dist*sind(orientation);
                dist3 = sqrt((projected2(2)-tail_centroid(2))^2+(projected2(1)-
tail_centroid(1))^2);
                if dist2 > dist3
                    CorrectOrientation = atan2d(centroid(2)-tail_centroid(2),centroid(1)-
tail_centroid(1));
```

```
                    dist = dist3;
                else
                    CorrectOrientation = atan2d(centroid(2)-tail_centroid(2),centroid(1)-
tail_centroid(1));
                    dist = dist2;
                end

                orientation = CorrectOrientation;
                orientation = 83.4+orientation;
            end % End if statement

            newline = [];
            try
                newline = [meanImg,stdevImg,double(medImg),data(k,2:end), ...
                    tail(k,2:end),aspect_ratio,dist,COV,orientation];

                label = predict(Mdl,newline);

                loc = a(k)-121;
                spread = -5:5;
                locs = loc + spread'.*ones(11,1);
                x = X(loc);
                x_smooth = X(locs);
                y = Y(loc);
                y_smooth = Y(locs);
                x_smooth = [ones(length(x_smooth),1) x_smooth];
                b = x_smooth\y_smooth;
                yCalc = b'*x_smooth';
                yCalc = yCalc';
                dist = sqrt((yCalc(end)-yCalc(1))^2+(x_smooth(end,2)-x_smooth(1,2))^2);
                t = 100*10^-6;
                v_avg = dist./t;
                p = P(loc);
                if p == 0
                    continue
                end
                point = p./v_avg;
                newError = (point-label)./point;
                if isnan(newError) || newError == 0 || newError == 1 || abs(newError)>100
                else
                    error = [error;(point-label)./point];
                    xpos = [xpos;x];
                    ypos = [ypos;y];
                    vel = [vel;v_avg];
                    pow = [pow;p];
                end
            catch
                continue
            end
        end
    end
    %end
```

REFERENCES

[1] B. Lane and H. Yeung, “Process Monitoring Dataset from the Additive Manufacturing Metrology Testbed (AMMT): ‘Three-Dimensional Scan Strategies,’” *Journal of Research of the National Institute of Standards and Technology*, vol. 124, 2019.

[2] M. Grasso and B. M. Colosimo, “Process defects and in situ monitoring methods in metal powder bed fusion: a review,” *Measurement Science and Technology*, vol. 28, no. 4, p. 044005, 2017.

[3] H. Yeung, Z. Yang, and L. Yan, “A melt pool prediction based scan strategy for powder bed fusion additive manufacturing,” *Additive Manufacturing*, vol. 35, p. 101383, 2020.

[4] M. Mani, B. Lane, A. Donmez, S. Feng, S. Moylan, and R. Fesperman, “Measurement Science Needs for Real-time Control of Additive Manufacturing Powder Bed Fusion Processes,” 2015.

[5] P. Witherell, S. Feng, T. W. Simpson, D. B. S. John, P. Michaleris, Z.-K. Liu, L.-Q. Chen, and R. Martukanitz, “Toward Metamodels for Composable and Reusable Additive Manufacturing Process Models,” *Journal of Manufacturing Science and Engineering*, vol. 136, no. 6, 2014.

[6] R. Cunningham, S. P. Narra, C. Montgomery, J. Beuth, and A. D. Rollett, “Synchrotron-Based X-ray Microtomography Characterization of the Effect of Processing Variables on Porosity Formation in Laser Power-Bed Additive Manufacturing of Ti-6Al-4V,” *Jom*, vol. 69, no. 3, pp. 479–484, 2017.

[7] M. Tang, P. C. Pistorius, and J. L. Beuth, “Prediction of lack-of-fusion porosity for powder bed fusion,” *Additive Manufacturing*, vol. 14, pp. 39–48, 2017.

[8] W. E. Frazier, “Metal Additive Manufacturing: A Review,” *Journal of Materials Engineering and Performance*, vol. 23, no. 6, pp. 1917–1928, 2014.

[9] W. E. King, A. T. Anderson, R. M. Ferencz, N. E. Hodge, C. Kamath, S. A. Khairallah, and A. M. Rubenchik, “Laser powder bed fusion additive manufacturing of metals; physics, computational, and materials challenges,” *Applied Physics Reviews*, vol. 2, no. 4, p. 041304, 2015.

