8-2023

In Situ Process Monitoring and Machine Learning Based Modeling of Defects and Anomalies in Wire-Arc Additive Manufacturing

Eduardo Miramontes
University of Tennessee, Knoxville, emiramon@vols.utk.edu

Recommended Citation

Miramontes, Eduardo, "In Situ Process Monitoring and Machine Learning Based Modeling of Defects and Anomalies in Wire-Arc Additive Manufacturing. " Master's Thesis, University of Tennessee, 2023.
https://trace.tennessee.edu/utk_gradthes/9971

To the Graduate Council:

I am submitting herewith a thesis written by Eduardo Miramontes entitled "In Situ Process Monitoring and Machine Learning Based Modeling of Defects and Anomalies in Wire-Arc Additive Manufacturing." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mechanical Engineering.

Bradley Howell Jared, Major Professor

We have read this thesis and recommend its acceptance:

Tony Schmitz, Anahita Khojandi

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:
I am submitting herewith a thesis written by Eduardo Miramontes entitled "In Situ Process Monitoring and Machine Learning Based Modeling of Defects and Anomalies in Wire-Arc Additive Manufacturing." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mechanical Engineering.

Bradley Howell Jared, Major Professor

We have read this thesis
and recommend its acceptance:

Tony Schmitz

Anahita Khojandi

Accepted for the Council:

Robert Hinde

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

In Situ Process Monitoring and Machine Learning Based Modeling of Defects and Anomalies in Wire-Arc Additive Manufacturing

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Eduardo Miramontes

August 2023

This Thesis is dedicated to Eduardo Miramontes Sr. and Maria Evans, my father and mother, who did everything they could to make this possible...

Acknowledgements

I would like to thank my advisor, Dr. Jared, for being a supportive mentor, and giving me direction when it was most needed. Joshua Penney, for teaching me many of the technical skills that made this thesis possible. Devon Goodspeed, for his insights on welding which proved invaluable to the completion of this work. Shems Eddine Bellhout for being a resource on the process of writing a thesis. Steven Williams for having the patience to be my lab safety buddy.

Somewhere, something incredible is waiting to be known

-Carl Sagan

Abstract

Wire Arc Additive Manufacturing (WAAM) has made great strides in recent years however, there remain numerous persistent challenges still hindering more widespread adoption. Defects in the parts produced degrade their mechanical performance. Inconsistency in the geometry of the weld beads or undesirable anomalies such as waviness, or humps can lead to loss of geometric accuracy and in extreme cases, when anomalies propagate to subsequent layers, build failure. Such defects can be mitigated by a controls framework, which would require a model that maps undesirable outcomes to information about the process that can be obtained in real time. This thesis explores the development of a multi-sensor framework for real time data acquisition and several approaches for arriving at such a model, employing well known machine learning methodologies including Random Forests, Artificial Neural Networks and Long Short Term Memory. The merits and drawbacks of these methods is discussed, and a physics based approach intended to mitigate some of the drawbacks is explored. The models are trained first on data obtained on a single build layer, and subsequently on a multi-layer wall.

Table of Contents

List of Tables

List of Figures

Nomenclature

$\bar{}$	Denotes a dimensional variable
ψ	Streamfunction
θ	Tangential coordinate
p	Pressure
r	Radial coordinate
u_r	Radial velocity
u_z	Axial velocity
u_θ	Tangential velocity
z	Axial coordinate

Chapter 1

Introduction

Additive manufacturing continues to grow apace, with the value of the industry projected to grow from \$16 billion this year to \$44 billion by 2027 — an average annual growth rate of 21% [1]. As the technology continues its rapid development, the range of capabilities continues to expand. The first commercially available systems only printed polymers and were only considered suitable for prototyping, hence the moniker *rapid prototyping*. Currently, systems capable of producing end-use parts in a diverse range of materials, including metals, composites, ceramics and organic material are commonplace. Metal additive manufacturing in particular is achieved by focusing an energy source to melt the feedstock material (typically powder or wire) in a path determined by slicing software. The standard commercially available metal additive process is powder based. Typically, powder based processes employ a highly focused laser or electron beam as the energy source. The most common powder based process is a powder bed process in which the beam scans the surface of a powder bed in the shape of a cross section of a CAD model and fuses the metal onto the previous layer. Then, the entire powder bed is shifted down by a constant offset and new powder is raked across the surface for the next layer.

Laser powder bed processes are a well developed and maturing class of additive manufacturing. It enables the fabrication of very complex parts with good mechanical properties. However, there are significant drawbacks. The process requires either a vacuum chamber for an electron beam, or an inert environment in the case of a laser beam. Both cases necessitate that the build take place inside a chamber, which severely limits the size of the parts that can be printed. Furthermore, material deposition rates are sluggish, with deposition rates ranging from 60-180 cm^3/hr [2]. The capital and operation costs are also very high and typically number in the millions of dollars.

These limitations are overcome by Wire Arc Additive Manufacturing (WAAM). WAAM, a relative newcomer to the metal additive field, employs a different deposition mechanism from powder based processes. The energy source that melts the metal is a Gas Metal Arc Welding (GMAW) torch instead of a laser or electron beam, while the feedstock is wire instead of powder. GMAW is a standard, and prevalent welding process that employs an electric arc to melt metallic wire that is fed through a torch. An inert environment is required, as in powder based processes, however, it only needs to be maintained locally, in the vicinity of the deposition. This can be achieved by blowing inert gas, known as the shielding gas, into the deposition zone, as shown in Figure 1. In WAAM, the GMAW torch is generally attached to a robotic manipulator, which is programmed to deposit material in a path corresponding to cross sectional layers of a CAD model.

By eliminating the need for an inert chamber, WAAM greatly expands the size of parts that can be built. In principle, the only limit is the reach of the robotic manipulator, though even this can be mitigated by adding more robotic manipulators. This opens the door to truly Large Area Additive Manufacturing (LAAM), which involves the manufacture of large scale parts that can be used for the automotive, aerospace, construction industries, among many others. An example of a large scale part fabricated by WAAM, an excavator arm produced at Oak Ridge National Labs, is shown in Figure 1.2. Such a part can be produced far quicker than powder based processes. The arm required 5 days of nonstop printing. Though lengthy, this is a huge improvement over powder bed printing [3].

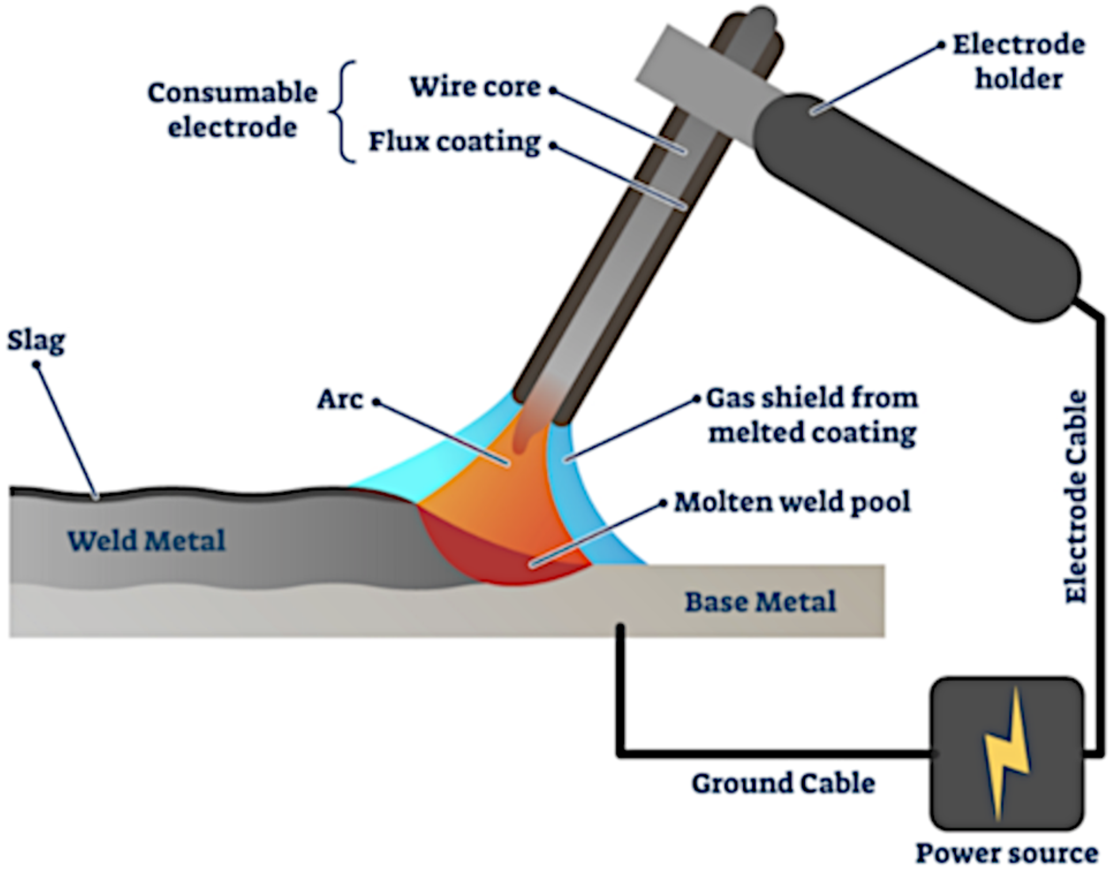


Figure 1.1: Diagram of GMAW process

While its potential is great, WAAM continues to held back by a number of significant drawbacks. These drawbacks can be categorized into two broad types: geometric inaccuracy, and mechanical defects. Geometric inaccuracy occurs in WAAM because it is characterized by high deposition rates and low resolution, which make it difficult to precisely control the geometry of finished parts. The shape of the individual deposition paths, known as beads, then becomes a limiting factor in the geometric accuracy of finished parts. Beads in WAAM are relatively large compared to powder based processes, which contributes to the high deposition rate, but on the flip side also causes low resolution. Furthermore, welding is a complex multi-physics process and as such is vulnerable to instabilities, which manifest as irregularities in the shape of the beads. WAAM beads commonly exhibit a number of irregularities such as humping, waviness, and uneven dimensions. These irregularities contribute to the low resolution of the process and loss of tolerance, and in some cases they can propagate to subsequent layers, become magnified, and lead to part failure.

The mechanical defects typically found in WAAM are porosity, lack of fusion, residual stress, and cracking. Porosity is the formation of regularly shaped spherical gas pores that are caused by material contamination or process instability. Voids in the part can also be classified as lack of fusion, which is irregularly shaped and results from insufficient melting between adjacent layers or deposition paths. Both porosity and lack of fusion can significantly reduce mechanical strength. These thermal cycles induce thermal expansion and contraction, which creates internal stresses inside the bulk geometry, known as residual stress. Residual stress leads to deformation, and in cases where it locally exceeds the tensile strength of the material, creates cracks. Cracks become weak points in the part which can expand and propagate when loaded, ultimately resulting in fracture. Collectively, these mechanical defects result in parts with mechanical properties inferior to conventional manufacturing, and exacerbate the geometric inaccuracy problem. This presents a formidable obstacle to the development of WAAM.

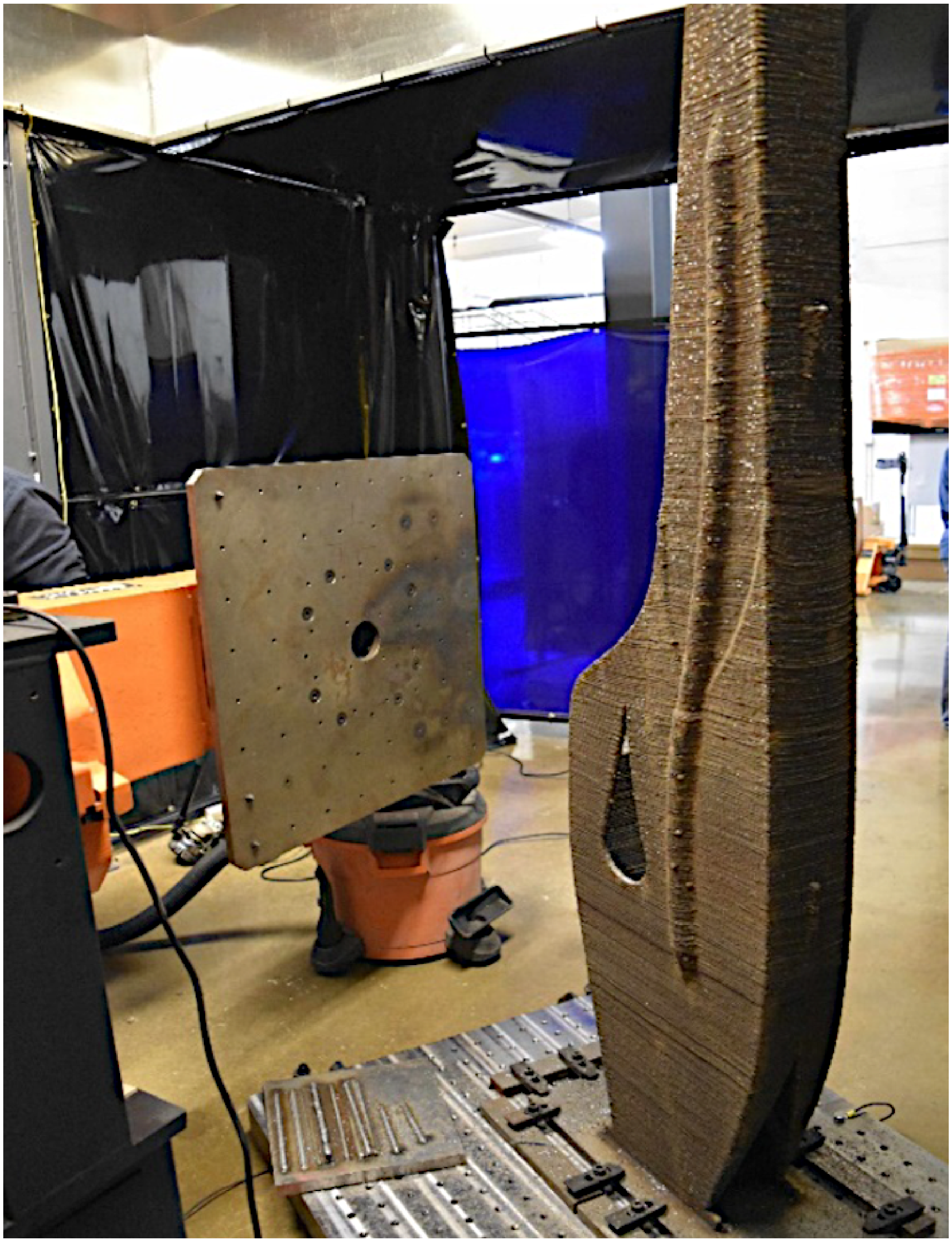


Figure 1.2: Large scale excavator arm manufactured by WAAM process [3]

Mitigating or even eliminating these defects requires three fundamental steps: sensing, prediction/detection and control. Sensing is the acquisition of relevant information that describes the present state of the process. As noted, WAAM is a complex, multi-scale, multiphysics process. Furthermore, most of the assortment of sub-processes involved e.g. arc ignition, droplet detachment, melt pool solidification, heating cycles, etc are thought to make significant contributions to the shape and mechanical properties of the finished part. Fully describing the state of the process at any given time requires knowledge of a multitude of physical quantities. This can only be accomplished by employing a variety of sensors designed to acquire information on corresponding physical sub-processes. The choice of sensors is of course informed by an fundamental understanding of the physics involved. Monitoring the arc necessitates voltage and current sensors, monitoring the thermal evolution of the whole part and the melt pool itself requires infrared imaging, and so on. Since the foundation of the process is welding, the experience of expert welders should be taken into account. Expert welders rely on sensory cues such as sound to inform their movements. This suggests that sensing modalities such as acoustic monitoring could provide valuable information on the stability of the process, and by extension the quality and consistency of material deposition.

The information obtained in the first step must then be mapped to the undesirable outcomes that one seeks to prevent, which is the prediction/detection step. This necessitates the development of a model capable of converting inputs extracted from the raw data into quantitative outputs representing the presence or lack thereof of the undesirable outcomes. Implicit in this step is the necessity of a data processing procedure. The amount of raw data generated by a comprehensive multi-sensor framework should be expected to be enormous. This large collection of data must be reduced by a procedure that extracts features from the data expected to be predictive. The model itself can be produced in multiple ways. One approach is physics based modeling which employs the fundamental physical laws that describe the process to generate computational models. This approach has serious drawbacks. Due to the complexity involved, an accurate model is not guaranteed, and producing an accurate model is very computationally intensive, which precludes deployment in real time control applications. A common alternative is machine learning, which is a powerful category of methods that has proven its worth in a wide variety of applications. Machine learning models, which while often computationally expensive on the front end, once trained can be deployed to production extremely fast, enabling real time prediction

Finally, if the state of the process is sufficiently well known and an accurate model mapping said state to the desired outcome has been developed, a controls framework can be developed. This would involve using the difference between the actual outcome and the desired outcome as an actuation signal for real time interventions. Such frameworks have been attempted for additive manufacturing processes, and WAAM in particular. Developing such a framework is beyond the scope of this work, however, an example of what can be accomplished with such a framework can be seen in Penney (2022) [4].

This work seeks to develop an approach for accomplishing the first two steps of that process: sensing and prediction. The first step is accomplished by developing a multi-sensor process monitoring framework that synchronously collects process relevant information while minimizing human intervention. The second step is accomplished by developing a data processing procedure that extracts useful information from the large trove of raw data. Features, or descriptive quantities are derived from computations performed on the raw data. These features are then mapped to outcomes of interest by employing different machine learning methodologies. This work will focus on modeling shape irregularites and porosity. The performance of the different machine learning methodologies will be assessed and compared. This was accomplished in two stages. The first is limited to bead on plate experiments, which were used to develop a framework and methodology for accomplishing the task. The lessons thus learned were then applied to a full build: a thick wall with infill pattern.

Chapter 2

Background

2.1 Wire Arc Additive Manufacturing

2.1.1 Retrospective

The concept that would become WAAM was first proposed in a 1925 patent filed by Ralph Baker. The patent described a method for forming decorative and ornamental objects by superimposing layers of metal deposited by a fusible metal electrode connected to a welding circuit. Baker highlighted the suitability of the method for mechanical control, which could be employed to create parts of predetermined, uniform shape [5]. This thread would be picked up by subsequent engineers to form the basis for WAAM.

Beginning in the 1980s, German manufacturing companies sought to find alternatives to forging for manufacturing large shafts, rotors, and turbines. They developed a solution known as shape welding (SW) which involves building large objects with arc welding deposition. Blohm+Voss Industries GmbH was able to manufacture a 79 ton cylinder in 6 weeks. Thyssen AG constructed a facility that employed multiple tandem weld heads to manufacture large cylindrical shells up to 11m long and 5.8 m in diameter [6]. Schmidt (1990) noted that for simple geometries, SW could not be seen as a serious alternative to forging unless a very high material deposition rate could be achieved. There was a clearer advantage in more complex geometries, where the impact of material deposition rate is diminished [7].

At this early stage, the need for a robust process monitoring and control framework was already recognized. One of the applications of SW was in the nuclear industry which is subject to strict standards necessitating reliable fabrication of parts with consistent mechanical properties. Kußmaul et. al (1983) described efforts to implement process sensing and control at the SW facility constructed by Thyssen-Krupp for manufacturing pressure vessels. The welding current, and arc power were continuously monitored, and a height sensing control was implemented to control the Contact To Workpiece Distance (CTWD) [6].

Research conducted at the Welding Engineering Research Centre (WERC) in Cranfield University produced a system with most of the components now considered key elements of WAAM: a robotic welding system, a CAD model slicing routine, and trajectory generation [8]. While this research was intended to develop a rapid prototyping system, process control, something not explored in their work, enables the same setup to build end use parts.

2.1.2 Gas Metal Arc Welding

GMAW is central to WAAM. In GMAW, a wire is continuously fed through a torch at a rate called the wire feed speed. Electrical current is passed through the wire while it is being fed through the torch which creates an electric arc between the wire and the work-piece. The electric arc heats up the tip of the wire up to its melting point at which point a droplet of molten metal begins to form. The droplet grows until the force of gravity overcomes the surface tension attaching it to the wire, at which point it is deposited onto the plate forming a small pool of molten metal called the melt pool. There are three primary mechanisms for transferring the droplet to the plate. The three main transfer modes are globular, spray and short circuit. It is the energy input that primarily determines which mode occurs.

Spray transfer requires the most energy. During spray transfer, high energy heats the droplet well past its melting point, which reduces the surface tension and makes it detach from the wire quickly. This results in small droplets being transferred to the work-piece surface at a high frequency. The next step down in energy is globular transfer, which follows a similar mechanism to spray transfer except that the lower energy means droplets grow bigger and detach at a slower rate. Thus, a single large droplet often with irregular shape forms on the electrode, and is pulled down by the force of gravity. The final transfer mode with the lowest energy requirement is short circuit. Short circuit is similar to globular in that droplets grow slowly but the energy input is so low that the droplet doesn't fall and instead bridges the gap between the electrode and the work-piece. The droplet contacting the work-piece results in a short circuit and an associated spike in current that finally causes the droplet to detach. Short circuit transfer is considered ideal for WAAM due to the low heat input [9].

A further improvement on short circuit transfer mode is to retract the wire during the short circuit, which suppresses the spike in current lowering the heat input still further.This is known as Cold Metal Transfer (CMT) which was developed by Fronius in 2004[10]. The effect of this mechanical intervention on the voltage and current waveforms of the arc is shown in Figure 2.1. The cycle has three distinct stages: peak current phase, background current phase and short circuit phase. During the peak current phase, the arc voltage is constant and a high current pulse heats the electrode to create a droplet. During background current phase, the current is decreased to prevent globular transfer. During the short circuit phase, the voltage is brought down to zero, and the wire feeder retracts the wire. This process has attracted keen interest for WAAM because of the lower heat input.

2.1.3 Process Physics

The physics underlying WAAM are highly complex, as they exhibit non-linear behavior, span multiple scales and draw upon multiple types of physics. Most of the mechanisms dominant in WAAM are the same as those behind GMAW. Fully describing that process requires coupling continuity equations of mass, charge, momentum, energy, Maxwell's equations and the generalized Ohm's Law in three dimensions [11]. These equations are all non-linear and can only be solved by computationally intensive numerical methods. WAAM further complicates the process because the shape of the beads and the microstructure of the deposited material are of paramount importance. The shape and mechanical characteristics of the deposited weld bead are affected by a number of forces such as gravity, surface tension and, electromagnetic force, arc pressure, and convective flow [12, 13].

Furthermore, previously deposited layers become the substrate for newer layers, which exerts a strong influence on their shape and material properties. A disturbance in one layer can create an uneven surface for the next layer, which can propagate and magnify the disturbance, leading to process instability and in extreme cases part failure [14]. Conversely, new layers exert an influence on previous ones because the bulk build volume is reheated every time new material is melted on top of it. This leads to thermal accumulation throughout the part which has detrimental effects such as waviness, increasing layer width, and modification of metal transfer mechanism. The repeated cycles of heating and cooling also have detrimental effects on the microstructure and lead to geometric changes on the part [15].

2.2 Process Monitoring

Process monitoring has been identified as a key to the success of metal additive manufacturing, and WAAM in particular. Because of WAAM's novelty there has been little work done documenting the range of available process monitoring techniques until relatively recently. However, GMAW has existed for decades and certain process monitoring techniques for that process are regarded as mature. Other methods are more novel both in GMAW and WAAM, as shown in Figure 2.2 [16].

Arc welding current and voltage have long been employed to measure arc stability and abnormal welding conditions such as contamination and low shielding gas[17]. More recently, arc signals have been employed to detect porosity in single pass weld beads. Thekkuden et. al correlated fluctuations in the arc current and voltage to instability from porosity using probability density functions and control charts [18]. Changes in the arc signature such as a broadening of the arc cycle have been tied to unstable deposition in WAAM, which is reflected in crooked deposition paths and waviness along the build direction [19].

Acoustic monitoring is a promising avenue because it is a low cost, non-contact method. It has primarily been explored to monitor the quality of welds in GMAW. Grad et. al used characteristics of the sound signal such as the signal kurtosis and the shape of the peak in the power spectral density function corresponding to the droplet detachment frequency to detect arc instability [20]. Pal et. al correlated kurtosis and sound pressure to metal transfer mode [21]. Research on application in WAAM is scant but its potential has been recognized by multiple researchers [16, 22].

Thermography has been in use for laser DED processes for some time now. Thermography has been used to monitor solidification and process deviations in SLM [23]. Mireles et. al used an infrared camera to observe purposefully seeded pores in LPBF process. They were able to successfully identify defects larger than 600μm [24]. In WAAM, Baier et. al used a thermographic line sensor to obtain temporally and spatially dependent temperature profiles which enabled them to detect geometric irregularities [25].

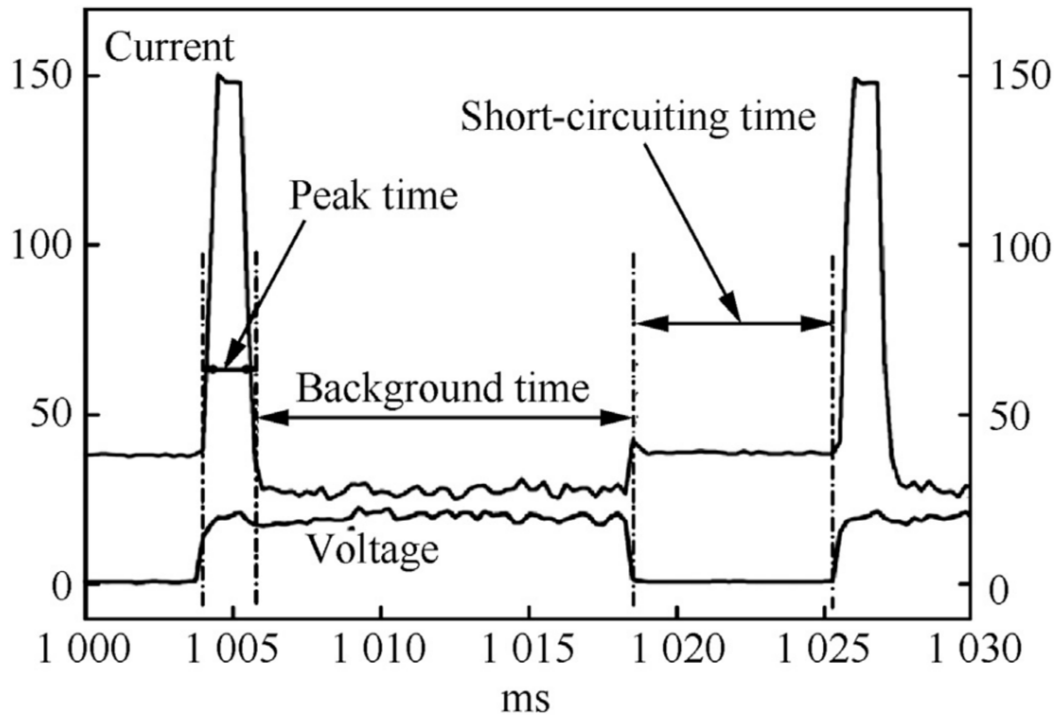


Figure 2.1: Voltage and current waveforms in CMT process [10]

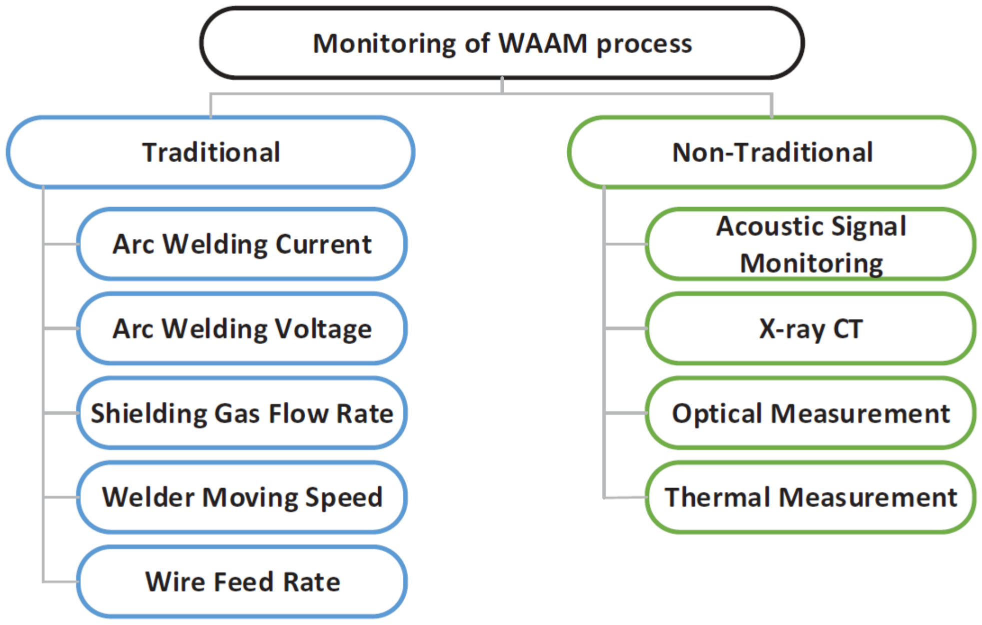


Figure 2.2: Types of process monitoring in WAAM [16]

2.3 Modeling

The central importance taken by identifying defects and irregularities in WAAM has prompted the development of models capable of mapping sensing data to these outcomes. The approaches taken fall into one of two camps: computational models and machine learning models. Interest in both approaches for WAAM is relatively recent so at times the literature must be supplemented with models developed for GMAW.

Ou (2020) developed a complete heat transfer and fluid flow model for multi-track deposition. By coupling equations of momentum, mass, and energy in three-dimensions they were able to model the evolution of deposit geometry [12]. Cadiou et. al (2020) developed a electromagnetic, heat transfer and fluid flow model specifically for WAAM with a CMT process. While they accomplished the task of solving the equations to model the dynamics at the melt pool label, their findings underscore the limits of the approach – the model took 15 days to simulate 6 seconds [26].

Machine learning models have also been explored. Wang et. al (2020) developed a model using analysis of variance (ANOVA) for predicting bead geometry in terms of height, width and depth of penetration based on process settings such as wire feed speed, substrate temperature and travel speed [27]. An Artificial Neural Network (ANN) model was employed in a similar approach to predict dimensions and contact angle in a CMT based process [28]. Panda et. al employ a genetic programming approach to map peak current, wire feed and travel speed to bead geometry [29]. These studies all construct a model by mapping process settings to geometric outcomes measured as averages which has the disadvantage of excluding real time process fluctuations from the model. Tang et. al. (2022) takes a process monitoring oriented approach by employing sensing data such as voltage, current, melt pool temperature and wire feed speed to construct a deep learning model for predicting the width and height of weld beads [30]. Penney (2022) utilizes arc voltage and arc current and a SVM model to predict contact to workpiece distance (CTWD), and by extension, layer height in WAAM. This model then formed the basis for a layer height control framework [4].

The aforementioned publications deal with models for bead geometry. Models that predict mechanical defects in WAAM are rarer, but some work has been performed in this sphere. Li et al (2022) employed a YOLOv3 algorithm, a variant of (Convolutional Neural Networks) CNNs that allows for localized object detection, to identify lack of fusion voids immediately after a layer has been deposited [31]. He et al (2021) used magneto-optical imaging to train a CNN to recognize cracks, lack of fusion and depressions in the deposition [32]. Jamnikar et al (2022) trained a CNN to predict microstructural characteristics such as alpha lath thickness and beta grain size based on visual, thermal, positional chemical and acoustic data [33]. It's also worth noting some of the approaches developed for LBPF systems, which have potential for application in WAAM. Smoqi et al (2022) developed several physics informed machine learning models including (K-means Nearest Neighbor) KNN, Support Vector Machine (SVM), Linear Regression (LR) and CNN to predict porosity based on physically relevant parameters that were obtained from melt pool signatures. This work is also notable for its experimental methodology, which used one large part with process conditions that vary along the build height. In this way, a large data set was obtained from a single build [34].

Chapter 3

Experimental Methods

3.1 WAAM System

The Wire-Arc Additive System employed in this work consists of a 6-axis KUKA KR-6 robotic manipulator fitted with a Fronius CMT capable weld torch. The torch is powered by a Fronius TransPuls Synergyic 5000 CMT Advanced power supply. The power supply can be toggled between different modes including Standard, CMT, and Pulsed, of which only the former two are investigated in this work. A mixture of 98% Ar and 2% CO2 was used for shielding gas. The welding torch is cooled by a 50/50 mixture of water and ethylene glycol. The feedstock is 0.035” AWS ER70S-3 Mild Carbon Steel welding wire made by Hobart. Material is deposited onto a 6”x12”x0.5” low carbon steel substrate which is secured to a worktable by clamps. The welding system is shown in Figure 3.1. For single pass weld studies, the tool path was programmed by jogging the robot to the desired positions and manually teaching the points. This is done for one bead, and all further beads are programmed by transforming the initial template. Multi-layer parts are made by resolving the CAD model into points, slicing the points into layers, and importing the result into Octopuz, an offline robot programming software.

