Adaptive Correction Strategy in Robotic Gas Tungsten Arc Welding for Additive Manufacturing

Marzieh Masoodi Nia

A Thesis

in

The Department

of

Mechanical, Industrial & Aerospace Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Mechanical Engineering) at

Concordia University

Montréal, Québec, Canada

August 2024

© Marzieh Masoodi Nia, 2024

This is to certify that the thesis prepared

By: **Marzieh Masoodi Nia**

Entitled: **Adaptive Correction Strategy in Robotic Gas Tungsten Arc Welding for Additive Manufacturing**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Mechanical Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Tsz Ho Kwok

_____ Examiner
Dr. Tsz Ho Kwok

_____ Examiner
Dr. Chunyan Lai

_____ Supervisor
Dr. Wen-Fang Xie

_____ Co- Supervisor
Dr. Javad Gholipour Baradari

Approved by _____
Martin D. Pugh, Chair
Department of Mechanical and Industrial Engineering

September 2024

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

ABSTRACT

Adaptive Correction Strategy in Robotic Gas Tungsten Arc Welding for Additive Manufacturing

Marzieh Masoodi Nia

Wire arc additive manufacturing (WAAM) is one of additive manufacturing (AM) methods and owns notable advantages like enabling production of large-scale components. However, the current WAAM has inherent drawbacks, such as heat accumulation and near-net-shape production issues that can lead to defects like geometrical deviations, porosity inside the weld and/or surface irregularities. It is noted that various process parameters are directly related to the above-mentioned production issues. Therefore, it is crucial to set various process parameters (based on geometry, processing changes when depositing, etc.) appropriately for achieving a good quality of product

In this project, we aimed to investigate the influence of various process parameters on the product quality and to automate the WAAM process using a vision system. To realize the objectives, we have developed an adaptive correction strategy to control the robotic welding machine, i.e. a Nertamatic power source that centralize the welding cycle while considering various welding parameters such as the robot path, deposited layer height, surface contamination, etc. To minimize operator intervention, a Cognex 3D A5000 series camera was employed to scan/monitor the deposited object layer by layer. The camera's ASCII output was used for mesh processing. An on-line control scheme has been developed to control the robot path according to the dimensions of previous layers and thus the robot's height was adaptively adjusted. An

algorithm was proposed to process the mesh data to detect and correct the inconsistencies by commanding the robot to stop in case of collisions or fill cavities in the case of insufficient height or underfills.

An experiment has been designed on a robotic WAAM machine where a TopTig gun was attached to the end effector of a 6-degree-of-freedom (DOF) ABB robot IRB4600, equipped with a 2-DOF IRBP_A500 table, to deposit material layer by layer. A nozzle is mounted with different diameter tungsten electrodes and fed with various wire materials. In this study, we tested a 3 mm diameter tungsten electrode and stainless-steel filler wire to assess the effects of overlap between the beads on the integrity of the deposited part. Simulations have been conducted in RobotStudio software to validate the recognizing deposited layer inconsistencies and the effectiveness of the path planning and welding machine settings, demonstrating the potential for deploying the adaptive correction strategy in the WAAM process.

Acknowledgments

First and foremost, I would like to express my heartfelt gratitude to my supervisors, Professor Wen-Fang Xie, Professor Priti Wanjara, and Professor Javad Gholipour. Their guidance and expertise were instrumental in helping me navigate and overcome the challenges faced throughout this project. Completing this journey would not have been possible without their knowledge and support.

I extend my deep appreciation to Xavier Pelletier, who not only taught me how to use the equipment but also served as an exceptional teacher and mentor throughout my life. It was a pleasure to learn and work alongside him, benefiting from his vast knowledge. I am truly indebted to Xavier for his unwavering support, even while he was under significant pressure with other projects.

I would also like to express my deepest gratitude to my cousin, Mohammad Mahdi Kermani, who supported me at every step of this project, particularly in programming. His guidance in software engineering and artificial intelligence was invaluable.

A huge thanks to Yasman Hedayatnasab and Alireza Saboukhi Saboukhi for their emotional and scientific support whenever I faced challenges in my project. I am also grateful to my dear friends, Razieh Shahedi, Mahsa Vaez, and Maryam Iraji, for their constant encouragement.

I also want to express my heartfelt thanks to Arash, who supported me during challenging times, holding my hand and never leaving me alone. I appreciate the help provided by Eric Poiriera in designing the fixtures. A special thanks to Mahan Sandoghi for his assistance.

Last but not least, I would like to sincerely thank my beloved parents for their unwavering support throughout my life.

Table of Contents

List of Figures

List of Tables

List of Abbreviation

Abbreviation	Full Form
AC	Alternating Current
AM	Additive Manufacturing
API	Application Programming Interface
ASTM	American Society for Testing and Materials
CCD	Charge-Coupled Device
CNC	Computer Numerical Control
DED	Direct Energy Deposition
DOF	Degree of Freedom
DC	Direct Current
FEA	Finite Element Analysis
FOV	Field of View
GUI	Graphical User Interface
HMI	Human Machine Interface
ISO	International Organization for Standardization
Kg	Kilogram
MPa	Mega Pascal
mm	Millimeter
M/S	Meter/Second
M/S^2	Meter per Second Squared
N.m	Newton Meter
PLC	Programmable Logic Controllers
WAAM	Wire Arc Additive Manufacturing
RPM	Revolution per Minute
TP	Teach Pendant
VSU	Voltage Supply Unit
PBF	Powder Bed Fusion
EBM	Electron Beam Melting

SLM	Selective Laser Melting
SLS	Selective Laser Sintering
DMLS	Direct Metal Laser Sintering
RAPID	ABB's Robot Programming Language
IP	Intellectual Property
CAPI	Controller Application Programming Interface
K-means	A clustering algorithm (no specific full form)

Chapter 1

1 Introduction

1.1 Overview

Manufacturing of metallic parts can be divided into three distinct categories which are: i) machining or subtractive manufacturing, ii) forming, and iii) additive manufacturing (AM) [1]. In the context of machining, it is a common traditional practice to remove material from the bulk of substrate to shape the desired end-user product, called subtractive manufacturing. The forming method is centered on the utilization of mechanical, chemical or thermal forces to shape the metal to the desirable profile. These two methods possess a footprint belonging to history over a millennium [1]. However, certain drawbacks and barriers, including design constraints, longer production times, material waste, and limitations in handling complex geometries, have necessitated the development of a more flexible technology known as AM. In the 1980s, a new method was introduced in which a 3D object could be produced via adding materials following a systemized approach. There was ongoing debate to assign name on such method; 3D printing, rapid prototyping, or rapid manufacturing were commonly used. Finally, in 2013, two main institutions of Standards, ISO and ASTM settled the case by calling this method, AM. Since then,

AM technology, i. e. the layer-by-layer material deposition has become a widely used technique to build final products in the manufacturing industry.

AM is classified into seven major categories as illustrated in Figure 1-1, offers several advantages over subtractive manufacturing, both intrinsic and extrinsic. These include optimized waste reduction. Additionally, it allows to produce near-net-shape objects, leading to cost and time efficiency. It is compatible with a wide range of materials and facilitates the production of large components and media [2]. From an economic perspective, AM does not require the costly tooling or dies needed for traditional forming processes. Furthermore, AM enhances deposition rates and enables the creation of diverse object morphologies. The improved buy-to-fly ratio [2] and aerospace-approved features [3], [4] are among its most notable benefits [3], [4]. All in all, it can be anticipated that AM is a kind of bidirectional process. Compared to forming methods, AM offers greater design flexibility, enabling the production of complex geometries with fewer post-processing steps, addressing one of the main challenges of traditional methods.

Despite these advantages, AM still has inherent drawbacks, such as heat accumulation and near-net-shape production issues that can lead to defects like geometrical deviations, porosity inside the weld and/or surface irregularities. It is noted that various process parameters are directly related to the above-mentioned production issues. Therefore, it is crucial to control various parameters during the production process.

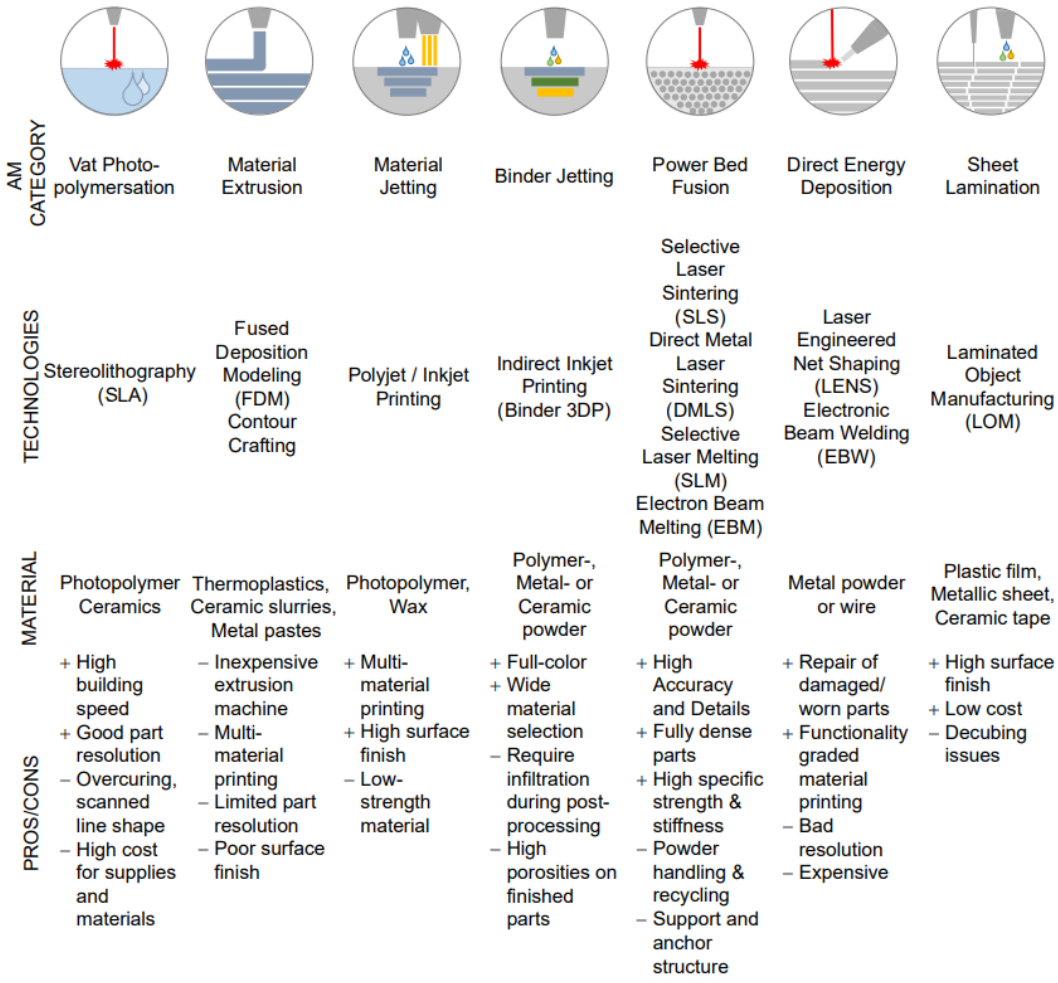


Figure 1-1 Classification of polymers, ceramic or wax, and metal AM [1].

Robots play a vital role in automation of industrial processes including AM, pick and place, welding, coating, machining etc. [5]. A robot of six-degree freedom enables the users to process WAAM autonomously and functions as a cornerstone step towards automation (**Figure 1-2**). The remaining barriers were to introduce quality control of end product and parameter tuning of the robot control system. By introducing a vision system in the robot control, these barriers are overcome with the best of its ability.

To minimize the human error, the vision system was introduced to the system for the first time in 1996 and its evolution initiated from two charge-coupled device (CCD) camera to three-dimensional (3D) camera scanner as of today industrial examples [6]. Different types of vision systems have been employed into the industrial manufacturing with respect to the dimension of

exposure (i.e. field of view and/or size of sample) which are: i) single dot laser[4], ii) line laser [7], [8], iii) two CCD camera [6], and iv) 3D scanner [9].

Figure 1-2. Apparatus of six-degree freedom 4600ABB robot (adopted from https://www.turbosquid.com/ko/3d-models/3d-abb-irb-4600-industrial-robot-02-1915018)

The aforementioned vision systems are mostly installed on robotic arm except 3D scanner. To scan the entire surface, the sample rate of these lasers is synchronized with the speed of robot. 2D cameras can be equipped with stationery or installed on the robot. Stationary ones are frequently used to determine the height of the sample.

Another breakthrough of AM is the ability to repair components. Thanks to its aerospace approved feature, a vast application to the repair of aerospace components is the emerged added value of such method (see **Figure 1-3** [10]. In this thesis, we will focus on one of AM methods, i. e., robotic WAAM machine guided by a 3D scanning camera and to investigate an adaptive correction strategy to improve the quality of product.