[10] H. Ko, P. Witherell, N. Y. Ndiaye, and Y. Lu, “Machine Learning based Continuous Knowledge Engineering for Additive Manufacturing,” *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 2019.

[11] Y.-A. Song and W. Koenig, “Experimental Study of the Basic Process Mechanism for Direct Selective Laser Sintering of Low-Melting Metallic Powder,” *CIRP Annals*, vol. 46, no. 1, pp. 127–130, 1997.

[12] I. Yadroitsev, L. Thivillon, P. Bertrand, and I. Smurov, “Strategy of manufacturing components with designed internal structure by selective laser melting of metallic powder,” *Applied Surface Science*, vol. 254, no. 4, pp. 980–983, 2007.

[13] H. Bikas, P. Stavropoulos, and G. Chryssolouris, “Additive manufacturing methods and modelling approaches: a critical review,” *The International Journal of Advanced Manufacturing Technology*, vol. 83, no. 1-4, pp. 389–405, 2015.

[14] G. Tapia and A. Elwany, “A Review on Process Monitoring and Control in Metal-Based Additive Manufacturing,” *Journal of Manufacturing Science and Engineering*, vol. 136, no. 6, 2014.

[15] B. Lane et. al, “Design, Developments, and Results From the NIST Additive Manufacturing Metrology Testbed (AMMT),” in *Solid Freeform Fabrication 2016: Proceedings of the 27th Annual International Solid Freeform Fabrication Symposium - An Additive Manufacturing Conference*, pp. 1145–1160.

[16] S. A. Khairallah, A. T. Anderson, A. Rubenchik, and W. E. King, “Laser powder-bed fusion additive manufacturing: Physics of complex melt flow and formation mechanisms of pores, spatter, and denudation zones,” *Acta Materialia*, vol. 108, pp. 36–45, 2016.

[17] W. Gander, G. H. Golub, and R. Strebel, “Least-squares fitting of circles and ellipses,” *Bit*, vol. 34, no. 4, pp. 558–578, 1994.

[18] T. M. Moges, W. Yan, S. Lin, G. Ameta, J. C. Fox, and P. W. Witherell, “Quantifying Uncertainty in Laser Powder Bed Fusion Additive Manufacturing Models and Simulations,” in *Solid Freeform Fabrication 2018: Proceedings of the 29th Annual International Solid Freeform Fabrication Symposium*, pp. 1913–1928.

[19] S. S. Razvi, S. Feng, A. Narayanan, Y.-T. T. Lee, and P. Witherell, “A Review of Machine Learning Applications in Additive Manufacturing,” *Volume 1: 39th Computers and Information in Engineering Conference*, 2019.

[20] M. Aminzadeh and T. R. Kurfess, “Online quality inspection using Bayesian classification in powder-bed additive manufacturing from high-resolution visual camera images,” *Journal of Intelligent Manufacturing*, vol. 30, no. 6, pp. 2505–2523, 2018.

[21] M. Islam, T. Purtonen, H. Piili, A. Salminen, and O. Nyrhilä, “Temperature Profile and Imaging Analysis of Laser Additive Manufacturing of Stainless Steel,” *Physics Procedia*, vol. 41, pp. 835–842, 2013.

[22] C. Gobert, E. W. Reutzel, J. Petrich, A. R. Nassar, and S. Phoha, “Application of supervised machine learning for defect detection during metallic powder bed fusion additive manufacturing using high resolution imaging.,” *Additive Manufacturing*, vol. 21, pp. 517–528, 2018.

[23] Z. Yang, Y. Lu, H. Yeung, and S. Krishnamurty, “From Scan Strategy to Melt Pool Prediction: A Neighboring-Effect Modeling Method,” *Journal of Computing and Information Science in Engineering*, vol. 20, no. 5, 2020.

[24] Z. Yang, Y. Lu, H. Yeung, and S. Krishnamurty, “Investigation of Deep Learning for Real-Time Melt Pool Classification in Additive Manufacturing,” *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 2019.

[25] S. Clijsters, T. Craeghs, S. Buls, K. Kempen, and J.-P. Kruth, “In situ quality control of the selective laser melting process using a high-speed, real-time melt pool monitoring system,” *The International Journal of Advanced Manufacturing Technology*, vol. 75, no. 5-8, pp. 1089–1101, 2014.