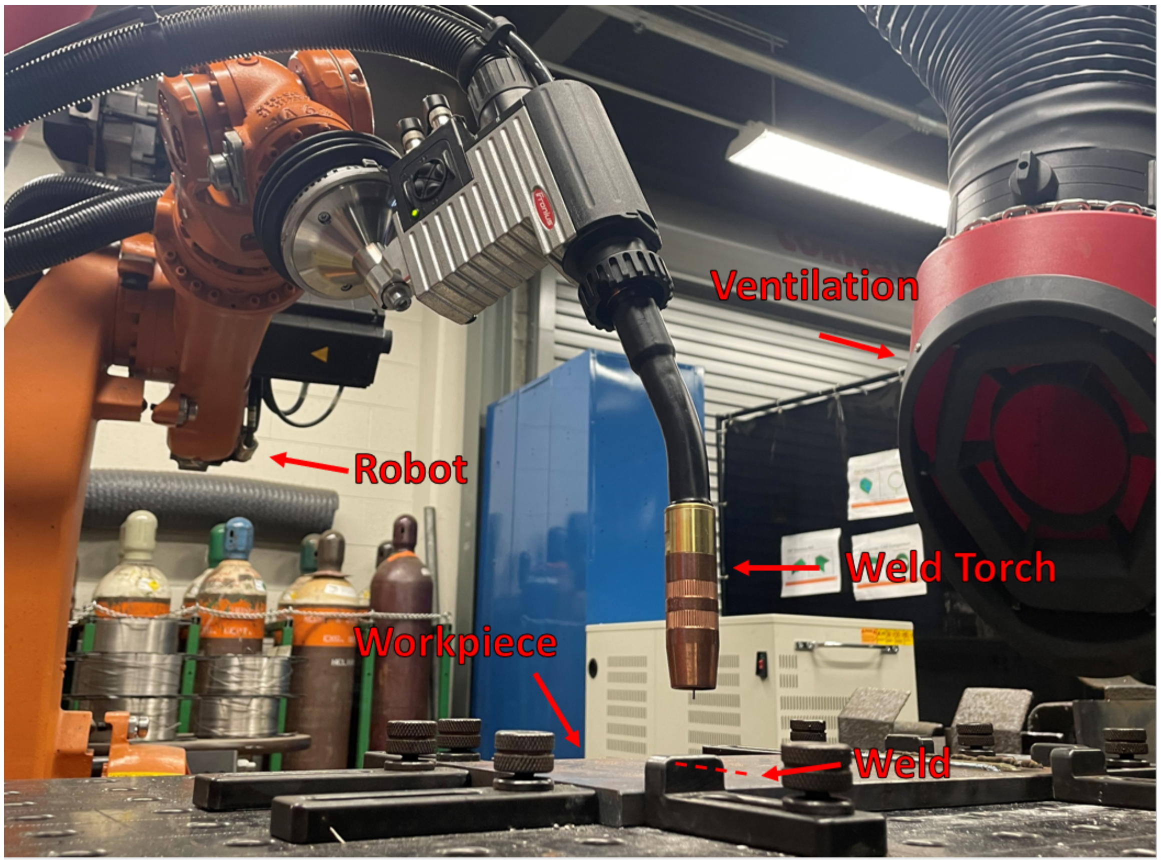


Figure 3.1: University of Tennessee's WAAM Development Cell

3.2 Multi-sensor Framework

A multi-sensor framework for online monitoring was developed for the University of Tennessee's WAAM development cell. The objective was to capture as much physically meaningful information about the process as possible. Arc sound, arc voltage and current, melt pool temperature, temperature distribution, wire feed speed, robot position and velocity were all monitored by various sensors. A diagram of the multi-sensor set-up and of the data streams collected by each respective sensor is shown in Figure 3.2. This suite of sensors generated an enormous amount of data. This data was subsequently converted into features by methods described in Chapter 4. After the pre-processing and feature extraction stages are completed, the data ultimately provided the inputs to the predictive models for bead shape and porosity.

3.2.1 Acoustics

The sound emitted by the weld arc is captured by a PCB 378B02 pre-polarized free-field omni-directional microphone. The raw analog signal is digitized by a PCB 485B39 digital ICP-USB signal conditioner. The audio recording system is shown in Figure 3.3. Sound is sampled at 48kHz, which is considered standard in the recording industry. This allows frequencies up to 24kHz to be captured, as per the Nyquist Sampling theorem – above the limit of human hearing.

3.2.2 Arc Measurements

The voltage and current waveforms of the welding arc were recorded by a Miller LEM Box, shown in Figure 3.4. To measure current, a lead was installed in series with the hot cable on the welding power source, while the other lead was connected to ground. Voltage was measured by connecting a pair of leads in parallel to the hot and ground cables on the power source. The waveforms were sampled at 20kHz, which is sufficient to capture the cycles corresponding to droplet formation and detachment.

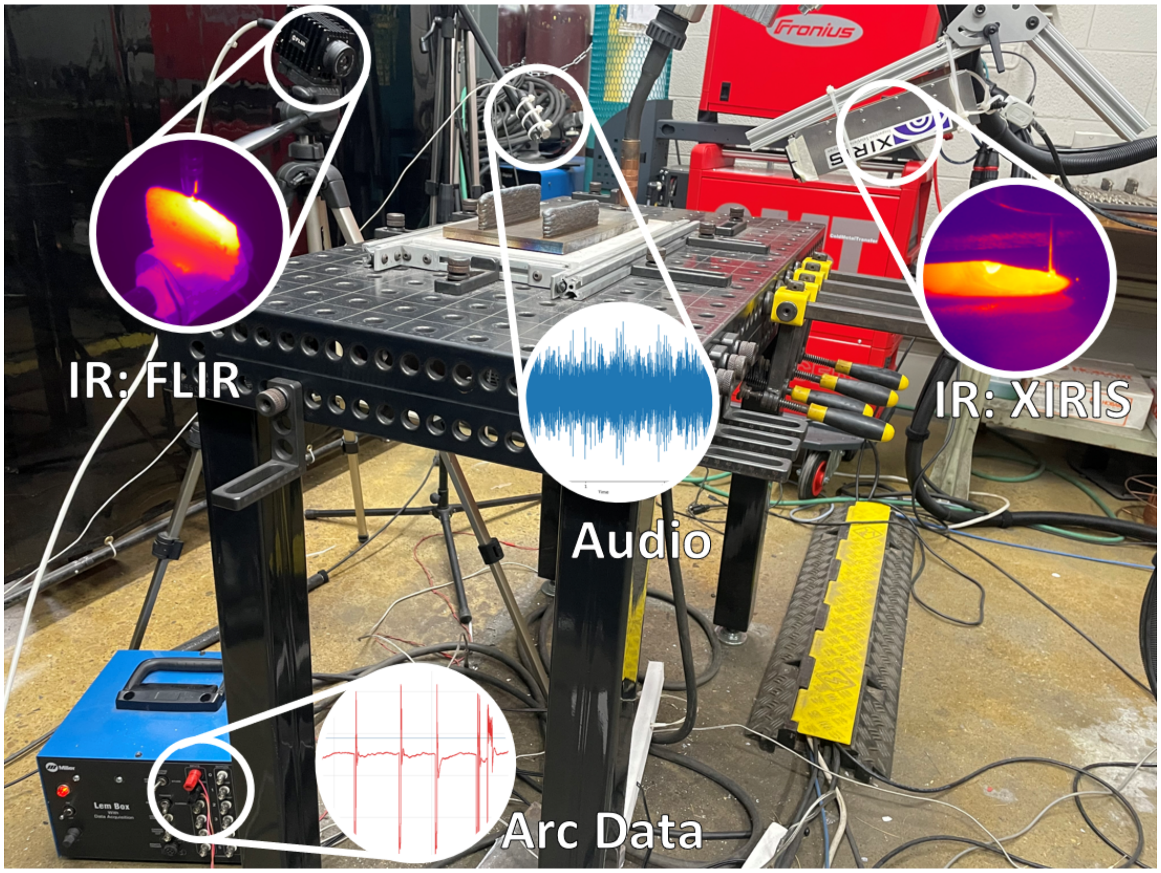


Figure 3.2: Multi-sensor framework

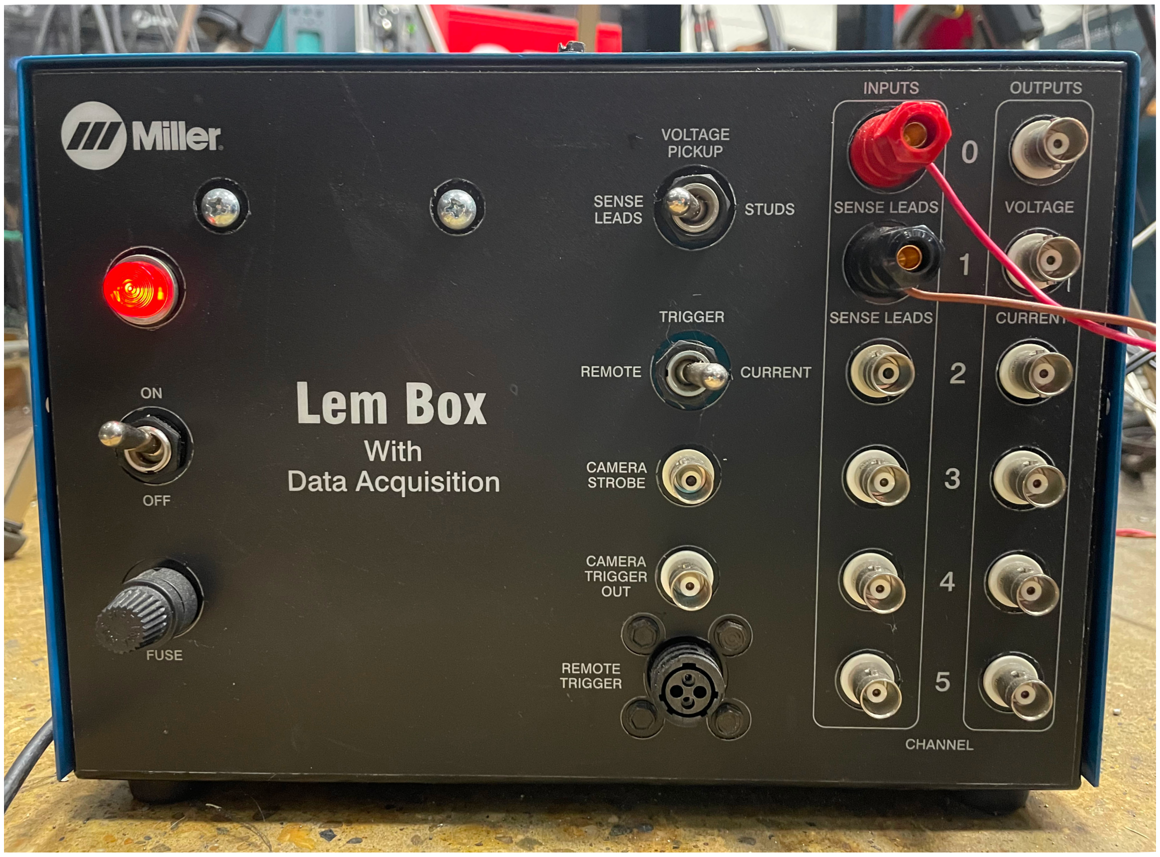
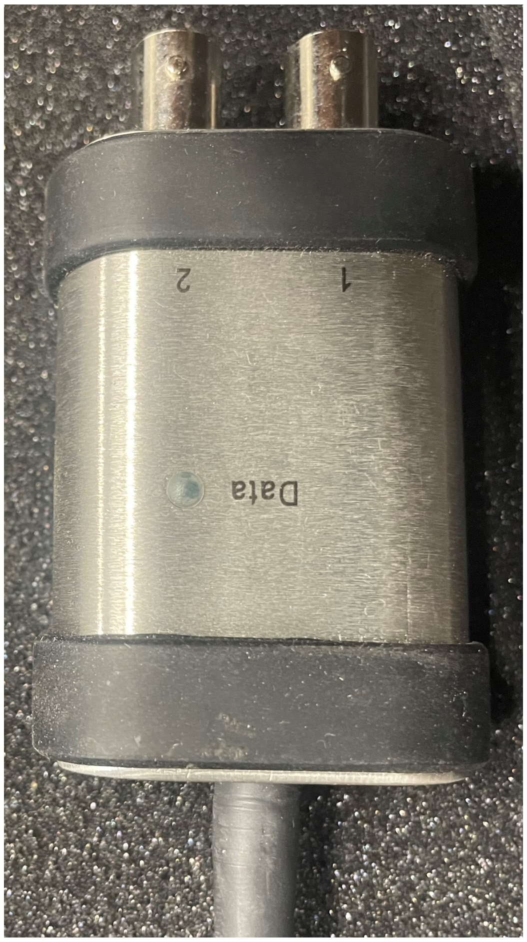
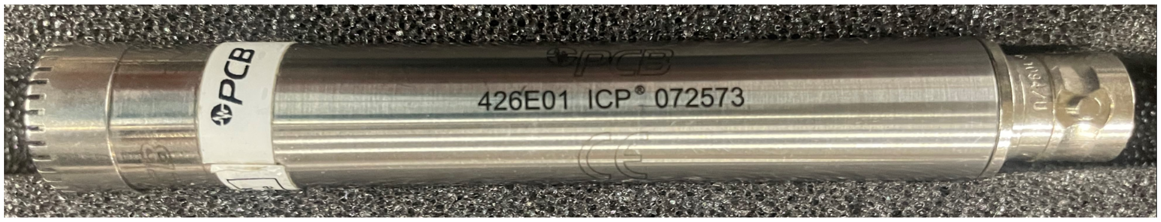


Figure 3.4: LEM Box



(a) Digital ICP-USB Signal Conditioner



(b) Omni-directional free field condenser microphone

Figure 3.3: Audio monitoring system

3.2.3 Thermography

The importance of thermal cycles to the quality of WAAM parts has been well documented [13,15]. Therefore, two infrared cameras were installed on the system to monitor the thermal behavior of the process. The FLIR A50, shown in Figure 3.5a, can detect temperatures from -20°C to 1000°C, and has an image resolution of 463 x 348 pixels. This temperature range does not allow for any meaningful information to be obtained in the vicinity of the arc, including the melt pool, therefore the primary use for this camera was to measure bulk temperature, the temperature of the substrate, and inter-layer cooling rates.

Steel melts at 1300°, so monitoring required a camera with a higher temperature range. That role was served by the XIRIS XIR-1800 Short Wave Infrared camera, shown in Figure 3.5b, which can capture temperatures from 350°C to 1800°C. Its image resolution is 640 x 512 pixels. Initially, the camera was stationary as it was mounted on the worktable, however, that mounting configuration proved undesirable because the camera provides low quality images of objects farther away than its working distance of 300 mm. Instead, the camera was mounted fixed relative to the weld torch, enabling to image the melt pool during robot motion. The arm that the camera attaches is adjustable on its azimuthal angle, which was utilized to provide different viewing angles. This torch tracking mounting configuration is shown in Figure 3.6.

After trial and error, it was determined that a slightly overhead view of the melt pool provided the optimal perspective. Since the camera is stationary relative to the torch tip position, the travel direction of the robot impacts how much of the melt pool region can be seen by the camera. When the robot is moving towards the camera, the camera faces the front of the melt pool at all times, which provides excellent visibility. Whereas when the robot moves away from the camera, the camera lags behind the melt pool by several inches, and can only see the back portion. When the robot travels in a direction perpendicular to the direction the camera points in, the side

(a) FLIR A50

(b) XIRIS XIR-1800

Figure 3.5: Infrared Sensors

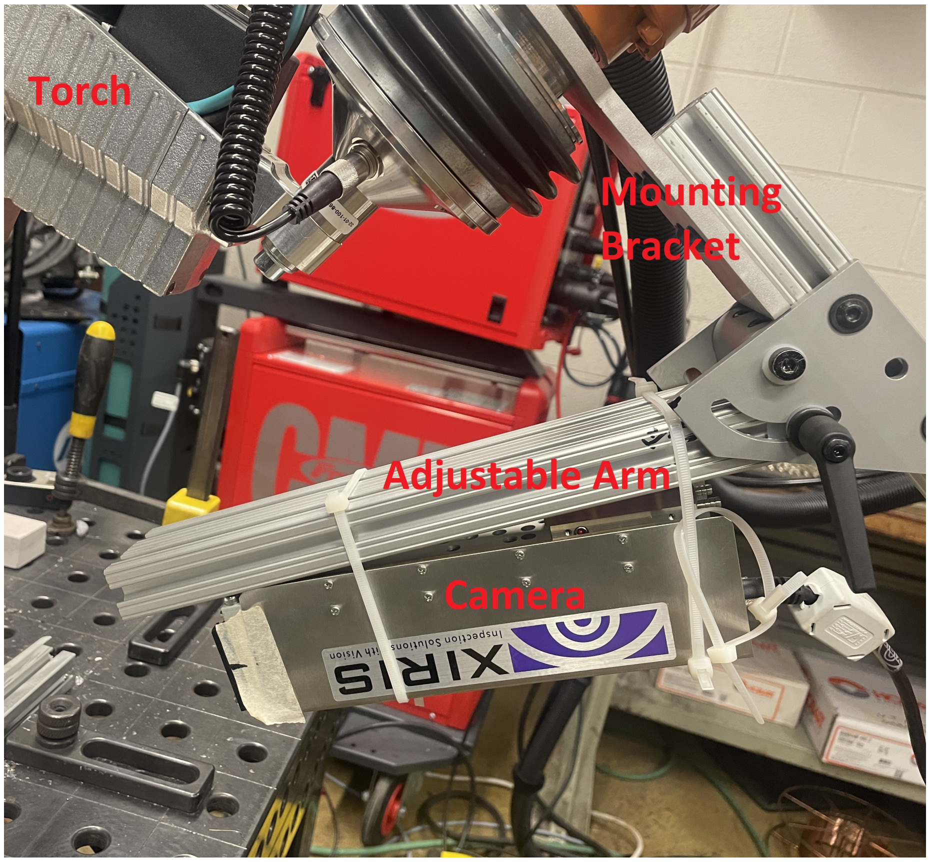
profile of the melt pool can be seen, which also provides excellent visibility. The side angle provides the added advantage of showing how far the melt pool drags behind the torch, which has been repeatedly shown to be predictive of shape irregularities such as humping [35]. Unfortunately, a singular viewing angle could not be guaranteed in more complex builds, so the camera was adjusted to look down at the melt pool from slightly above, so that even when the robot travels away from the camera, the back as well as a substantial portion of the front of the melt pool could be visible. Figure 3.7 shows the resulting viewing angle, which enables for crisp, detailed images of the melt pool region to be obtained.

3.2.4 Robot Data

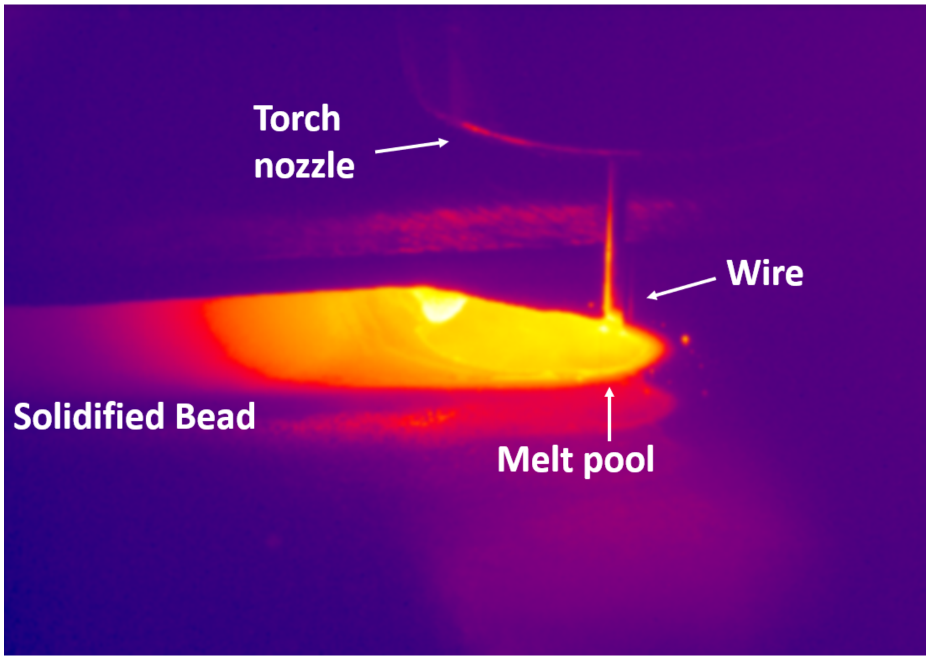
Useful process information could also be obtained from the robot. The x, y and z components of the end effector position in the base frame were each sampled at 60 Hz. The primary use for this data was to compute the robot travel speed by computing the derivatives of each position component to obtain the travel velocity vector, then finding its magnitude. The position data later proved useful during the thick wall study, which is comprised by layers with an in-fill pattern, and therefore a range of different robot travel directions. The wire feed speed is monitored internally by the Fronius controller, but gets routed through the robot and packaged into the same message as the position data. The robot can also send boolean signals that indicate when the torch is live. This process active signal was employed to synchronize the acquisition of the disparate data streams, which is discussed in the following section.

3.3 Data Acquisition

The multitude of sensors employed in this work required an accompanying multitude of data acquisition tools. Most of the sensors employed included accompanying proprietary data acquisition software, or at least software recommended by the



(a) Mounting Configuration for XIR-1800



(b) XIR-1800 viewing angle as it tracks the deposition path

Figure 3.6: XIRIS Mounting Configuration and Resulting Viewing Angle

manufacturer. Managing this multiplicity of different software tools was deemed to be unwieldy. Rather than contending with half a dozen different computer programs during operation, a streamlined data acquisition framework was developed by placing as much data acquisition under a a unified software architecture and user interface. This entailed developing a platform that could communicate with a multitude of sensors, read the data generated by said sensors, display it in a visual interface, and store it. LabVIEW, the system design and visual programming platform developed by National Instruments was chosen for this task.

3.3.1 LabVIEW Interface

Since its release in 1987, LabVIEW has been widely employed by engineers for data acquisition, instrument control and automation. One of the primary advantages conferred by LabVIEW is its ability to communicate with a wide array of devices from a diverse range of manufacturers. National Instruments data acquisition devices can communicate with almost any type of sensor. However, those devices are often very expensive and have long lead times. Luckily, most manufacturers provide LabVIEW development kits for their devices free of charge. This enables us to import modules that communicate and read from our different sensors into the same workspace. Programming logic can then be built around those modules to control the timing and manner in which data is obtained. This capability was employed to unify and streamline most of the various sensors into a single interface, which is shown in the Appendix.

Acoustic data from the microphone, arc data from the LEM Box, and welding power supply data were all integrated into the data acquisition interface. Sensor settings such as sample rates, sensor sensitivity, bits per sample, etc can all be adjusted by the user before or during operation. Since all of the interesting information occurs while the arc is active and depositing material, an automated recording system that initiates data acquisition simultaneously with the ignition of the welding arc was desirable. This is true whether the sample in question is a single pass weld or a multi-layer build. This was accomplished by basing the timing of recording on a boolean variable corresponding to the welding torch being inactive or active. The moment the welding torch becomes active, recording begins and continues until the moment that the torch shuts off, at which point recording stops, the current data files are closed, and new ones are opened for the subsequent layer. In this way, a set of data files is generated for every continuous deposition path made by the robot. Furthermore, all sensors integrated into the LabVIEW interface record data synchronously, which significantly simplifies data analysis since one can be certain that the different time series cover an identical time span.

Data storage was also subjected to the same streamlining directive. Generating a multitude of different files for each time series was deemed undesirable from a data management perspective. The LabVIEW platform provides a ready made solution to this problem in the form of the Technical Data Management Streaming (TDMS) format. The TDMS file format allows for a variety of different data streams, as well as metadata to be stored in the same file. The different data streams can be organized by user defined groups. This functionality was employed to organize the data by sensor. Arc data and welding power supply were organized into their own groups and stored as TDMS files. Acoustic data was not included in the TDMS files because subsequent analysis required that it be formatted as a wave file.

3.3.2 Robot Communication

Communication with the KR-6 robot plays a critical role in the multi-sensor framework because it not only provides some of the data employed in the study, it also provides the trigger that controls the flow of logic for the entire data acquisition interface. The link between the robot and the computer was established by utilizing KUKA's Robot Sensor Interface (RSI) technology package. Data is transmitted via the Ethernet UDP/IP protocol, which is a network protocol based on the exchange of data packets. No connection needs to be established prior to sending or receiving data, only a destination needs to be defined. The robot controller initiates the exchange by transmitting a data packet, and subsequently continues to send data packets at the sensor cycle rate, which is 12 ms. The data packets are received by LabVIEW, which has a built in UDP communication module. The data is received by the module in the form of a formatted string, which is scanned to extract the relevant data.

3.3.3 Infrared Data

The infrared cameras were more difficult to integrate into the LabVIEW interface. Lack of standardization among IR camera manufacturers complicates communication with the devices, which lack the availability of ready made drivers for LabVIEW. Due to this limitation, the cameras were not integrated into the multi-sensor LabVIEW interface. Instead, the proprietary software developed by FLIR and XIRIS was used to read and store data from the IR Cameras: FLIR Research Studio for the FLIR A-50 and WeldStudio Pro for the XIRIS XR-1800.

FLIR Research Studio and WeldStudio Pro both provide a range of analysis tools for making sense of the data. Both allow for Regions of Interest (ROI) to be highlighted which can take the shape of lines, rectangles, and circles. The temperature distributions inside these ROIs can be analyzed by computing statistics such as the minimum, maximum, mean, and standard deviation, and plots showing the temporal evolution of these metrics can be generated. Spatial plots of the distribution in a single frame can also be generated. WeldstudioPro further provides additional features such as blob detection, which allows for regions which temperature readings in between an upper and lower threshold to be outlined, as shown in Figure 3.9. This feature was employed to monitor the melt pool region.

There are significant drawbacks to using the manufacturer's proprietary software. Chief among them is the limited ability to interact with the raw data, which is especially true of FLIR Research Studio. While the software does provide analysis tools, their capabilities are limited. For example users can highlight a ROI and generate plots showing the distribution of temperature within that region. However, this can only be done for a single frame, which precludes the possibility of easily generating data that shows the temporal evolution of the temperature. Given that most of this work is predicated on time series analysis, this posed a significant hurdle. Reading the data from the saved files with external software or programming languages such as python is not trivial either. The software suites only store data in proprietary file types that are not easily accessed by third party software. FLIR Research Studio stores data as sequence flies (.seq) while XIRIS stores data in the .xir format. These file types proved difficult to work with, which forced reliance on the analysis tools provided by the manufacturer software suites. While there are Software Development Kits (SDK) available for the infrared cameras that would enable one to communicate the devices and access the raw data using custom programming solutions, pursuing that route was not feasible in the time allotted for this thesis.

3.4 Sample Characterization

The primary objective of this work is to develop models that enable real-time prediction of bead shape and porosity. This requires that reliable metrics for those characteristics be established and measured in every sample. These measurements must be easily related back to the sensing data that was obtained in the same location in space as the measurement. The approach taken in this project involves splitting up both sensing and characterization data into overlapping windows and correlating the windowed sensing data with corresponding windows in the characterization data.

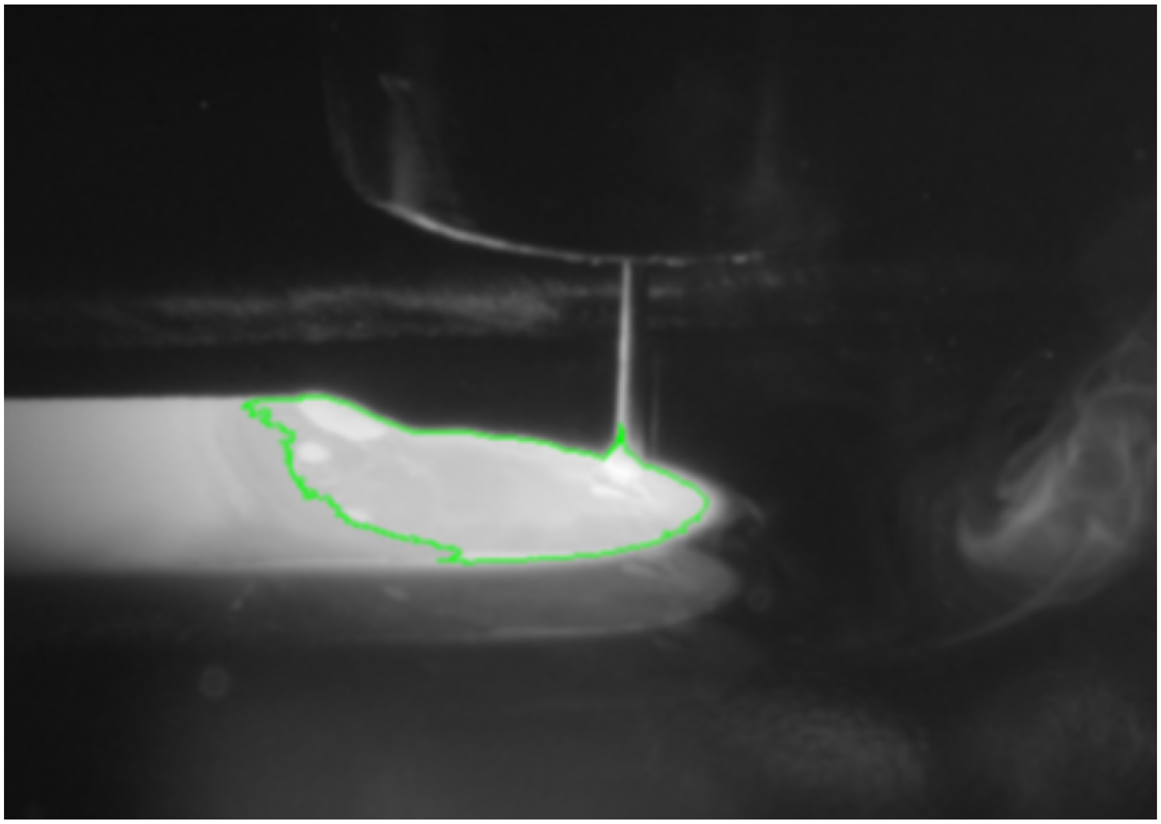


Figure 3.7: Blob detection feature in XIRIS software

3.4.1 Keyence

Measurements of the geometry and surface quality of the single pass beads were performed by a Keyence VR-5000 Optical Profiler, shown in Figure 3.9. The VR-5000 is a wide area 3D optical profiling measurement system. It works by emitting structured light through multiple double telecentric lenses. When the light bands bounce off the sample they are distorted, which allows the system to infer height via triangulation. This generates a 3D point cloud of height values, which can resolved back into a 3D model of the sample, or analyzed using the tools provided by Keyence's analyzer software.

The average height and average width of the beads are obtained by extracting multiple profiles along the length of the bead, as shown in Figure 3.10, and taking their average. A cross-section diagram of a single bead demonstrating how the beads are measured is shown in Figure 3.11. There were few available tools for characterizing the amount of waviness and humping present in the beads, so a complete height profile at the peak along the length of the bead was also obtained. The height profile and the cross section it was taken from is shown in Figure 3.12. No analogous profile could be generated for the width, so alternative methods for generating one had to be found. The solution was to pass the height data provided by the VR-5000 through an edge detection algorithm to generate a digitized copy of the bead footprint. This method is discussed at length in Chapter 4.

3.5 Experimental Design

This work is divided into two primary phases. The first involves predictive models trained on data acquired from single pass beads. A model developed from data acquired from a 30 layer in-filled thick wall comprises the second stage. The methods described in this thesis were first developed and refined on the single pass bead data because it greatly simplifies analysis and data processing.