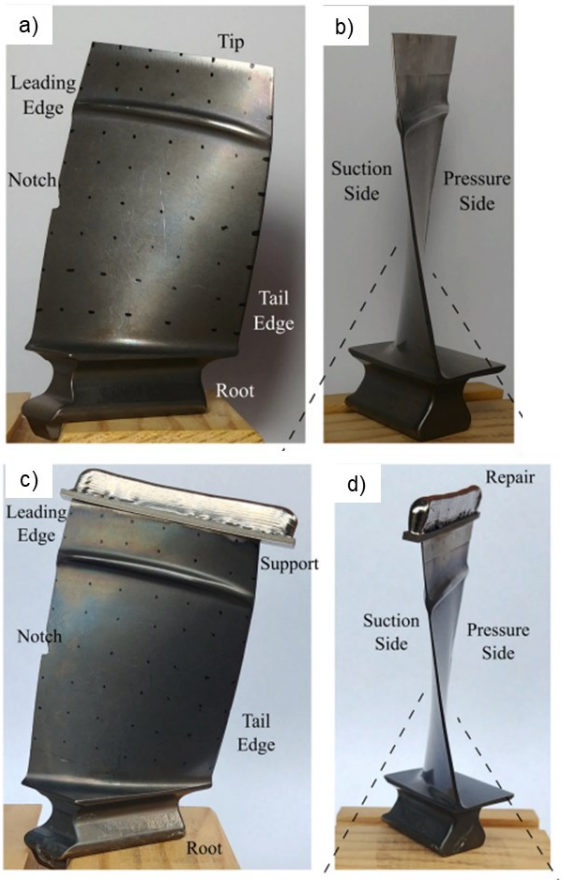


Figure 1-3. (a & b) pre-repair of defected blade; (c & d) post-repair of the blade [10]

1.2 Problem Statement

The introduction of vision system to AM was a great breakthrough in the manufacturing industry. However, experts are still faced the following challenges: i) low resolution and demand of post-processing, ii) the limitations imposed by inbuilt offline control in robotic software, which restricts real-time path planning, process feature correction, and adjustment of process parameters [11].

Taking these concerns into account, it is possible that cavity and irregularities would be created through deposition and such a trend is extended to the next layer of deposition that may introduce the robot gun's collision as well as causing generation of unfavorable porosity (See **Figure 1-4**). The schematic of irregularities and cavity is illustrated in **Figure 1-5** in which the blue region represents cavity and red one describes the irregularities.

Another barrier is the heat accumulation [11], [12]. In case this effect occurs, the deposited layers would not build up and droplets formation for one side might take place instead of increase in height of deposited layers (**Figure 1-6**).

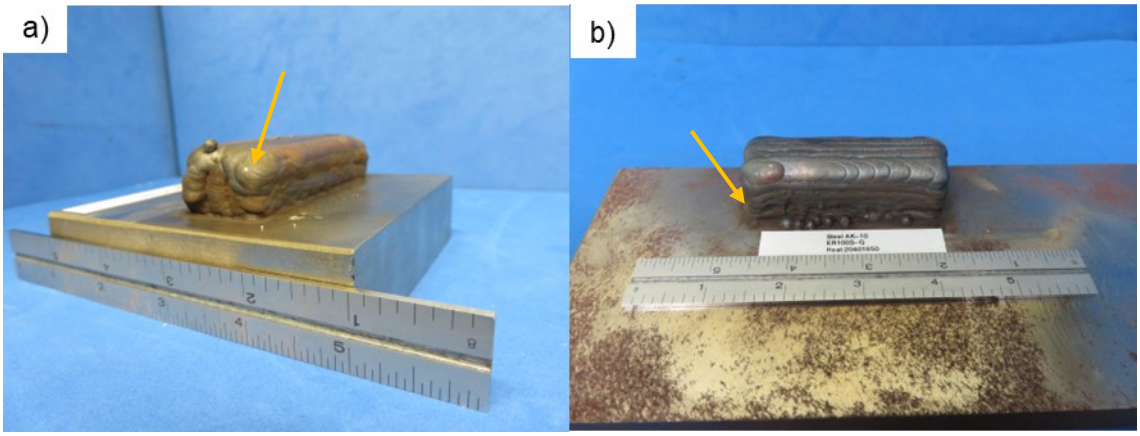


Figure 1-4 Examples of creation of (a) irregularity; (b) cavity.

Additionally, the present technologies of welding can operate under passive and offline control. Therefore, the preliminary experiments are imperative in order to find the optimal parameters which will be used and maintained constant during AM process.

Path planning is another issue that conventional practices are facing. The programming of the robots does not define inherited and built-in path planning and the common missing feature is the absence of path planning add-on. An add-on is a package of codes or modules that is integrated into pre-existing software to address its drawbacks, eliminate limitations, and enhance its capabilities, all without altering the original software's core code. This thesis aims to fill the gap by designing a path planning add-on to determine the path based on the measured data from the vision system.

Oxygen and other elements such as nitrogen, carbon and hydrogen may play a role as contamination species in the terms of exposing to reach melted first layer. With an increase in height of gun head, the torch may be exposed to such contaminants from the atmosphere to higher extent causing the generation of turbulence (**Figure 1-7**). Consequently, atmospheric elements contaminates the melting pool which results in undesirable porosity [13] and also discoloration of the surface. In laser and especially electron beam methods, it is vital to undertake the additive process in a vacuum chamber, whereas WAAM can be operated at ambient pressure and in an

open chamber. However, the torch requires shielding gas and separate shielding for the head of the welding gun. Laser systems can also operate with just shielding gas, whereas electron beam systems require more stringent conditions; however, a special electron beam gun has recently been developed for low vacuum environments. Typically, laser operations occur within a chamber due to the eye hazards associated with class 4 lasers. In contrast, WAAM (Wire Arc Additive Manufacturing) differs in this regard, as the hazards from the arc can be effectively mitigated using curtains and appropriate eyewear

To meet the above-mentioned challenges, the research community is seeking the correlation between various process parameters and the quality of the welding product and an adaptive correction scheme to automate the WAAM process to enhance the quality of the manufactured or repairing products.

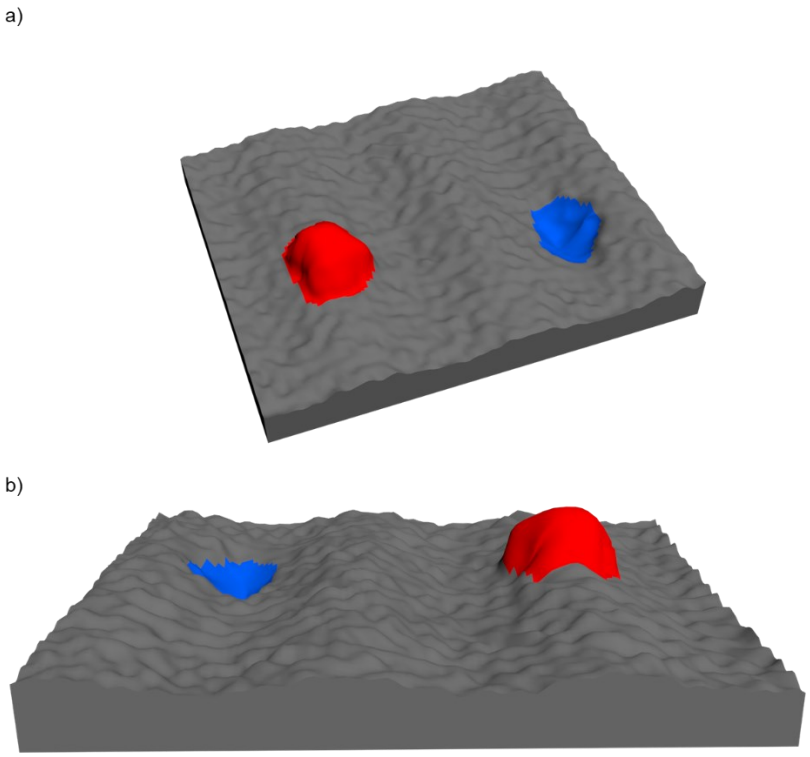


Figure 1-5 Schematic of irregularities (red) and cavity (blue)

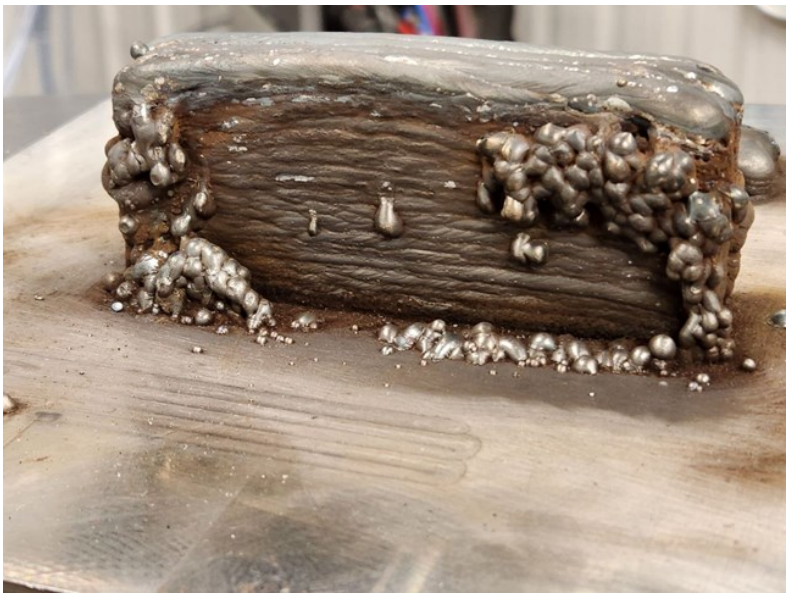


Figure 1-6 Droplet formation during layer deposition

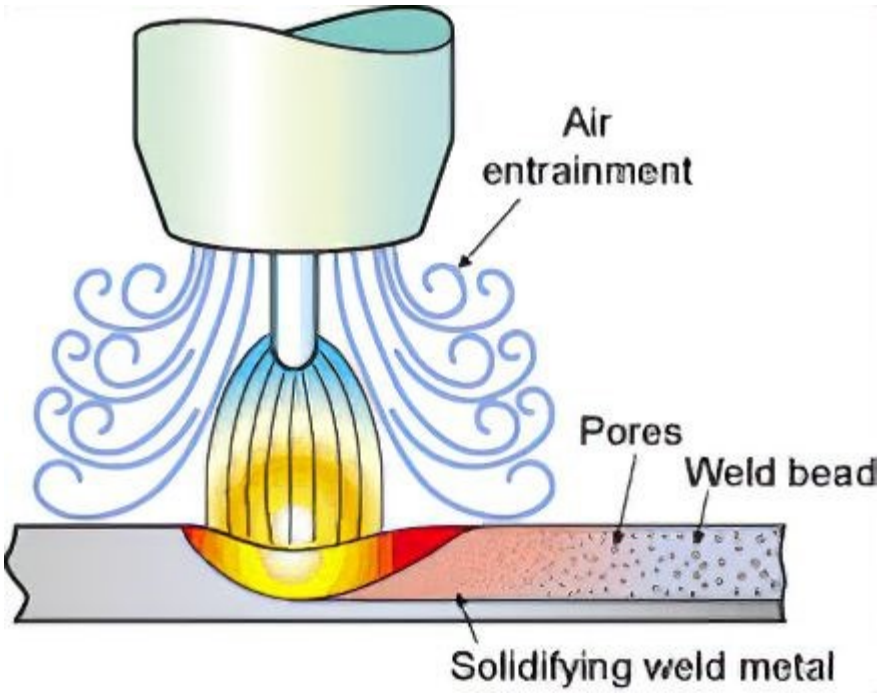


Figure 1-7 Schematic of generated turbulence due to increase in height of gun head (adopted from: https://www.twi-global.com/technical-knowledge/job-knowledge/defects-imperfections-in-welds-porosity-042)

1.3 Scope and Objectives

The scope of this research is focusing on developing a scheme to automate the WAAM process to enhance the quality of the manufactured parts. The research work will be conducted throughthree phases which are: Phase I: Programming, Phase II: Operation of process, and Phase III: Experimental Test. The initial point of the project is assessing the performance of an offline open-loop control system of AM and the final aim is to transform the system to an online closed-loop control one. The rationale of this research is to tackle the barriers of an offline system such as the demand for the time and instrument to measure the height of each deposited layer as well as any cavities, irregularities, surface contaminations, etc. (described in detail in Section 1.2). Automation of the AM process is the missing chain and acts as the motivation of this research.

To begin with, Phase I focuses on programming different devices and modules which are: robot, vision system, and Graphical User Interface (GUI). To run the process, it is essential to synchronize the parameters of the welding system with robot ones where Phase II comes in. The

novelty of operation lies in implementing programming to one of the state-of-the-art technologies of WAAM which is patented TOPTIG device located at National Research Council, Canada.

Then, the algorithm validation emerges as the critical step to ascertain whether the height of each layer can be autonomously adjusted with the previous deposited layer or not. In such context, the essence of Phase III: Experimental tests comes into consideration.