[26] Y. Lu, S. Choi, and P. Witherell, “Towards an Integrated Data Schema Design for Additive Manufacturing: Conceptual Modeling,” *Volume 1A: 35th Computers and Information in Engineering Conference*, 2015.

[27] L. Ma, M. M. Crawford, and J. Tian, “Local Manifold Learning-Based k -Nearest-Neighbor for Hyperspectral Image Classification,” *IEEE Transactions on Geoscience and Remote Sensing*, 2010.

[28] M. Hasanlou and F. Samadzadegan, “Comparative Study of Intrinsic Dimensionality Estimation and Dimension Reduction Techniques on Hyperspectral Images Using K-NN Classifier,” *IEEE Geoscience and Remote Sensing Letters*, vol. 9, no. 6, pp. 1046–1050, 2012.

[29] R. Kohavi, “International Joint Conference on Artificial Intelligence (IJCAI),” in *A Study of Cross-Validation and Bootsrap for Accuracy Estimation and Model Selection*, 1995.

[30] Y. Liu, S. Li, H. Wang, W. Hou, Y. Hao, R. Yang, T. Sercombe, and L. Zhang, “Microstructure, defects and mechanical behavior of beta-type titanium porous structures manufactured by electron beam melting and selective laser melting,” *Acta Materialia*, vol. 113, pp. 56–67, 2016.

[31] M. Letenneur, A. Kreitcberg, and V. Brailovski, “Optimization of Laser Powder Bed Fusion Processing Using a Combination of Melt Pool Modeling and Design of Experiment Approaches: Density Control,” *Journal of Manufacturing and Materials Processing*, vol. 3, no. 1, p. 21, 2019.

[32] M. Letenneur, V. Brailovski, A. Kreitcberg, V. Paserin, and I. Bailon-Poujol, “Laser Powder Bed Fusion of Water-Atomized Iron-Based Powders: Process Optimization,” *Journal of Manufacturing and Materials Processing*, vol. 1, no. 2, p. 23, 2017.

[33] L. E. Criales, Y. M. Arısoy, B. Lane, S. Moylan, A. Donmez, and T. Özel, “Laser powder bed fusion of nickel alloy 625: Experimental investigations of effects of process parameters on melt pool size and shape with spatter analysis,” *International Journal of Machine Tools and Manufacture*, vol. 121, pp. 22–36, 2017.

[34] https://www.nist.gov/system/files/documents/el/isd/NISTAdd_Mfg_Report_FINAL-2.pdf

[35] Grasso, M., & Colosimo, B. M. (2017). Process defects and in situ monitoring methods in metal powder bed fusion: A review. *Measurement Science and Technology, 28*(4), 044005. doi:10.1088/1361-6501/aa5c4f

[36] https://www.astm.org/Standards/F2792.htm

[37] Herzog, D., Seyda, V., Wycisk, E., & Emmelmann, C. (2016). Additive manufacturing of metals. *Acta Materialia, 117,* 371-392. doi:10.1016/j.actamat.2016.07.019

[38] A. Vasinonta, J. L. Beuth, and M. L. Griffith, “Process maps for controlling residual stress and melt pool size in laser-based SFF processes,” in Solid Freeform Fabrication Proceedings, Austin, TX, 2000, pp. 200–208.

[39] G. B. M. Cervera and G. Lombera, “Numerical prediction of temperature and density distributions in selective laser sintering processes,” Rapid Prototyp. J., vol. 5, pp. 21–26, 1999.

[40] R. B. Patil and V. Yadava, “Finite element analysis of temperature distribution in single metallic powder layer during metal laser sintering,” Int. J. Mach. Tools Manuf., vol. 47, pp. 1069–1080, 6.

[41] T. Chen and Y. Zhang, “Numerical simulation of two-dimensional melting and resolidification of a two-component metal powder layer in selective laser sintering process,” Numer. Heat Transf. Part Appl., vol. 46, pp. 633–649, 2004.

[42] A. V. Gusarov, I. Yadroitsev, P. Bertrand, and I. Smurov, “Heat transfer modelling and stability analysis of selective laser melting,” Appl. Surf. Sci., vol. 254, no. 4, pp. 975–979, Dec. 2007.