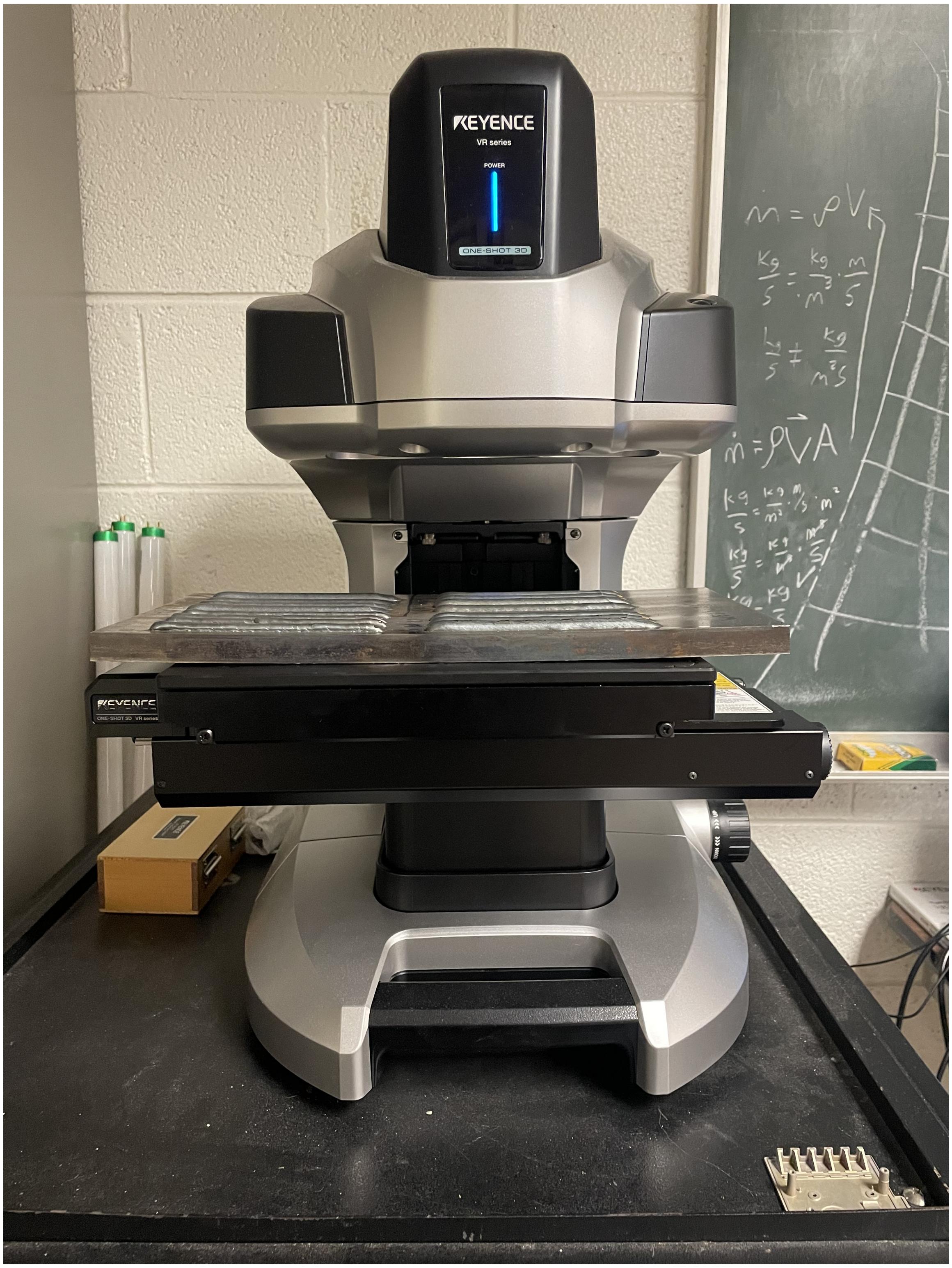


Figure 3.8: Keyence VR-5000

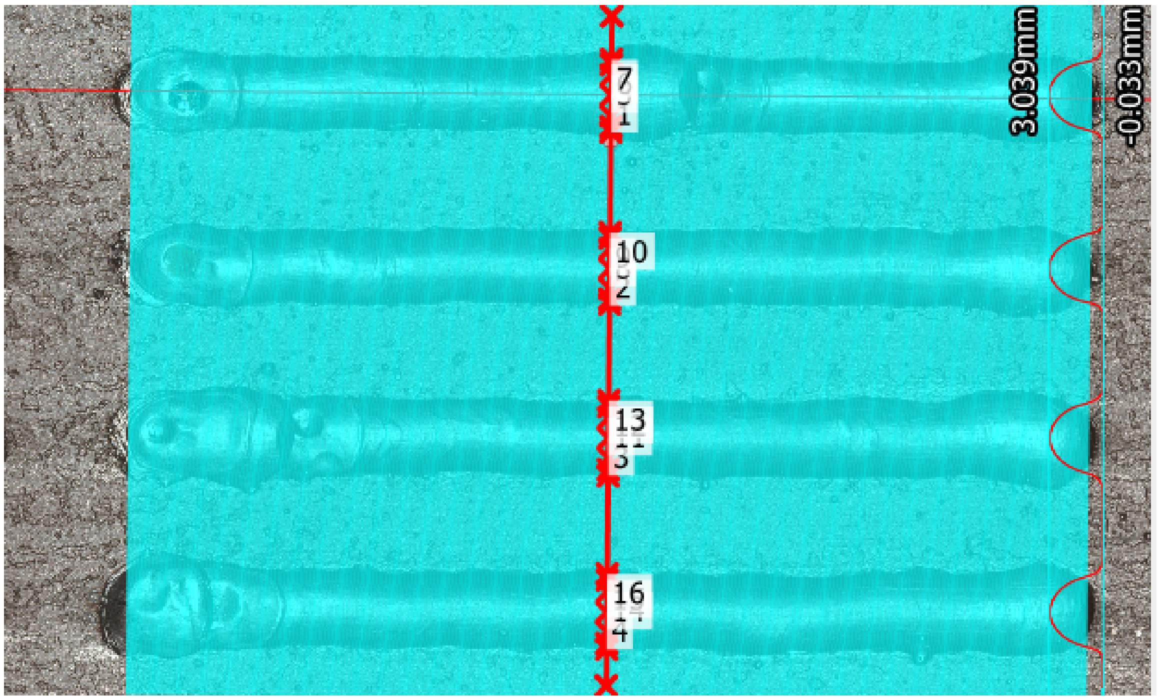


Figure 3.9: Image of bead cross sections used to compute average width and height

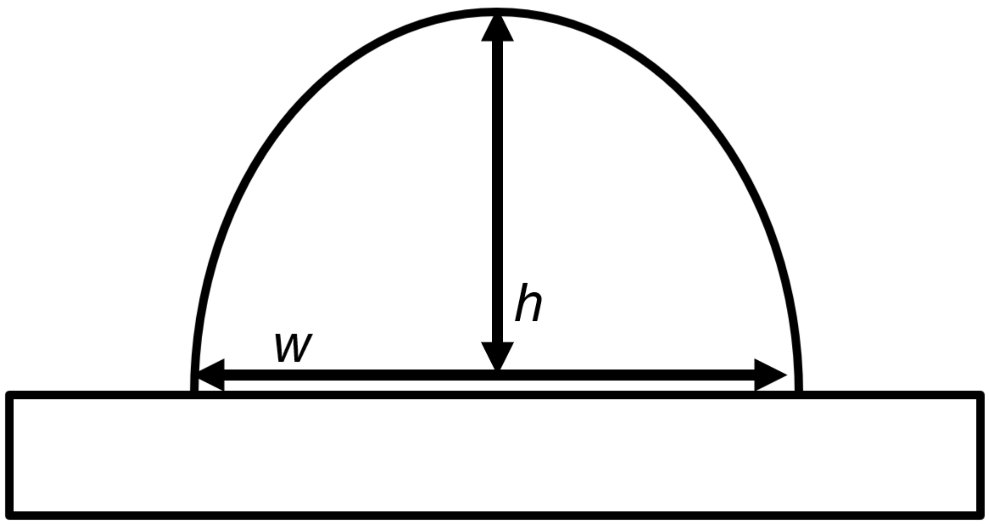


Figure 3.10: Diagram of bead cross section showing height and width measurements

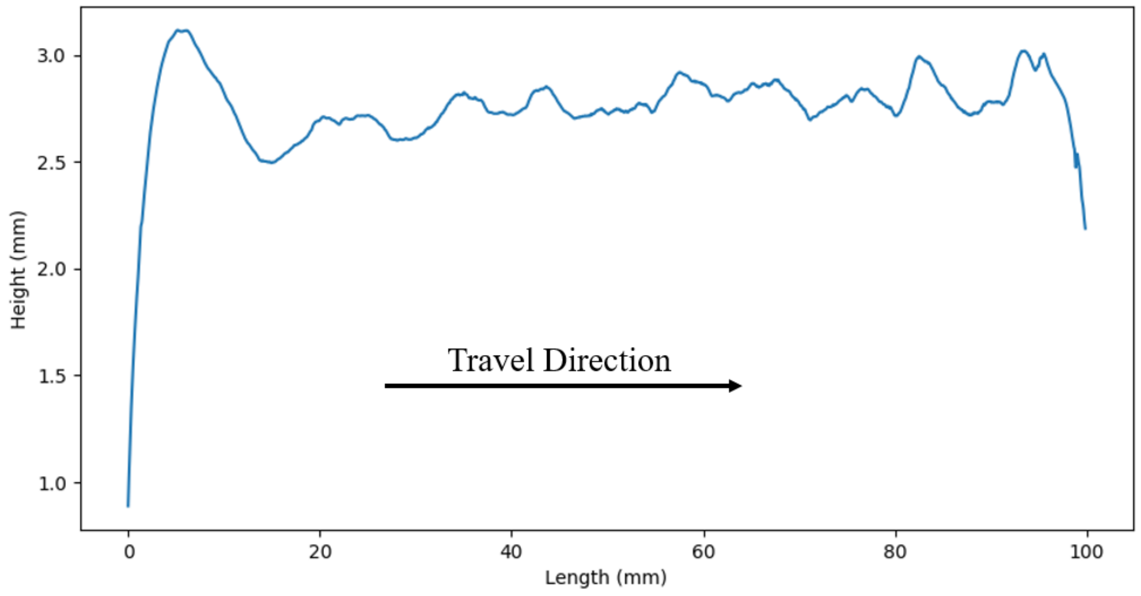


Figure 3.11: Sample bead profile

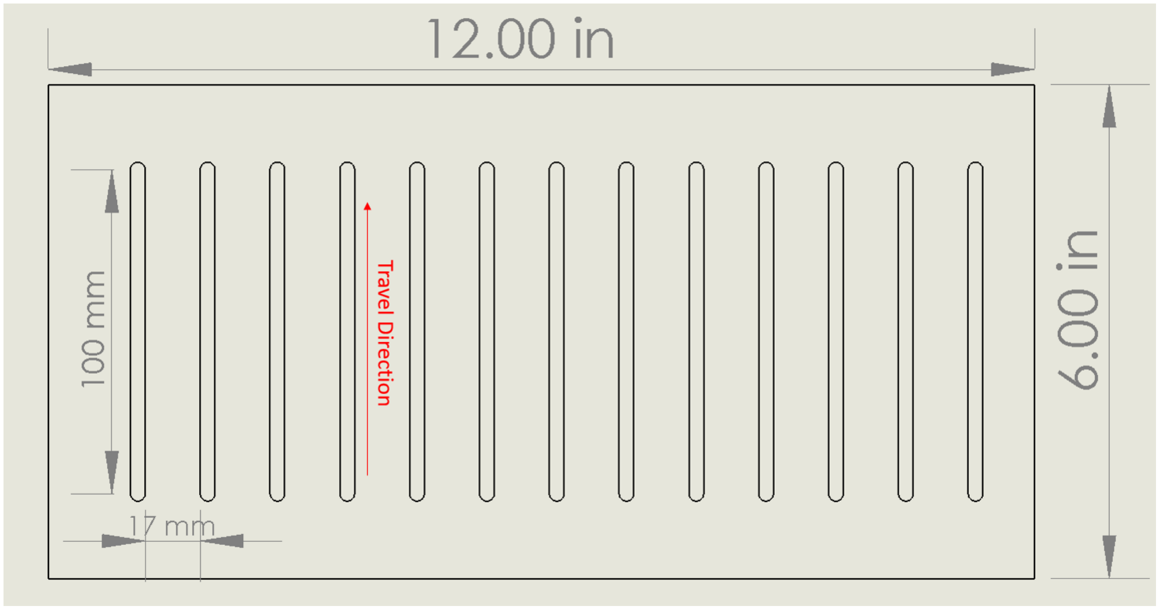
A single pass is simply a straight line, which therefore means all of the input data and output metrics can be readily mapped to each other as long as start and stop points are synchronized. The experience gained, and the methodology devised for the single pass bead studies were then applied to an infilled wall, which considerably increases complexity since not only does it contain multiple layers, but each layer is itself comprised of multiple paths, which are not continuous with each other.

3.5.1 Substrate preparation

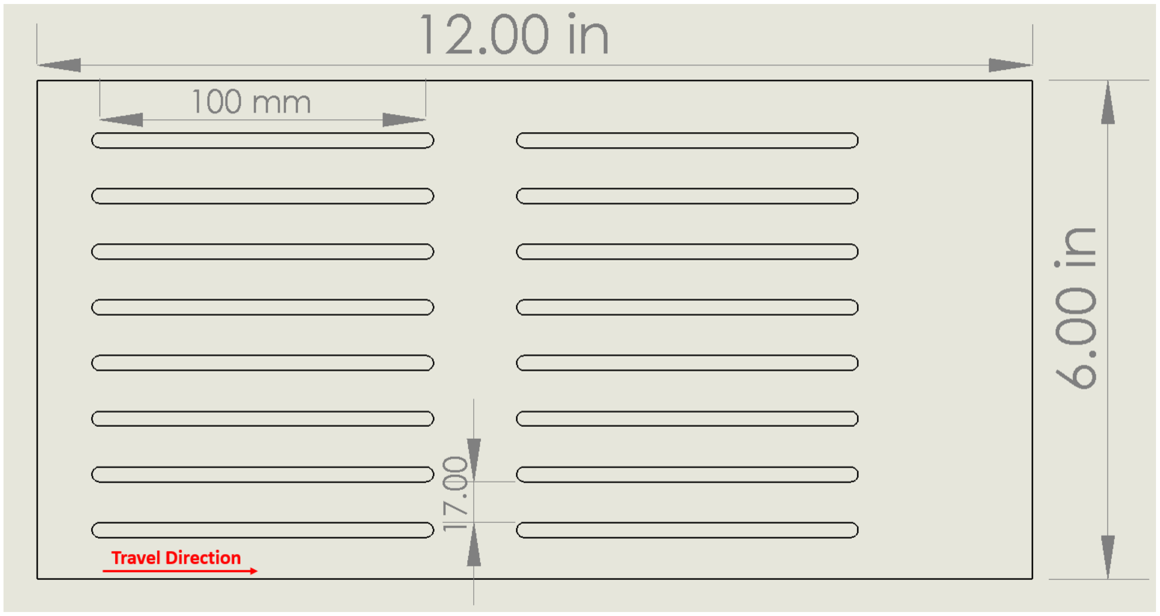
The steel plates arrive from the manufacturer covered in impurities such as grease, so they must be cleaned prior to use. First the surface of each build plate must be ground down until the shiny metal surface is revealed. Once the layer of impurities has been removed, a shop towel soaked in acetone is used to clean the surface. This step ensures the removal of grease, oil, oxidants, etc which can create instability in the arc and thus reduce the quality of the welds.

3.5.2 Single Pass Beads

The single pass beads were made by programming the robot to move in a 100 mm long straight line. During the first few runs of these experiments, the travel direction of the robot is defined to be in the x direction, or parallel with the short side of the plate, as shown in Figure 3.15. However, when these samples were scanned by the Keyence optical profiler, a substantial amount of data drop off was noticed. This problem was solved by rotating the travel direction by 90 degrees to make the robot travel in the y direction, or parallel to the long side of the plate. This eliminated the data drop off problem, giving us scans that mapped the entire geometry. The effect that sample configuration had on the completeness of the scans is outlined in Figure 3.16. The samples were spaced 17 mm apart, a number that was arrived at by trial and error in an attempt to strike a balance between isolating each sample from the effects of the arc on subsequent runs, and minimizing material utilization.

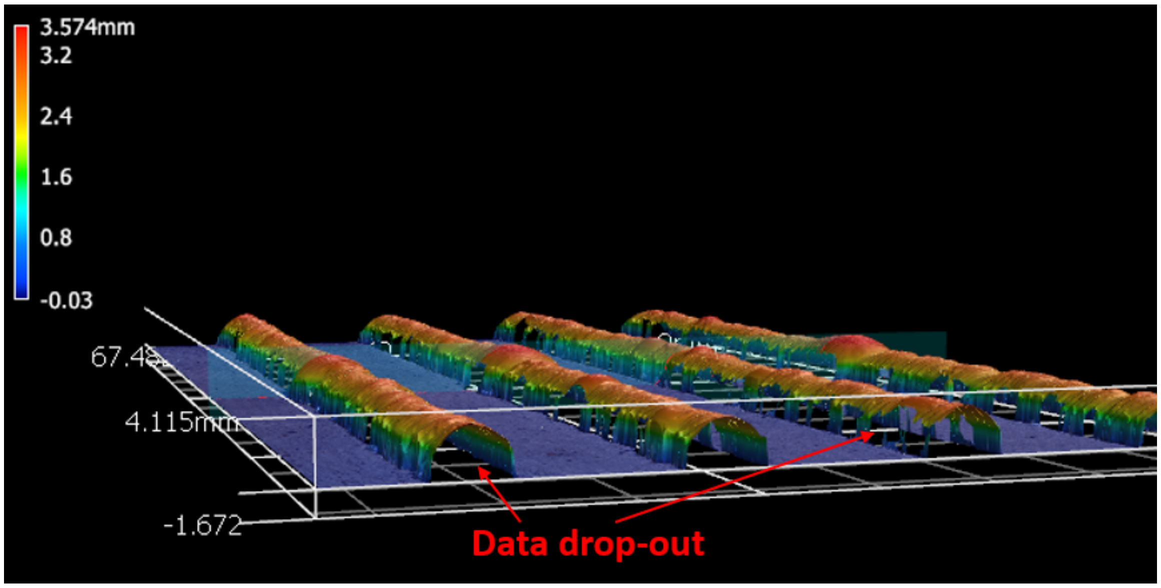


(a) Beads arranged parallel with short side of plate

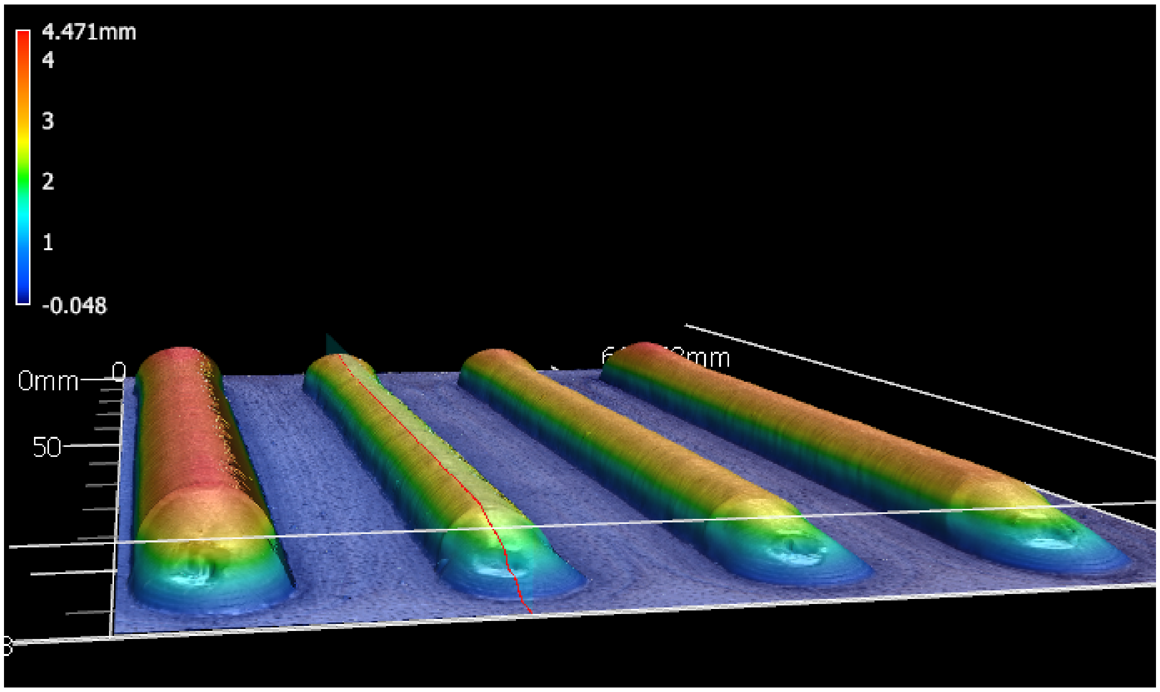


(b) Beads arranged parallel with long side of plate

Figure 3.12: Bead on plate configurations



(a) Data drop-off observed in the configuration from Fig. 3.13a



(b) Data drop-off eliminated after switching to the configuration from Fig. 3.14b

Figure 3.13: Data drop-off dependency on bead configuration

Three distinct data sets were generated. In each data set, the effects of a different set of process settings and process type are studied. Data Set 1 was created using a standard MIG process while Data Sets 2 and 3 were created by using CMT. In each data set, three main process settings are varied to create a range of outcomes for each individual sample. In Data sets 1 and 2 those process settings are wire feed speed, travel speed and arc correction. In Data set 3, arc correction is maintained constant and the contact to work piece distance (CTWD) was varied instead. Wire feed speed refers to the speed at which wire is fed to the welding torch, and travel speed refers to the velocity of the robot tool tip while it welds on the programmed path; both have units of in/min. Wire feed speed and travel speed are two of the most common settings most commonly adjusted by professional welders and are also extensively studied in the literature. Arc correction, defined on the Fronius controller as a percentage, is parameter that adjusts the arc length. Arc length, as discussed in Chapter 2 is defined as the distance between the electrode and the work piece. On Standard mode, the arc correction is highly correlated to the voltage, and can be considered approximately equivalent to varying the voltage. This is generally true of CMT as well but the relationship between the two is obfuscated in a CMT process due to the fact that the wire feed speed varies throughout the duration of the weld according to a control law.

Multiple levels for each setting were chosen, and a full factorial combination was generated. This ensures that the entirety of the process space is represented in the data set. Four levels were chosen for wire feed speed and travel speed while three were chosen for arc correction and CTWD. Wire feed speed levels range from 100-200 in/min and vary in 50 in/min increments.Travel speed levels range from 6-12 in/min and vary in 2 in/min increments. Arc correction levels range from -15%-15% and vary in 15% increments. Table 1 shows the process settings, the levels for each process settings and the process type of all 3 data sets. Three settings, with 4 levels on two settings and 3 levels on the remaining one generate 48 experiments using the full factorial method; however, some of these samples were deemed unsuitable for additive manufacturing due to excessive energy input that produces very flat beads. The nominal energy input for each of the combinations of settings was computed, and experiments where the energy input exceeded 1500 J/mm were discarded. Additionally, Data Set 3 involved increasing the CTWD, which in some instances caused arc ignition failure, which prevented certain combinations of settings from being implemented. After discarding unsuitable parameter sets, Data Set 1 consists of 46 beads, Data set 2 consists of 45 beads and Data set 3 consists of 43 beads, as shown in Table 2.

3.5.3 Infilled Wall

The initial design for the thick wall was made in Solidworks and is shown Figure 3.14. The design is a rectangular wall with dimensions 1in wide, 4.5in long and 3.75in tall. This design was sliced in the hyperMILL ADDITIVE software, which also generates a tool path and infill strategy. The infill strategy for each layer is composed of concentric rectangles and three lines placed inside the inner rectangle, as shown in Figure 3.15. The tool path data generated by hyperMILL is then imported to Octopuz, which translates the data into a path plan for the KR-6 robot. A digital model of the cell including the robot, the worktable and workpiece is created in Octopuz, which allows for the exact spatial configuration of the components to be defined. Base and tool frames that match those programmed on the robot itself are defined in the software.

The base frame for this build required closer consideration. The time series data obtained from the multi-sensor framework will only be meaningful if it can be matched with the material deposited at time t in the time series. Since the robot position as a function of time $p(t)$ is known, the x and y coordinates of the deposition path are automatically known. Matching the z coordinate requires more work, because the tool tip hovers over the material at a distance equal to the CTWD, however being able to pin point two of the three coordinates makes the robot positional data an effective tool in this task. By tracking the time steps of $p(t)$ one can determine the time t that the torch was at (x, y, z) which therefore enables one to determine the arc current, arc voltage, audio amplitude, etc corresponding to the material that was deposited at point $(x, y, z - CTWD)$. This, however, is only possible if the coordinate systems of the robot matches the coordinate system of the CT scan spatial data. Fortunately, there is some freedom to decide where the origin of the coordinate systems should be placed. The robot coordinate system can be placed arbitrarily, but the CT scan spatial data coordinate system requires a feature such as a corner or notch. The corner of the build plate might be an obvious place for the origin if not for the fact that the plate undergoes significant warping due to the heat accumulation which is most severe at the corners. This means that when the origin is defined in the CT scan software, it will be offset from the robot base frame origin. Instead, the midpoint of the long edge was chosen due to repeated observations of the plate undergoing minimal warping there.

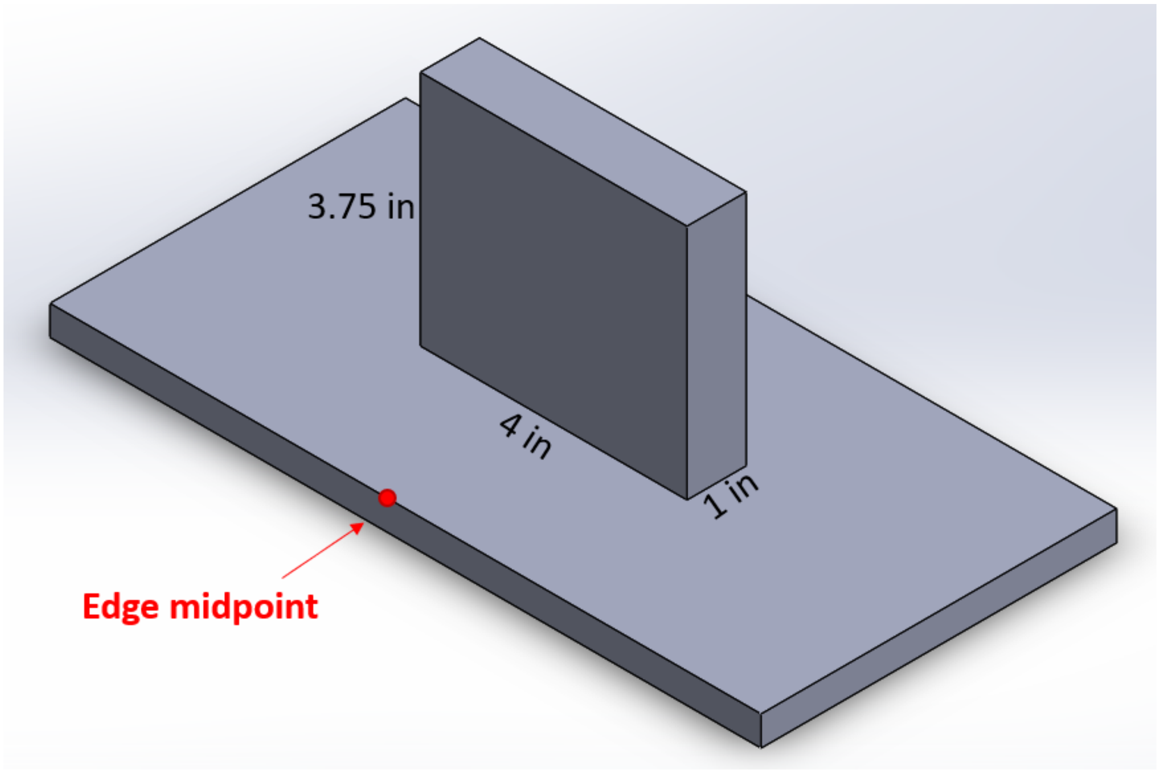


Figure 3.14: Thick Wall CAD model

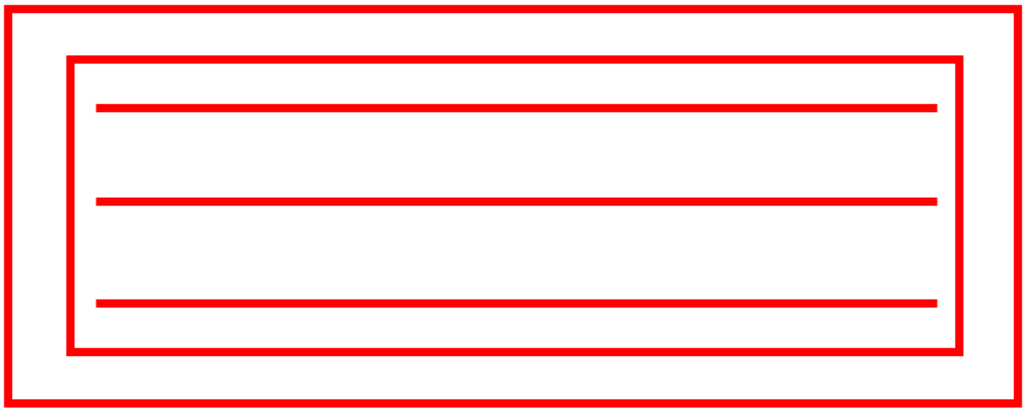


Figure 3.15: Infill strategy for each layer of the thick wall

Table 3.1: Process Conditions for Dataset 1 (Standard)

Bead	**Wire Feed Speed**	**Travel Speed**	**Arc Correction**
1	100	6	-30
2	100	6	-8
3	100	6	15
4	100	8	-30
5	100	8	-8
6	100	8	15
7	100	10	-30
8	100	10	-8
9	100	10	15
10	100	12	-30
11	100	12	-8
12	100	12	15
13	150	6	-30
14	150	6	-8
15	150	6	15
16	150	8	-30
17	150	8	-8
18	150	8	15
19	150	10	-30
20	150	10	-8
21	150	10	15
22	150	12	-30
23	150	12	-8
24	150	12	15
25	200	6	-30
26	200	6	-8
27	200	6	15
28	200	8	-30
29	200	8	-8
30	200	8	15
31	200	10	-30
32	200	10	-8
33	200	10	15
34	200	12	-30
35	200	12	-8
36	200	12	15
37	250	6	-30
38	250	8	-30
39	250	8	-8
40	250	8	15
41	250	10	-30
42	250	10	-8
43	250	10	15
44	250	12	-30
45	250	12	-8
46	250	12	15

Table 3.2: Process Conditions for Dataset 2 (CMT)

Bead	Wire Feed Speed	Travel Speed	Arc Correction
1	100	6	0
2	100	8	0
3	100	10	0
4	100	12	0
5	150	6	0
6	150	8	0
7	150	10	0
8	150	12	0
9	200	6	0
10	200	8	0
11	200	10	0
12	200	12	0
13	250	6	0
14	250	8	0
15	250	10	0
16	250	12	0
17	100	6	-15
18	100	8	-15
19	100	10	-15
20	100	12	-15
21	150	6	-15
22	150	8	-15
23	150	10	-15
24	150	12	-15
25	200	6	-15
26	200	8	-15
27	200	10	-15
28	200	12	-15
29	250	8	-15
30	250	10	-15
31	250	12	-15
32	100	6	15
33	100	8	15
34	100	10	15
35	100	12	15
36	150	6	15
37	150	8	15
38	150	10	15
39	150	12	15
40	200	8	15
41	200	10	15
42	200	12	15
43	250	8	15
44	250	10	15
45	250	12	15

Table 3.3: Process Conditions for Dataset 3 (CMT)

Bead	Wire Feed Speed	Travel Speed	CTWD
1	100	6	12
2	100	8	12
3	100	10	12
4	100	12	12
5	150	6	12
6	150	8	12
7	150	10	12
8	150	12	12
9	200	6	12
10	200	8	12
11	200	10	12
12	200	12	12
13	250	6	12
14	250	8	12
15	250	10	12
16	250	12	12
17	100	6	24
18	100	8	24
19	100	10	24
20	100	12	24
21	150	6	-15
22	150	8	-15
23	150	10	-15
24	150	12	-15
25	200	6	-15
26	250	8	-15
27	250	10	-15
28	250	12	-15
29	100	6	15
30	100	8	15
31	100	10	15
32	100	12	15
33	150	6	15
34	150	8	15
35	150	10	15
36	150	12	15
37	200	6	15
38	200	8	15
39	200	10	15
40	200	12	15
41	250	8	15
42	250	10	15
43	250	12	15

Chapter 4

Model Development

4.1 Machine Learning

Physics-based modeling involves making priori assumptions about the underlying mechanisms of a studied phenomenon, or about the nature of the relationship between the variables that influence it. Machine learning differs by assuming nothing about the nature of the process, instead employing algorithms to uncover patterns in the data, which has been compared to the way that humans learn [35]. A robust physics based model is preferable when the physics equations accurately describe the behavior of a system, and when the computational time required for the simulation is acceptable for the application. WAAM fails to meet both criteria. The underlying physics are highly complex and multifaceted. The existing models, discussed in Chapter 2, are far from complete and often complex. Increasing the complexity of computational models increases the computational resources needed to solve the physics equations, and therefore also the computational time. This is not suitable for real-time defect prediction, which requires the output to be generated on a time scale equal to or faster than the underlying physics. Machine learning models are an attractive alternative because they do not require one to have complete knowledge of the physical laws governing the process. While the computational demands for training the model can be on par with physics based models, once trained, a machine learning model is very computationally cheap to implement.

Machine learning problems can be classified into two broad types: supervised and unsupervised. Supervised learning problems are those where outcomes are assigned a known label, and every labeled outcome has a corresponding quantitative observation, x_i. The entire set of observations is the input vector x

$$x_i = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \tag{4.1}$$

If there are multiple different types of observations, which are called features, the input is a matrix X whose columns correspond to features e.g. average voltage, with its rows being individual observations for each feature. Output labels can be categorical as in classification problems, or they can be continuous as in regression problems. The other type is unsupervised learning, where the outcomes have no labels. This type is optimal for discovering hidden patterns. For WAAM defect prediction, we are interested in predicting labeled outcomes such as surface waviness, porosity, bead straightness, etc, which makes it a supervised learning problem. These outcomes are all labeled by metrics devised to quantitatively describe them e.g. porosity density for voids or peak to valley ratio for surface waviness. The objective then is to find a model $f(X) = Y$ that maps the feature vectors X to the output metric vector Y.

However, it is also possible to re-frame this regression problem as a binary classification problem by defining a threshold on the output metric. Instead of directly predicting the peak to valley metric for surface waviness, one could define a threshold based on the distribution of data below which we consider surface waviness to be negligible, and above which we consider it to be significant.

4.1.1 Sequences

A common approach for developing machine learning based predictive models in additive manufacturing is to generate numerous small parts and measure a characterization metric such as porosity averaged across the entire part. Sensing data, then, is too averaged for the entire part, This approach can be effective at predicting overall characteristics of a part, however additive manufacturing can exhibit stochastic behavior, and as such fluctuations in the process cause defects and irregularities to arise locally. In other words, there is no guarantee that a defect will always arise in the same place. A local defect such as a tendril like lack of fusion void can still become the nucleus for a larger crack, and therefore is of great interest. Furthermore, when sensing data is averaged over large time scales, a wealth of information is lost. The welding current waveform, for example, exhibits cyclical fluctuations that occur on a microsecond timescale. Changes in the pattern of the arc cycles have been demonstrated to correlate to defects in the weld [19]. If the objective is real-time prediction, then a model should have the ability to predict defects locally.

The classical machine learning problem is one where observations are sampled independently from the same underlying probability distribution $P(X)$ where X is a vector of observations. However there are many problems that do not adhere to this assumption because each observation depends on the others. Such data is referred to as sequential. Time series are a type of sequence since observations at one time step are not independent from those at previous time steps. Additive manufacturing can be thought of as a sequential problem. Sensing data is often obtained as a time series, which is certainly sequential. Material deposition itself along with its shape and mechanical characteristics can also be thought of as sequential. Additive deposition after all is a function of time, and furthermore highly dependent on the properties of deposition that came before.