The main research objective of this thesis is to develop a fully automated WAAM process using a vision system. The adaptive correction strategy includes the determination of the optimized position, number and the type of clamps to minimize the probability of collision between the gun and clamps and prevent warping of the base materials. Also, the path planning needs to be integrated with online programming of the robot and an algorithm for mesh analysis is expected to find the cavities and irregularities.

1.4 Achieved Contributions

The major contributions of this research work are summarized as follows:

- **Optimization of Welding Machine Inputs:** We optimized the inputs of the welding machine, leading to three predefined settings for infilling large, medium, or small size objects. This optimization improves process efficiency and consistency.
- **Path Planning Methodology:** We identified a feasible method for path planning, applicable even for complex shapes. This methodology enhances the precision and adaptability of the manufacturing process.
- **Offline RobotStudio Programming**: We developed an offline RobotStudio robot program capable of depositing multi-line, multi-layer objects for basic shapes such as rectangular and hollow cylinders. This program facilitates the creation of consistent and high-quality components.
- **Nozzle Shield Development:** We designed a shield for the nozzle to prevent contamination of the weld pool and oxidation of the object and tungsten. This shield extends the lifespan of the nozzle and improves weld quality.

- **G-Code Conversion:** We identified the program for converting G-codes to RAPID language, streamlining the process of translating design files into executable robot instructions.
- **Virtual Testing Environments:** We developed virtual environments in RobotStudio for testing codes and in RoboDK for testing planned path. These environments allow for collision avoidance and the identification of singularities, enhancing safety and reliability.
- **Gripper Design:** We designed a gripper for the calibration object, used for calibrating the 3D camera with respect to the robot workspace. This design improves the accuracy and repeatability of the calibration process.
- **3D Camera Programming:** We programmed the 3D camera to integrate with the manufacturing process, enabling real-time monitoring and adjustments.
- **Mesh Processing Program:** We developed a mesh processing program for analyzing the point cloud of the scanned object to correct the next layer path plan. This program enhances the accuracy and quality of the final product.
- **Adaptive Correction Strategy:** In simulation, we dynamically adjusted parameters, to reduce errors and enhance overall quality, with the potential to expand the method's applicability to complex manufacturing environments.

1.5 Thesis Structure

This thesis consists of five chapters. The outline of the thesis is given as follows.

- Chapter 1 provides a comprehensive overview of the research topic and the problem statement. It sets the stage for the thesis by defining the scope of the study and outlining the key objectives to be achieved.
- In Chapter 2, a thorough review of the existing literature is presented. It covers the current AM techniques, path planning methods, simulation software, and vision systems. This review establishes the context for the research by identifying gaps and opportunities in the existing literature.
- Chapter 3 details the materials used, and the methodologies adopted for all steps of the project, from deposition to path planning. It includes descriptions of experimental setups, protocols, and

the software or tools utilized in the research. This provides the foundation for the experimental work.

- Chapter 4 discusses the results obtained from mesh analysis programming and microstructure examinations. It includes a detailed analysis of the data, interpretation of findings, and comparison with existing literature. The discussion highlights the significance of the results and their implications for the field of study.
- Chapter 5 summarizes the key findings of the research, highlighting the conclusions drawn from the study. Additionally, recommendations for future studies are provided, suggesting potential areas for further investigation to build on the work presented in this thesis.

Chapter 2

2 Literature review:

2.1 Additive Manufacturing

Additive manufacturing (AM) is one of the most closely aligned tools with bottom-up manufacturing methods, where a structure can be constructed based on a design using a 'layer-by-layer' approach [1]. Traditional methods such as casting, forging, and machining do not offer the same degree of unprocessed freedom in manufacturing versatile objects, including composite, complex, and hybrid structures that require immediate precision and control [1–3]. AM broadens its applicability to a diverse range of materials, including ceramic, polymeric, and metallic substances, as well as combinations thereof, in the fabrication of hybrid, composite, or functionally graded materials (FGMs)[1].

It is crucial to evaluate how ASTM standards integrate with current AM technologies. There are seven ASTM categories to consider: i) binder jetting (BJ); ii) directed energy deposition

(DED); iii) material extrusion (ME); iv) material jetting (MJ); v) powder bed fusion (PBF); vi) sheet lamination (SL); and vii) vat photopolymerization (VP). These categories are listed in **Table 2.1**. In this study, DED and PBF will be thoroughly discussed, as both methods involve the melting of material.

2.1.1 Power bed fusion

PBF process enables the user to import 3D solid CAD model data and fabricate parts following the concept of layer-by-layer addition. Heat energy sources of this method play a pivotal role in classifying PBF processes. These sources can be electron or laser beam where lasers are more commonly used in industry. The types of PBF processes are:

- Selective Heat Sintering (SHS)
- Selective Laser Melting (SLM)
- Electron Beam Melting (EBM)
- Selective Laser Sintering (SLS)
- Direct Metal Laser Sintering (DMLS)

Sintering processes by laser (e.g. DMLS, SLS) also includes two subcategories which are metal laser sintering (mLS) and polymer laser Sintering (pLS). The mechanism underlaying all PBF processes is spreading powder over previously formed layers. A roller or blade mechanism carries out such a role. Briefly, the first step is spreading the powder and involves fusing the powders via as beam to solidify them layer-by-layer into the final components.

EBM stands as one of the most effective methods of PBF thanks to six major merits [1,3] that can bring to the AM process.

- **Higher Energy Efficiency:** Electron beams are far more energy-efficient compared to laser-based methods. This feature can predominantly contribute to lessening operating cost
- **Reduced Residual Stresses**: Operation at higher temperatures due to lower heat transfer in vacuum environment, required in EBM processing, leads to minimizing the residual stress through the built parts. Subsequently, the demand of post-processing can be pronouncedly decreased.

- **Material Utilization**: High-melting-point materials such as titanium and nickel-based superalloys are the most suited to incorporate EBM with. This property turns EBM to an adaptable method for aerospace and medical applications.
- **Faster Build Speeds:** Rapid reflection of electron beam permits to build part much faster compared to laser-based ones.
- **Less Need for Support Structure:** Thanks to the higher process temperature, the demand of support structures faces significant decrease for the built parts with EBM. Consequently, material waste and post-processing can be remarkably decreased.
- **Homogeneous Material Properties:** The porosity of parts would be pronouncedly reduced and the composition of the parts are homogenized due to complete melting of powders through EBM. Reduced porosity poses remarkable enhancement in durability of the parts and the overall strength.
- **Vacuum environment:** minimizes oxidation of powder within the build chamber, allowing for improved powder reuse, which is crucial for high-value materials.

Although EBM is one of the most effective methods, its high cost can be a barrier for scale-up and future use compared to DED methods such as WAAM. The higher cost of EBM is primarily due to the requirement of a vacuum chamber, which significantly increases both operational and capital expenses, whereas DED methods do not require vacuum conditions, leading to lower costs [4]. The key to WAAM lies in its low capital costs, due to arc technology, and reduced operational costs, thanks to local shielding.

2.1.2 Direct energy deposition

DED process is a reliable method for creating tailored surfaces and arbitrary shapes through line-by-line deposition of metallic materials. This capability allows for the fabrication of heterogeneous materials with desirable characteristics and properties by using different materials simultaneously for deposition. The simpler apparatus of DED, compared to PBF, facilitates the development of hybrid processes. Consequently, DED processes have garnered significant attention from researchers in recent years [5]. DEDs can be classified based on two criteria of: i) type of feedstock and ii) type of energy source (See Figure 2-1) [6].

Table 2.1. Summary of ASTM category adaptation with AF, modified from [1].

ASTM Cat.	Basic Principle	Example technology	Advantages	Disadvantages	Materials	Build volume (mm×mm×mm)
DED	Focused thermal energy melts materials during deposition	Laser deposition (LD) Laser Engineered Net Shaping (LENS) Electron beam •Plasma arc melting	High degree control of grain structure High quality parts Excellent for repair applications	Surface quality and speed requires a balance Limited to metals/metal based hybrids	Metals Hybrid	Versatile X = 600–3000 Y = 500–3500 Z = 350–5000
PBF	Thermal energy fuses a small region of the powder bed of the build material	Electron beam melting (EBM) Direct Metal Laser Sintering (DMLS) Selective Laser Sintering/Melting (SLS/SLM)	Relatively inexpensive Small footprint Powder bed acts as an integrated support structure Large range of material options	Relatively slow Lack of structural integrity Size limitations High power required Finish depends on precursor powder size	Metals Ceramics Polymers Composites Hybrid	Small *X* = 200–300 *Y* = 200–300 *Z* = 200–350
BJ	Liquid binder/s jet printed onto thin layers of powder. The part is built up layer by layer By glueing the particles together	3D inkjet technology	Free of support/substrate Design freedom Large build volume High print speed Relatively low cost	Fragile parts with limited mechanical properties May require post processing	Polymers Ceramics Composites Metals Hybrid	Versatile (small to large) X = <4000 Y = <2000 Z = <1000
ME	Material is selectively pushed out through a nozzle or orifice	Fused Deposition Modelling (FDM)/Fused Filament Fabrication (FFF), Fused Layer Modelling (FLM)	Widespread use Inexpensive Scalable Can build fully functional parts	Vertical anisotropy Step-structured surface Not amenable to fine details	Polymers Composites	Small to medium *X* = <900 *Y* = <600 *Z* = <900
VP	Liquid polymer in a vat is light-cured	Digital Light Processing (DLP)	Large parts Excellent accuracy Excellent surface finish and details	Limited to photopolymers only Low shelf life, poor mechanical properties of photopolymers Expensive precursors/Slow build process	Polymers Ceramics	Medium *X* < 2100 *Y* < 700 *Z* < 800
SL	Sheets/foils of materials are bonded	Laminated Object Manufacturing (LOM) Ultrasound consolidation/Ultrasound Additive Manufacturing (UC/UAM) Stereo Lithography (SLA)	High speed, Low cost, Ease of material handling	Strength and integrity of parts depend on adhesive used Finishes may require post processing Limited material use Limited to photopolymers	Polymers Metals Ceramics Hybrids	Small *X* = 150–250 *Y* = 200 *Z* = 100–150
MJ	Droplets of build materials are deposited	3D inkjet technology Direct Ink writing	High accuracy of droplet deposition Low waste Multiple material parts Multicolour	Support material is often required Mainly photopolymers and thermoset resins can be used	Polymers Ceramics Composites Hybrid Biologicals	Small *X* = <300 *Y* = <200 *Z* = <200

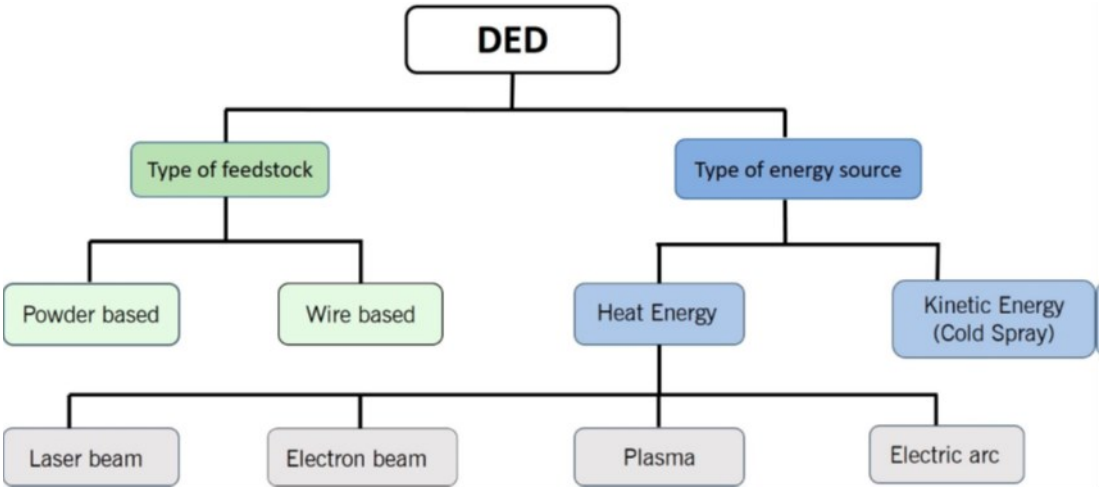


Figure 2-1 Classification of DED processes via type of feedstock and type of energy source [6].

2.1.2.1 Wire Arc Additive Manufacturing (WAAM)

The overall mechanism of Directed Energy Deposition (DED) involves fusing materials by melting them during the deposition process. This focused thermal energy, necessary for melting and depositing the material simultaneously, can be supplied via laser, electron beam, electric arc, or plasma. The melted material is then regulated to be deposited according to a pre-defined CAD model through a layer-by-layer mechanism. Once deposited, the material solidifies into the desired structure.

The focus of this research was WAAM. This method uses the energy source of electric arc and metal wire as feedstock and is known as one of the modern methods of AM [7]. WAAM can be competitive with other AM methods thanks to [8–11]:

1. Lower Equipment Costs and Material Utilization: the cost of the device using for electric arc as a source of energy is much more cost effective than available DED methods with the source of electron beam and laser. Equally important, the wire used for WAAM is commonly less costly than metal powders that are typically used for other DED methods [7].