[43] A. Hussein, L. Hao, C. Yan, and R. Everson, “Finite element simulation of the temperature and stress fields in single layers built without-support in selective laser melting,” Mater. Des., vol. 52, pp. 638–647, Dec. 2013.

[44] L. Dong, A. Makradi, S. Ahzi, and Y. Remond, "Three-dimensional transient finite element analysis of the selective laser sintering process," J. Mater. Process. Technol., vol. 209, pp. 700–706, Jan. 2009.

[45] J. Yin, H. Zhu, L. Ke, W. Lei, C. Dai, and D. Zuo, "Simulation of temperature distribution in single metallic powder layer for laser micro-sintering," Comput. Mater. Sci., vol. 53, no. 1, pp. 333–339, Feb. 2012.

[46] M. Shiomi, A. Yoshidome, F. Abe, and K. Osakada, "Finite element analysis of melting and solidifying processes in laser rapid prototyping of metallic powders," Int. J. Mach. Tools Manuf., vol. 39, no. 2, pp. 237–252, 1999.

[47] R. Li, Y. Shi, J. Liu, H. Yao, and W. Zhang, "Effects of processing parameters on the temperature field of selective laser melting metal powder," Powder Metall. Met. Ceram., vol. 48, no. 3–4, pp. 186–195, Mar. 2009.

[48] L. Wang, S. D. Felicelli, and J. Craig, "Thermal modeling and experimental validation in the LENS process," in Solid Freeform Fabrication Proceedings, Austin, TX, 2007, pp. 100–111.

[49] M. Pavlov, M. Doubenskaia, and I. Smurov, "Pyrometric analysis of thermal processes in SLM technology," Phys. Procedia, vol. 5, pp. 523–531, 2010.

[50] J.-P. Kruth, J. Duflou, P. Mercelis, J. Van Vaerenbergh, T. Craeghs, and J. De Keuster, "On-line monitoring and process control in selective laser melting and laser cutting," in Proceedings of the Laser Assisted Net Shape Engineering 5 (LANE), 2007, vol. 1, pp. 23–37.

[51] T. Craeghs, F. Bechmann, S. Berumen, and J.-P. Kruth, "Feedback control of Layerwise Laser Melting using optical sensors," Phys. Procedia, vol. 5, pp. 505–514, 2010.

[52] T. Craeghs, S. Clijsters, E. Yasa, and J.-P. Kruth, "Online quality control of selective laser melting," in Solid Freeform Fabrication Proceedings, Austin, TX, 2011, pp. 212–226.

[53] T. Craeghs, S. Clijsters, J.-P. Kruth, F. Bechmann, and M.-C. Ebert, "Detection of Process Failures in Layerwise Laser Melting with Optical Process Monitoring," Laser Assist. Net Shape Eng. 7 LANE 2012, vol. 39, no. 0, pp. 753–759, 2012.

[54] Y. Ning, Y. Wong, J. Y. Fuh, and H. T. Loh, "An approach to minimize build errors in direct metal laser sintering," Autom. Sci. Eng. IEEE Trans. On, vol. 3, pp. 73–80, 2006.

[55] P. Aggarangsi, J. L. Beuth, and M. L. Griffith, "Melt pool size and stress control for laser-based deposition near a free edge," in Solid Freeform Fabrication Proceedings, Austin, TX, 2003, pp. 196–207.

[56] G. Bi, A. Gasser, K. Wissenbach, A. Drenker, and R. Poprawe, “Characterization of the process control for the direct laser metallic powder deposition,” Surf. Coat. Technol., vol. 201, no. 6, pp. 2676–2683, Apr. 2006.

[57] J.-P. Kruth, M. Badrossamay, E. Yasa, J. Deckers, L. Thijs, and J. Van Humbeeck, “Part and material properties in selective laser melting of metals,” in Proceedings of the 16th International Symposium on Electromachining, Shanghai, China, 2010.

[58] Guo, Q., Zhao, C., Qu, M., Xiong, L., Escano, L. I., Hojjatzadeh, S. M., . . . Chen, L. (2019). In-situ characterization and quantification of melt pool variation under constant input energy density in laser powder bed fusion additive manufacturing process. *Additive Manufacturing*, *28*, 600-609. doi:10.1016/j.addma.2019.04.021

[59] https://scikit-learn.org/stable/modules/cross_validation.html