A common approach to solving sequential models is the sliding window method. The sliding window method involves re-framing the sequential model as a traditional supervised learning problem by splitting the input sequence(s) up into N overlapping windows, x_i, each of which map to an output value y_i. This creates a supervised learning problem with N examples, (X_i, y_i). If the output y is itself a sequence or collection of sequences then the predictions y_i can be concatenated together to reform the sequence. Here, the output y refers to chosen metrics that describe bead shape or defects. The advantage of this method is that it enables us to use any traditional machine learning method that works on supervised learning problems [36] Three widely used methods will be explored for WAAM defect prediction.

4.1.2 Neural Networks

One of the most powerful and widely used methods is neural networks. Neural networks are networks of connected nodes, called neurons. Neurons are aggregated into layers. The simplest formulation consists of an input layer, one or more hidden layers and an output layer, shown in Figure 4.1. Each neuron stores a data value x_i, and outputs a signal to connected neurons in subsequent layers. The output of a neuron is the data scaled by a weight w_i and offset by a bias b_i such that its output is given by

$$y_n, i = w_i x_n, i + b_i \tag{4.2}$$

The neuron output is then passed through an activation function which is often non-linear to enable modeling of complex problems. Common activation functions include sigmoid, tanh, and linear. The weights and biases contain the information ”learned” by the network during training. A forward pass through the network computes an output which is compared to a loss function. The gradients of the weights and biases with respect to the loss are then computed which provides the direction in which they should be adjusted for the next iteration. The magnitude that they should be adjusted by is a tunable parameter called the learning rate.

4.1.3 Random Forests

A random forest is an ensemble of decision trees. Decision trees are a machine learning method where the objective is to predict target data by splitting the input data through a series of questions. The simplest formulation of this process is to ask yes or no questions, forming a binary split e.g. ”is the voltage greater than 10V?” Each of those questions is contained in a node, while the answers point to child nodes, in this way, forming a hierarchical branching pattern. A prediction is arrived at by answering the questions at each node starting from the root node, all the way through the child nodes until a node without children is reached [37]. The stop criteria is based on node impurity, or the homogeneity of the labels at the node. An example decision tree is shown in Figure 4.2. Splits are determined by examining the relationship between each feature and the output to identify which binary split would result in a prediction that reduces the loss function the most. This process continues until all nodes encompass a satisfactory number of output observations. A random forest combines the predictions from a collection of randomized trees by averaging the predictions for regression, or majority vote for classification. This method was been proven to be robust and overcomes the tendency of lone decision trees to overfit.

4.1.4 Long Short Term Memory

When dealing with sequential data, one drawback of simple neural networks is that the influence that outputs have on subsequent outputs in the series is ignored. This weakness is addressed by Recurrent Neural Networks, which are trained in a sequential manner, i.e. one time step after another, and use the output from the previous steps as inputs for subsequent steps. The influence that events in the past has on the present is therefore preserved. The most robust version of a Recurrent Neural Network is Long Short Term Memory (LSTM) which in addition to incorporating previous outputs as inputs, can store select information for many time steps, thus giving it a sort of long term memory that simple Recurrent Neural Networks lack. Additionally, LSTM

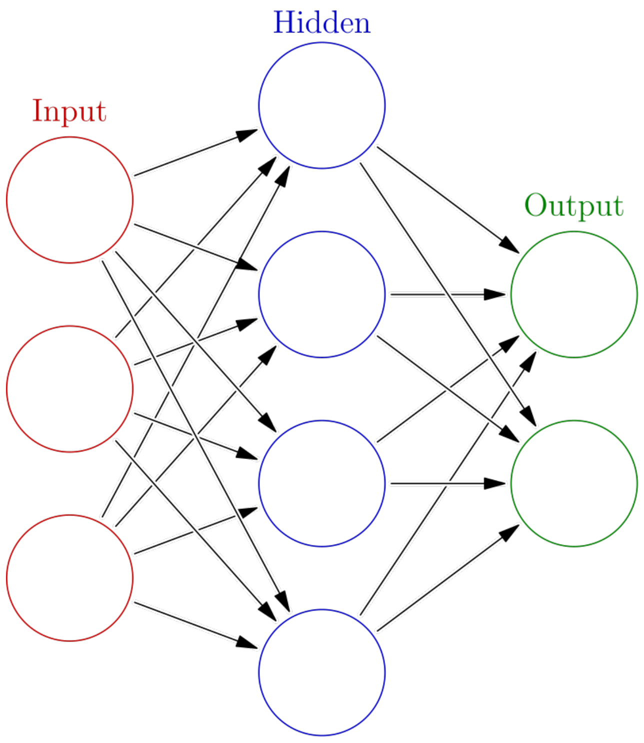


Figure 4.1: Neural Network

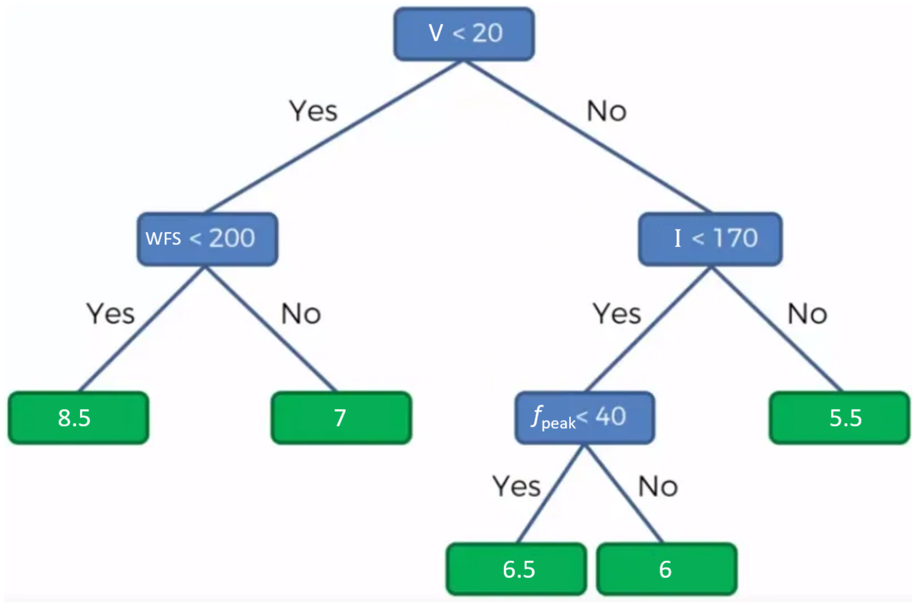


Figure 4.2: Example decision tree. Nodes are labeled with parameters studied in this work. V is voltage, I is current, WFS is wire feed speed and f_{peak} is peak frequency. This does not correspond to an actual decision tree.

includes a mechanism by which to determine which pieces of information should be remembered and for how long, allowing some information to be ”forgotten” when it proves advantageous to the model. The sort of sequential time series data obtained throughout this work is well suited to such a model architecture, and it makes physical sense to include a mechanism for remembering previous measurements of height, width or shape consistency since the bead shape in the past naturally influences the bead shape in the present and future.

4.2 Output Metrics

The objective at hand is to predict shape irregularities and porosity, which informs the choice of output metric Y. Modeling bead shape could be accomplished in one of two ways: by predicting the dimensions of the bead themselves, or by predicting a metric intended to describe the consistency of the deposition shape. Both approaches will be explored. Modeling porosity on the other hand only requires a metric that describes the amount of porosity contained in a given voxel of the part.

4.2.1 Shape

The raw characterization data for bead shape obtained from the Keyence Optical Profiler comes in two forms. One is a one dimensional height profile that corresponds to the outline of a cross section produced by a bisecting a weld bead along its center line, as shown in Figure 3.12. The other is a pictorial representation of the height variation that uses color to represent the variation in height throughout a 2D grid, where each pixel represents a step size in x and y. This raw data is utilized to characterize the shape of the beads. The height profile can be directly employed as an output metric, where the objective becomes modeling the height of the bead.

Modeling the occurrence of shape irregularities such as humping, waviness or crookedness requires a modified approach. One way to measure the amount of waviness is the peak to valley difference, which has been used to measure wall waviness and bead waviness in AM [38]. The peak to valley difference is simply the difference between a peak and a valley in a given segment, as shown in Figure 4.3.

Another way to measure irregularity is the standard deviation. A bead with a flat, consistent profile will have a low standard deviation while a bead with a lot of variation in the geometry, especially larger irregularities such as humps will have a high standard deviation.

4.2.2 Edge Detection

The heat maps obtained from the optical profiler are utilized to develop another set of shape metrics. Since the profiler software did not offer a way to measure how the width of a bead evolves along the length of the path, one had to be developed. The proposed solution follows a procedure inspired by Choudhury et. al (2021) [19]. First an outline of the bead footprint is obtained by passing an image of the top view through an edge detection algorithm. Then a center line is computed by implementing a skeletonization algorithm. The center line is utilized to compute an estimate of the bead width at every length step. Choudhury et. al also describe how the shape of the center line, namely, how much it deviates from an ideal straight line path can itself be an indication of process stability, so the deviation of the center line from an ideal straight line path was converted into an output metric for use in the model.

Edge detection provided an attractive option because imaging of the beads was readily available, and implementation of such an algorithm is relatively simple. Canny edge detection was chosen from a lineup of possible candidates for its robustness. Canny edge detection is a multi-stage edge detection algorithm first described in Canny (1986) [39].

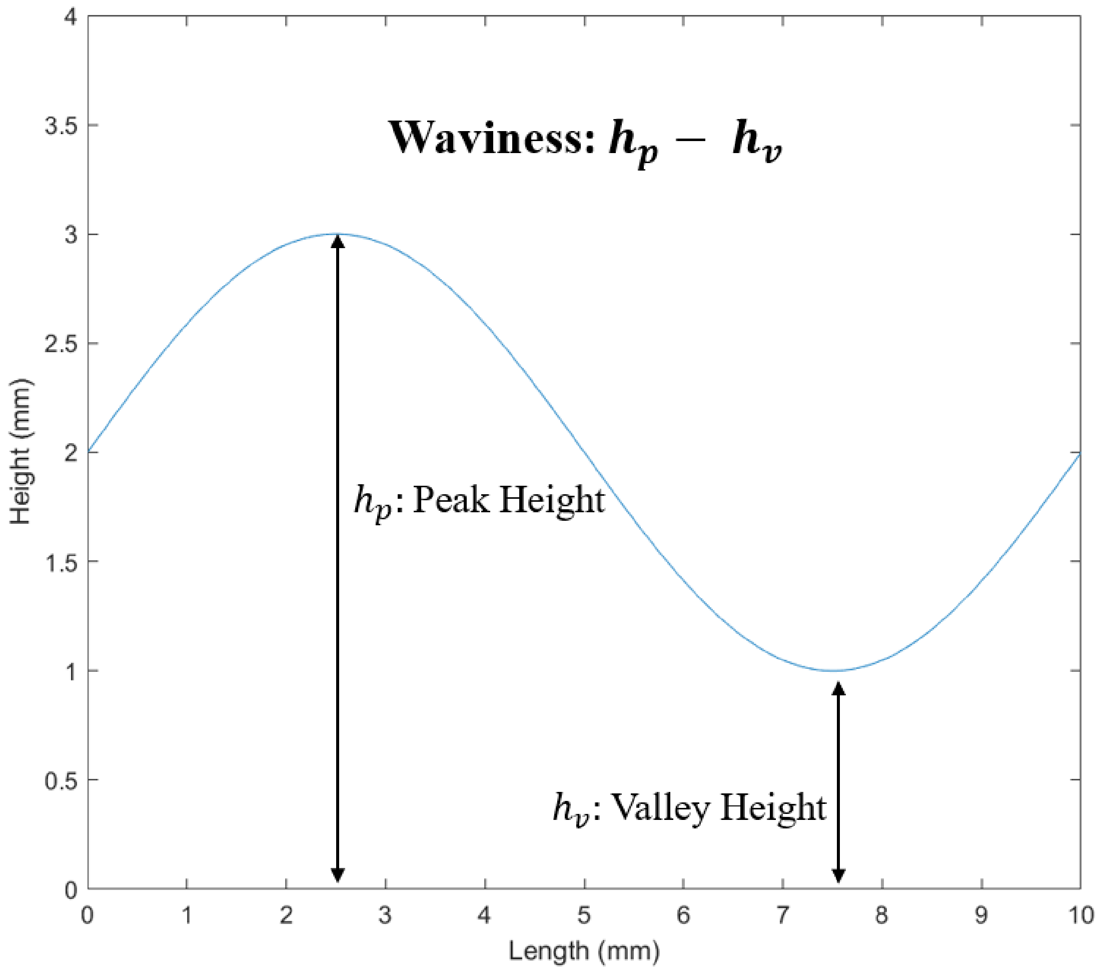


Figure 4.3: Definition of Peak-to-Valley Difference

first, the image is converted to grayscale. Then, a Gaussian blur is applied to the image in order to reduce noise. which makes it less likely that irrelevant noisy features in the image will be detected as an edge. Edges in the image correspond to changes in the intensity of the pixels. Thus the next step is to compute the intensity gradient matrix of the image. This is accomplished by convolving the image with Sobel filters for the vertical K_y and horizontal direction K_x

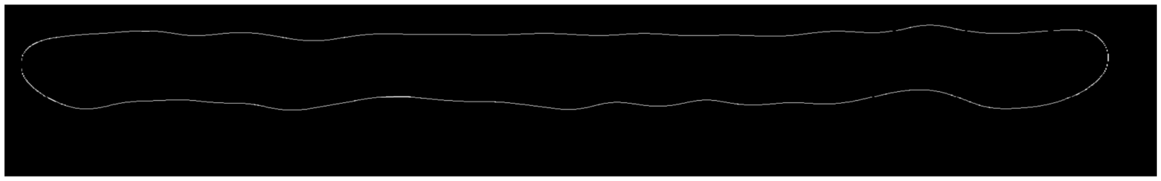
$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, K_y = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \tag{4.3}$$

This yields the intensity derivatives I_x and I_y. The magnitude G and the slope θ of the gradient are found by

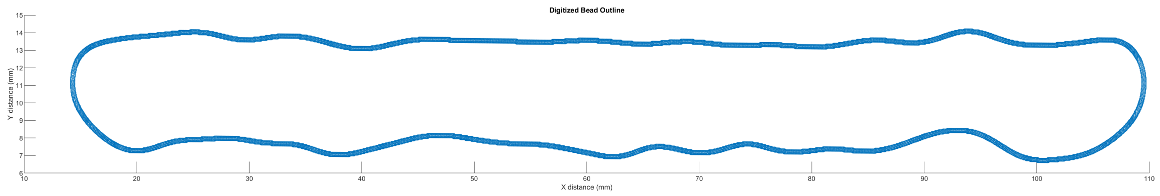
$$\|G\| = \sqrt{I_x^2 + I_y^2}\theta = \arctan \frac{I_y}{I_x} \tag{4.4}$$

To ensure that edges returned aren't fuzzy, a non-maximum suppression algorithm is implemented. The algorithm checks every point in the intensity matrix and identifies the pixel in the edge direction, obtained from the θ matrix, with the maximum intensity. Next, a double threshold is imposed to separate the pixels into strong, weak and non-relevant. The upper threshold identifies strong pixels that definitely contribute to the edge, the lower threshold identifies irrelevant pictures that do not contribute and those pixels with intensities in between the threshold are called weak pixels which may or may not contribute. These thresholds are tunable parameters. Weak pixels are converted to strong ones if they are bordered by another strong pixels – this is called hysteresis.

The Canny Edge Detection algorithm is applied to the heat maps obtained from the Keyence, which yields a bitmap of the bead outline where edge pixels are 1 and the background is 0, as shown in Figure 4.4a. Each pixel corresponds to a step change in the vertical and horizontal directions. The size of that step change is obtained from



(a) Bead outline bitmap



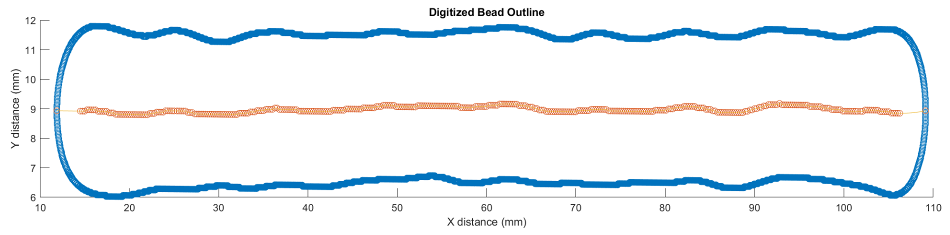
(b) Digitized bead outline

Figure 4.4: Canny edge detection algorithm results

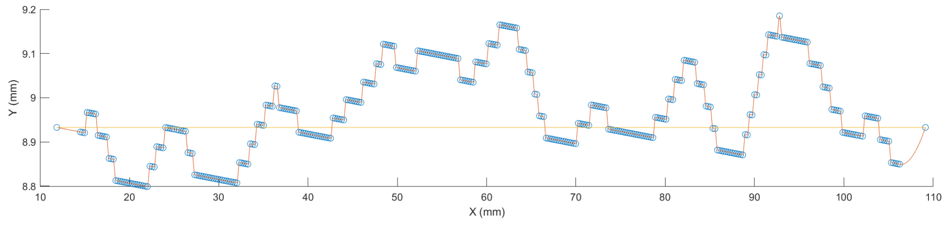
the optical profiler. The bitmap of the bead outline is converted into a Cartesian plot by finding all of the row and column indexes that correspond to a value of 1 (i.e. part of the edge) and multiplying them by the distance per pixel obtained from the Keyenece. The row indexes then become the x coordinates while the column indexes become the y coordinates. The plot of the bead outline obtained by this transformation is shown in Figure 4.4b.

The coordiantes for the center line of the bead are then obtained by utilizing the Hamilton-Jacobi skeletonization algorithm described by Siddiqi [40, 41]. The algorithm works by computing the average outward flux of the position vector field gradient $q = (x, y)$. The shape is then thinned by sequentially removing points until a threshold is reached. The algorithm results in the center line shown in Figure 4.5a. The shape of the center line corresponds with process stability; a crooked zig zagging center line suggests instability while a straight line points to a stable process. This observed relationship with process stability can be quantified by plotting an ideal straight center line on top of the actual center line, and comparing the two, as shown in Figure 4.5b. The standard deviation of the difference between the actual and ideal center line is employed as one of the output metrics for the predictive models.

The bead center line was used to obtain the bead width. For every point on the center line, the vertical distance to the top side of the outline and the vertical distance to the bottom side of the center line are computed. The width is the difference between the distance to the top and the distance to the bottom. Some of the beads are at an angle because of how they were positioned in the scanner. Before this measurement can be made, the bead must be in the horizontal position. Some of the bead images are angled because of how they were positioned in the scanner. This angle is corrected by applying a transformation matrix with an angle equal to the angle of the ideal center line. The width measurement methodology is shown in Figure 4.6.

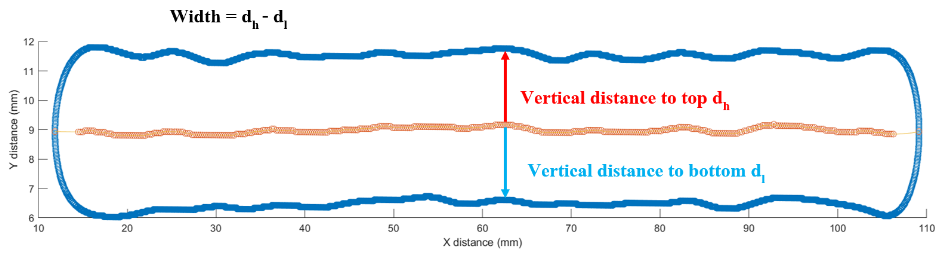


(a) Bead outline with center line

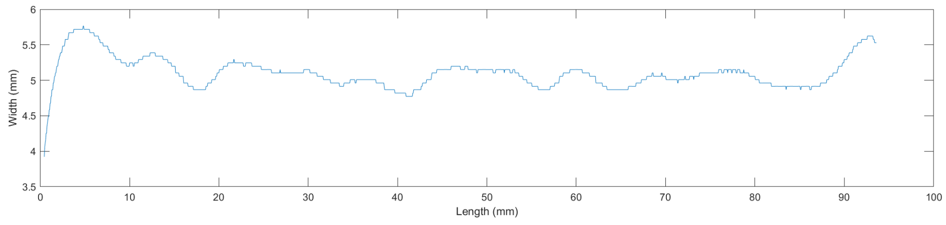


(b) Actual bead center line vs ideal center line

Figure 4.5: Acquisition of bead center line


Width = d_h - d_l
Vertical distance to top d_h
Vertical distance to bottom d_l
Y distance (mm)
X distance (mm)

(a) Diagram of vertical measurements made to obtain bead width



(b) Bead width as a function of length

Figure 4.6: Acquisition of bead center line

4.3 Pre-processing

The raw time series are not suitable for use in predictive models. Most of the disparate time series are sampled at different rates, which means the time series vectors can have vastly different sizes. Waveform data contains significant noise. Additionally, despite efforts to synchronize data acquisition with the start and end of the process, the time series still do not align with the shape profiles. Furthermore, the bead geometry data covers the entire bead, not merely the stretch corresponding to the robot travel. These challenges call for a pre-processing procedure that trims, and cleans up the data to ensure that the disparate time series data matches up as precisely as possible with each other and with the geometric data.

4.3.1 Wavelet Based Denoising

High resolution signals typically contain noise from a variety of sources. Electric signals are affected by electromagnetic interference, current drawn from other devices in the circuit, power main hum, among many others. The noise problem is particularly severe for the acoustic data because the microphone picks up environmental noise produced by equipment in the workcell. Hums and whirring of motors, pumps and worst of all the ventilation system which is quite loud. This makes the audio signal highly noisy, making it difficult to work with. The noise could be somewhat reduced by strategically placing the microphone as close as possible to the torch, but even then the noise remains. Mitigating this problem further requires a method for denoising the acquired signal.

One denoising method is wavelet based denoising. A wavelet is a small wave with its energy concentrated in time. An example wavelet is shown in Figure 4.4a. Wavelets can be used to decompose a signal into component parts, much like a Fourier transform, except the basis functions are wavelets instead of sines and cosines [42]. Much like a Fourier transform, a wavelet transform can be used to eliminate certain frequency terms by imposing thresholds on their coefficients. However, Fourier transforms are created by a one dimensional sequence of coefficients, whereas wavelets expansion map the signal to a two dimensional array of coefficients, which allows for efficient localization in both the time and frequency domains. This allows for the separation of components that overlap in both the time and frequency domains. Once a wavelet expansion is found, thresholds are then imposed on the coefficients for the wavelet functions to reduce or eliminate the magnitude of noisy terms [43]. The wavelet based denoising procedure was applied to both the audio and arc voltage and current waveforms. The effect of denoising on an audio signal is shown in Figure 4.8

4.3.2 Synchronization

A successful machine learning model requires that all observations x_i are paired with the correct outputs y_i. For the sensor time series data, this means that time series observations synchronized so that all observations correspond to the same t_i, and all time series observations $x(t_i)$ should be aligned with the shape profile section that was deposited at time t_i. As noted, this is not the case for the raw time series data due to the fact that the computer receives a process active signal before the torch actually starts moving and it continues to receive it a short time after it stops moving. This triggers data acquisition slightly early and pauses it slightly late, thus adding a small interval of time to the sensor time series corresponding to the torch being stationary i.e. while no deposition is occurring. Figure 4.9 shows this effect. The cause is that RSI sends a process active signal during the pre-and post flow time despite the fact that the arc has not yet struck. Pre-flow and post-flow time are a small interval of time programmed into the robot path, typically less than 1 second before arc ignition or after arc stop, where the torch hovers over the programmed path start and blows shielding gas over the location where the arc will strike, thus ensuring that there is enough shielding gas where the melt pool is going to land. This interval of time needs to be trimmed from the time series data.

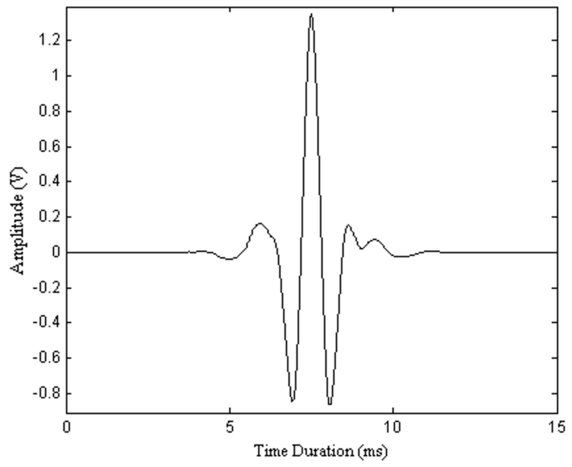


Figure 4.7: Wavelet

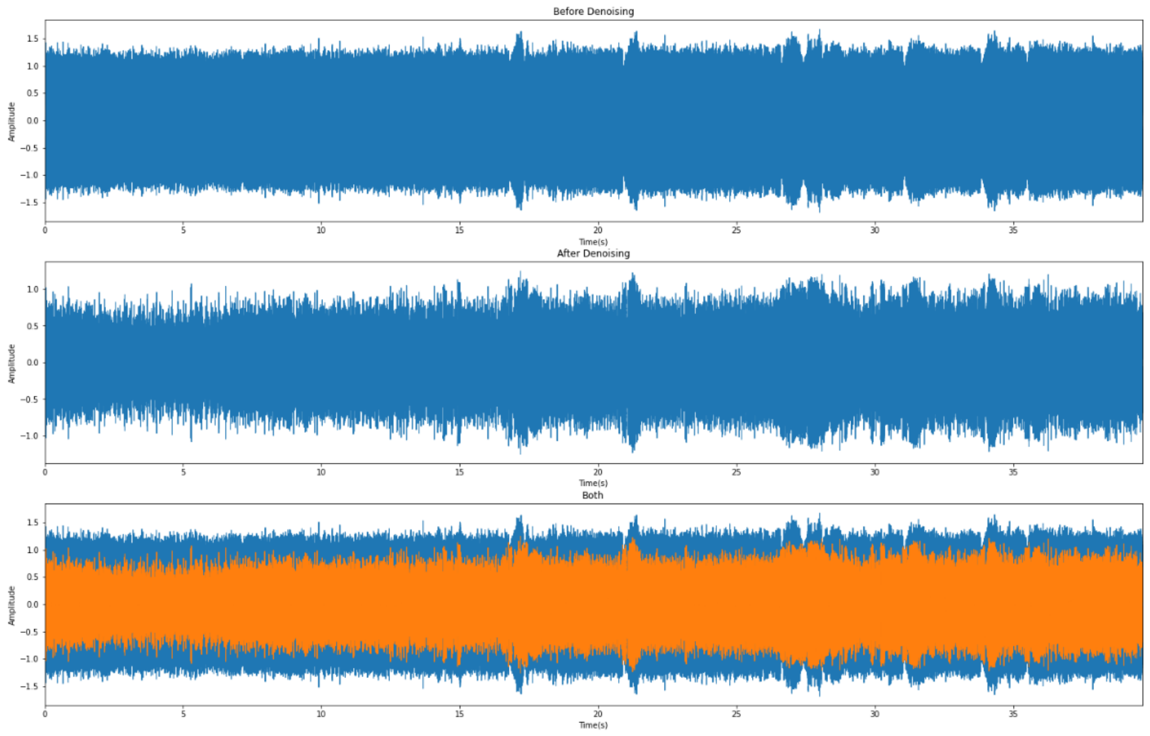


Figure 4.8: Effect of wavelet based denoising on audio signal

Since the characterization data is obtained post-process, correlating positions along the deposition profile to moments in time is difficult. There is no time series for shape that provides information about which parts of a shape profile were deposited when. However, there are some known facts about the deposition which can facilitate such an alignment. Firstly, the the position of the torch with respect to time is known. Since the beads only differ from this position by a distance equal to the CTWD, the robot position data provides a reasonable way to correlate positions along the characterization data to moments in time.

For bead studies, this process is further simplified by the fact that the travel path is one dimensional. Shape metrics are likewise obtained in the form of 1-D profiles. Deposition is assumed to only occur while the torch is moving. Then, the only remaining task is to identify the locations on the weld bead that correspond to the start and end of deposition. This can be accomplished by identifying distinguishing features on the bead. It was observed that a small crater always forms on the bead at approximately the same location where the torch stops moving. Furthermore, this feature can be readily identified on the optical profiler. Keyence's software enables us to locate the position of this feature on the height profile, as shown in Figure 4.10. Since the travel distance is programmed to be 100 mm, the start position can be found by subtracting 100 mm from the position of the end point.

The start and end points of a weld bead correspond to transient process regimes that are very different from the conditions during continuous deposition. This difference is reflected in the shape of the bead at the end caps. The end cap at the start of the arc tends to be bulgier and rounder than the rest of the bead, while the at the opposite end, where the arc shuts off, the bead exhibits a downward slope. Trimming the shape profiles to the start and end positions does not automatically resolve this issue. The start/end slopes still show up in the profile, so more trimming is required. The start of the bead is trimmed to the peak of the bulge, while the end of the bead is trimmed to the start of the downward slope, as shown in Figure 4.11a. The resulting trimmed bead profile is shown in Figure 4.11b.

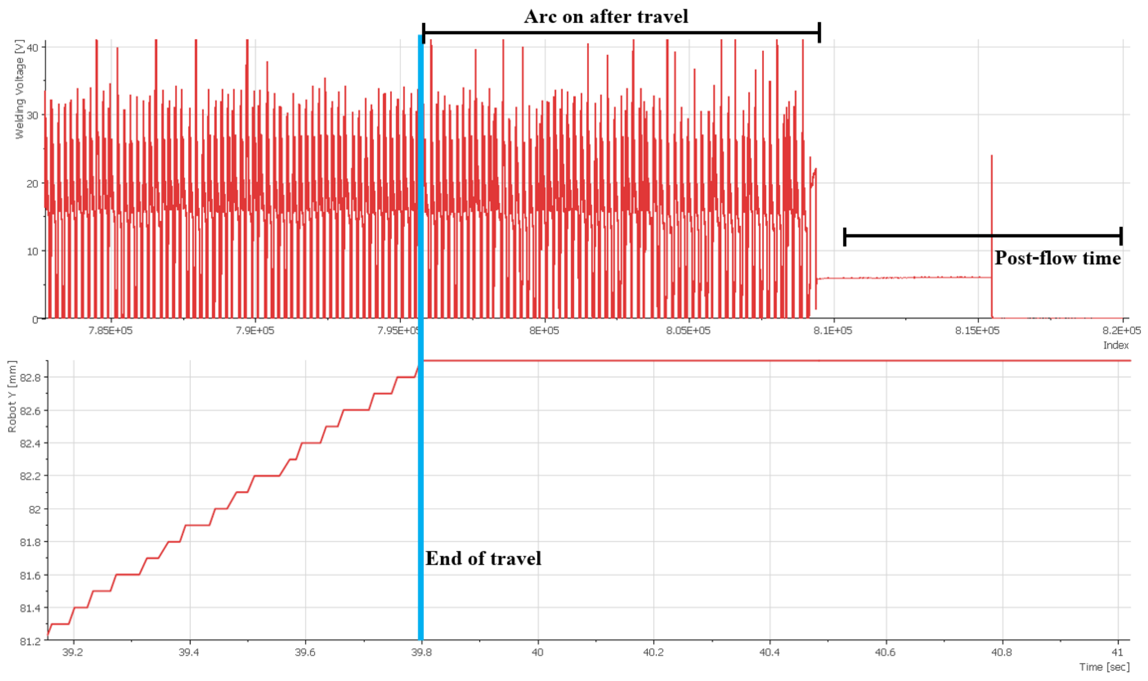
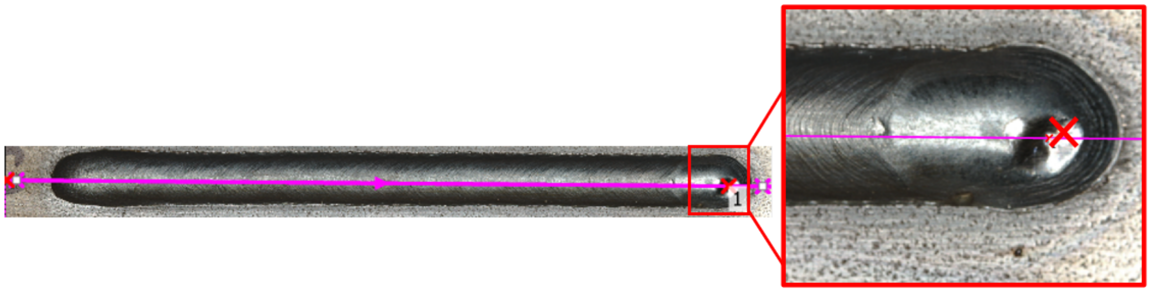
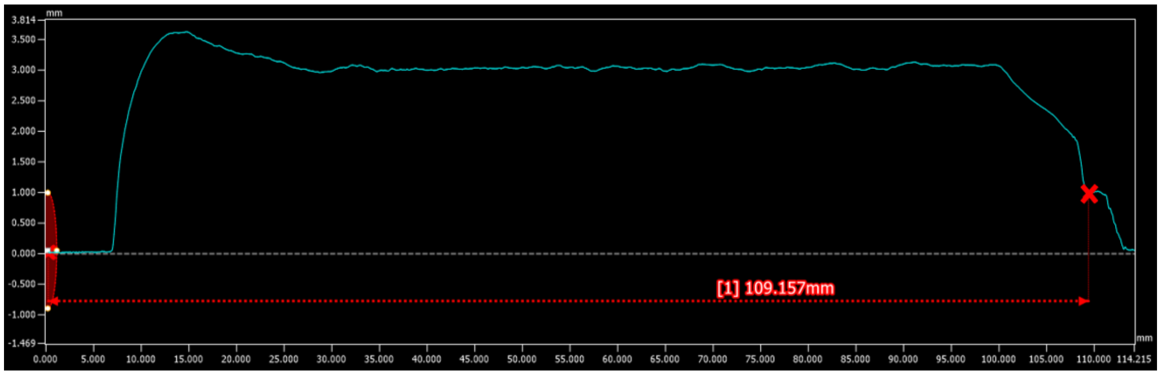


Figure 4.9: Sensor data continues after robot travel stops



(a) Position of crater on the bead



(b) Resulting trimmed profile

Figure 4.10: Horizontal measurement from zero position to crater

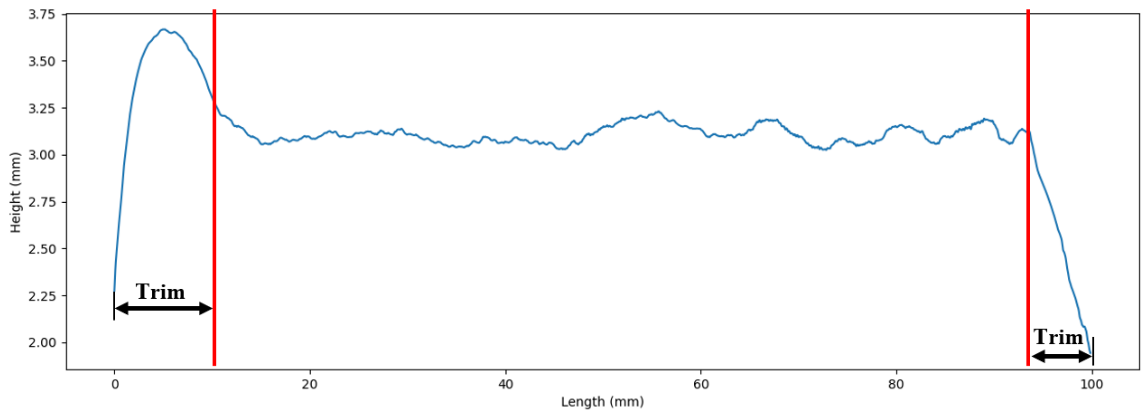
4.3.3 Windowing

In order to implement the approach proposed in Section 4.1.2, all sensor time series and output data are partitioned into overlapping windows, as shown in Figure 4.11. Each overlapping window is defined by a window size n_w in number of samples, a hop length h, which is the number of samples that each window $x_{x,w,i}$ advances by to arrive at the subsequent window $x_{w,(i+1)}$. For the windows to be overlapping $h > n_w$. The amount of overlap between windows can be characterized by a percentage p where n_o is the number of overlapping samples.