2. Structural Integrity: The components that would be produced through WAAM pose competitive mechanical properties and meets high structural integrity and there would be room to post-process via heat treatments to enhance these properties[12].
3. Scalability: The most notable benefit of WAAM is its ability to produce large structures via a means of cost-effective technology and complex parts can be built without constraints defined for size in PBDs.
4. High Deposition Rate: Such properties result in lessening production times and be particularly advantageous for large parts.
5. Repair and Maintenance: WAAM is an effective refurbishing and repairing tool by adding material to damaged or worn parts. At some point that replacing parts can be costly, repair turns out to be crucial for enduring the service life.
6. Reduced Material Waste: Compared to machining, WAAM led to remarkably less waste.
7. Hybrid Manufacturing: The ability to combine WAAM and robotic machining can remarkably enhance the strength of each method for production of parts. It points out that WAAM is a compatible method with different manufacturing techniques [11].
8. Material Flexibility: It is feasible to produce metal composite materials by simultaneous feeding of vast range of materials such as nickel-based alloys, aluminum, steels, and titanium.

Overall, these properties facilitate the application of WAAM in industries that require precise and structurally sound materials, including aerospace, maritime, and automotive sectors.

2.2 Vision system

In AM technology, accurately scanning and reconstructing the point cloud is crucial. This step is essential for both building a model from scratch and for repair processes. Noisy or inaccurate scanning can result in deviations that require parts to be redeposited or ground down, and in repair processes, it can lead to failures in mechanical properties or, in the worst-case scenario, significant divergence between the repaired part and the original intended part. Three factors determine the accuracy of the point cloud [13] (see Table 2.2):

1. The type of scanner selected for scanning.

2. The point extraction method used to identify the object's points relative to the environment.
3. The alignment method employed to reconstruct the 3D point cloud from the scans.

Table 2-2. 3D scanning technology advantages and drawbacks.

Technology	Time of acquisition	Resolution	Resilience	Handling
Contact based	High	High	Very High	Low
Photogrammetry	Medium	Low	Low	Very High
Laser	Low	Very High	Low	Low
Time-of-flight	Very Low	Medium	Medium	High
Structured light	Very Low	High	Low	Low

2.2.1 Contact based 3D scanning

In this method, a touch probe attached to the robot head or another robotic arm contacts the object to determine its coordinates in space. For example, a contact-based 3D scanner, such as a coordinate measuring machine (CMM) [39], operates on this principle. Each time the probe touches the object, a point is recorded. The main advantage of this method is its flexibility, especially for shiny or transparent objects, as it does not rely on cameras or laser beams. Consequently, lighting conditions in the environment are not a concern. However, because the resolution depends on the number of points acquired, this process can be time-consuming [13].

2.2.2 Photogrammetry

Photogrammetry can be performed using a single 2D camera [14], operating in passive mode, or with a combination of two cameras to create a stereo vision system, which allows for the recognition of a region of interest [15]. When using multiple images from different views, they must be stitched together in the final step. A simple camera or a charge-coupled device (CCD) can be used to capture the images. After capturing the images, software merges them by matching pixels that correspond to the same physical points. The resolution of the images determines the accuracy of this method. However, the process can be time-consuming, depending on the number and size of the images [16].

2.2.3 Laser 3D scanning

We have three different types of laser systems. In point laser applications, the quality of the point cloud depends on the speed and interval of sampling from the surface [17]. In this method, the laser should be installed on the gun or another robotic arm to scan the surface [17]. Another method is the laser line, which offers greater accuracy [18].

The 3D laser scanner operates using trigonometric triangulation to generate point cloud data for the scanned surface of deposited objects. The principle behind this method involves projecting a point onto the surface of an object, with a sensor capturing the reflected beam from the surface. Given the predefined distance between the laser and the sensor, and knowing the angle of the laser beam's reflection, the coordinates of the points are calculated. Although 3D scanners provide higher precision, they are not suitable for highly reflective surfaces on materials like aluminum [19]

2.2.4 3D Scanning with combination of two cameras and a laser

By adding a laser to two angled CCD cameras, stereovision scanning is achieved. In this method, the torch is controlled using an ant colony algorithm. A linear laser beam detects the position and orientation of the weld seam [20].

2.2.5 Structured Light 3D Scanning

One of the most popular 3D scanning methods widely used in industries is structured light 3D scanning. According to 3DINSIDER, structured light 3D scanning is “a 3D scanning technology that uses a single light source to project multiple lines onto an object, all of which are tracked simultaneously by one or more cameras.” The key difference between this system and conventional laser scanning is that structured light projects multiple lines simultaneously, while traditional lasers emit individual dots of light one by one. The lines can be white, blue, or green, and the pattern typically consists of multiple parallel stripes [21].

The cameras capture the shape of these patterns, and the distance between the points and the sensor is used to generate the 3D model. The complex calculations required for this process are performed within the 3D scanning device. Among the advantages of this method are its high

accuracy and fast scanning speed. Unlike traditional laser scanners, which need to launch the laser multiple times to gather point cloud data, structured light 3D scanning collects all this information in a single shot.

However, this method also presents challenges that must be addressed. Shadows or light sources can interfere with the scanning process, as this method relies on optical sensors. Reflections from shiny and reflective materials can lead to data loss and incorrect triangulation or point coordinate placement. Applying a dark, non-reflective coating to the object can help mitigate these issues.

Table 2.2 outlines the different characteristics of scanning technologies. The repair and build process of a 3D object requires a scanning method that is both fast and accurate. Consequently, a structured light 3D scanner represents the best solution for this purpose.

2.3 Volume identification and 3D Model Reconstruction

After gathering information from the 3D model, the next step is to rebuild the object. This process can be challenging due to the massive amount of data, with millions of scanned points and their characteristics stored in a matrix containing their coordinates. For some predefined geometries, some methods have been developed to isolate the object's point cloud from the background. Bokhabrine et al. applied the Gaussian Image (GI) method to find the normal vectors of each point within a unit sphere [21]. Subsequently, the RANSAC (Random Sample Consensus) algorithm was used to extract the normal vectors. By selecting only specific normal vectors from the start, the impact of noise and environmental errors was minimized. In the final step, the Iterative Closest Point (ICP) algorithm was employed to align the scans, define overlaps, and recreate the object's point cloud. This method generates an accurate and robust point cloud within the region of interest (ROI) but is limited to predefined geometries introduced in a Gaussian environment [13].

To overcome the limitations of the Gaussian algorithm, machine learning tools can be utilized. Machine learning algorithms are widely used in face recognition and object detection in the automotive and shipbuilding industries. These algorithms can identify and extract complex shapes and geometries from point clouds. Huang et al. applied a support vector machine (SVM) for segmenting the point cloud [22]. The classifier was defined by a 33-dimensional Fast Point Feature

Histogram (FPFH). The algorithm was trained offline on over 200,000 points labeled as planes, pipes, edges, and thin pipes, and then used for clustering. In the next step, the Flood-Fill algorithm grouped sets of points, assigning them to specific lists. Through this process, specific object points can be identified and extracted, reducing the effects of noise by selecting neighboring points associated with a specific object. However, SVM algorithms are limited to the shapes they were trained on and are only effective for those specific shapes. Additionally, this method only works on a single scan to extract points on the object's surfaces and cannot handle reconstruction from multiple scans.

2.4 Online control

Conventional control methods are not well-suited to today's industrial needs because they lack flexibility and are computationally intensive. Therefore, industries require more adaptable and efficient control strategies that can meet the demands of hybrid and complex environments.

2.4.1 Online Robot Trajectory Planning and Programming Support System

Robot trajectory and programming are divided into three categories: traditional, offline programming, and online programming. The traditional approach to robot trajectory planning relies immensely on manual mathematical computations to determine the precise path of the robot. This method often requires significant expertise and time, making it less efficient for complex or dynamic environments. Offline programming involves writing the robot's program, uploading it to the robot controller, testing the code in a simulator or the real world, and then stopping the process to make modifications and corrections. In contrast, online programming allows the operator to access the controller in real-time and edit the code while the robot is running, without the need to stop the process.

A trajectory planner combined with Voronoi generation creates a support system that minimizes the need for manual programming, enhances real-time control, and improves path planning [23].

2.4.2 Incremental Online Sparsification for Real-Time Model Learning

To achieve successful real-time robot control, the method must quickly adapt to changes, support instant learning, and be computationally efficient. Gaussian Processes Regression (GPR) is highly accurate but requires significant computational resources, which can lead to instability in the process.

Local Gaussian Processes Regression (LGP) addresses this by using multiple local models for different regions of space. This approach offers the advantage of maintaining high accuracy while keeping computational costs low. In this method, real-time control of the robot is enhanced through data assignment, model learning, and prediction.

The results demonstrate a significant improvement in tracking accuracy, fast computation, and high stability in real-time control [24].

2.4.3 Online Control Programming Algorithm

The system components include a pretraining online module that gathers and analyzes the gesture features of operators, creating an individual library for each user. For real-time gesture recognition, the system employs a cascade classification algorithm that delivers acceptable precision. This method allows the user to rewrite the robot's path planning while the process is running.

Building on the gesture recognition library, the system captures real-time images and, after analyzing them, replans the task accordingly. The flexibility of path planning in dynamic environments and the ability to accommodate various operator inputs are key advantages of this method [25].

2.4.4 Neural Network System for Online Controller Adaptation

The system structure contains both real and imaginary worlds, with a Forward Model (FWD) based on a neural network used for simulating robot movement. This setup allows the control system to adapt in real-time without interruption.

In the imaginary training phase, the error signal from the simulation is used for continuous adjustment, enabling real-time adaptation without halting the ongoing process. Experimental results demonstrate the system's effectiveness in handling disturbances and trajectory adjustments [26].

2.4.5 Local Gaussian Processes Regression for Real-Time Model-Based Robot Control

To achieve successful real-time control, an algorithm is needed that can precisely predict both robot movements and changes in a dynamic environment. Local Gaussian Process (LGP) [24], by combining the regression capabilities of both local and global methods, provides high accuracy and strong computational performance. Since LGP can easily adapt to new data in real-time, it is an ideal choice for dynamic industries [24].

2.4.6 Controlling an Industrial Robot Using a Graphic Tablet

The system utilizes a graphical tablet to convert path coordinates into robot trajectories, enabling intuitive and impulsive robotic programming. This setup includes an embedded digital section for offline testing and real-time compensation, facilitated through Ethernet communications. Users command the robots by drawing paths on the tablet, which are then analyzed by the robot controller's processors. These paths are translated into control commands that the controller executes, demonstrating high accuracy and responsiveness in both simulation and real-world tests [27].

Despite the advantages, real-time and online robot control presents challenges that are currently being addressed, such as ensuring user safety, accommodating new inputs, and managing the high computational demands. The proposed systems enhance the versatility and precision of the process. However, future studies should focus on improving system stability, particularly as the integration of vision systems and online control can introduce instability.

2.5 Mechanical Properties

One of the biggest challenges in the AM process is achieving the required mechanical properties, especially since many parts produced by this method are used in critical and sensitive

industries. Additionally, the selected materials must be weldable. Common materials for this purpose include steel, stainless steel, aluminum, and titanium [28–30].

The thermal cycle experienced by the deposited material can be summarized as heating the wire by tungsten, melting, solidification, cooling, and then remelting [31,32]. Numerous studies have attempted to observe and test the factors affecting the welding process, such as the microstructure of the weld and the resulting mechanical properties. Key factors include shielding gas, path planning, and post-heat treatment.

Research indicates that the required tensile strength can be achieved by performing a heat treatment after the welding process [33]. According to Suryakumar's studies [34], hardness is not significantly affected by thermal treatment, except for the last layers where thermal cycles influence hardness. Increasing the welding machine's current does not alter the tensile strength in any direction. In the case of steel, cold metal transfer (CMT) outperforms gas metal arc welding (GMAW) by exhibiting greater hardness and higher ultimate tensile strength [28]. Moreover, a combination of hot-formed parts and WAAM deposited parts from the titanium alloy Ti-6Al-4V yields properties identical to those produced by standard forging processes [30].

2.6 Tool-Path Generation Strategies for Additive Manufacturing

AM technology provides an efficient method for generating complex structures layer by layer. To achieve high deposition rates and cost efficiency in WAAM, the tool path plays a vital role in determining the weld bead characteristics, which in turn impacts the quality of the deposited surface, mechanical properties, and microstructure of the object [35–37]. The path plan is specifically defined as the movement of the tungsten relative to the base plate. Figure 2-2 illustrates six different types of path plans.

The line or raster infilling method covers regions with parallel beads, which can deviate by 90, 60, or 45 degrees relative to the base material. This is the most common 3D printing method due to its capacity for filling arbitrary models with simple path planning codes. However, its disadvantage is low accuracy at the edges because the deposition paths are inconsistent at the edges and do not align well with the nozzle movement [35,38].