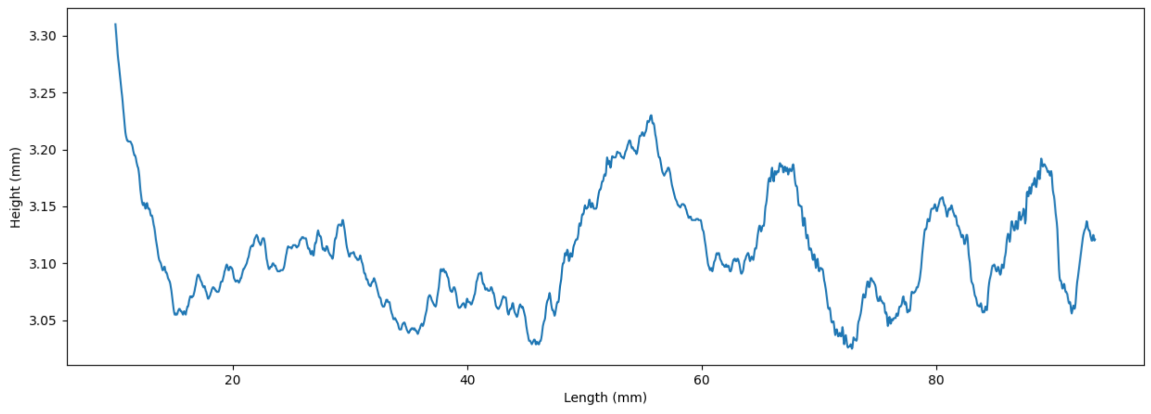
$$p = \frac{n_o}{n_w} \tag{4.5}$$

It is critical for windows from one time series to align with the windows from the other time series as well as the windows from the output data. The time series windows must all correspond to the same time range while all the time series windows must correspond to the concurrently deposited segments of material. While this could be easily accomplished if all of the data was sampled at the same rate, the multi-sensor approach taken precluded that possibility. Ideally, all sensors would produce data at an identical sample rate, which would make windowing trivial. However, acquiring sensors that could measure all of the disparate physical phenomena of interest at an identical sample rate proved to be impractical. Consequently, the windowing procedure had to account for the differing sample rates between time series.

The efforts taken to synchronize the start and stop points of the weld undertaken in the previous section go a long way towards resolving this issue. Having ensured that all the sequences start and end at the same moment in time and the same position in space, matching windows can be obtained by dividing all of the sequences into N equally sized windows. This was accomplished by establishing a relation between the number of windows, the window size, and the percent overlap. The number of windows and percent overlap are arbitrary parameters from which the window size and hop length are computed. The relationship between these parameters is given by



(a) Height profile plot showing the bulge near the start and the slope near the end that are trimmed from the profile



(b) Resulting trimmed profile

Figure 4.11: Trimming the bead height profile

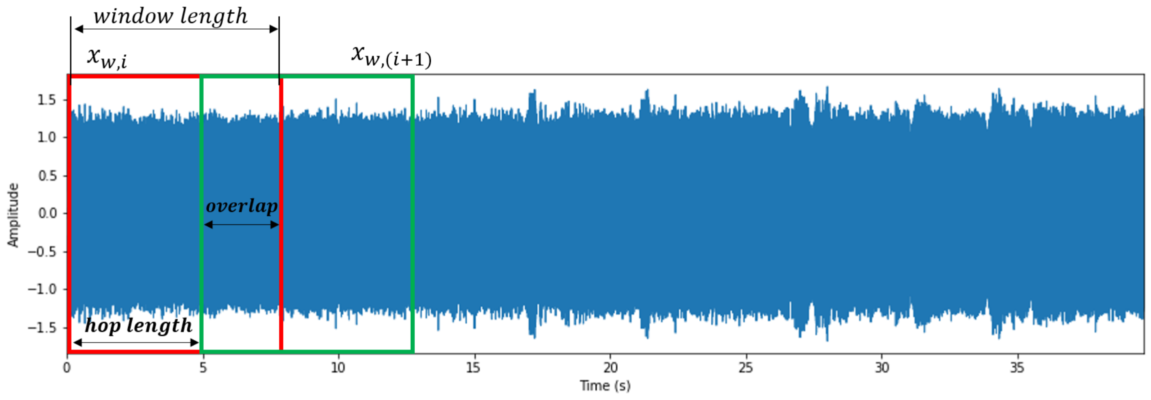


Figure 4.12: Diagram of signal showing how overlapping windows are defined. Window length is the number of samples in a window, hop length is the number of samples skipped to get to the next window and overlap is the overlap between windows

$$N = floor(\frac{n_x - n_w}{h}) + 1 \tag{4.6}$$

where n_x is the length of the entire series. This equation is solved for n_w, using the fact that $h = n_x - n_x p$ to obtain

$$n_w = \frac{n_x}{N - pn_w + p} \tag{4.7}$$

Thus, N and p are arbitrary parameters that are used to compute the window length n_w. N windows per bead means that N * number of samples will be available for the model. It should be noted that increasing the number of windows per bead will not necessarily improve the model performance. The performance may improve at first but will eventually decline because the n_w in the lowest sample rate time series becomes very small. In this study, the limiting factors were infrared data and shape characterization data, both of which were sampled at much lower rates than the audio and arc current/voltage. The the audio and current/voltage signals had several hundred thousand samples, while the melt pool temperature time series obtained from the XIRIS typically had upwards of 4000 samples. The height and width profiles obtained from Keyence data similarly had upwards of 3000 samples. Additionally, some samples are truncated from those siglas as described in preceding sections, decreasing the number of samples available for analysis still further. This imposes a limitation on the number of windows. Trial and error suggested that N = 10 windows would be an optimal number of windows for this particular data set. With 10 windows, a height profile or melt pool temperature time series with about 3000 samples, would have a window size of about 300. Increasing the number of windows beyond that, and therefore reducing the window size, did not prove to be advantageous.

4.4 Feature Extraction

After each 1D time series is reframed into a collection of overlapping windows, represented as a matrix where the number of rows is N and the number of columns is n_w. The windowed data is utilized to compute data features. For each feature, each window yields one observation. Each bead then yields N observations for each feature. The total number of samples in the data set is equal to N * number of beads.

4.4.1 Welding Process Data

Wire feed speed and travel speed, both welding process settings, are used as features. The travel speed in bead studies is always constant, therefore there is no need to subject travel speed data to a windowing procedure. The same travel speed is simply repeated N times. The same is true of the wire feed speed in Data set 1, which uses standard mode, since it is always constant. However, for Data sets 2 and 3 where the wire retraction causes significant fluctuations in wire feed speed, it is useful to compute sliding metrics other than the mean.

4.4.2 Waveform Data

Waveform data, such as acoustics and electrical signals, can be analyzed in either the time domain or the frequency domain. Time domain features are obtained by performing calculations on the windowed time series. One way to characterize a wave form in the time domain is by computing statistical metrics. The mean is the most obvious one but higher order statistics such as the standard deviation and kurtosis can also yield valuable information. For audio waveforms, only the kurtosis is computed because a number of authors have identified it as a predictive factor for arc stability [20, 21]. Kurtosis is also computed for voltage and current. The mean voltage and current are also obtained due to its central role in determining the heat input.

Over the course of experimentation, it was observed that the loudness of the arc varied significantly depending on process conditions. A way to measure that is to employ the root mean square (RMS). RMS is often used in signals with negative fluctuations, since these would reduce the mean (the average of a sinusoid for example is 0). In audio processing, the RMS can be thought of as the effective amplitude or effective loudness. The RMS of a discretized signal is found by

$$RMS = \sqrt{\frac{1}{K}\sum_{k=0}^{N} x(k)^2} \tag{4.8}$$

Another time domain metric is the zero crossing rate. The zero crossing rate is often used in audio processing to measure the smoothness of a signal, which useful for determining the pitch and timbre of sounds.

$$ZC = \frac{1}{2}\sum_{k=0}^{N} |sgn(S(k)) - sgn(S(k+1))| \tag{4.9}$$

Frequency domain analysis requires the signal to be transformed to the frequency domain which is achieved by applying a Fast Fourier Transform (FFT)

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}\,dt \tag{4.10}$$

When dealing with discretized data, such as the data sampled from sensors, a Discrete Fourier Transform (DFT) is used instead.

$$X(k) = \sum_{n=0}^{N-1} x(t)exp\left(-j\frac{2\pi kn}{N}\right) \tag{4.11}$$

where k is a frequency bin, and N is the number of samples. The problem with the traditional Fourier series is that it removes the time dimension, which prevents one from studying the evolution of the frequency domain features over time. A solution to this problem is to apply the DFT to windowed segments of the time series, which is exactly the premise behind the Short Time Fourier Transform (STFT). STFT is an efficient algorithm for computing the Fourier Transform of a windowed signal. This can be described by the equation

$$X(f) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j2\pi ft}\,dt \tag{4.12}$$

where $w(t-\tau)$ is a windowing function which can be rectangular, Hann, Gaussian, among others. This work will employ a rectangular window. A diagram demonstrating how the STFT works is shown in Figure 4.13. The STFT yields a matrix where each row or column corresponds to the frequency spectrum for a window of the time series. These spectral windows can be used to derive frequency domain features. These are mostly chosen based on intuition and information from the literature. A spectral centroid is the frequency that corresponds to the center of energy, analogous to the concept of a center of mass. It is calculated by the formula

$$f_c = \frac{\sum_{k=0}^{N} S(k)f(k)}{S(k)} \tag{4.13}$$

where $S(k)$ is the amplitude of the frequency spectrum at frequency bin k and $f(k)$ is the frequency at bin k. Once the spectral centroid is known, one can compute a measure for the spectral range surrounding the spectral centroid. This is known as the spectral bandwidth, which is a weighted mean of the distances of the frequency bands from the spectral centroids. The spectral bandwidth is found by the equation

$$f_b = \left(\sum_{k=0}^{N} S(k)\left(f(k) - f_c\right)^p\right) \tag{4.14}$$

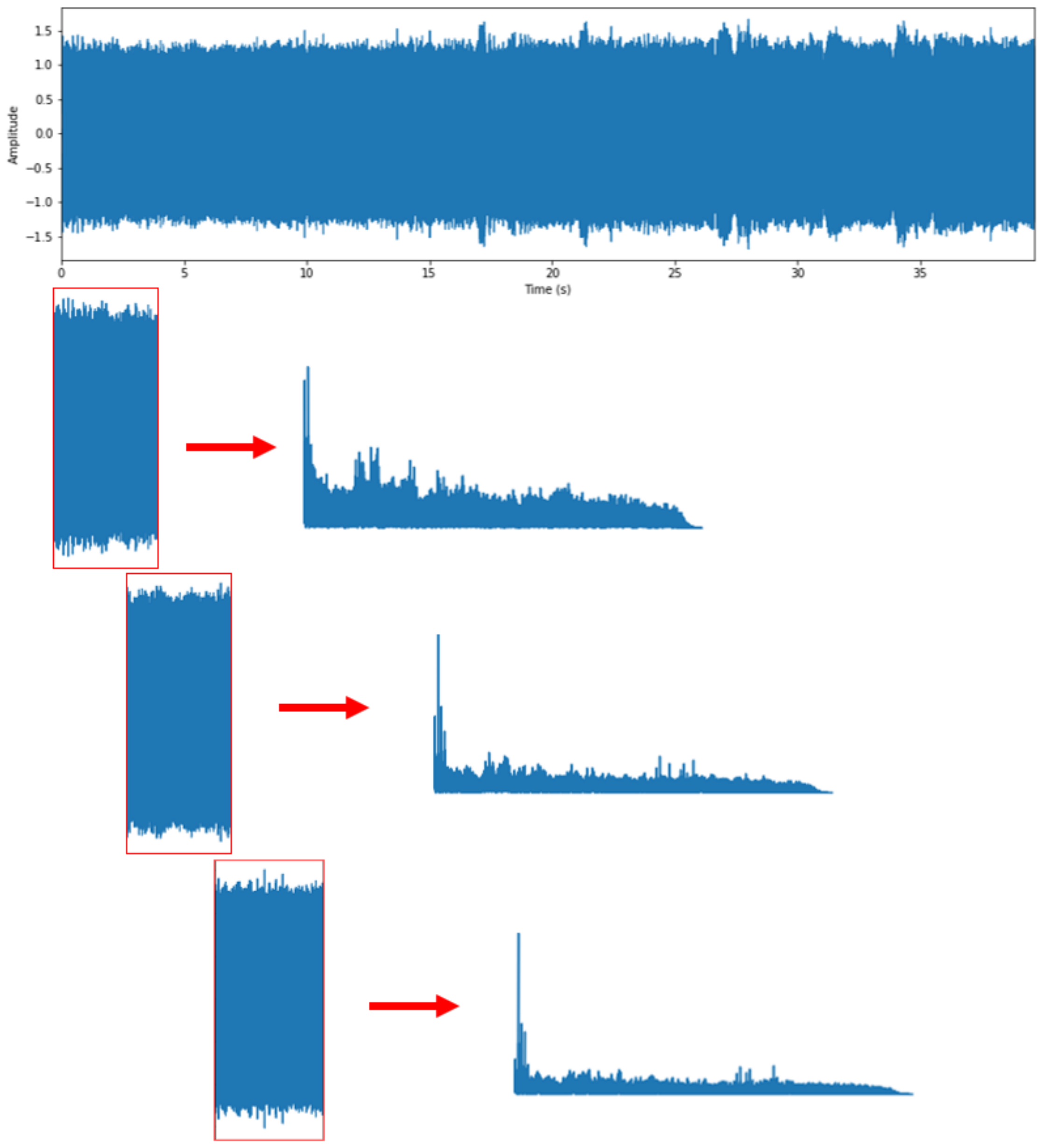


Figure 4.13: Diagram of working principle behind Short Time Fourier Transform. Frequency spectrums are computed for each overlapping window.

Finally, the concept of onset envelope is imported from signal processing literature. Onset is the occurrence of distinct and perceptible events in an audio signal such as a beat, or musical note. These events correspond to transient behavior in the signal such as a sudden shift in spectral energy. Though this concept was initially developed for musical processing, the concept of onsets is broadly applicable, as there many other contexts where one might be interested in detecting distinct events in an audio signal. Acoustic weld monitoring is such an application because the sound of welding always has discernible features such as pops, crackling, and sizzling. It is very likely that these types of sounds correspond to measurable shifts in the spectral energy of the signal, so that the methods developed for onset detection in music [44, 45] should have value in this application.

4.4.3 Thermography

The literature makes it clear that the thermal characteristics of the melt pool and of the inter-layer temperature both have a strong influence on defect formation and bead shape. Infrared data will be utilized to derive features that measure both of these traits. Bead on plate studies do not have inter-layer temperatures per se, since material is deposited directly on the substrate for a single pass. However, the temperature of the build plate itself should affect the wetting and wetting solidification behavior much as a layer would. It is also variable during experiments, since other welds on the plate increase its temperature. Therefore the base plate temperature directly under the travel path is used as a stand in for inter-layer temperature. The blob detection feature discussed in Section 3.3.3 can be used to measure the temperature of the melt pool. That data is directly obtained from the analysis software.

The type of time series data that could be obtained from the FLIR was very limited, as discussed in Section 3.3.3. Only statistics of the temperature such as the maximum, mean, or standard deviation could be exported as time series. This did not prove helpful in the bead on plate studies because the recording encompasses the time during which the arc was on. The arc is much hotter than the maximum temperature the camera can record, so clipping occurs. However, the average temperature of the base plate prior to material deposition could still be obtained. This temperature was measured by creating a linear ROI that is co-linear with the travel axis on the plate. With the line in place, a snapshot of the moment before arc ignition is taken. At that snapshot, a spatial profile of the temperature variation within that line is obtained, as shown in Figure 4.14. The average of the temperature profile was computed to obtain the base plate temperature.

4.4.4 Summary

The five independent sensor time series in addition to the spatial temperature line profile are analysed by the procedures described in the preceding subsections. This feature extraction procedure results in a set of 18 features, as shown in Table 5.1. These features, with the exception of Travel Speed and Base Temperature are computed on a per windowed segment basis. Every overlapping window of a bead shape profile is associated with these 18 features, by way of the windowing procedure and feature extraction procedure applied to the corresponding windows of the time series data. The input to the model, therefore, is a matrix with the number of rows equal to the number of segments per bead * number of beads and the number of columns equal to the number of features.

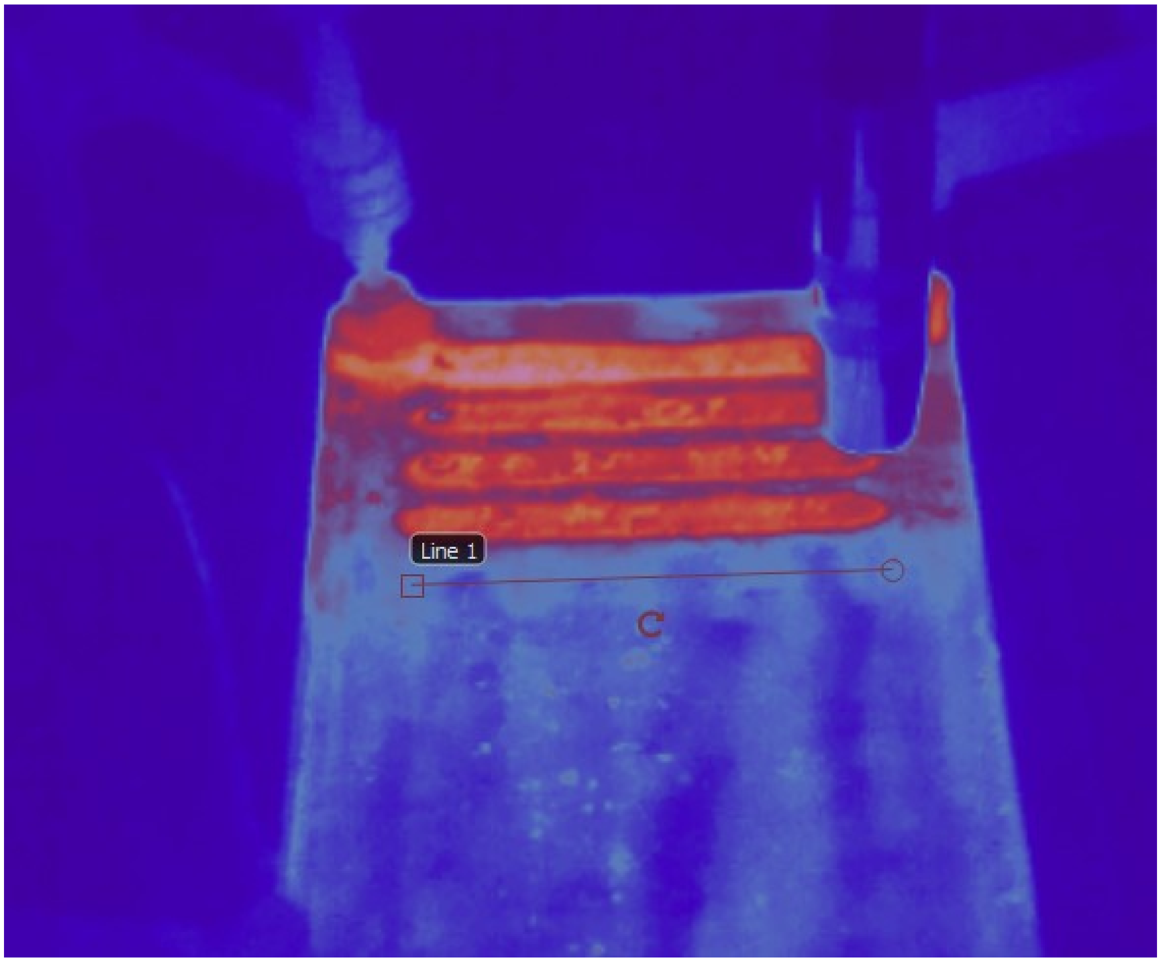


Figure 4.14: Snapshot of FLIR thermal imaging immediately prior to arc ignition. Line is placed on the spot where the bead is going to be

Table 4.1: Features Derived from Multi-Sensor Framework

Feature	**Raw Data Source**
Welding Voltage Mean	Welding Voltage
Welding Voltage Std	Welding Voltage
Welding Voltage Kurt	Welding Voltage
Peak Frequency	Welding Current
Welding Current Mean	Welding Current
Welding Current Std	Welding Current
Welding Current Kurt	Welding Current
Wire Feed Speed Mean	Wire Feed Speed
Wire Feed Speed Std	Wire Feed Speed
Melt Pool Temperature	Melt Pool Temperature
Audio Kurtosis	Audio
Spectral Centroids	Audio
Spectral Bandwidth	Audio
Root Mean Squared	Audio
Zero Crossing Rate	Audio
Onset Strength	Audio
Travel Speed	Robot Position
Base Temperature	Temperature Line Profile

4.5 Model training

4.5.1 Feature Pruning

Reducing the dimensionality of the input space prior to training is desirable due to the so called curse of dimensionality, which postulates that adding dimensions to the input space rapidly increases the number of observations needed to fully describe it. Consequently, for most machine learning problems adding dimensions only improves a model's performance up to a certain point, after which it begins to degrade. One method for rejecting features prior to training is to identify features that are highly correlated. The correlations between each of the input features can be visualized using a correlation matrix, as shown in Figure 4.15. If there are two or more features that are more than 95% then all but one features in that set are considered redundant and all but one of the correlated features are rejected. Out of the features shown in Table 4.1, only Welding Current Mean and Wire Feed Speed Mean were correlated at 95% or higher. Wire Feed Speed Mean was then rejected.

4.5.2 Standardization

A common problem in machine learning is input features have significantly different scales. The data obtained in this work provides an example. The various types of frequency domain features discussed in the previous section have values ranging in the thousands. Meanwhile, average welding voltage ranges from 0 to 20 V. The difference in scale is multiple orders of magnitude. In some algorithms this difference in scales severely handicaps the models because the features with values on a higher scale impact the model more, giving a false impression that they are more important to the outcome than other features. To address this problem, the values of each feature must be re-scaled so that they all share a common scale, without altering the overall distribution of the values. There are two common strategies for addressing this problem: normalization and standardization. Normalization means re-scaling the values to fit between 0 and 1. Standardization means re-scaling the data to make it have a mean of 0 and a standard deviation of 1. Standardization is useful in situations where the data is normally distributed, or the model assumes such a distribution. None of the data features were normally distributed, thus normalization was used.

4.6 Model validation

4.6.1 Validation Techniques

Once a model is trained, it must be tested on data it hasn't yet seen. This is usually done by setting some of the data aside during training, training the model on a subset of the data and then using the model to predict the unseen data. The drawback of this approach is that it does not guarantee a well performing model. Even if the model performs well on the test set, it could be because the output observations in the test set were easier. This problem can be mitigated by training the model on multiple different subsets of the data so that model sees the entire data set. This approach is called cross-validation. K-fold cross validation is a type of cross validation where the data is split into subsets called folds. During training, the model is reinitialized and trained on multiple iterations, where for each iteration a different subset is set aside to be the test set. The number of iterations will be equal to the number of folds. Once every fold has been the test set once, the loop is completed. A diagram of K-fold cross validation is shown in Figure 4.16.

4.6.2 Feature Selection

Another way to combat the curse of dimensionality is to implement feature selection. This differs from feature pruning in that it is only done after a model is trained rather than before. The model itself is utilized to obtain a feature ranking based on the impact of each feature to the model. Once such a ranking is obtained, the least impactful features can then be eliminated, and the model can be retrained with the new reduced feature set. Not all machine learning methods lend themselves to feature selection. Some methods such as Random Forests have a built in feature importance ranking mechanism. Random Forests can produce a estimate of a feature's importance based on the mean decrease in node impurity. In other words, it's a measure of how much splitting on that feature gets the model closer to a final prediction, averaged across all trees. It is possible to estimate feature importance for other model types, however. One method is computing the decrease in accuracy when that feature is removed from the model.

SHAP (SHapley Additive exPlanations) Values have emerged as one of the best methods for improving machine learning interpretability, and they can be employed in a wide array of machine learning methods, in particular neural networks such as the ones developed in this study. SHAP Values are a concept imported from game theory where they are used to show the contribution of individual players to the outcome of a game. In machine learning, they are used to quantitatively describe the contribution of individual features to the outcome of a model. This is achieved by finding subsets of the feature set called coalitions, where the set of all features is known as the grand coalition. For each possible coalition, the difference in the model score is computed before and after feature i is removed. This parameter is computed and averaged across all non-repeating permutation of possible coalitions in the entire feature set,

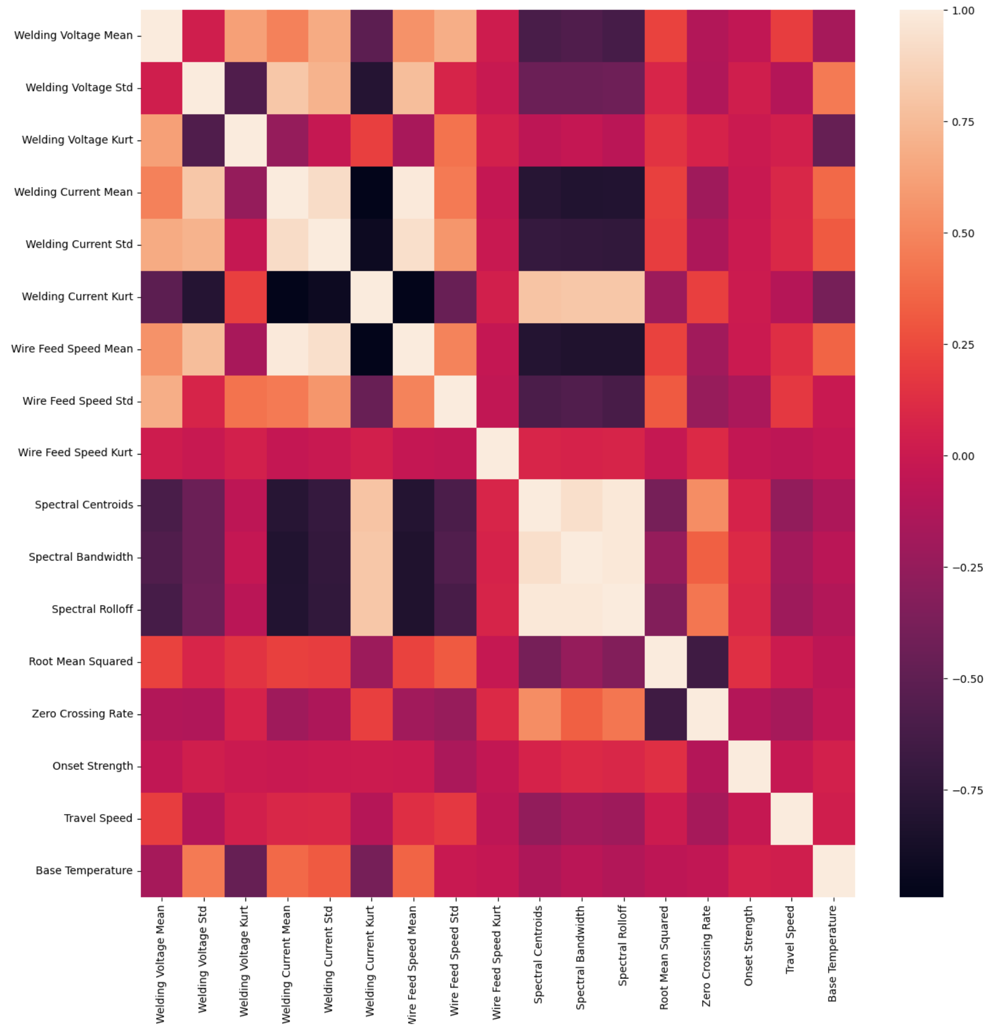


Figure 4.15: Correlation Matrix. Legend is the color bar on the right. Scale is unitless and consists of Pearson correlation coefficients.

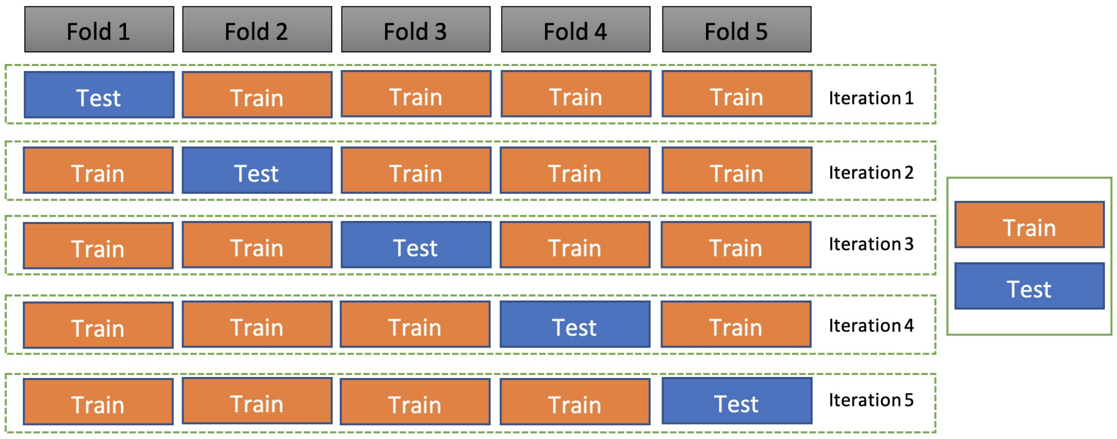


Figure 4.16: 5 Fold K-Fold cross validation

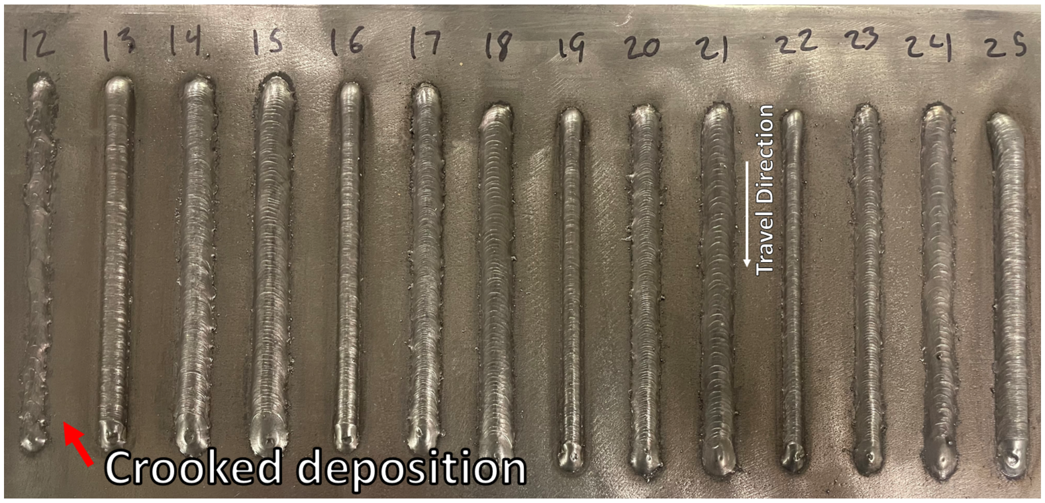
Chapter 5

Single Layer Model

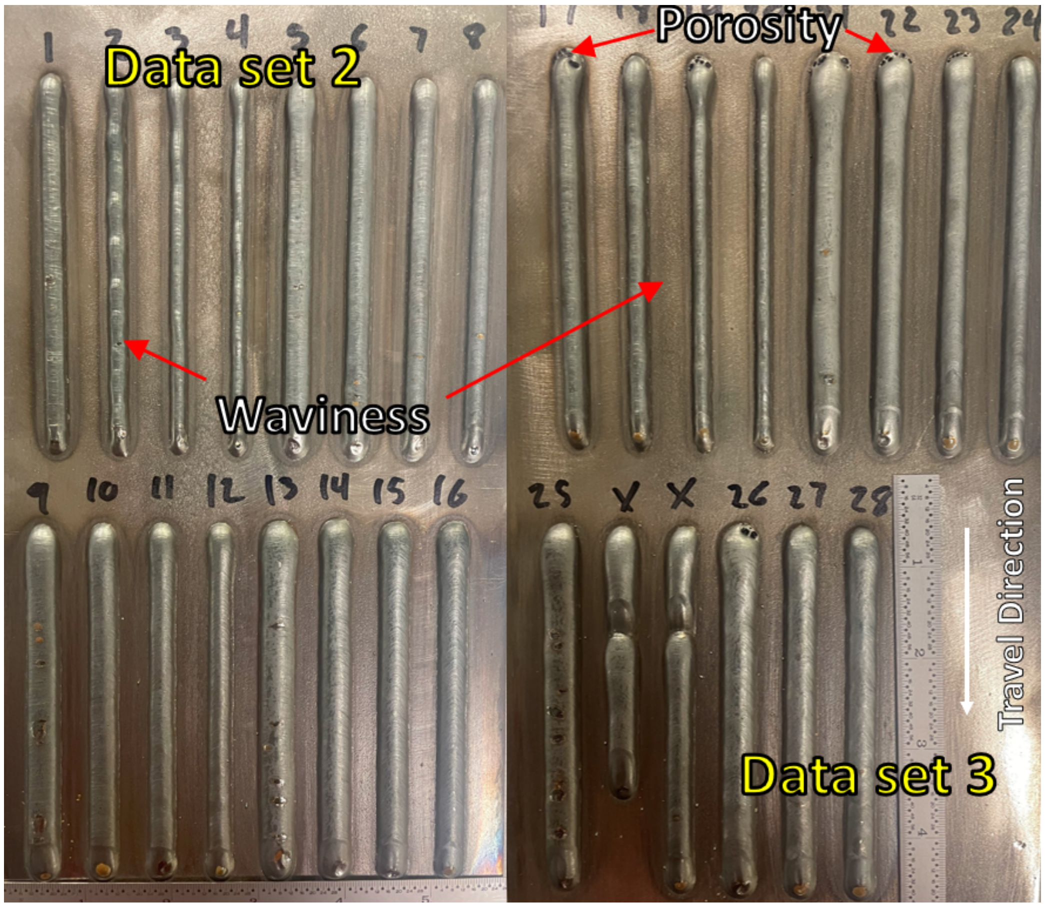
5.1 Experimental Results

Between the three data sets, 134 beads were deposited. These beads exhibited a diverse array of shapes and dimensions. Visual inspection reveals that shape irregularities such as waviness and crookedness discussed in Chapter 2 of this work are well represented. Figure 5.1 shows a summary of the type of irregularities found in the bead samples. Predictive models are developed and trained on data belonging to one of these data sets at a time. The results of the models developed from the three data sets will be compared and the differences in model performance will be discussed.

The attempt to purposefully generate porosity on the bead surface by increasing the CTWD was only partially successful. As Figure 5.2 shows, porosity did indeed appear on some of the beads with a CTWD of 18 mm and 24 mm. However, this porosity was only visible on the bulge that forms immediately after arc ignition. This result is not useful for the objective of developing a model that maps time series sensor data to a porosity metric. The portion of the end-cap where porosity formed in all the beads is outside of the robot travel path, so there is no time span in the



(a) Beads from Data Set 1, produced by a Standard process



(b) Beads from Data Set 2 and 3

Figure 5.1: Examples of irregularities and defects found on single pass beads

sensor data that corresponds to visible surface porosity. Consequently, developing a predictive model for porosity from bead studies was abandoned as an objective.

Shape irregularities on the other hand were abundant. The shape properties of the beads from all three data sets is shown in Table 5.1. In order to summarize the shape properties of entire data sets, the shape metrics computed for every windowed segment of every bead in the data set were averaged and are reported as a single value. On average, the beads from Data Set 1, which were produced by a Standard process, were wider and shorter than those from the CMT data sets. Data Set 1 beads also exhibited significantly lower straightness than the other data sets, with the center line deviation being more than double that of Data Set 2, and 65% higher than Data Set 3. Coupled with the observation that the arc sound during the most crooked Data Set 1 beads was more akin to a fast popping sound rather than the fast sizzle/crackling usually associated with stable welds, suggests a greater prevalence of process instability during the Standard process. The rough surface of most of these welds, such as the one shown in Figure 5.3 likely also resulted from that fact. On the other hand, Data Sets 2 and 3 had greater shape variation in terms of Height Standard Deviation. This likely reflects the prevalence of wavy beads, such as the one shown in Figure 5.4, in that data set. The same is not entirely true for Height Peak-to-Valley Difference (HPVD), where Data Set 3 did have a larger HPVD than Data Set 1, but Data Set 2 did not.

The shape metrics were plotted against length to obtain the profile. Since these measurements are all for sliding length segments of the shape profiles, the length had to be windowed as well. The sliding shape metrics are therefore plotted against the sliding mean of the length. Confidence that these metrics are reliable descriptors of shape irregularity was established by comparing them to images of the beads themselves, as shown in Figure 5.5. Both beads shown are among the waviest beads in the entire study, which is reflected in the well above average values for the HPVD and Height Standard Deviation found along the length of these beads. Consistent bead profiles on the other hand had very low values for both of those metrics.

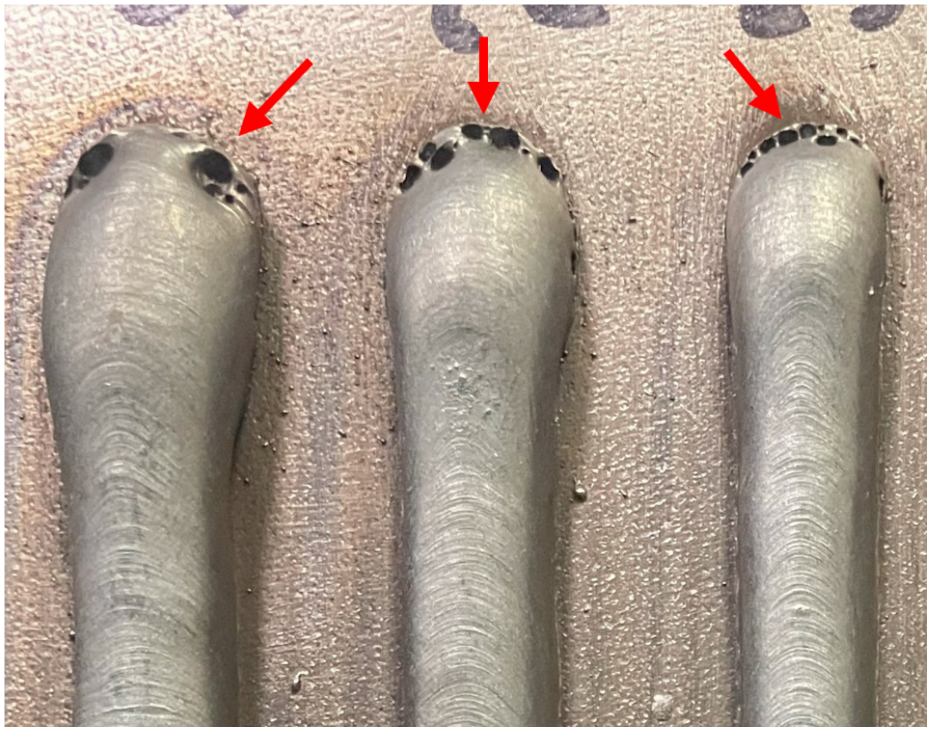


Figure 5.2: Porosity developed on the end cap of some beads in Data Set 3

Table 5.1: Features Derived from Multi-Sensor Framework

Dataset	**Height (mm)**	**Width (mm)**	**Peak-to-Valley (mm)**	**Height St. Dev. (mm)**	**Center Line Deviation (μm)**
Dataset 1	2.232	7.618	161.8	65.72	145.7
Dataset 2	3.388	7.18	153.9	68.48	72.83
Dataset 3	3.740	7.329	165.4	75.89	85.24

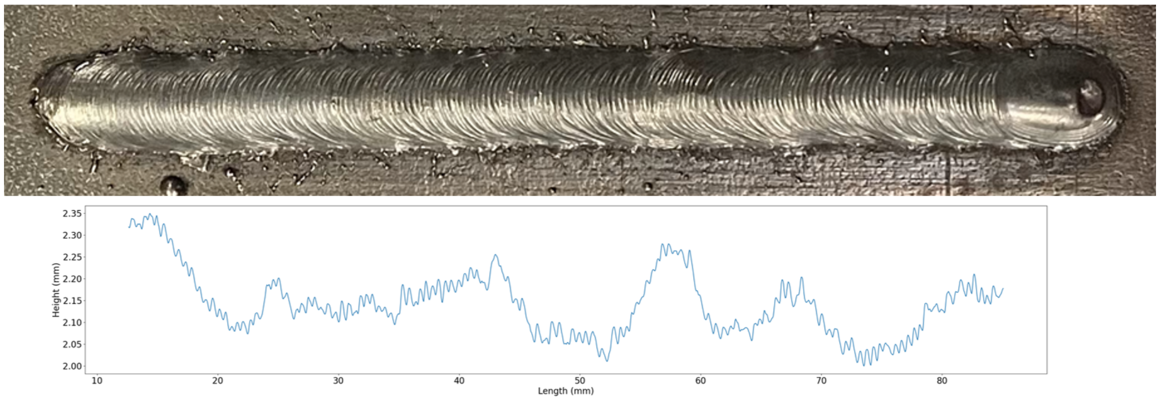


Figure 5.3: Rough bead surface (Bead 32) from Data Set 1

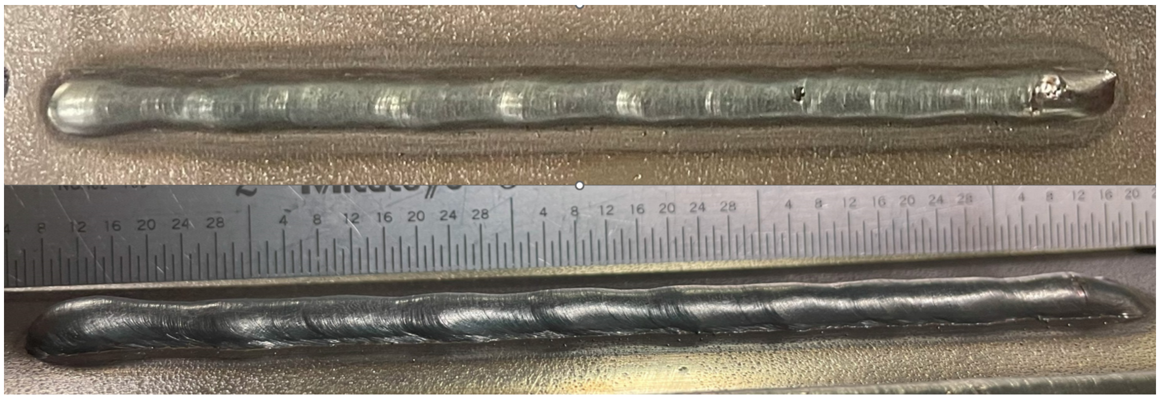


Figure 5.4: Example of wavy bead (Bead 2) from Data Set 2

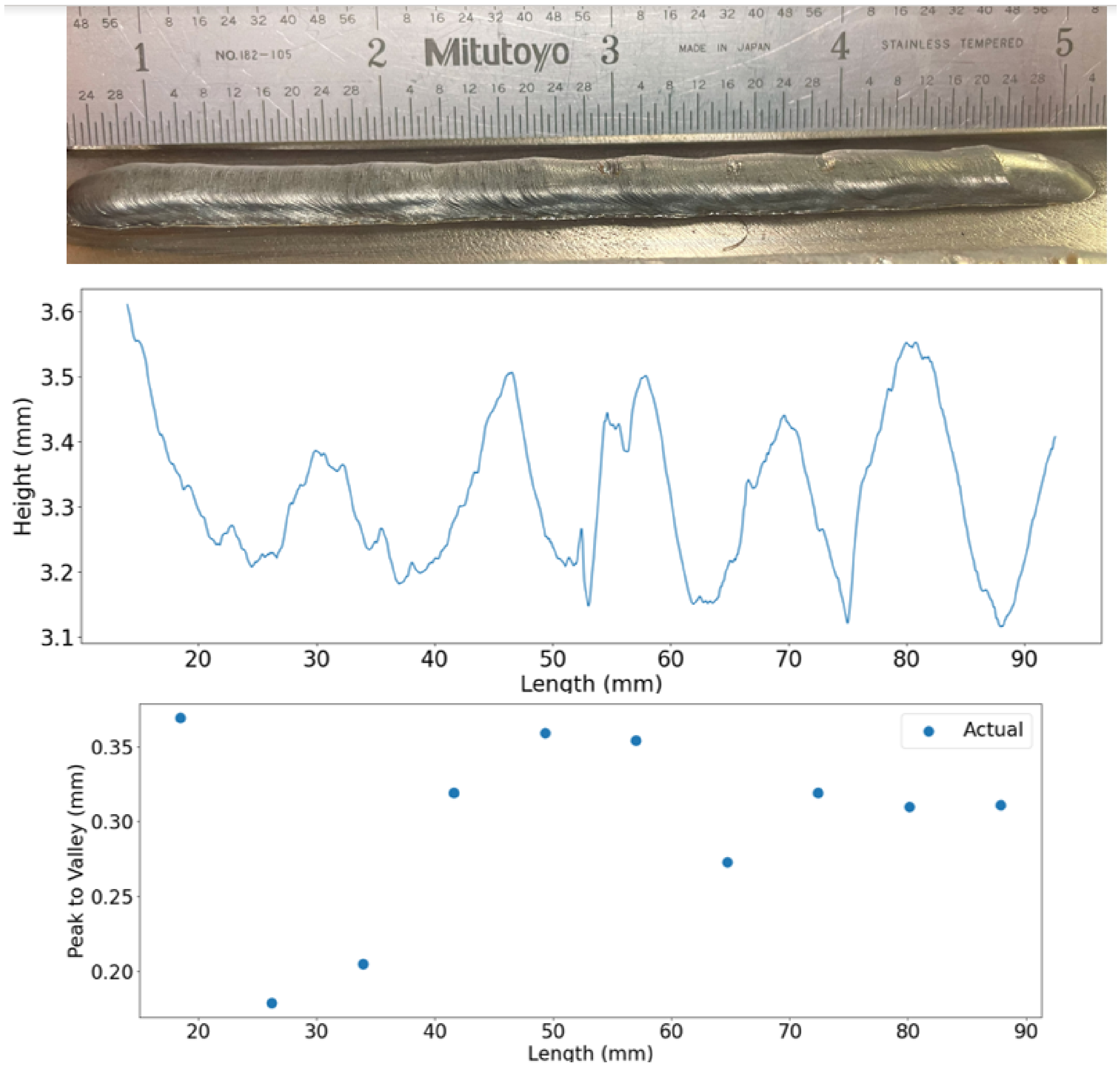


Figure 5.5: Actual bead (Bead 37) from Data Set 2 compared to its height profile and peak to valley profile

5.2 Model Performance

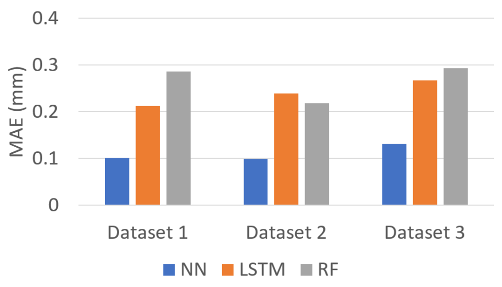
Five shape metrics, three machine learning models and three data sets yields a total of 45 different shape prediction models. This sprawling array of models can be summarized by comparing the model scores, in this case reported as mean absolute error (MAE). The differences in model performance between each data set and machine learning model for each shape metric are shown in Figure 5.6. The simple neural network (NN) performed the best across the board, regardless of metric or data set. Overall, random forests (RF) performed the worst. When predicting width or height RF performance was very close to NNs and LSTM, even competitive as in the case of Data Set 2. When predicting profile consistency, however, RF was consistently the worst, and often by large margins. The reason for RF under-performance is not clear but is likely related to the discontinuous nature of RF predictions. The LSTM models performed favorably compared to RF and NN, however, no improvement over simple NNs was observed. This failure could be explained in two ways. One is that this is a non-traditional application for LSTMs, since LSTMs are typically used for applications where there the input data is not processed into features. Secondly, training and hyper-parameter tuning and training is more difficult for LSTMs; moreover, LSTMs introduce more hyper-parameters into the picture such as lock back period. Thus it could simply be that the optimal hyper-parameters for these models have not been found.

To better visualize the model performance, the measurements can be plotted against the model predictions. This is done for every model in the study. These comparison plots demonstrate the close agreement between measurement and prediction of Height and Width for NN and LSTM, as shown in Figure 5.7. This close agreement exists in neural network models from all data sets. On the other hand, the RF models were able to predict height and width, but with significantly greater dispersion, as shown in Figure 5.8.

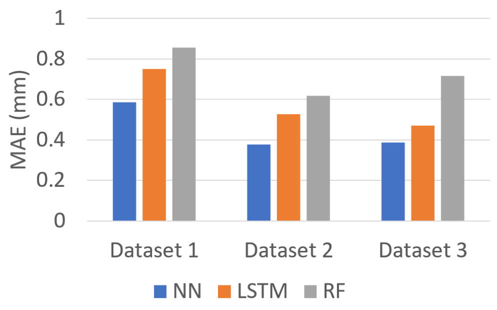
The predictions for metrics measuring shape consistency exhibit significantly more error compared to predicting width and height. The relative errors for shape consistency were approximately 7x higher than those for height and width. This is hardly surprising considering shape consistency is a higher order property, making it harder to predict. Nevertheless, NN and LSTM performed relatively well for all data sets, as shown in Figure 5.9. RF, however, performed very poorly for all shape consistency metrics, as shown in Figure 5.10. The RF models exhibit the greatest error toward the higher end of the measured shape metrics, which renders such models unworkable for real time prediction. The objective is to predict shape irregularities, which in this context translates to above average values for shape consistency metrics. This requires a model that can predict the upper ranges of the shape metrics with greater accuracy.

Another way to visualize model performance is to reconstruct the shape profiles, as seen in Figure 5.11. The actual bead profile with respect to length is compared to the predicted bead profile. The predicted height profile is compared to the actual height profile for two beads in data set 3. The bead profile from Figure 5.11a shows a very close agreement between measurements and predictions, except for the first measurement. Figure 5.11b shows more spotty agreement, but the first measurement error still stands out. This observation was repeated for nearly all beads in all three data sets. This persistent failure to predict the first measurement in the profile suggests that this region corresponds to a transient where the model is incomplete or inaccurate. Its physical manifestation is a bulge found at the front end of all the beads, which despite best efforts was not sufficiently trimmed off. Indeed, many of the trimmed height profiles still have a sizeable slope at the start of the profile.

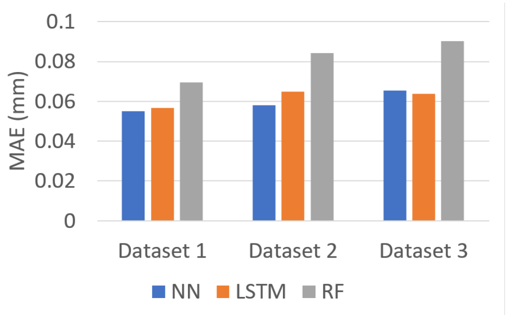
Shape profiles for shape consistency metrics were also reconstructed. Figure 5.12 demonstrates that for some beads, shape consistency model predictions also exhibit close agreement with observation. The first windowed measurement (i.e. closest to length = 0) is sometimes the worst, similar to the Width and Height profiles, though



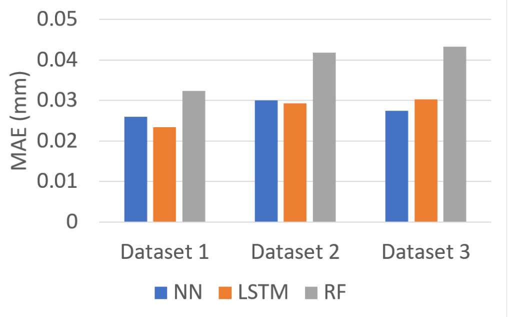
(a) Model performance summary for Bead Height



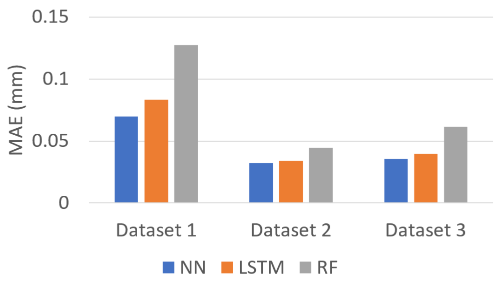
(b) Model performance summary for Bead Width



(c) Model performance summary for Peak to Valley Difference

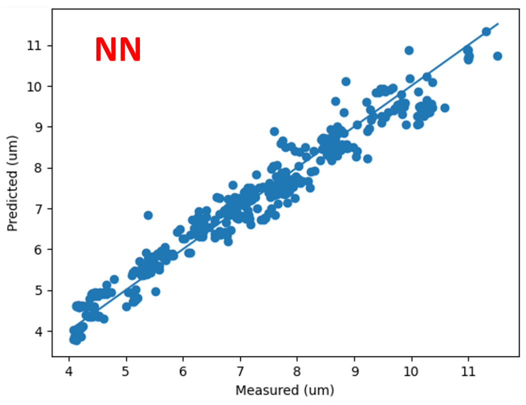


(d) Model performance summary for Bead Height Standard Deviation

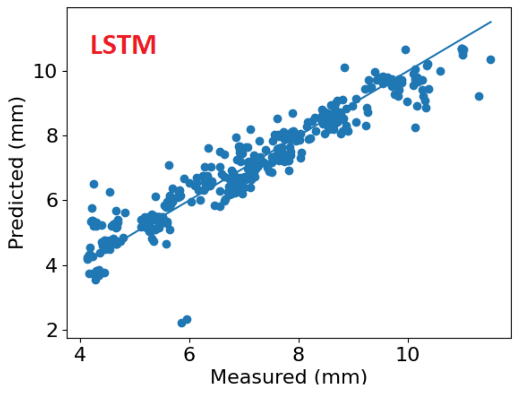


(e) Model performance summary for Center Line Deviation

Figure 5.6: Model performance summaries for all runs

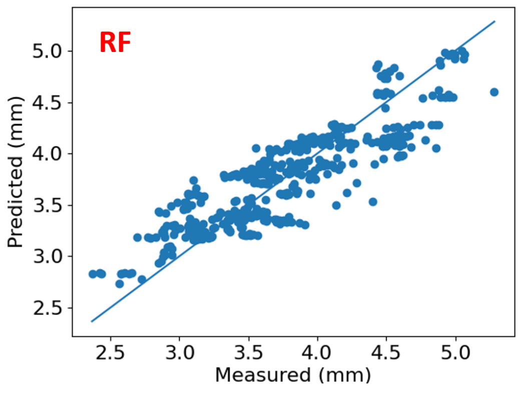


(a) Performance of NN model for predicting Bead Height for Data Set 2.

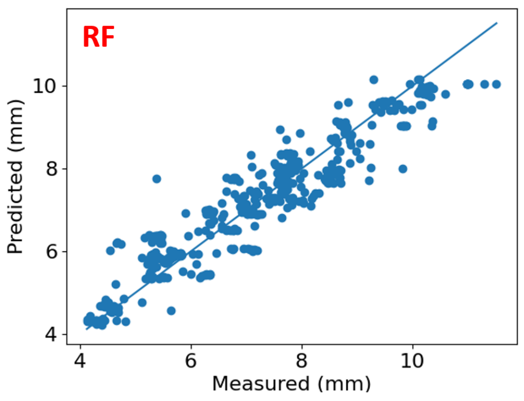


(b) Performance of LSTM model for predicting Bead Width on Data Set 2

Figure 5.7: Neural net based models predict height and width very well. The individual scatter points represent a measured value, shown on the horizontal axis and its corresponding prediction, shown on the vertical axis. The solid line represents a 1 to 1 ratio between measurements and predictions so points along that line represent perfect predictions.

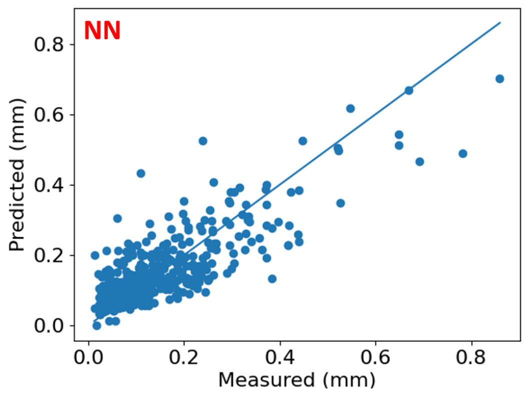


(a) Performance of RF model for predicting Height on Data Set 3

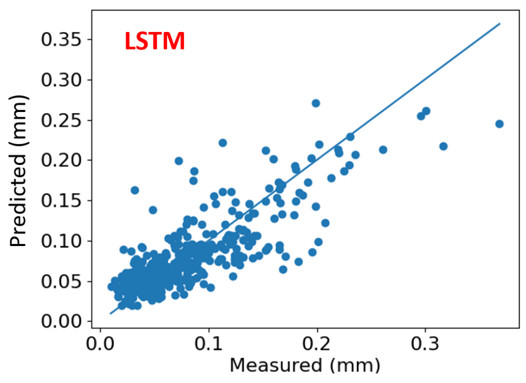


(b) Performance of RF model for predicting Width on Data Set 3

Figure 5.8: Performance of RF models for predicting Height and Width

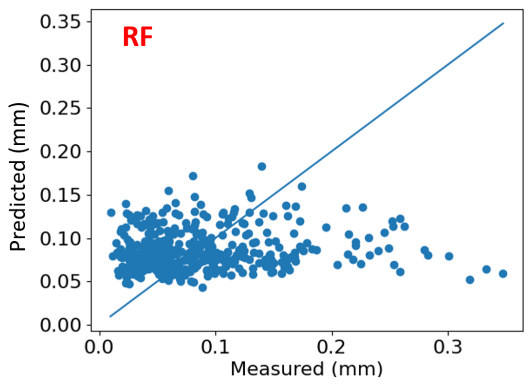


(a) Performance of NN model when predicting Center Line Deviation on Data Set 1

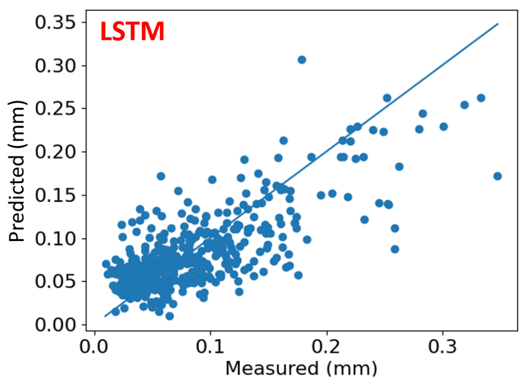


(b) Performance of LSTM model when predicting Height Standard Deviation on Data Set 13

Figure 5.9: NNs and LSTMs displayed moderate performance when predicting bead shape consistency metrics



(a) Performance of RF model when predicting Center Line Deviation on Data Set 3



(b) Performance of LSTM model when predicting Center Line Deviation on Data Set 3

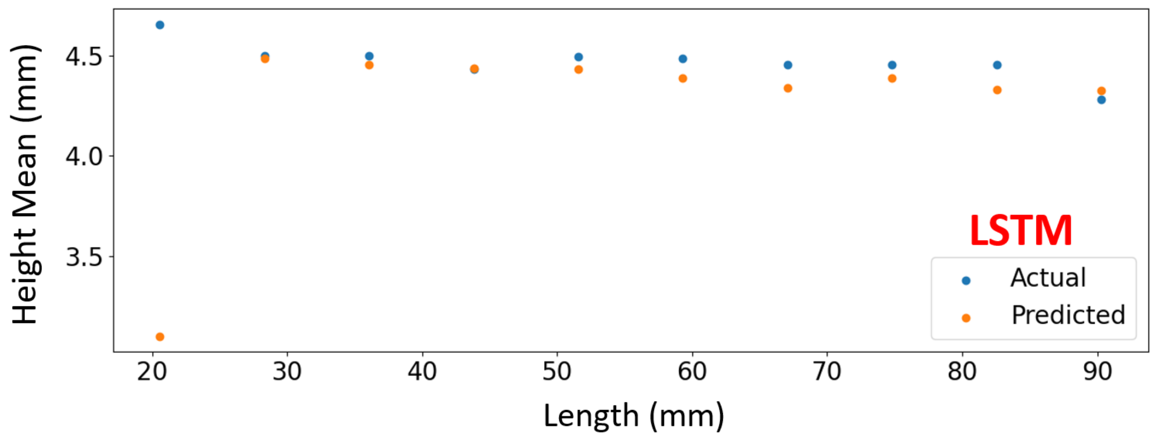
Figure 5.10: Comparison of model performance between RF and LSTM for the same output metric and data set. RF performance is vastly inferior

this is not nearly as prevalent compared to the universality of that phenomenon in the width and height models.

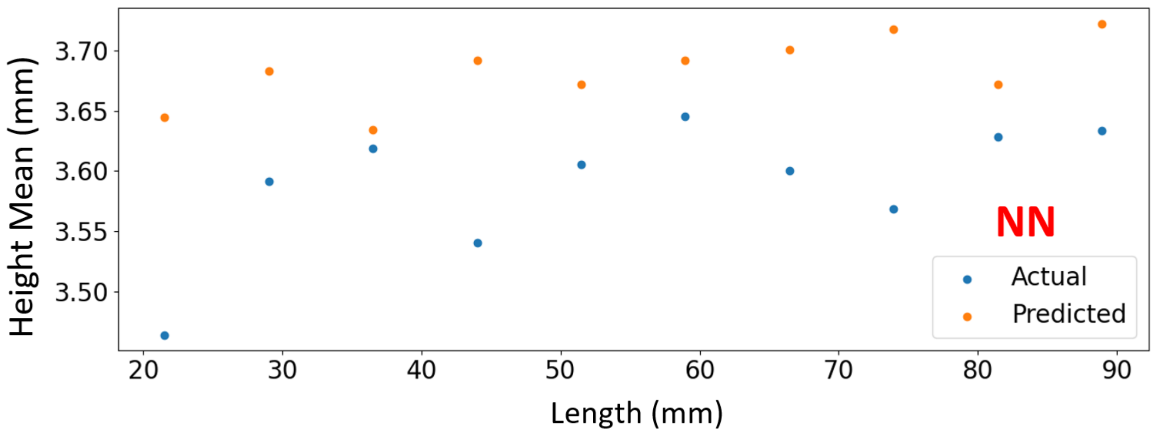
5.3 Feature Importance

A feature importance ranking for each model was obtained by employing SHAP Values for NN and permutation feature importance in RF. To streamline the process, no feature importance was obtained for LSTM. The feature importance rankings for Peak to Valley Difference in Data Set 3 are shown for both RF and NN in Figure 5.13. Interestingly, there is some degree of overlap between the two, despite the difference in the methods that generated them. A handful of features such as Spectral Centroids, Onset Strength and Welding Voltage Standard Deviation all rank near the top in both lists while features such as Root Mean Squared and Melt Pool Temperature rank near the bottom of both.

There was also a noticeable difference in the features found near the top of the ranking between Height and Width models and shape consistency models. Figure 5.14 shows a comparison between the feature importance ranking for the Height model and the Height Standard Deviation model for the same data set. Audio features could consistently be found in the top 5 features of most shape consistency models. On the other hand, almost all the top 5 features of width and height models were features derived from process variables such as travel speed, arc current/voltage and wire feed speed. In fact, travel speed was by far the most predictive feature in nearly all height and width NN models, with the average impact on model being double or more that of the next most important feature. However, this does not mean that travel speed alone can predict all the outcomes since the travel speed was constant throughout the entire path of a single bead (though it did vary between beads, hence its predictive power), which means it cannot account for the variations in bead dimensions throughout the bead.