The zigzag pattern is an improvement over the raster pattern, linking the lines along the x- and y-axis to create a continuous line [38,39]. This consistency results in better displacement of the robotic arm, reduces the stop-and-start syntax, and minimizes the likelihood of creating irregularities at each start point [38,40]. However, it still suffers from a lack of accuracy along the border lines, and there remains a possibility of overfilling at turning points [35,38,41].

To address border accuracy, the contour path plan is applied. It prevents overfilling and enhances outline accuracy, making it ideal for shapes with complex boundaries [35,38]. However, the gaps between beads and inefficient movement make the contour pattern less suitable for WAAM applications, which require continuous paths [35,41].

A spiral tool path, commonly used in NC machining, is beneficial for removing material in 2D geometries. Although applicable to AM, it is limited to fractal space-filling curves like Hilbert curves, which cover areas without intersecting themselves. This tool path reduces shrinkage and distortion but is not ideal for WAAM due to its uneven directional paths [42].

By combining zigzag and contour patterns, the benefits of both can be leveraged to enhance the quality of the deposited part. For example, inner paths can be deposited using the zigzag method, which promotes better joint formation between weld beads, increases production speed, and improves utility. Meanwhile, outer border lines can be produced using the contour pattern for its inherent accuracy, resulting in better surface quality. This hybrid pattern effectively meets WAAM production requirements by optimizing the number of passes, simplifying the process, and increasing speed [37].

WAAM, with its high deposition rate, is particularly suited for large-scale objects, making it ideal for producing components with rib-web structures in the aerospace industry. These lightweight structures, typically produced by machining methods, often result in a high buy-to-fly ratio and significant material waste [43]. Rib structures, used in heat transfer equipment, chemical processes, and synthetic processes, are traditionally forged, incurring high tool costs due to the need for specialized dies [44,45]. The advantages of the WAAM process include reducing material waste, tool investment, and production time, making it a viable alternative to conventional production methods [35].

Constant path planning reduces both the number of code lines and the actual travel distance of the tool, thereby improving surface quality. Hilbert path planning algorithms unlock patterns that can achieve constant deposition but may increase heat accumulation and distortion of the base material and the object itself [35]. Another approach involves breaking down 2D geometries into monotone polygons, producing closed zigzag patterns for each polygon, and then merging them into a single continuous path.

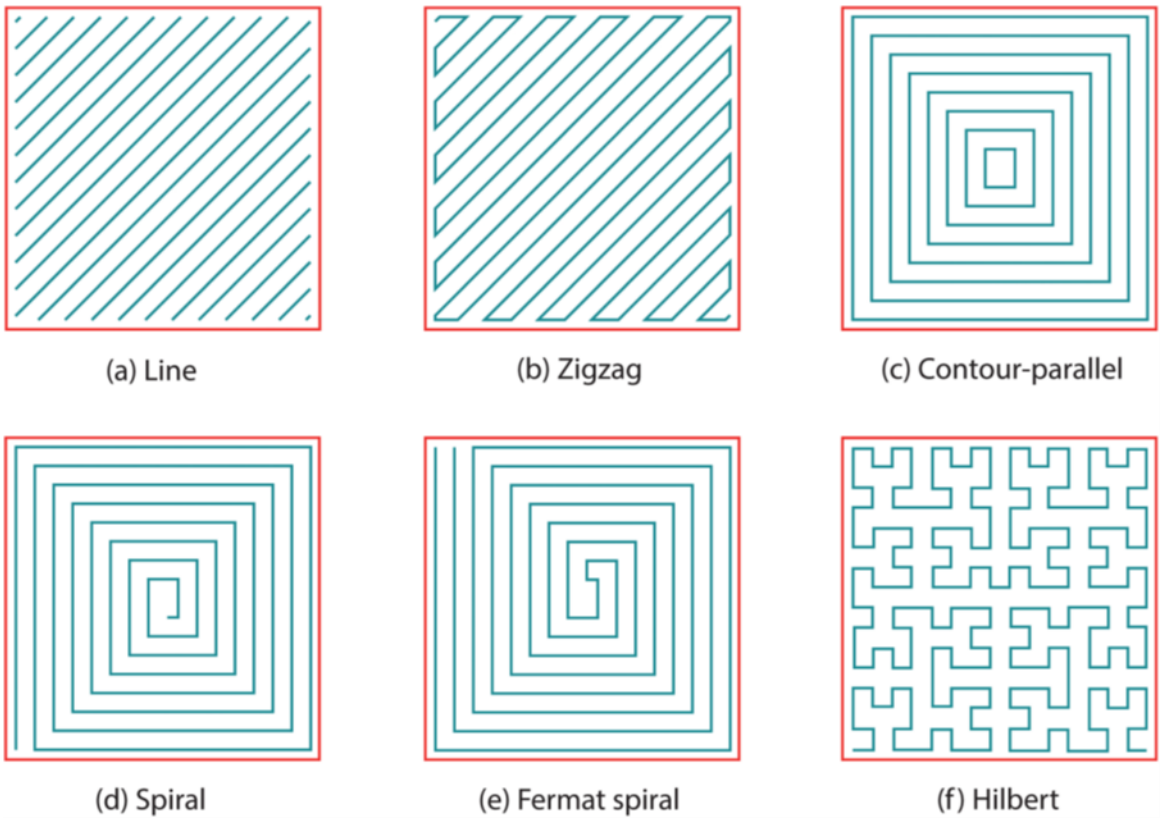


Figure 2-2 Six filling patterns: (a) line; (b) zigzag; (c) contour-parallel; (d) spiral [29,30]; (e) Fermat spiral [31]; (f) Hilbert [32]. The red line represents the boundary of [46].

2.7 K-Means Algorithm

The rapid advancements in techniques for scientific data collection in the era of big data have led to the systematic accumulation of vast quantities of data across numerous data-capturing sites.

There has been exponential growth in the development of various data analysis methodologies, among which the K-means algorithm stands out as one of the most popular and straightforward clustering techniques [47]. The widespread applicability of the K-means algorithm across numerous clustering application domains can be attributed to its straightforward implementation and low computational complexity. However, despite its advantages, the K-means algorithm faces several challenges that can negatively impact its clustering performance.

2.7.1 Challenges of the K-Means Algorithm

One of the primary challenges of the K-means algorithm is the need for users to specify the number of clusters in a given dataset a priori during the initialization process. This requirement can be particularly problematic for large datasets where determining the optimal number of clusters to start with is complex and challenging. Additionally, the initial cluster centers in the K-means algorithm are selected randomly, which makes the algorithm's performance highly sensitive to this initial selection. Random initialization frequently leads to limited local convergence because of the algorithm's inherently greedy behavior.

Moreover, the K-means algorithm uses the Euclidean distance metric to determine the similarity between data objects. While this is effective for spherical clusters, it limits the algorithm's robustness in detecting clusters of other shapes and poses significant challenges in identifying overlapping clusters. These limitations have prompted numerous research efforts aimed at improving the performance and robustness of the K-means algorithm.

2.7.2 Enhancements and Variants of K-Means

Over the years, researchers have proposed various enhancements and variants to address the limitations of the K-means algorithm. These efforts are aimed at improving the algorithm's initialization process, enhancing its robustness to different cluster shapes, and increasing its ability to detect overlapping clusters. Some of the notable enhancements include methods for better initialization of cluster centers, alternative distance metrics, and hybrid approaches that combine K-means with other clustering techniques.

2.7.3 Overview and Taxonomy of K-Means and Its Variants

The current work provides a comprehensive overview and taxonomy of the K-means clustering algorithm and its variants. It covers the historical development of the K-means

algorithm, current trends in its application and development, open issues and challenges faced by the algorithm, and recommended future research perspectives.

2.7.4 Historical Development and Current Trends

The K-means algorithm has a long history, dating back to its initial development in the 1950s. Since then, it has undergone numerous modifications and improvements to enhance its performance and applicability. Current trends in K-means research focus on addressing its limitations through various innovative approaches, including advanced initialization methods, the incorporation of different distance metrics, and the integration of K-means with other clustering techniques.

2.7.5 Open Issues and Challenges

Despite the numerous enhancements, several open issues and challenges remain in the application of the K-means algorithm. These include the need for a priori specification of the number of clusters, the algorithm's sensitivity to initial cluster centers, and its limitations in detecting clusters of non-spherical shapes and overlapping clusters. Addressing these challenges requires ongoing research and development efforts.

2.7.6 Recommended Future Research Perspectives

Future research on the K-means algorithm should focus on developing more robust initialization methods, exploring alternative distance metrics, and designing hybrid clustering approaches that can address the algorithm's current limitations. Additionally, there is a need for more research on methods for determining the optimal number of clusters in large datasets and improving the algorithm's scalability.

2.8 Comparative Analysis of Existing WAAM Systems

Adaptive correction strategies have been explored through various methods and concepts. This section outlines recent advancements in the field and identifies existing research gaps that this thesis aims to address. Yuan et al. [48] utilized offline robot programming and a pre-determined algorithm for collision-free path planning; however, the absence of real-time control presents a key research gap. Incorporating a 3D camera for closed-loop control could dynamically adjust the deposition process, enabling real-time monitoring and correction. Integrating a second

robot for machining and automating mesh analysis for clustering are forward-thinking enhancements that could add significant value to the strategy proposed by Yuan et al. [48], which primarily focused on layer planning and collision avoidance.

In a study conducted by the Reisch research group at the Technical University of Munich [49], a post-process was developed utilizing a multivariate sensor framework to gather data on gas flow, voltage, current, acoustics, and thermal imaging. While this approach offers significant contributions to the field, it lacks dynamic, real-time layer adjustments during deposition. Additionally, the research does not adequately address material stability or clamping methods. The primary gap to be addressed is the development of a real-time adaptive system, as opposed to a pre-planned multivariate sensor framework and fixed path planning. Although the authors incorporate thermal imaging into their monitoring system, real-time visual monitoring of the nozzle tip remains absent. This could be improved by integrating a transparent shield to facilitate continuous observation.

Coutinho et al. [50] also focused on a post-process technique using the Robot Operating System (ROS), where process variables such as wire feed, voltage, current, and travel speed are monitored. While effective, this method relies on predefined trajectories and sensor data, limiting its adaptability to irregularities that require real-time vision systems. The controller system employed by the authors primarily depended on joint position and velocity adjustments, which, although allowing for online trajectory corrections, does not enable real-time detection and correction of surface imperfections. Such capabilities remain unfeasible with the current controller setup.

Dharmawan et al. [51] implemented a reinforcement learning framework where the model iteratively learned to predict and correct surface height errors during the process. This approach addresses the challenges of Multi-Layer Multi-Bead (MLMB) deposition by offering a solution for correcting geometric errors between layers. While the strategy proved effective, the optimization process is time-consuming due to the required training period and iterative learning nature of the model. Additionally, suboptimal performance may occur during the early layers as the model adapts to making accurate adjustments. The method also heavily relies on the initial setup and multiple iterations to become fully operational. Integrating a real-time vision system

with a 3D Cognex camera could enable immediate error correction, offering a more efficient and responsive solution to process deviations as they arise.

In another study on online control, Lizarralde et al. [52] focused on a coordinated motion control approach using a positioning table to align the torch and surface along a pre-planned trajectory. While this system offers room for improvement in terms of real-time adaptability and the potential for multi-robot coordination, their process relied on task-priority kinematic control to ensure adherence to pre-planned trajectories, which notably increased stability. However, the system still lacks a real-time closed-loop control to enable dynamic performance. Additionally, clamping and stability were not parameters addressed in their research

2.9 Summary

This chapter presents a comprehensive literature review on main topics related to this thesis, including AM, vision systems, and online robot path planning. It investigates the advantages and challenges of AM methods such as PBF and WAAM, emphasizing their potential in producing complex, high-integrity structures. The chapter also discusses advanced 3D scanning technologies and real-time control strategies, highlighting their critical role in enhancing precision and efficiency in AM processes. The potential research work in this thesis project is pointed out.

Chapter 3

3 Material and methods

3.1 Introduction

In this Chapter, different parts of the hardware and software of the setup and operation are discussed with the aim of providing a full methodological picture for online control of process with the use of 3D scanner. First, the parameters of welding are optimized, then the offline programming of the robot is developed. Consequently, the best method of path planning is determined. Finally, the robot is transformed to be operational one by online control system. Thanks to the implemented vision system, the path plan of the robot is modified autonomously and online without any interruptions in the process.

3.2 Equipment

3.2.1 Welding machine

The power source of welding machine used in this study is Nertamatic plus machine (**Figure 3-1**), Air Liquide, Montreal, Canada. The controllable and adjustable parameters of welding machine are: current decrement time [s], wire retraction time [s], post-gas time [s], torch pre-gas

[s], wire feed rate [cm/min], current keep time [s], wire stop time [s], wire start time [s], current increment time [s], time of wire slope [s], current [A], reverse pre-gas time [s] prewelding time [s], and prewelding current [A]. Prior to parameter adjustment, a preliminary step existed for assigning current regime (AC, DC, or pulse), programming # to providing interface with welding robot, and finally the command for I/O of wire feed.