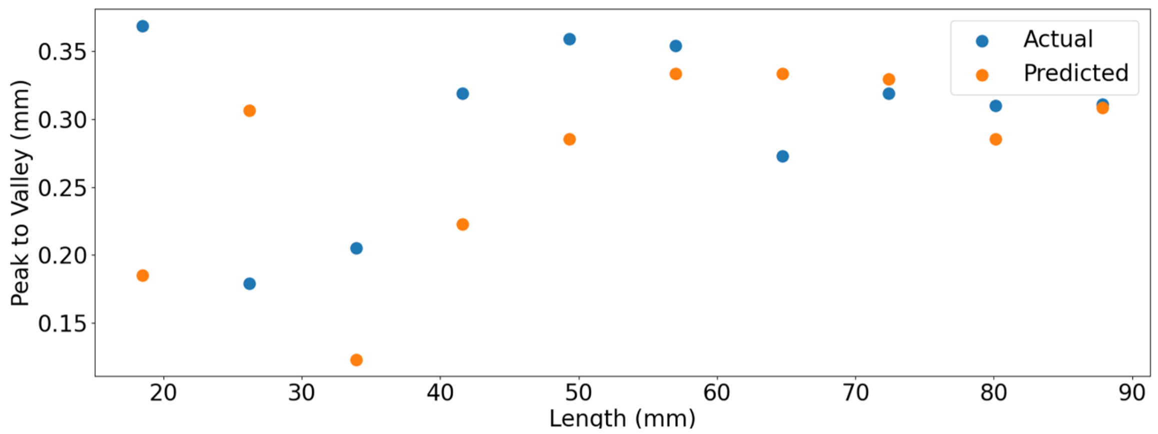


(a) Reconstructed bead height profile compared to actual bead height profile for Bead 33 from Data Set 3

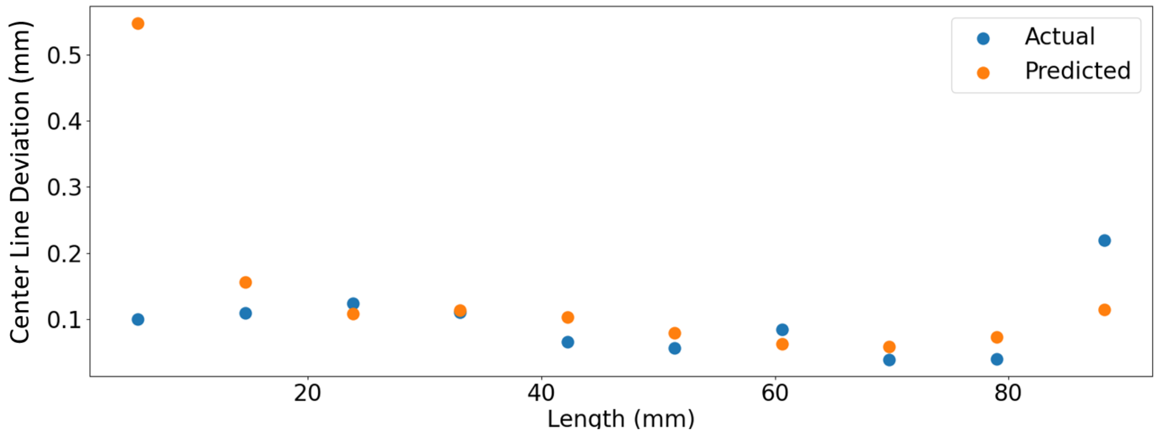


(b) Reconstructed bead height profile compared to actual bead height for Bead 15 from Data Set 3

Figure 5.11: Reconstructing the bead height profiles using model predictions yields reasonably close agreement with the measured profile. Note the large error for the first measurement

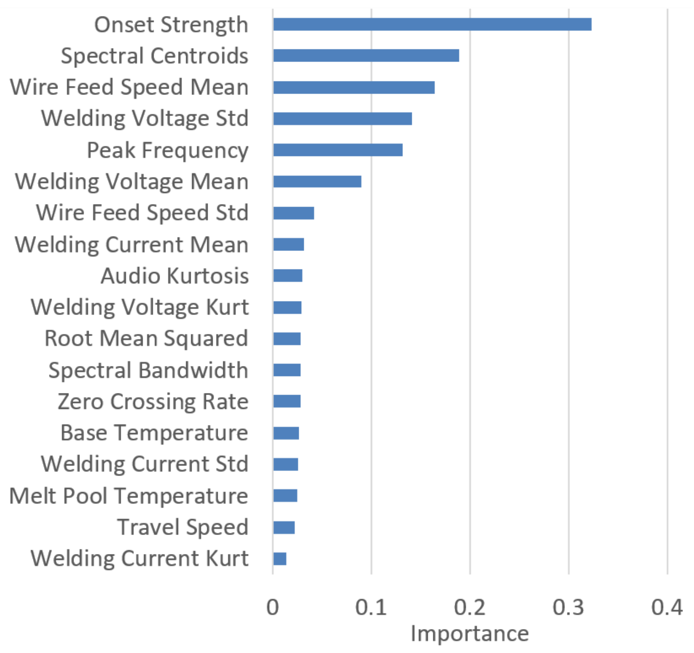


(a) Reconstructed peak to valley difference profile compared to actual peak to valley difference profile for Bead 37 from Data Set 2

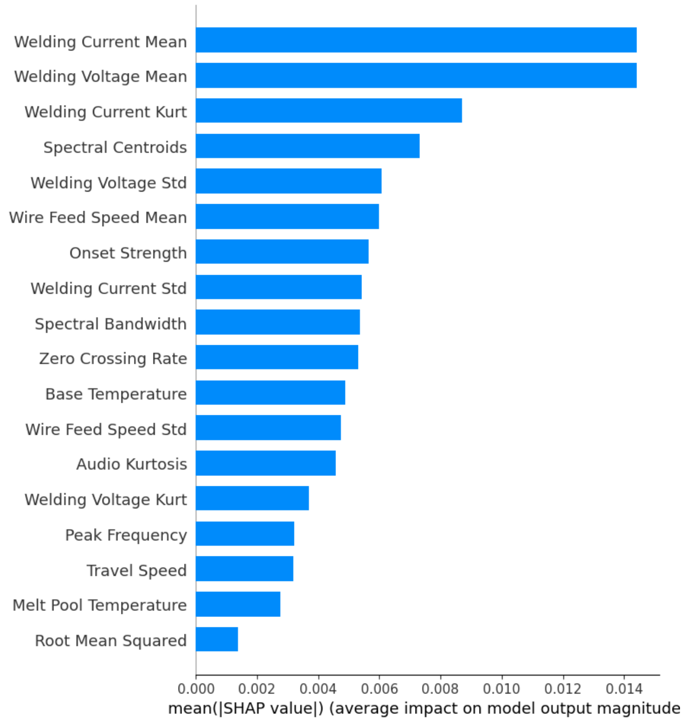


(b) Reconstructed center line deviation profile compared to actual center line deviation for Bead 42 from Data Set 1

Figure 5.12: In terms of bead shape consistency metrics, models also yielded very close agreement with measured values for many beads



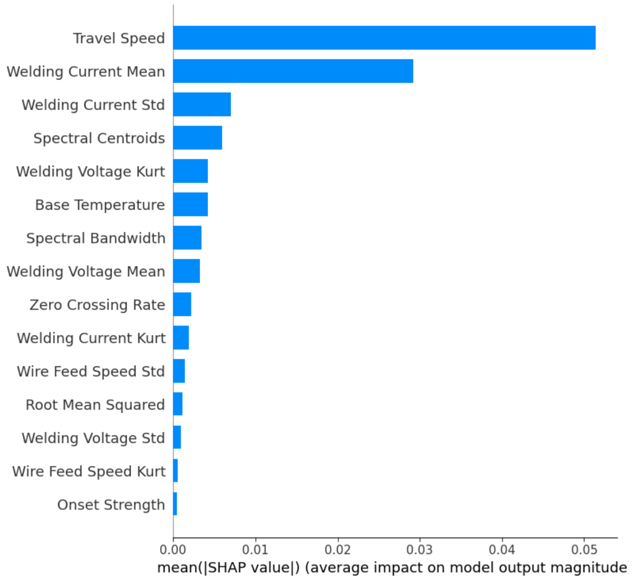
(a) Permutation Feature Importance for Random Forest



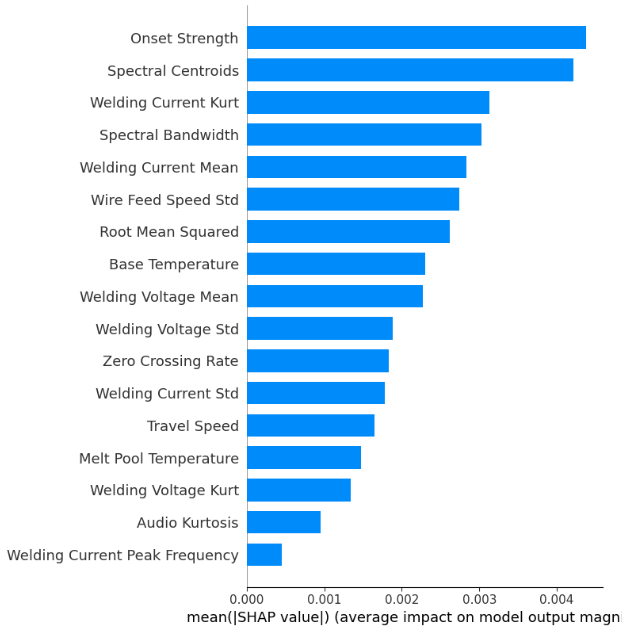
(b) SHAP Value Feature Importance for Neural Network

Figure 5.13: Feature importance ranking for Data Set 3 Peak to Valley in RF and NN

An attempt was made to utilize the feature importance rankings to improve the model performance by eliminating poorly performing features. This was first attempted with SHAP values on NN models. The bottom 2 features for the Peak to Valley Difference model in Data Set 2 were removed and the model was retrained. In this model, the least importance features were found to be Root Mean Square and Melt Pool Temperature, so those features are removed from the set and the model is retrained. Unfortunately, no improvement in model performance was observed. In fact, the performance slightly decreased. The comparison plot before and after is shown in Figure 5.15. The difference between the two is barely perceptible. Alternatively, features can be eliminated one feature at a time, with the model being retrained after a feature is discarded. This process is known as Recursive Feature Elimination (RFF). This process was tested on a RF model for bead height. The best results were obtained after 10 features were eliminated, leaving 7 features in the model. RFF minimally improved the RF model as shown in Figure 5.16.

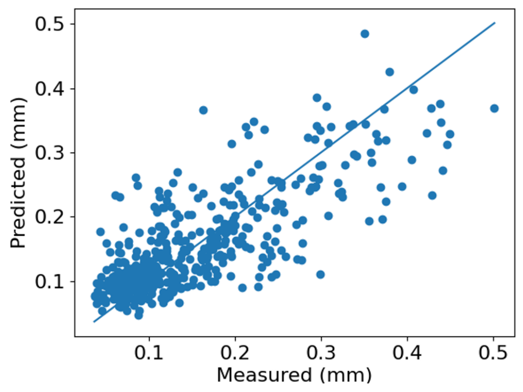


(a) Reconstructed peak to valley difference profile compared to actual peak to valley difference profile for Bead 37 from Data Set 2

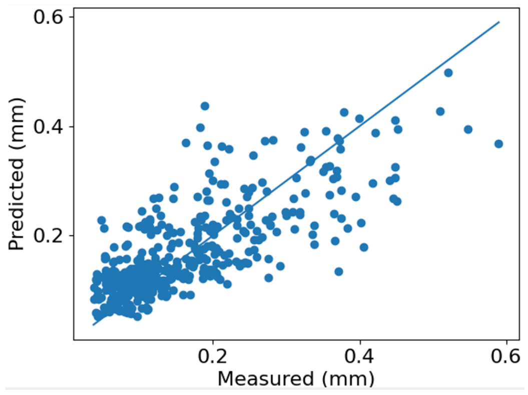


(b) Reconstructed center line deviation profile compared to actual center line deviation for Bead 42 from Data Set 1

Figure 5.14: In terms of bead shape consistency metrics, models also yielded very close agreement with measured values for many beads

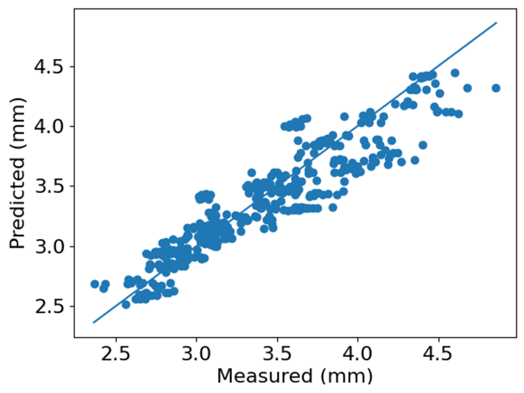


(a) Model performance of Data Set 2 Peak to Valley Model Before Feature Elimination. MAE = 0.058 mm

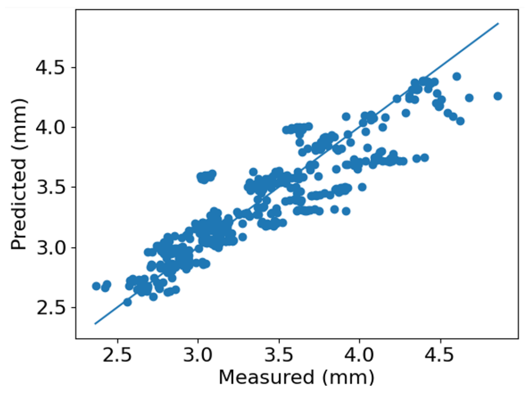


(b) Model performance of Data Set 2 Peak to Valley Model After Feature Elimination. MAE = 0.070 mm

Figure 5.15: No improvement in model performance observed after eliminating two features



(a) Random Forest model performance of Data Set 2 Bead Height Model Before Recursive Feature Elimination. MAE = 0.08204 mm



(b) Random Forest model performance of Data Set 2 Bead Height Model After Recursive Feature Elimination. MAE = 0.07888 mm

Figure 5.16: No improvement in model performance observed after eliminating two features

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Overall, the approach developed in this work was largely successful at predicting bead shape and bead shape consistency. Models for predicting height and width performed very well across every data set and machine learning method. Models for predicting shape consistency as described by Height Peak to Valley Difference, Height Standard Deviation, and Center Line Deviation yielded promising results, but still show room for improvement. The difference in model performance between the two categories is hardly surprising given that predicting the change in the bead shape as opposed to the bead shape itself makes it a higher order problem. Nevertheless, the neural network based models, both simple and recurrent, were still able to give reasonable approximations for bead shape consistency. Random Forests on the other hand performed very poorly in this domain and as such likely does not have a place in predicting bead shape consistency. A simple neural network was consistently the best performing model regardless of outcome metric or data set. This contradicts the expectation that the mechanism for incorporating past events that underlies LSTM would improve predictions. However, this expectation frequently does not pan out for a number of reasons. Hyper-parameters are more difficult to tune in LSTMs and there are more of them that need to be tuned. It is highly likely that all 15 LSTM models in this study were not optimally tuned. Furthermore, all of the sequences fed to the LSTM models were rather short, each only containing 10 time steps (windows) which likely did not provide enough time steps for the long term memory mechanism to be impactful.

Feature importance analysis yielded some interesting findings. The fact that shape consistency metrics had audio features well represented near the top of the rankings, while Height and Width models primarily had welding process variables near the top demonstrates the validity of the multi-sensor approach. Clearly, depending on the outcome of interest, data from disparate sources can be highly predictive. The comparatively poor performance of thermal data should be taken with a grain of salt. For one, in several of the models, Base Temperature and Melt Pool Temperature were at least in the middle of the pack in terms of feature importance. Secondly, there were a couple problems with this data. The Base Temperature data was not a function of time, it was only a spatial profile in the snapshot immediately before arc ignition. Therefore, on some segments of the bead, the actual Base Temperature was probably very different at the time when it was actually deposited. Obtaining the necessary spatio-temporally varying data from the radiometric files is possible but proved to be too unwieldy a task for this project. Secondly, matching the start and end points of the melt pool temperature data was challenging since this data was not recorded synchronously with the rest. Melt pool imaging performed on a different software package, and recording had to be activated separately from the other time series.

6.2 Future Work

6.2.1 Model Improvements

It is not likely that the performance of bead shape consistency models is sufficient for deployment to a real time prediction framework. However, the author recommends several measures that could be immediately implemented to improve the model performance, which could not be implemented within the time frame allotted to this project. The first is to make the beads longer. The deleterious effect of including the beginning and end portions of the bead on the model performance was noted, and attempts were made to address it by trimming them off. However, because only 100 mm of bead were available, a balance had to be struck between trimming off sections and preserving enough data to prevent the data sets from becoming too sparse. This dilemma would be eliminated by increasing the bead length enough to allow large sections from the beginning and end to be trimmed off while still leaving enough data to train a machine learning model. This would ensure that the region of interest in the bead corresponds to a truly steady state process which is easier to work with than the transient behavior found near the start and stop positions.

Another improvement on this work would be to find a better way to precisely locate the start and end positions of the torch on the weld beads. While locating the end crater on the bead is a reasonable way to locate the end position of the torch, it is hardly precise and can be off by as much as 1 mm. That represents 1% of the total travel distance which is enough to degrade the model performance. A better method for locating the start and end points of the travel path would be to perform a cold run of the welding program, and make some kind of physical mark on the plate itself that corresponds to the start and end positions of the torch. Such a physical mark can be detected by metrology equipment such as the Keyence microscope, making it relatively easy to compute its coordinates. Related to this problem is the lack of synchronization between the IR cameras and the rest of the data. This would be solved by integrating the IR cameras into the data acquisition control code for the audio, robot and arc measurements. This can either be done with LabVIEW if possible, or if not, by developing a custom interface using the IR cameras' SDKs.

Finally, improving the sampling rate for the IR cameras, the welding data obtained from the robot, and the shape profiles would be a significant boost to the model performance. The low sampling rate from those sensors severely limited the amount of windowed segments that could be extracted from one set of profiles and time series. Only 10 segments per bead could be obtained, which leaves much to be desired. Significantly increasing the number of segments would both increase the number of samples available to the machine learning models, it would also decrease the time scale for each of the data features, enabling events that occur on a very small time scale to be considered by the models.

6.2.2 Thick Wall Model

The model development approach developed in this work can easily be extended to multi-layer builds. In a multi-layer build, time series data is obtained for each layer, as shown in Figure 6.1. Much of the same data preparation and feature extraction procedure described in Chapter 4 is applicable here. Each sensor time series can be windowed like the bead time series were, and features can be computed for each window. Since each layer time series is divided to into N windowed segments, if there are M layers, then the build will have NM segments, corresponding to NM time steps.

A multi-layer build can be characterized in terms of shape and defects. Ongoing work at the University of Tennessee's WAAM Development Cell, in partnership with ZEISS is investigating characterization of porosity and lack of fusion defects obtained by CT Scan. The CT scans produce a 3D image of voids and pores inside the bulk build volume which can be converted into boundary surfaces of the void shapes. Coordinates of these void boundaries can be obtained, enabling spatial analysis of the void defects that develop during deposition. A 3D map of the void locations in cartesian rectangular coordinates can then obtained. The time series windows should be matched with overlapping spatial windows in this 3D map. That is to say, time series data obtained at time range $t_i - t_{i+1}$ should be matched with the material that was deposited from $t_i - t_{i+1}$.

Matching these overlapping windows requires knowledge of the exact location in space where material deposited at time t resides. Obtaining this information is the most difficult challenge involved in model development for larger builds because of the variability of the bead shape. The robot position data provides most of the answer, since it reliably gives the x and y positions of the torch during deposition. Determining the z coordinates of the bead only requires that the CTWD be subtracted from the z coordinates of the robot position data. The CTWD can be obtained from the voltage and current wave forms relatively easily [4]. This yields a line at the center of the bead. A voxel can then be created around this line by defining ranges of the x, y, and z coordinates that can fully encompass any size bead. These overlapping voxels can be directly correlated to time series windows that occurred at the same time when material in that voxel was deposited. All that remains is to assign an output metric to these voxels such as porosity density. This continuation of the project described in this work is currently underway.

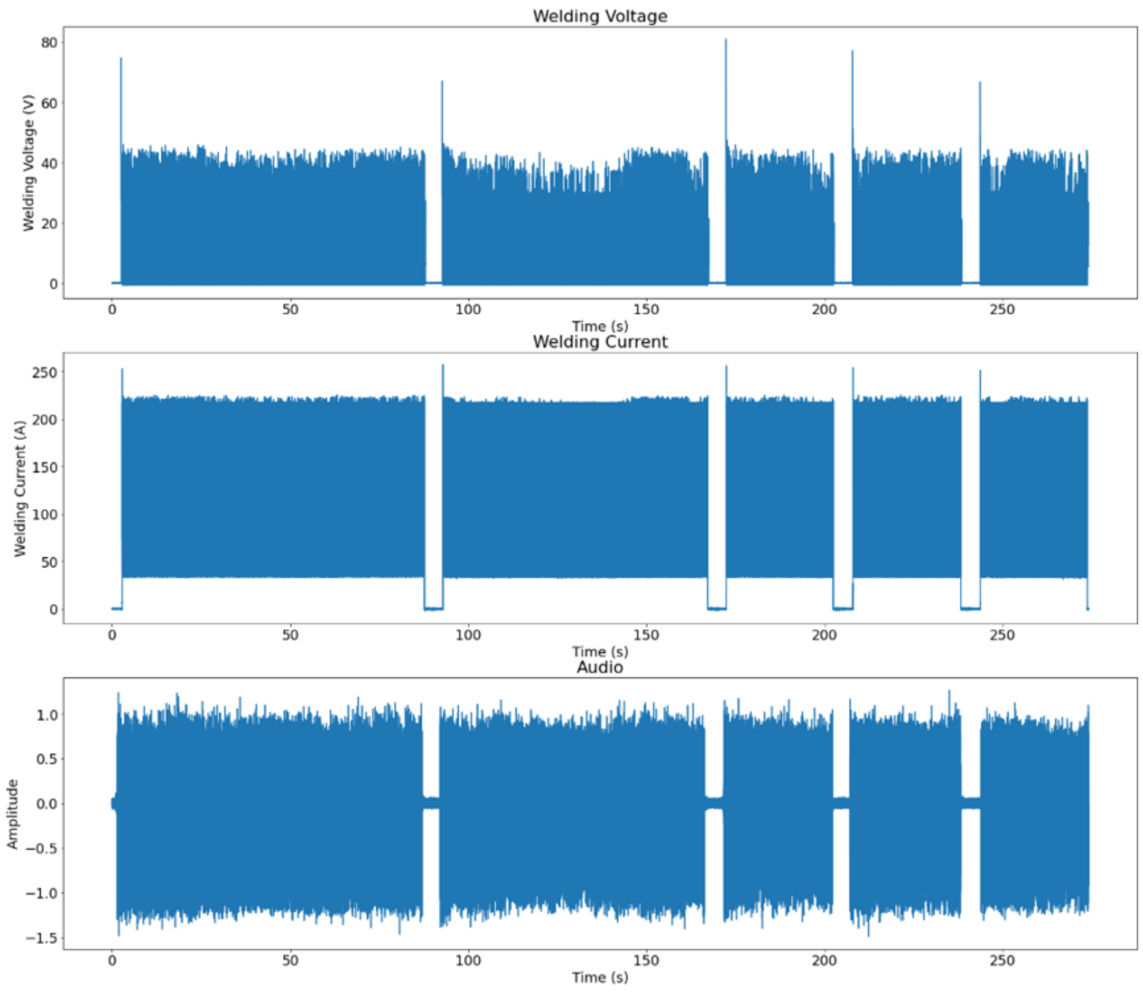


Figure 6.1: Layer time series data

Bibliography

[1] "Global additive manufacturing market: Analysis by printer type, by material, by application, by component, by region size and trends with impact of COVID-19 and forecast up to 2027," tech. rep., Daedal Research, January 2023. 1

[2] A. Khorasani, I. Gibson, J. K. Veetil, and A. H. Ghasemi, "A review of technological improvements in laser-based powder bed fusion of metal printers," *The International Journal of Advanced Manufacturing Technology*, vol. 108, pp. 191–209, 2020. 2

[3] A. Nycz, M. W. Noakes, B. Richardson, A. Messing, B. Post, J. Paul, J. Flamm, and L. Love, "Challenges in making complex metal large-scale parts for additive manufacturing: A case study based on the additive manufacturing excavator," in *Solid Freeform Fabrication 2017: Proceedings of the 28th Annual International Solid Freeform Fabrication Symposium*, 2017. 2, 5

[4] J. Penney, *Modelling, Sensing, and Control of Weld Beads in Gravity Aligned and Non-Gravity Aligned Orientations for Wire Arc Additive Manufacturing*. PhD thesis, The University of Tennessee, 2022. 7, 15, 103

[5] R. Baker, "Method of making decorative articles," U.S. Patent 1533300A, 1920. 8

[6] K. Kussmaul, F. W. Schoch, and H. Luckow, "High quality large components 'shape welded' by a SAW process," *Welding Journal*, vol. 62, no. 9, pp. 17–24, 1983. 8, 9

[7] J. Schmidt, H. Dorner, and E. Tenckhoff, "Manufacture of complex parts by shape welding," *Journal of Nuclear Materials*, vol. 171, pp. 120–127, 1980. 9

[8] F. Ribeiro, J. Norris, and R. S. McMaster, "Practical case of rapid prototyping using gas metal arc welding," 1994. 9

[9] P. Kah, R. Suoranta, and J. Martikainen, "Advanced gas metal arc welding processes," *The International Journal of Advanced Manufacturing Technology*, vol. 67, p. 655–674, 2013. 10

[10] S. Selvi, A. Vishvaksenan, and E. Rajasekar, "Cold metal transfer (CMT) technology - an overview," *Defence Technology*, vol. 14, pp. 28–44, 2018. xi, 10, 13

[11] J.F.Lancaster, "The physics of welding," *Physics in Technology*, vol. 15, no. 2, 1984. 11

[12] W. Ou, G. Knapp, T. Mukherjee, Y. Wei, and T. DebRoy, "An improved heat transfer and fluid flow model of wire-arc additive manufacturing," *International Journal of Heat and Mass Transfer*, vol. 167, 2021. 11, 14

[13] J. Norrish, J. Polden, and I. Richardson, "A review of wire arc additive manufacturing: development, principles, process physics, implementation and current status," *Journal of Physics D: Applied Physics*, vol. 54, no. 47, 2021. 11

[14] D. A. Bristow, "Layer-to-layer dynamics in ded and their relation to stability and geometric accuracy in 5-axis parts," in *American Society for Precision Engineering 2022 Summer Topical Meeting: Advancing Precision in Additive Manufacturing*, 2022. 11

[15] D. Jafari, T. H. Vaneker, and I. Gibson, "Wire and arc additive manufacturing: Opportunities and challenges to control the quality and accuracy of manufactured parts," *Materials Design*, vol. 202, 2021. 11

[16] F. Xu, V. Dhokia, P. Colegrove, A. McAndrew, S. Williams, A. Henstridge, and S. T. Newman, “Realisation of a multi-sensor framework for process monitoring of the wire arc additive manufacturing in producing ti-6al-4v parts,” *International Journal of Computer Integrated Manufacturing*, vol. 31, no. 8, pp. 785–798, 2018. xi, 11, 12, 13

[17] T. Quinn, C. N. M. C. Smith, E. Blachowiak, and R. Madrigan, “Arc sensing for defects in constant-voltage gas metal arc welding,” *Weld Journal*, vol. 78, pp. 322S – 328S, 1999. 12

[18] D. T. Thekkuden, A. Santhakumari, A. Sumesh, A.-H. I. Mourad, and K. Rameshkumar, “Instant detection of porosity in gas metal arc welding by using probability density distribution and control chart,” *The International Journal of Advanced Manufacturing Technology*, vol. 95, p. 4583–4606, 2018. 12

[19] S. S. Choudhury, S. K. Marya, and M. Amirthalingam, “Improving arc stability during wire arc additive manufacturing of thin-walled titanium components,” *Journal of Manufacturing Processes*, vol. 66, pp. 53–69, 2021. 12, 47, 52

[20] L. Grad, J. Grum, I. Polajnar, and J. M. Slabe, “Feasibility study of acoustic signals for on-line monitoring in short circuit gas metal arc welding,” *International Journal of Machine Tools and Manufacture*, vol. 44, no. 5, pp. 555–561, 2004. 12, 68

[21] K. Pal, S. Bhattacharya, and S. K. Pal, “Investigation on arc sound and metal transfer modes for on-line monitoring in pulsed gas metal arc welding,” *International Journal of Machine Tools and Manufacture*, vol. 210, no. 10, pp. 1397–1410, 2010. 12, 68

[22] C. Xia, Z. Pan, J. Polden, H. Li, Y. Xu, S. Chen, and Y. Zhang, "A review on wire arc additive manufacturing: Monitoring, control and a framework of automated system," vol. 57, pp. 31–45, 2020. 12

[23] H. Krauss, C. Eschey, and M. Zaeh, "Thermography for monitoring the selective laser melting process," in *2012 International Solid Freeform Fabrication Symposium*, 2012. 12

[24] J. Mireles, S. Ridwan, P. A. Morton, A. Hinojos1, and R. B. Wicker, "Analysis and correction of defects within parts fabricated using powder bed fusion technology," vol. 3, no. 3, 2015. 12

[25] D. Baier, A. Bachmann, and M. F. Zaeh, "Towards wire and arc additive manufacturing of high-quality parts," *Procedia CIRP*, vol. 95, pp. 54–59, 2020. 12

[26] S. Cadiou, M. Courtois, M. Carin, W. Berckmans, and P. L. Masson, "3d heat transfer, fluid flow and electromagnetic model for cold metal transfer wire arc additive manufacturing (cmt-waam)," *Additive Manufacturing*, vol. 36, 2020. 14

[27] C. Wang, H. Bai, C. Ren, X. Fang, and B. Lu, "A comprehensive prediction model of bead geometry in wire and arc additive manufacturing," *Journal of Physics: Conference Series*, vol. 1624, no. 2, 2020. 14

[28] Z. Wang, S. Zimmer-Chevret, F. Léonard, and G. Abba, "Prediction of bead geometry with consideration of interlayer temperature effect for cmt-based wire-arc additive manufacturing," *Welding in the World*, vol. 65, p. 2255–2266, 2020. 14

[29] B. Panda, K. Shankhwar, A. Garg, and M. M. Savalani, "Evaluation of genetic programming-based models for simulating bead dimensions in wire and arc additive manufacturing," *Journal of Intelligent Manufacturing*, vol. 30, p. 809–820, 2019. 14

[30] S. Tang, G. Wang, H. Song, R. Li, and H. Zhang, "A novel method of bead modeling and control for wire and arc additive manufacturing," *Rapid Prototyping Journal*, vol. 27, no. 2, pp. 311–320, 2021. 14

[31] Y. Li, H. Mu, J. Polden, H. Li, L. Wang, C. Xia, and Z. Pan, "Towards intelligent monitoring system in wire arc additive manufacturing: a surface anomaly detector on a small dataset," *The International Journal of Advanced Manufacturing Technology*, vol. 120, p. 5225–5242, 2022. 15

[32] X. He, T. Wang, K. Wu, and H. Liu, "Automatic defects detection and classification of low carbon steel waam products using improved remanence/magneto-optical imaging and cost-sensitive convolutional neural network," *Measurement*, vol. 173, 2021. 15

[33] N. D. Jamnikar, S. Liu, C. Brice, and X. Zhang, "In situ microstructure property prediction by modeling molten pool-quality relations for wire-feed laser additive manufacturing," *Journal of Manufacturing Processes*, vol. 79, pp. 803–814, 2022. 15

[34] Z. Smoqi, A. Gaikwad, B. Bevans, M. H. Kobir, J. Craig, A. Abul-Haj, A. Peralta, and P. Rao, "Monitoring and prediction of porosity in laser powder bed fusion using physics-informed meltpool signatures and machine learning," *Journal of Materials Processing Technology*, vol. 304, 2022. 15

[35] L. Breiman, "Statistical modeling the two cultures," *Statistical Science*, vol. 16, no. 3, pp. 199–231, 2001. 45

[36] T. G. Dietterich, "Machine learning for sequential data: A review," in *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops SSPR 2002 and SPR 2002 Windsor, Ontario, Canada, August 6–9, 2002 Proceedings*, 2002. 48

[37] C. Kingsford and S. L. Salzberg, "What are decision trees?," *Statistical Science*, vol. 26, no. 9, pp. 1011–1013, 2008. 49

[38] H. Geng, J. Li, J. Xiong, X. Lin, D. Huang, and F. Zhang, "Formation and improvement of surface waviness for additive manufacturing 5a06 aluminium alloy component with gtaw system," *Rapid Prototyping Journal*, 2018. 52

[39] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 6, pp. 679–698, 1986. 52

[40] P. Dimitrov, J. Damon, and K. Siddiqi, "Flux invariants for shape," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings*, 2003. 56

[41] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker, "Hamilton-jacobi skeletons," *International Journal of Computer Vision*, vol. 48, p. 215–231, 2002. 56

[42] C. S. Burrus, "Wavelets and wavelet transforms," 2015. 58

[43] G. Chen, W. Xie, and Y. Zhao, "Wavelet-based denoising: A brief review," in *2013 fourth international conference on intelligent control and information processing (ICICIP)*, IEEE, 2013. 59

[44] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions On Speech and Audio Processing*, vol. 13, no. 5, pp. 1035–1047, 2005. 72

[45] S. Böck and G. Widmer, "Maximum filter vibrato suppression for onset detection," in *Proc. of the 16th Int. Conf. on Digital Audio Effects (DAFx). Maynooth, Ireland (Sept 2013)*, vol. 7, 2013. 72

Appendices

Appendix A

LabVIEW Code

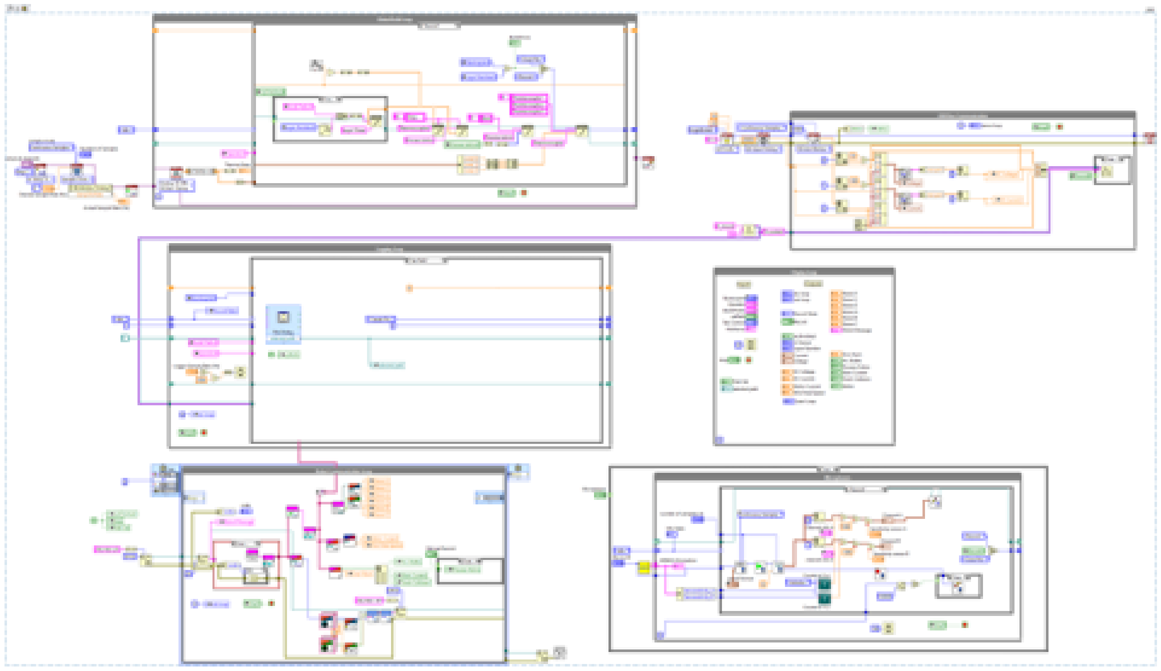


Figure A.1: LabVIEW Data Acquisition VI

Appendix B

Python Code

Main Code

```python

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import os

import plotting

import random_forests as rf
import data_processing as dp
import segmentation as seg
import neural_network as nn

from tkinter import filedialog


def get_shape(NumPts,path,attribute):
    hfilename = "Height Profile Bead "
    wfilename = " Width Profile.csv"
    BeadShapes = []
    for i in range(1, NumPts + 1):
        tempBead = dp.BeadShape(i)
        if attribute == 'Height':
            dataHeight = pd.read_csv(path + '\\Height Profiles\\' + hfilename + str(i) + '.csv')
            X = dataHeight["X(mm)"].tolist()
            Z = dataHeight["Z(mm)"].tolist()
            tempBead.add_height(X, Z)
            tempBead.profile_trim(path, i)
            del dataHeight, X, Z
        elif attribute == 'Width':
            dataWidth = pd.read_csv(path + '\\Width and Centerline\\Bead' + str(i) + wfilename)
            L = dataWidth['Length'].tolist()
            Width = dataWidth['Width'].tolist()
            CenterLineDeviation = dataWidth['CenterLineDeviation']
            tempBead.add_width(L, Width,CenterLineDeviation)
            del dataWidth, L, Width, CenterLineDeviation

        BeadShapes.append(tempBead)

    return BeadShapes

#Main Program

#Enter main problem parameters
path = filedialog.askdirectory()
outputMainPath = os.path.join(path, 'Output')


attribute = 'Height'
metric = 'Std'
heightMetrics = ['Mean','Std','Peak to Valley']
widthMetrics = ['Mean','CenterLine']


NumPts = 43

num_windows = 10
percent_overlap = 0.15


##############################################
#Get Bead Shape Profiles
###########################################

BeadShapes = get_shape(NumPts, path, attribute)

##################################################################
# Extract from TDMS
##################################################################

#Extract settings from csv file
Settings = pd.read_csv(path + '\\Settings.csv')
#Extract time series data from tdms file
Beads = dp.extract_labview(path,NumPts)
#Extract IR data from csv files
Beads = dp.extract_IR_data(path,Beads,NumPts)
#Extract feature units
units = pd.read_csv(path + '\\feature units.csv')

#################################
#Denoise Waveforms
######################################

for bead in Beads:
    #bead.denoise('lemData','Welding Voltage')
    #bead.denoise('lemData', 'Welding Current')
    bead.denoise('audio', 'Audio')

#Plot bead profiles prior to pre-processing
for beadshape in BeadShapes:
    plotting.plot_profile(beadshape,'Height Profile',outputMainPath)


############################################################
```

```
#Pre-processing
############################################################

for bead in Beads:
    i = bead.number
    print(i)

    bead.add_settings(Settings.iloc[i-1,:])
    bead.remove_prepost_time()

    if attribute == 'Height':
        xtrim, xendtrim = BeadShapes[i - 1].trim_slopes('Height', 'x')
        bead.trim_profile_time(xtrim, xendtrim)


#########################################
#Plotting
#########################################

#Plot bead profiles after trimming
for beadshape in BeadShapes:
    plotting.plot_profile(beadshape,'Trimmed Height Profile',outputMainPath)

for bead in Beads:
    plotting.plot_spectrograms(bead,outputMainPath, percent_overlap, num_windows, sr = 22050, group = 'audio',
     waveform = 'Audio')
    plotting.plot_spectrograms(bead, outputMainPath, percent_overlap, num_windows, sr=20000, group='lemData',
     waveform='Welding Voltage')
    plotting.plot_spectrograms(bead, outputMainPath, percent_overlap, num_windows, sr=20000, group='lemData',
     waveform='Welding Current')


for bead in Beads:
    plotting.plot_timeseries(bead,outputMainPath,'weldData','lemData','audio','meltTemps')

# ############################################
# # Create data frame and output dictionary
# ##########################################

outputName = attribute + ' ' + metric

try:
    outputPath = os.path.join(outputMainPath, outputName)
    os.mkdir(outputPath)
except:
    pass

X,Y, Segments = seg.segment_assemble(Beads, BeadShapes, num_windows, percent_overlap, attribute = attribute,
     metric = metric)
X = X.drop('Wire Feed Speed Kurt',axis = 1)

#Plot feature correlations with respect to output
for (columnName, columnData) in X.iteritems():
    plotting.plot_segmentXY(X, Y, columnName, outputName, outputPath,units)

#Obtain basic stats about output metric
Y = np.array(Y)
Y_stats = pd.DataFrame(columns = ['Mean','Range','Max','Min'])
Y_stats['Mean'] = [np.mean(Y)]
```

```
Y_stats['Range'] = [max(Y) - min(Y)]
Y_stats['Max'] = [max(Y)]
Y_stats['Min'] = [min(Y)]
Y_stats.to_csv(outputPath + '\\output_stats.csv')

#Plot Feature Distributions
for (columnName, columnData) in X.iteritems():

    try:
        dist_path = os.path.join(outputMainPath, 'Feature Distributions')
        os.mkdir(dist_path)
    except:
        pass
    plotting.plot_histogram(columnData,columnName,dist_path)

print('Output %s ranges from %.6f to %.6f. The range is %.6f' % (outputName,min(Y), max(Y), (max(Y)-min(Y))))


# # #####################################
# # #Eliminate Highly Correlated Features
# # ######################################

#Compute correlation matrix
corr = X.corr()
plt.rcParams.update({'font.size': 27})
fig_corr = plt.figure(figsize = [35,35])
ax = fig_corr.add_subplot()
sns.heatmap(corr , ax = ax)
fig_corr.figure.savefig(outputMainPath+'\\Correlation_Matrix.png')

columns = np.full((corr.shape[0],), True, dtype=bool)
cols = X.columns.tolist()


# Remove one of every pair of columns that are 95% correlated
print("Dropping data points that are 95% correlated to existing data:")
for i in range(corr.shape[0]):
    if i > 0 :
        for j in range(i+1, corr.shape[0]):
            if corr.iloc[i,j] >= 0.95:
                if columns[i] and columns[j]:
                    print(str(cols[j]) + " " + str(j) + ": (95%+ correlated to " + str(cols[i]) + " " + str(i) +
     ")")
                    X = X.drop(str(cols[j]), axis = 1)
                    columns[j] = False
                elif columns[j]:
                    columns[j] = True

# ###########################
# ##Machine Learning Model ###
# ###########################
n,bins = plotting.plot_histogram(Y,outputName, outputPath)

del Beads

#Run model

modelType = 'LSTM'
nn_epochs = 1600
```

```
classify_epochs = 50
lstm_epochs = 10


if modelType == 'LSTM':
    segsPerBead = 10
    numBeads = NumPts
else:
    segsPerBead = None
    numBeads = None
print('Model Initializing')


try:
    ml_outputPath = os.path.join(outputPath, modelType)
    os.mkdir(ml_outputPath)
except:
    pass

if modelType == 'RF':
    Y_pred = rf.regression_RFE(X, Y, ml_outputPath, outputName)
else:
    Y_pred = nn.neuralNetworkMain(X, Y, ml_outputPath, modelType=modelType, epochs=nn_epochs, lr=1e-4,
                                   foldSplits = 25, numBeads = numBeads, segsPerBead = segsPerBead)

BeadShapes = seg.prediction_assignment(Segments,BeadShapes, Y_pred)

for beadshape in BeadShapes:
    plotting.plot_output(beadshape,ml_outputPath,attribute,metric)
```

Data Preparation Functions

```

from nptdms import TdmsFile
import numpy as np
import pandas as pd
import math
from math import floor
import segmentation as seg
import librosa

from skimage.restoration import denoise_wavelet

import audio_features as af


class BeadShape:

    def __init__(self,beadNum):
        self.beadNum = beadNum
        self.x = None
        self.Height = None
        self.L = None
        self.Width = None
        self.CenterLine = None
        self.peaktoval = None
        self.Std = None
        self.numWindows = None
        self.overlap = None
        self.xtrim = None
        self.predictions = []

    def add_height(self,X,Height):
        self.x= np.array(X)
        self.Height = np.array(Height)

    def add_width(self,L,Width,CenterLine):
        self.L = np.array(L)
        self.Width = np.array(Width)
        self.CenterLine = CenterLine

    def profile_trim(self,path,beadNum,attributeX = 'x',attributeProfile = 'Height'):

        x = getattr(self,attributeX)
        z = getattr(self,attributeProfile)

        startEndPts = pd.read_csv(path + '\\Start end points.csv')
        start = startEndPts['Start'][beadNum - 1]
        end = startEndPts['End'][beadNum - 1]

        filter_idx = np.where(np.logical_and(x >= start, x <= end))

        z = z[filter_idx]
        x = x[filter_idx]

        x = x - min(x)

        setattr(self,attributeX,x)
        setattr(self,attributeProfile,z)

    def segment(self,overlap,num_windows,attribute = 'Height',metric = 'Mean'):

        outputName = attribute + metric

        if metric == 'CenterLine':
            temp = getattr(self, metric)
            attribute = metric
        else:
            temp = getattr(self, attribute)

        temp = temp[np.isfinite(temp)]
        L = temp.size
        lw = seg.calculate_winlength(L, overlap, num_windows)
        windowedProfile = seg.segment_axis(temp, lw, percent_overlap=overlap, end="cut")

        setattr(self,attribute+ ' Windows',windowedProfile)

        if metric == 'Mean':
            y = np.mean(windowedProfile, axis=1)
            setattr(self,outputName,y)

        elif metric == 'Std':
            y = self.local_stdev(attribute+ ' Windows')
            setattr(self, outputName,y)

        elif metric == 'Peak to Valley':

            y = self.peak_to_valley(attribute+ ' Windows')
            setattr(self, outputName,y)

        elif metric == 'CenterLine':

            y = self.local_stdev(metric+ ' Windows')
            setattr(self, outputName,y)

        delattr(self,attribute)

        return y

    def local_stdev(self,attribute):

        measureWindows = getattr(self,attribute)
        avg = measureWindows.mean()

        stdev = []

        for x in measureWindows:
            sum = 0
            for xi in x:
                sum = sum + (xi - avg) ** 2

            stdev.append(math.sqrt(sum / x.size))

        stdev = np.array(stdev)
        return stdev

    def peak_to_valley(self,attribute):

        windowedProfile = getattr(self, attribute)
        windowedProfile = windowedProfile
        peak2valley = []

        for window in windowedProfile:
            peak = max(window)
            valley = min(window)
            peak2valley.append(peak - valley)

        self.peaktoval = peak2valley

        return peak2valley

    def trim_slopes(self,attribute,attributeX):
        X = getattr(self,attributeX)
        profile = getattr(self,attribute)

        peak = max(profile[0:floor((profile.size)/2)])


        peakIdx = np.where(profile == peak)[0][0]

        xtrim = X[peakIdx]

        newXtrim = xtrim + 7
        diff = abs(X - newXtrim)
        startIdx = np.where(diff == min(diff))[0][0]
        xtrim = X[startIdx]

        for i in range(-1, -profile.size, -1):
            if (profile[i] > profile[i - 1]) & (profile[i] > 0.82*np.mean(profile)):
                endIdx = i
                break
            elif profile[i] < 0.82 * np.mean(profile):
                continue

        xendtrim = X[endIdx]

        profile = profile[startIdx:endIdx]
        X = X[startIdx:endIdx]

        setattr(self, attribute, profile)
        setattr(self, attributeX, X)

        return xtrim, xendtrim


class Bead:

    allBaseTemp = []
    alltravelSpeed = []
    allwfs = []
    allArc = []
    allctwd = []

    def __init__(self, number, lemData,weldData,robData, audio):
        self.number = number
        self.lemData = lemData
        self.weldData = weldData
        self.robData = robData
        self.audio = audio

        self.meltTemps = None
        self.lineProfile = None
        self.baseTemp = None

        self.travelSpeed = None
        self.wfs = None
        self.arcCorrection = None
        self.ctwd = None
        self.arcCorrection = None


        self.predictions = []
        self.segments = []

        self.audioFrames = {}
        self.allBeadStats = None
        self.lemDataStats = {}
        self.weldDataStats = {}
        self.meltTempStats = {}

    def add_settings(self,settings):
        self.travelSpeed = settings['Travel']
        self.wfs = settings['WFS']

        keysList = list(settings.keys())

        if keysList[2] == 'Arc Correction':
            self.arcCorrection = settings['Arc Correction']
            Bead.allArcCorrection.append(self.arcCorrection)
        elif keysList[2] == 'Arc Correction':
            self.ctwd = settings['CTWD']
            Bead.allctwd.append(self.ctwd)

    def add_infrared(self,meltPoolVals,lineProfile):
        meltTemps = {}
        self.lineProfile = lineProfile
        self.baseTemp = np.mean(lineProfile)
        Bead.allBaseTemp.append(self.baseTemp)

        numSamples = meltPoolVals.size
        time = self.robData['Time']
        startTime = time[0]
        endTime = time[-1]
        del time
        meltTime = np.linspace(startTime,endTime,numSamples)
        meltTemps['Melt Pool Temperature'] = np.array(meltPoolVals)
        meltTemps['Time'] = np.array(meltTime)

        self.meltTemps = meltTemps

    def add_line_profile(self,lineProfile):
        self.lineProfile = lineProfile
        self.baseTemp = np.mean(lineProfile)
        Bead.allBaseTemp.append(self.baseTemp)

    def segment(self,overlap,numWindows):
        lemData = self.lemData
        weldData = self.weldData
        robData = self.robData

        lemData = seg.data_split_stats(lemData,overlap,numWindows)
        weldData = seg.data_split_stats(weldData, overlap, numWindows)
        robData = seg.data_split_stats(robData, overlap, numWindows)

        self.lemData = lemData
        self.weldData = weldData
        self.robData = robData

    def remove_prepost_time(self):

        robotY = self.robData['Robot Y']
        robotX = self.robData['Robot X']
        robotZ = self.robData['Robot Z']
        time = self.robData['Time']

        travelDirection = robotY
        print(len(travelDirection))
        for i in range(0, len(travelDirection)):
            if travelDirection[i] != travelDirection[i + 1]:
                startIdx = i
                break
        for i in range(-1, -len(travelDirection), -1):
            if travelDirection[i] != travelDirection[i - 1]:
                endIdx = i
                break

        travelDirection = travelDirection[startIdx:endIdx]
        travelDirection = travelDirection - min(travelDirection)

        robotY = travelDirection

        self.robData['Robot Y'] = robotY
        self.robData['Robot X'] = robotX[startIdx:endIdx]
        self.robData['Robot Z'] = robotZ[startIdx:endIdx]

        del robotX,robotY,robotZ

        startTime = time[startIdx]
        endTime = time[endIdx]
        self.robData['Time'] = time[startIdx:endIdx]

        del time

        timeAudio = af.add_time(self.audio['Audio'], endTime + 0.5)
        self.audio['Time'] = np.array(timeAudio)
        del timeAudio

        try:
            N_melt = self.meltTemps['Melt Pool Temperature'].size
            meltTime = np.linspace(startTime, endTime + 0.5, N_melt)
            self.meltTemps['Time'] = np.array(meltTime)
            del meltTime
            self.meltTemps = trim_times(startTime, endTime, self.meltTemps)

        except:
            pass

        self.weldData = trim_times(startTime, endTime, self.weldData)
        self.lemData = trim_times(startTime, endTime, self.lemData)
        self.audio = trim_times(startTime, endTime, self.audio)

    def trim_profile_time(self,xtrim,xendtrim):

        robotY = self.robData['Robot Y']
        time = self.robData['Time']
        diffStart = abs(robotY - xtrim)

        print(xtrim,xendtrim)
        print(robotY[-1])
        idxStart= np.where(diffStart == min(diffStart))[0][0]

        diffEnd = abs(robotY - xendtrim)
        idxEnd = np.where(diffEnd == min(diffEnd))[0][0]

        startTime = time[idxStart]
        endTime = time[idxEnd]

        try:
            self.meltTemps = trim_times(startTime, endTime, self.meltTemps)
        except:
            pass


        self.weldData = trim_times(startTime, endTime, self.weldData)
```

```python
        self.lemData = trim_times(startTime, endTime, self.lemData)
        self.audio = trim_times(startTime, endTime, self.audio)


    def filter_blips(self):

        xtemp = self.robData['Robot X']
        filter_idx = np.where(xtemp > 500)

        if filter_idx.size == 0:
            return 0
        else:
            self.robData['Robot X'] = np.delete(xtemp, filter_idx)
            del xtemp
            self.robData['Robot Y'] = np.delete(self.robData['Robot Y'], filter_idx)
            self.robData['Robot Z'] = np.delete(self.robData['Robot Z'], filter_idx)
            self.robData['Time'] = np.delete(self.robData['Time'], filter_idx)

            self.weldData['Wire Feed Speed'] = np.delete(self.weldData['Wire Feed Speed'], filter_idx)
            self.weldData['Time'] = np.delete(self.weldData['Time'], filter_idx)

    def delete_item(self,attribute,*itemKeys):

        attributeDict = getattr(self,attribute)
        for arg in itemKeys:
            attributeDict.pop(arg)

        setattr(self,attribute,attributeDict)

    def merge_data(self,*attributes):

        for arg in attributes:
            try:
                dataStats = getattr(self, arg)
                self.allBeadStats = self.allBeadStats | dataStats
            except:
                allBeadStats = {}
                dataStats = getattr(self, arg)
                allBeadStats = allBeadStats | dataStats
                self.allBeadStats = allBeadStats

    def segment(self,percent_overlap,num_windows):
        # Window audio data
        audio_size = self.audio['Audio'].size
        lw_audio = seg.calculate_winlength(audio_size, percent_overlap, num_windows)
        hop = floor(lw_audio - lw_audio * percent_overlap)
        # Store windowed audio features back in object
        self.audioFrames = af.extract_basic_features(self.audio['Audio'], 22050, hop, lw_audio)


        # Delete unwanted items from bead object
        self.delete_item('weldData', 'Time')
        self.delete_item('lemData', 'Time')

        try:
            meltTempStats = {}
            self.delete_item('meltTemps', 'Time')
            meltTempWindows = \
            seg.segment_axis(self.meltTemps['Melt Pool Temperature'], num_windows, percent_overlap=
    percent_overlap,
                             end="cut")[0]
            meltTempStats['Melt Pool Temperature'] = meltTempWindows
            self.meltTempStats = meltTempStats
            self.meltTempStats
        except:
            pass


        # Window the rest of data and store back into object
        self.lemDataStats = seg.arc_features(self.lemData, percent_overlap, num_windows)
        self.weldDataStats = seg.data_split_stats(self.weldData, percent_overlap, num_windows)


        try:
            self.merge_data('lemDataStats','weldDataStats','meltTempStats')
        except:
            self.merge_data('lemDataStats', 'weldDataStats')

        del self.audio, self.lemData, self.weldData

    def denoise(self, group, waveform):

        wavedict = getattr(self,group)
        X_denoise = denoise_wavelet(wavedict[waveform], method='VisuShrink', mode='soft', wavelet_levels=3,
                                    wavelet='sym8', rescale_sigma='True')
        wavedict[waveform] = X_denoise
        setattr(self,group,wavedict)


def beadNumber(i):
    if i < 10:
        beadnumstr = "0" + str(i)
    else:
        beadnumstr = str(i)

    return beadnumstr


def get_LEMtime(data):
    time = []
    sample_rate = 20000
    n_samples = len(data['Welding Voltage'])
    dt = 1/sample_rate

    for i in range(1,n_samples+1):
        time.append(dt*i)

    return time


def extract_IR_data(path,beads,num):
    try:
        pathBlob = path + '\\IR Data\\Blob Detection\\'
        pathLine = path + '\\IR Data\\Line Profiles\\'
        for i in range(1, num + 1):
            beadnumstr = beadNumber(i)

            filenameLine = 'Bead' + beadnumstr + '_basetemp'
            filenameBlob = 'Bead' + beadnumstr + '_BlobDetection'

            temp_dfLine = pd.read_csv(pathLine + filenameLine + '.csv')
            temp_dfBlob = pd.read_csv(pathBlob + filenameBlob + '.csv')

            blobTemps = temp_dfBlob['TemperatureMean'].values
            lineTemps = temp_dfLine['Bead' + beadnumstr + '.seq:Line 1 [C]:mean:vert'].values

            beads[i - 1].add_infrared(blobTemps, lineTemps)
    except:
        pathLine = path + '\\IR Data\\Line Profiles\\'
        for i in range(1, num + 1):
            beadnumstr = beadNumber(i)
            filenameLine = 'Bead' + beadnumstr + '_basetemp'
            temp_dfLine = pd.read_csv(pathLine + filenameLine + '.csv')
            lineTemps = temp_dfLine['Bead' + beadnumstr + '.seq:Line 1 [C]:mean:vert'].values
            beads[i - 1].add_line_profile(lineTemps)

    return beads


def normalize_data(X):

    for (columnName, columnData) in X.iteritems():
        featureVector = columnData.values
        featureMax = featureVector.max()
        featureMin = featureVector.min()
        normFeature = (featureVector - featureMin)/(featureMax - featureMin)
        X[columnName] = normFeature

    return X


def trim_times(startTime,endTime,data):

    time = data['Time']

    idx = np.where(np.logical_and(time >= startTime, time <= endTime))[0]

    for key in data:
        series = data[key]
        data[key] = series[idx]
    return data


def extract_labview(path,NumPts):

    beads = []

    for i in range(1, NumPts+1):

        TempRobData = {}
        TempWeldData = {}
        TempLEMData = {}
        Audio = {}

        beadnumstr = beadNumber(i)
        beadPath = path+'\\LabVIEW\\Bead' + beadnumstr

        temp_file = TdmsFile.read(beadPath + '\\Bead01.tdms')

        RobGroup = temp_file["Robot Data"]
        WeldGroup = temp_file["Welding Data"]
        LEMGroup = temp_file['LEM Box']

        for channel in RobGroup.channels():
            TempRobData[channel.name] = channel[:]

        for channel in LEMGroup.channels():
            TempLEMData[channel.name] = np.array(channel[:])

        for channel in WeldGroup.channels():
            TempWeldData[channel.name] = np.array(channel[:])

        LEMtime = get_LEMtime(TempLEMData)
        TempLEMData['Time'] = np.array(LEMtime)
        del TempWeldData['Motor Current']
        #Get Audio
        tempAudio, SR = none = librosa.load(beadPath + '\\Bead01.wav')
        tempAudio = af.clipEndAudio(tempAudio)

        Audio['Audio'] = tempAudio
        beads.append(Bead(i, TempLEMData, TempWeldData, TempRobData, Audio))
        del tempAudio

    return beads
```

Machine Learning Functions

```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
import os

from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Dropout, LSTM
from tensorflow.python.keras.optimizer_v2.adam import Adam
from tensorflow.python.keras.constraints import MaxNorm

from sklearn.model_selection import KFold, TimeSeriesSplit, LeaveOneOut, StratifiedKFold
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import LeaveOneOut, KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
from sklearn.inspection import permutation_importance

import shap


def neuralNetwork(x_shape):
    model = Sequential([
        Dense(units=x_shape[1], input_shape=(x_shape[1],), kernel_constraint=MaxNorm(8), kernel_initializer='
normal',
              activation='relu'),
        Dense(units=12, kernel_constraint=MaxNorm(8), kernel_initializer='normal', activation='relu'),
        Dropout(0.2),
        Dense(units=1, activation='linear')
    ])

    return model


def neuralNetworkClassify(x_shape):
    model = Sequential([
        Dense(units=x_shape[1], input_shape=(x_shape[1],), kernel_constraint=MaxNorm(5), activation='relu'),
        Dense(units=10, kernel_constraint=MaxNorm(5), activation='relu'),
        Dropout(0.8),
        Dense(units=1, activation='sigmoid')
    ])

    return model


def recurrentNeuralNetwork(segsPerBead,x_shape):
    model = Sequential()
    model.add(LSTM(20, input_shape=[segsPerBead, x_shape[1]], stateful=False, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(Dense(units=1, activation='linear', kernel_initializer='normal'))

    return model


def plot_NN(Y_true,Y_pred,outputPath):
    plt.rcParams.update({'font.size': 16})
    compPlot = plt.figure()
    line = np.linspace(min(Y_true), max(Y_true), 100)
    plt.plot(line, line)
    plt.scatter(Y_true, Y_pred)
    plt.xlabel('Measured (mm)')
    plt.ylabel('Predicted (mm)')
    compPlot.savefig(outputPath + '\\ComparisonPlot.png')
    plt.show()


def compute_shap(model,X_train,X_test):

    randsamples = X_train[np.random.choice(X_train.shape[0], 150, replace=False)]
    explain = shap.DeepExplainer(model, randsamples)
    shap_vals = explain.shap_values(X_test[0:15])

    return shap_vals[0]


def neuralNetworkMain(X, Y, outputPath, modelType = 'NN', epochs = 700 ,lr = 1e-4, foldSplits = 25, numBeads =
None, segsPerBead = None):

    # Get feature names and current number of features
    X_features = X.columns
    number_of_features = len(X_features)

    scaler = MinMaxScaler(feature_range=(0, 1))
    X = X.to_numpy()
    X = scaler.fit_transform(X)

    x_shape = X.shape
    cvscores = []
    Y_pred = []
    Y_true = []

    if modelType == 'NN':
        modelInit = neuralNetwork(x_shape)
        kfolds = KFold(n_splits=foldSplits)
    elif modelType == 'NN Classify':
        kfolds = StratifiedKFold(n_splits= foldSplits)
        modelInit = neuralNetworkClassify(x_shape)
    elif modelType == 'LSTM':
        kfolds = KFold(n_splits=foldSplits)
        X = X.reshape(numBeads, segsPerBead, x_shape[1])
        Y = Y.reshape(numBeads, segsPerBead)
        modelInit = recurrentNeuralNetwork(segsPerBead, x_shape)

    shap_vals_folds = np.zeros([15,x_shape[1]])

    for trainIdx, testIdx in kfolds.split(X,Y):
        print('Test Indexes:')
        print(testIdx)

        X_train = X[trainIdx]
        X_test = X[testIdx]

        model = modelInit

        model.compile(
            optimizer=Adam(learning_rate=lr),
            loss="mean_squared_error",
            metrics="mse")

        model.fit(X[trainIdx], Y[trainIdx],
                  epochs=epochs,
                  verbose=0,
                  shuffle=True)

        train_scores = model.evaluate(X[trainIdx],Y[trainIdx], verbose = 1)
        scores = model.evaluate(X[testIdx],Y[testIdx], verbose = 1)
        print(scores)
        yFoldPred = model.predict(X[testIdx])

        if modelType == 'LSTM':
            yFoldPred = yFoldPred.reshape(1,segsPerBead*len(testIdx))[0]
            yFoldTrue = Y[testIdx].reshape(1,segsPerBead*len(testIdx))[0]
        else:
            shap_vals = compute_shap(model,X_train,X_test)
            shap_vals_folds = np.add(shap_vals_folds,shap_vals)
            yFoldTrue = Y[testIdx]

        Y_pred.extend(yFoldPred.tolist())
        Y_true.extend(yFoldTrue.tolist())

        cvscores.append(scores[1])
        print('Current split iteration training score: %.6f' % (train_scores[1]))
        print('Current split iteration validation: %.6f' % (scores[1]))
        print('Model overall validation score: %.6f' % (np.mean(cvscores)))


    finalScore = np.mean(cvscores)

    if modelType == 'NN':
        #Compute average shap values across all folds
        shap_vals_avg = shap_vals_folds / foldSplits
        shap_avg = np.mean(shap_vals_avg, axis=0)
        sorted_idx = shap_avg.argsort()
        plt.figure()
        shap.summary_plot(shap_vals_avg, features=X_train, feature_names=X_features, plot_type="bar", max_display
    =30)
        plt.savefig(outputPath + '\\Shap Values.png')
        print(shap_avg)

        least_valuable_feature = X_features[sorted_idx[0]]
        print('The least valuable feature is: %s' % (least_valuable_feature))

    modelSummary = pd.DataFrame()

    modelSummary['Epochs'] = [epochs]
    modelSummary['Learning Rate'] = [lr]
    modelSummary['Cross Val Score'] = [finalScore]
    modelSummary.to_csv(outputPath + '\\ModelSummary.csv',index = 0)
    print(modelSummary)

    print('Final MSE score: %.5f' % (np.mean(cvscores)))
    print(cvscores)

    config = model.get_config()
    print(config)

    plot_NN(Y_true, Y_pred, outputPath)

    return Y_pred

def regression_RFE(X,Y,outputPath, metric):
    X_new = X

    kfolds = KFold(n_splits = 23)
    kfolds.get_n_splits(X_new)
    print('Leave-One-Out Splits Acquired')

    counter = 1

    X_new = X

    X_features = X_new.columns.to_numpy()
    number_of_features = len(X_features)

    summary_df = pd.DataFrame(columns=['Number of Features', 'MAE', 'Least Important Feature'])
    all_num_features, all_mae, all_least_import = list(), list(), list()

    try:
        new_path = os.path.join(outputPath, 'Feature Importance')
        os.mkdir(new_path)
    except:
        pass

    while number_of_features >= 7:
        print('RFE Loop Iteration: %d' % counter)

        y_pred, y_true = list(),list()

        print('Training loop begins')
        for train_index, test_index in kfolds.split(X_new):
            X_train, X_test = X_new.iloc[train_index], X_new.iloc[test_index]
            Y_train, Y_test = Y[train_index], Y[test_index]

            regressor = RandomForestRegressor(n_estimators=300, min_samples_leaf=3)

            regressor.fit(X_train, Y_train)
            y_hat = regressor.predict(X_test).tolist()
            y_pred.extend(y_hat)
            y_true.extend(Y_test)

        print('Training loop ends')
        feat_imp = permutation_importance(regressor, X_train, Y_train)
        combined_feature_importance = feat_imp.importances_mean
        sorted_idx = combined_feature_importance.argsort()

        curr_score = mean_squared_error(y_true, y_pred)

        file_suffix = metric + '_' + str(number_of_features) + "features"
        print(len(y_pred))
        line = np.linspace(min(y_true), max(y_true), 30)
        comp_plot = plt.figure()
        plt.scatter(y_true, y_pred)
        plt.plot(line, line)
        plt.xlabel('Measured (mm)')
        plt.ylabel('Predicted (mm)')
        comp_plot.savefig(new_path + '\\CompPlot_' + file_suffix + '.png')

        importances_df = pd.DataFrame(columns=['Feature', 'Importance'])
        importances_df['Feature'] = X_features
        importances_df['Importance'] = combined_feature_importance
        importances_df.to_csv(new_path + '\\Importances_ ' + file_suffix + '.csv')

        least_valuable_feature = X_features[sorted_idx[0]]
        X_new = X_new.drop(least_valuable_feature, axis=1)

        all_num_features.append(number_of_features)
        all_mae.append(curr_score)
        all_least_import.append(least_valuable_feature)

        new_score = mean_squared_error(y_true, y_pred)
        print('Iteration %d......Current score: %.5f' % (counter, curr_score))
        print("Current Number of features %d" % number_of_features)
        X_features = X_new.columns.to_numpy()
        number_of_features = len(X_features)
        print('Least valuable feature: %s .... MSE: %.5f ' % (least_valuable_feature, new_score))
        print("New number of features %d" % number_of_features)

        counter = counter + 1

    summary_df['Number of Features'] = all_num_features
    summary_df['MAE'] = all_mae
    summary_df['Least Important Feature'] = all_least_import
    summary_df.to_csv(new_path + '\\Summary_ ' + file_suffix + '.csv')

    return y_pred
```

Vita

Eduardo Miramontes was born in Mexico City in December 1995. He lived there for the first 8 years of his life until in 2004 he moved to the United States, in Madison, WI to live with his dad's side of the family. In 2010, he moved to the state of Tennessee to live with his mom. He graduate high school with the Class of 2013 and was accepted to the University of Tennessee where he first attended in the Fall of 2013. He initially enrolled as an Aerospace Engineering major before switching to Mechanical Engineering after a year and a half. Eduardo graduated with a Bachelor of Science in Mechanical Engineering in December of 2017. In November 2018 he began his stint at Oak Ridge National Labs as a Post-Bachelor Research Associate. At ORNL, Eduado worked with Dr. Costas Tsouris to develop, characterize and test an additively manufactured multi-functional packed bed/heat exchanger designed to improve the capture efficiency of carbon capture by Methanolamine (MEA). Eduardo left ORNL in the summer of 2020, at which point he applied and was accepted to the engineering graduate program. He joined Dr. Bradley Jared's research team as a Graduate Research Assistant starting in the Spring 2021 Semester.