The welding gun that was used in this study was TOP-TIG device, Montreal, Canada located at National Research Council, Montreal, Canada. The head of TOP-TIG gun is mounted with two narrow pipes providing coolant with a groove for wire feeding and a tungsten core positioned perpendicular to the inert cylinder of the gun in center. As illustrated in **Figure 3-2**, a black pipe is also connected to the gun for introducing the shielding gas to concentrate the heat and to prevent the oxidation and contamination taking place through welding pool. The function of core is to provide potential difference between the tip and surface and consequent heat generation. Such heat would be used to melt the wire. In addition, the wire feeds at a 45° angle corresponding to core. It is noteworthy to mention that wire can be fed in a wide range of materials selection (e.g., Al. Ti. Stainless steel etc.) which will be fed from a spool (**Figure 3-3**).

Figure 3-1 Nertamatic plus machine used for welding.

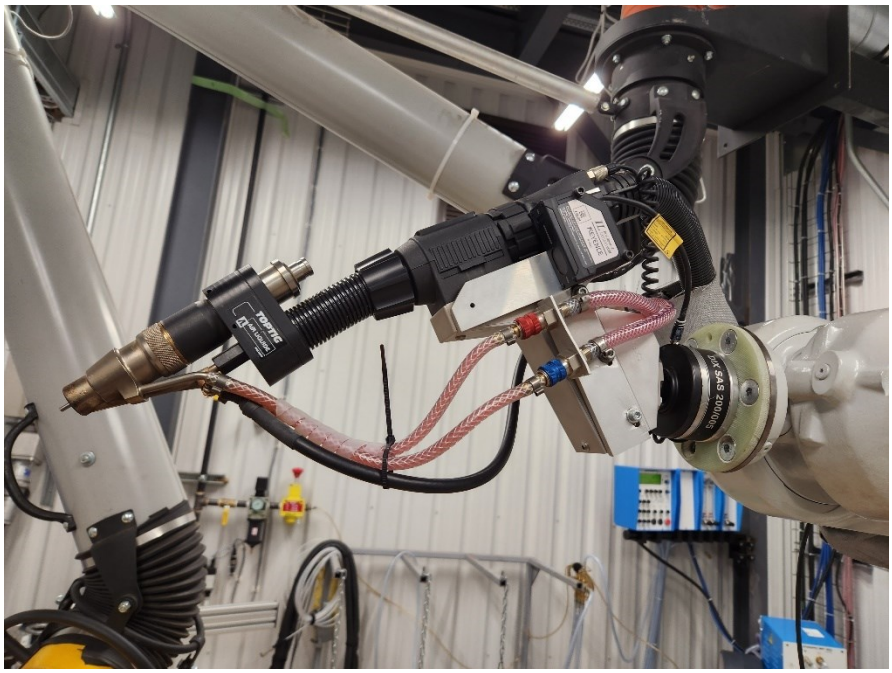


Figure 3-2 TOP-TIG gun equipped with two pipes for coolant.

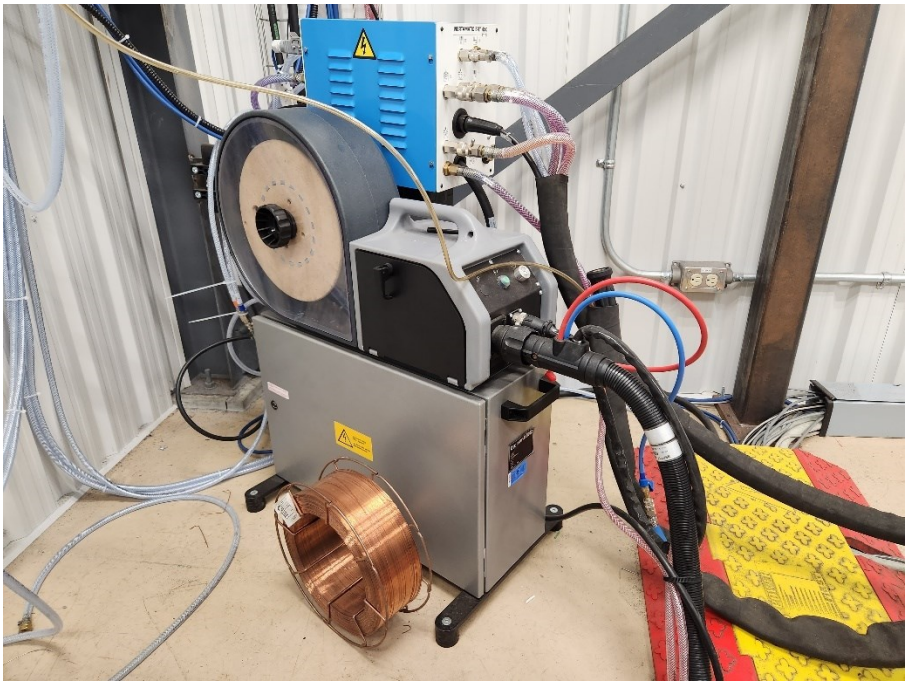
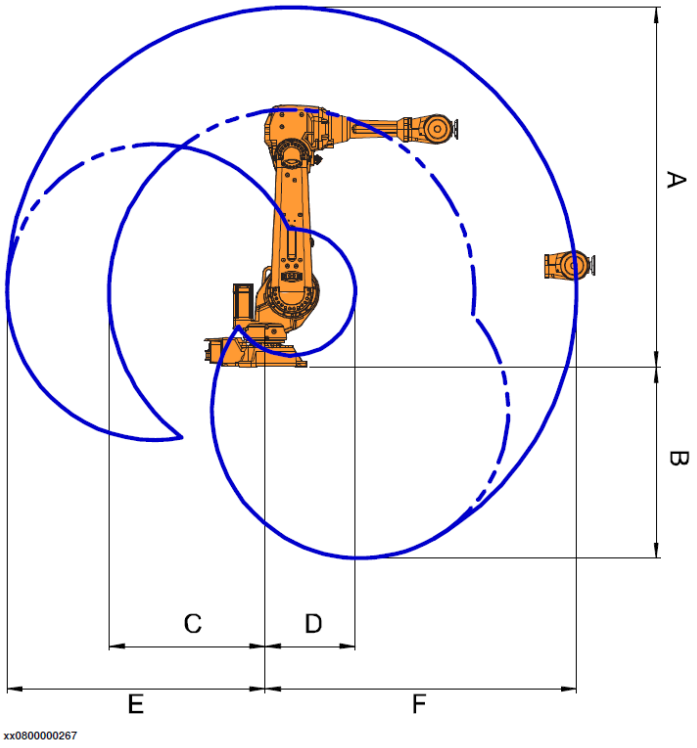


Figure 3-3 Wire spool connected to gun and Nertamatic plus machine.

3.2.2 Robot

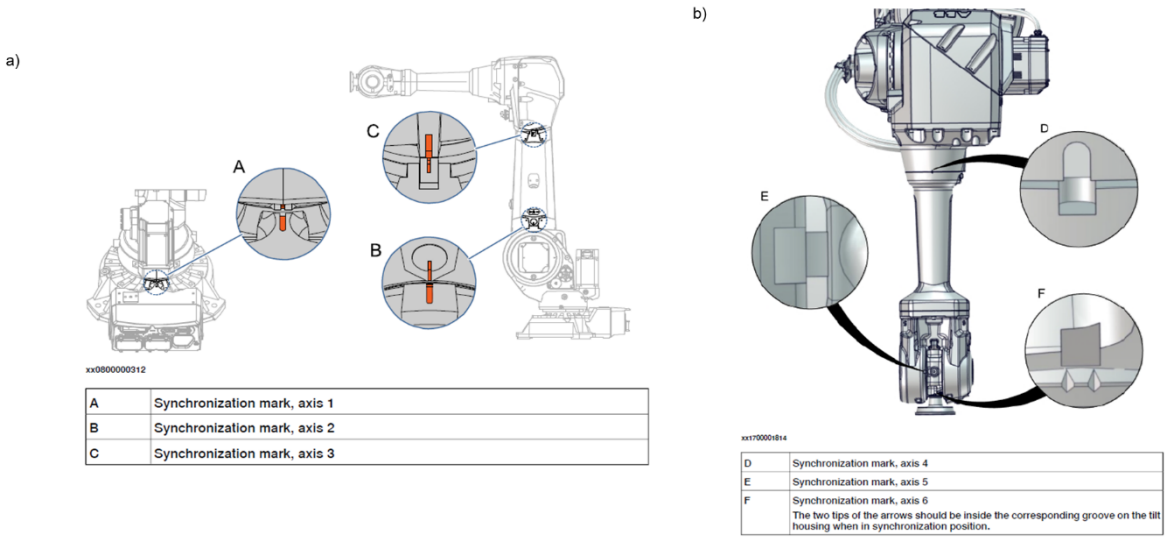
The utilized robot in this study was IRB 4600-45/2.05 (ABB, Västerås, Sweden) and the specifications were six-degree of freedom (DOF) plus two more degree of freedom provided from robotic table (IRBP A-500, ABB, Västerås, Sweden), capable of carrying 45 kg load, short cycle time, and ultra-wide working range. Figure **3-4** displays the working range of the robot as well as its singularity. The extreme positions of the robot arm are specified at the wrist center.

The calibration of robots was carried out by synchronization marks and synchronization position for the axes (**Figure 3-5**). Th robotic table (IRBP A-500, ABB, Västerås, Sweden), provides two degrees of freedom through tilting around the x axis and rotation around the z axis (**Figure 3-6**). In addition, the controller that used for the robot was IRC5, ABB, Västerås, Sweden. The robot is controlled by RobotStudio software with programming language of RAPID programming stand on C sharp coding.



Variant	Pos. A	Pos. B	Pos. C	Pos. D	Pos. E	Pos. F
IRB 4600 - 60/2.05	2371 mm	1260 mm	1028 mm	593 mm	1701 mm	2051 mm

Figure 3-4 Working range floor mounted robot (adopted from https://search.abb.com/library/Download.aspx?DocumentID=3HAC040585-001&LanguageCode=en&DocumentPartId=&Action=Launch).



A	Synchronization mark, axis 1
B	Synchronization mark, axis 2
C	Synchronization mark, axis 3

D	Synchronization mark, axis 4
E	Synchronization mark, axis 5
F	Synchronization mark, axis 6 The two tips of the arrows should be inside the corresponding groove on the tilt housing when in synchronization position.

Figure 3-5 Illustration of calibration synchronization method (adopted from https://search.abb.com/library/Download.aspx?DocumentID=3HAC040585-001&LanguageCode=en&DocumentPartId=&Action=Launch)

Figure 3-6 ABB robotic table, IRBP A-500, ABB, Västerås, Sweden (adopted from https://search.abb.com/library/Download.aspx?DocumentID=ROB10080EN_R3&LanguageCode=en&DocumentPartId=&Action=Launch).

3.2.3 Human Machine Interface (HMI)

Synchronization of welding machine and robot requires an interface called Human Machine Interface (HMI). The advantages of HMI can be summarized as i) controlling robot, welding machine and dust collector and ii) diagnosing the fault and errors taken place in the control loop. HMI permits to perform at a preliminary run known as dry run in which the motion of robot could be examined with disabling the welding deposition commands.

As described in Section 3.2.2, each program would be assigned to a number and by calling these numbers via HMI, we can run the welding gun and robot with the predefined setting. It can turn on and off the dust collector which served as collecting any toxic gaseous effluents and smoke producing during welding **(Figure 3-7**).

It is noteworthy to mentioning that the robot is placed in a cell which is divided into two sections and these two sections can collaborate with each other. Therefore, the HMI controls both sections of the cell and the rearm come into play once we plan to start a process (**Figure 3-8**)

Figure 3-7 The photo of HMI screen.

Figure 3-8 The photo of rearming the cell in HMI.

This section of HMI (rearming) checks the doors (open/close) and the door remains open under collaboration mode of two sections of the cell. On the other hand, in the mode singular run of a section, the door is maintained closed for safety purposes. The state of robot motors and I/O signals would be checked by this module as well. If there is an error, it permits the user to check the details of the signals.

3.2.4 Vision system

A 3D camera scanner was used as an area scanner in this study (3D-A5030, Cognex, Natick, Massachusetts, USA) and the technology of such camera is 3D light burst that enables the user to capture the photos rapidly (maximum 1000 ms). This technology creates a blue light pattern for scanning the object and providing the output of full field of view (FOV) 3D point cloud image with more than 1.5 million data points.

This camera is comprised of a light projection and two GigE vision cameras. A software, called A3D, would be used for adjusting variables of image capture. Another software that would be used for image and mesh processing is VisionPro. The format of output data that was used in

this research was 3D cloud of point in ASCI format. The working range specifications of the vision system is illustrated in **Figure 3-9** and the inert structure can be found in **Figure 3-10**.

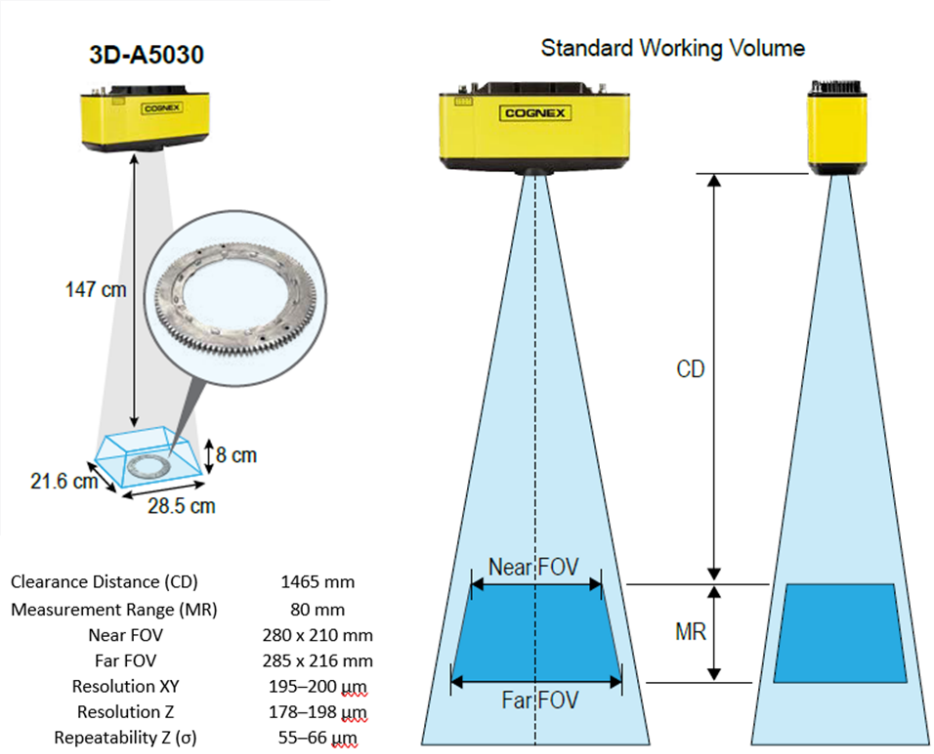


Figure 3-9 Working range and specification of the camera (revisualized from instructions of the Cognex device, Cognex Copr.).

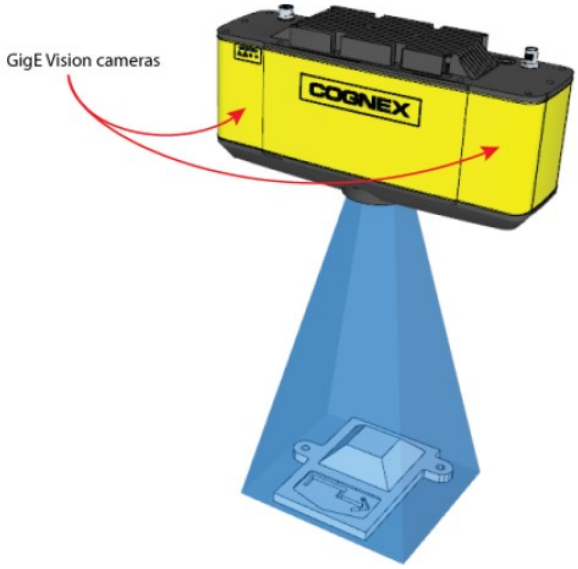


Figure 3-10 Insert structure of vision system. (revisualized from instructions of the Cognex device, Cognex Copr.).

3.3 Programming

3.3.1 Robot programming

Robot programming is included as two parts which are: i) offline programming that carried out by RobotStudio 2020, and ii) online control of the system that programmed via PC SDK, which allows end users add their own interface to the controller of the robot to develop custom applications.

In RobotStudio (offline programming), the codes are divided to three sections of i) control of robot motions, ii) control of welding gun, and iii) communication to HMI. One of the methods used for deposition of a part is the predefined "teach" method. In such a method, two points for the start and end point of a line would be defined for a robot and the line generated by these points are extended in y axis and z axis by serval sublayers (first in y-axis then z-axis for the next layer).

Another method to generate a path plan is to use machining in RoboDK. The robot and utilized table are al inbuilt simulated model in RoboDK and it is only required to call these models. However, the welding gun requires customization, and it should be designed in an individual CAD software (e.g. Catia) then it would be imported to RoboDK software. The next step is to initiate path plan by importing the CAD file in the format of .SDL on a proposed zone of deposition placed on the table. The software of RoboDK can detect singularity of robot and prevent them from taking place. In addition, the predefined barrier can be avoided upon activation of collision detector (codes can be found in Appendix 1).

As mentioned in Chapter 1, it is not feasible to correct dimensions and path planning in the course of the process in offline control. To make such correction possible in real time, it is required to implement the Application Programming Interface (API). APIs act like telephone lines and SDKs are like houses which permit to connect with ABB controller in order to control the robot in the soft real-time application. To set up the PC to robot controller, we can use two techniques: i) ethernet network and ii) direct connection to controller service port. The latter can be assigned by fixed or automatic IP. In addition, the PC SDK provides APIs for developing PC applications based on C# and all codes were developed by Visual Studio 2019 Software.

Equally important, it is mandatory to control the controller resources by a single client at a time in order to modify the RAPID data The two means of access are read-only (default) or write

access and Mastership is essentially used to have the leverage of write access. To secure the write access and Mastership simultaneously, it is crucial to run in automatic mode to maintain first come, first served priority without additional permission requirements. The reason for this is the priority for write access is given to flexPendant in the manual mode which is not favorable due to the consistent need to give permission for the access. **Figure 3-11** displays a part of Controller Application Programming Interface (CAPI) object model as a part controller API.

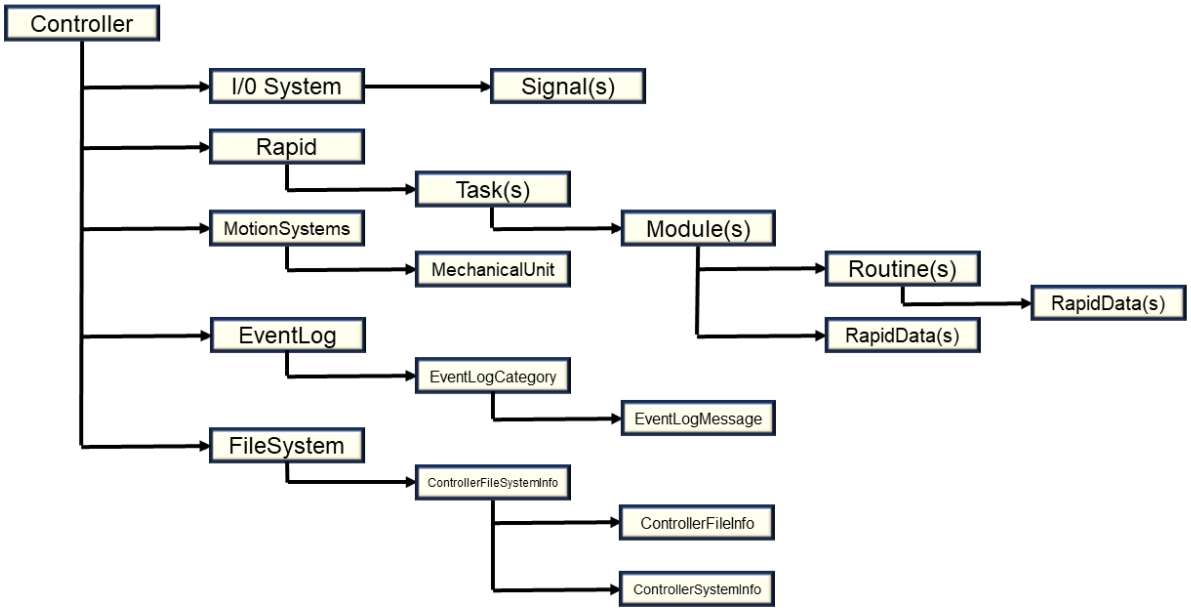


Figure 3-11 Components of CAPI object model (adopted from help center of ABB robot: https://developercenter.robotstudio.com/api/robotstudio/articles/How-To/Add-Ins/Creating-a-RobotStudio-Add-In.html).

Communication between clients and codes can be built by a GUI. The user interface of the application is illustrated in **Figure 3-12**.

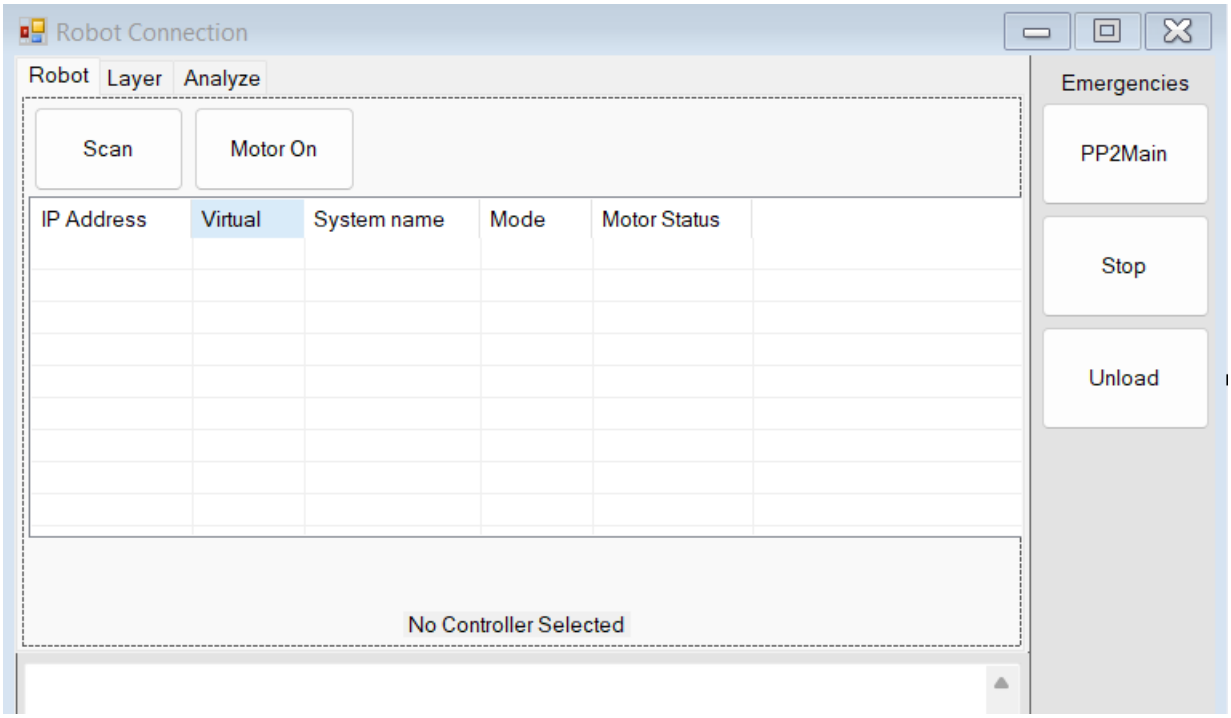


Figure 3-12 The user interface of the application.

The first step for programming is to add the required ABB libraries with the extension of .dll to the reference section of the C# programming. The required libraries are all provided in Appendix 1. The addition of these libraries enables us to search the network in order to find all available controllers, as well as operate the commands. It is essential to consider whether the selected controller is virtual or real, though the desirable one for programming is the real controller. Creation of an object per each component such as NetworkScanner, controller, and the structure of ABB robot library is mandatory due to the fact that C# programming is an object-oriented language.

To establish a link from a PC SDK software to the controller, it is required to utilize the Netscan feature within the Discovery domain. It is essential to generate a NetworkScanner entity and execute a scan command.

For the PC SDK to connect, the user either needs RobotStudio or Robot Communications Runtime set up on the PC that houses the PC SDK program. If RobotStudio is not present, one can set up Robot Communications Runtime from <PCSDK>\Redistributable\RobotCommunicationRuntime.

To determine available controllers in the network, a user can employ the NetworkScanner's functions like Scan, Find, GetControllers, and GetRemoteControllers. Utilizing a Controller object allows the user to tap into various sectors of the robot controller, such as I/O signals, RAPID, the file system, and log notifications.

To access a robot controller, the PC SDK software needs to sign in to the controller first. The Logon method's UserInfo parameter provides a DefaultUser attribute that can be utilized. Every robot system is typically set up with this user by default.

To obtain writing permissions for certain controller areas, the application must seek Mastership. The Rapid domain, encompassing tasks, programs, modules, routines, and variables in the robot system, is an example. The Configuration domain is another such area.

It is crucial to relinquish Mastership following a modification action. One method to achieve this is by means of the 'using' statement, leading to the Mastership object's automatic disposal after the block concludes. Alternatively, the Mastership can be released within a Finally block, which runs post the Try and Catch segments.

To interact with RAPID data, the user first needs to establish a RapidData object, using the data's declaration path in the controller as an argument. If one is unsure of this path, the SearchRapidSymbol function can be employed to find the RAPID data. We have two models to access the RAPID data which are Direct Access and Hierarchical Access.

Direct Access is more memory-efficient and quicker, making it the preferred choice if there is no subsequent need for task and module objects. For instance, to craft a RapidData object referring to the "reg1" instance within the USER module, one would use direct access.

Hierarchical Access: Should one require the task and module objects, a hierarchical approach might be more effective. The GetRapidData function is available in the Rapid, Task, and Module classes.

To implement the path plan on the selected robot, a user clicks on *load Robot file* in the layer, Tab. Consequently, a set of functions would be run and the *mod* file generated from RoboDK would be called and modified. As a result, a modified file labelled “.mod” would be sent directly

to the robot controller which has the required command for robot motions and controlling robot torch. The next step is to start the robot by clicking on *Start Robot* button.

3.3.2 Path analysis

Upon pressing the Start button (See **Figure 3-12**) upon establishing connection, the first layer would be deposited based on the first robot path planning assigned to the .stl file of the object. Consequently, the layer tab enables the user to adjust the table height, base height, and the number of the layer (**Figure 3-13**). Under the layer tab and the section of 'Select a Task', the selection of Load Robot File is feasible and would be applicable for the case of running the robot path file which is generated based on the .stl. The other available selection under this section is 'Load Correction File' and enables the user to run a file thar can carry out the correction of the path plan based on the previous layer analysis. The reason that it is possible to change the layer# is the fact that at some instances there is no need to rectify and correct the path plan and the next layer can directly proceed with a redo the previous layer. The intelligence of this program is established in a way that the robot could select the desired layer based on its number.

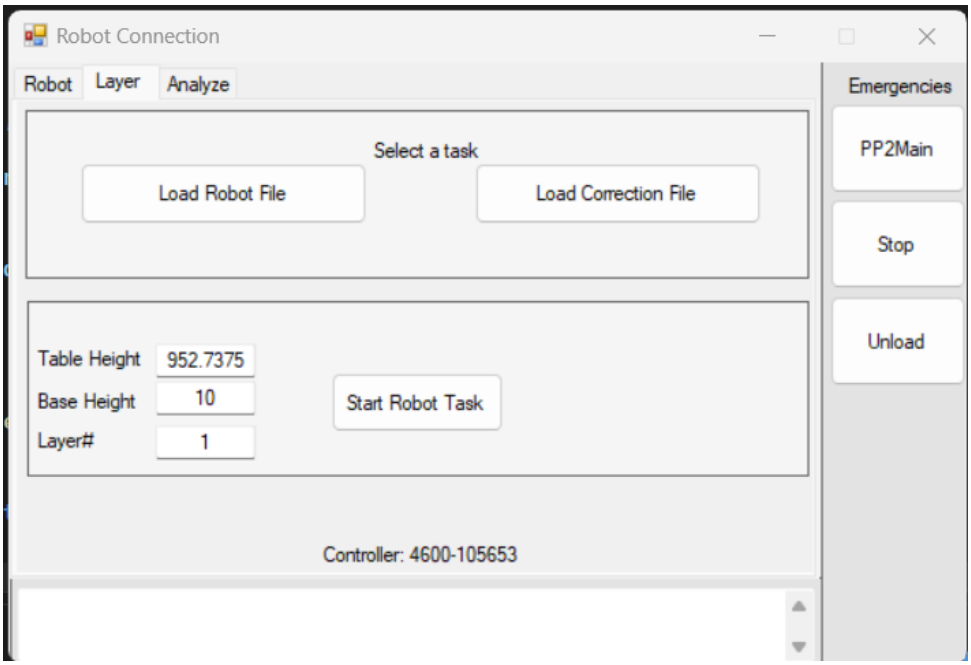


Figure 3-13 The layer of robot connection.

3.3.3 Mesh analysis

After the deposition of one layer, the robot should wait for the next command for the next layer. During this wait time, the user captures the picture of the surface of the material using the Cognex camera. With the selection of ‘Read Congex File’ under ‘Analyze’ Tab, the program would display the cavities and collisions in the windows illustrated in Error! Reference source not f ound.. The experimental results aid in the approximation of the range of height for detection of cavities and collisions.

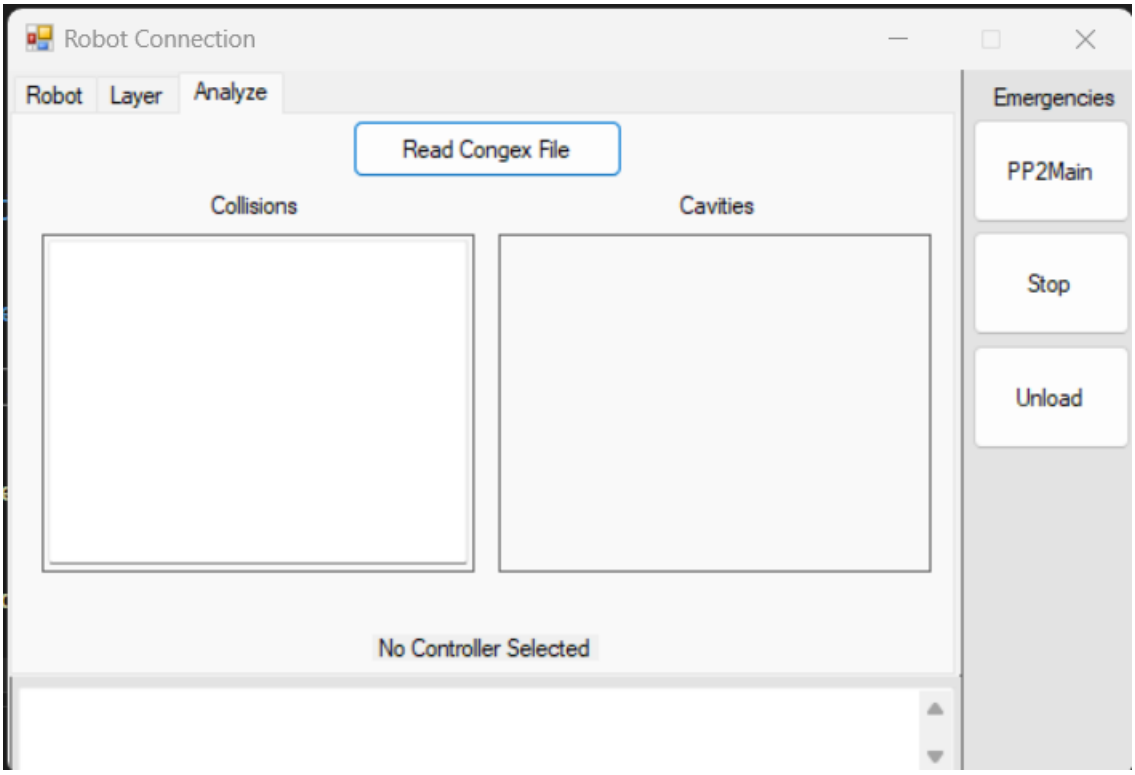


Figure 3-14 The interface for mesh analysis of each layer.

3.4 Summary

Chapter 3 describes the materials and methods utilized in this research, centering on the integration of hardware and software components for achieving online control with a 3D scanner. The chapter outlines the optimization of welding parameters, development of offline robot programming, and the transition to online control, ensuring continuous and precise operations. It also provides a comprehensive overview of the equipment used, including the welding machine, robot, and vision system, all of which are synchronized through a developed interface to facilitate seamless and autonomous path adjustments during the process.

Chapter 4

4 Developed Methodologies

4.1 Introduction

In this chapter, the aim is to present the developed adaptive correction strategy in WAAM process. The correlation between various process parameters with the cavities and irregularities are fully investigated. On-line identification of cavities and irregularities using a vision system is developed and new paths are generated based on the previous layer's topography data. The results of the microstructure tests and the analysis of the programming are presented. Additionally, the effectiveness of setting small beads in terms of their penetration into the base material is validated and their role in the fabrication of an object is discussed.

4.2 Adaptive Correction Strategy

4.2.1 Optimization of Welding Device Setting

One of the fundamental steps to ensure that the WAAM process yields superior quality is to optimize the welding device settings. Each bead size (small, medium, large) requires specific and distinct settings to maintain consistency throughout the deposition process. For example, small

beads need more precise control to prevent defects such as incomplete fusion and porosity. This control involves adjusting the wire feed speed and welding current to appropriate settings to avoid the aforementioned defects. In contrast, medium and large beads require optimized travel speeds and relatively higher energy inputs to ensure satisfactory bonding between layers. Achieving consistency and repeatability in the manufacturing process depends on identifying and documenting the appropriate settings. Such actions ensure that the end products meet the desired mechanical and structural properties.

4.2.2 Post-processing

Another crucial step is post-processing, which is as important as the deposition process itself. Converting CAD models into G-codes and data is a multi-stage process that directly impacts the quality of the end products. Open-source 3D printer software provides the opportunity to optimize and customize the post-processing stages required for the project. For instance, implementing a model using a line-directional path with a 90-degree angle and a 0.5 mm overlap results in a smooth and uniform surface finish. This approach enhances the aesthetic, dimensional quality, and surface integrity of the final product.

Additionally, the selection of Cura software is a pivotal stage in refining the post-processing phase. This software enables the definition and control of several parameters throughout the manufacturing process, offering greater flexibility. Equally important, it allows fine-tuning of parameters such as deposition speed, cooling rates, and layer thickness, as well as customization of post-processing steps. The structured approach of Cura software ensures meticulous planning and execution of each stage of post-processing, leading to superior quality finished products that meet industry standards. In this research, we used Cura software as a slicer to generate G-code for the robot path. Cura demonstrated superior performance compared to other slicers, such as Slic3R, Simplify3D, and Tinkercad.

4.2.3 Obtaining precise 3D point clouds

Precise and accurate dimensional data are criteria to guarantee that the manufactured parts meet design specifications. Obtaining detailed 3D point clouds is a technology that provides measurements of the length, width, and depth, which are crucial for quality verification and control. The tool used to obtain these 3D point clouds is the Cognex A Series 3D Camera, thanks to its superior ability to generate them. 3D scanners can obtain high-resolution data, aiding in the

detection of any deviations from target dimensions and permitting timely corrections. This level of accuracy is vital in industries like automotive and aerospace, where precision is paramount.

The next important stage is calibrating the 3D camera with the ABB robot, an essential step to provide seamless integration between the sensing and deposition systems. Appropriate calibration ensures that the robot accurately interprets the 3D camera data, leading to precise path planning and material deposition. The desired layer-by-layer build is a result of this successful integration. Successful calibration underscores the significance of compatibility between different components of the WAAM system, as any misalignment or error in calibration can compromise structural integrity and result in defects.

4.2.4 Calibration

To successfully guide a robot using a vision system, calibration plays a crucial role. This is especially important in welding processes, where the required accuracy is on the order of microns, as compared to the pick-and-place industries. Calibration ensures that the working spaces of the camera and robot are aligned by performing transformations and rotations on the vision system's data. The two most common methods of calibration are hand-eye and stereo vision systems.

4.2.4.1 Traditional Hand-Eye Calibration Methods

4.2.4.1.1 Classic Approach: Homogeneous Transformation Equation

The primary method for hand-eye calculations involves solving the following homogeneous equation [53]:

$$AX=XB \tag{4.1}$$

where:

X = Represent the unknown hand-eye transformation matrix

A = Represents the robot's forward kinematics

B = Represents the camera's extrinsic parameters.

After applying these formulas, we obtain a set of linear equations resulting from the decoupling of rotation and translation. Instabilities may arise due to the interaction of translation and rotation matrices during multiplication [54,55].

4.2.4.1.2 Low-Cost Hand-Eye Calibration Method

One of the advantages of the hand-eye calibration method is its low cost. The aim of this method is to align the frames of the object and camera with the robot frame (the base coordinate frame). This process involves calculating the robot-world transformation using a point laser on the robot end-effector, followed by determining the hand-eye transformation matrix [54].

For calibration, the following steps must be followed to calibrate each piece of equipment and derive the required parameters:

1. Camera Calibration: The intrinsic and extrinsic parameters of the camera are obtained using the Caltech Matlab toolbox. A calibration object with a grid pattern is used for this purpose.
2. Coordinate System Convention: The right-hand rule is used to define the coordinate systems for the world, camera, robot wrist flange, and robot base. This step ensures consistency in the convention, and transformation matrices between the frames are determined.
3. Robot-World Calibration: The robot-world transformation matrix is derived by moving the robot to three touchpoints on the calibration object.
4. Hand-Eye Calibration: This step involves using the robot-world transformation matrix and images captured by placing the robot in different positions. The rotational and translational components are averaged to obtain the final transformation matrix [31,32]. The accuracy of this method is 1mm.

4.2.4.1.3 Simultaneous Calibration of Stereo Vision System

The proposed method for the simultaneous calibration of a stereo vision system begins with the concurrent calibration of both the vision system and the welding robot. This method is ideal for industrial applications due to its high accuracy.

The methodology for this approach is as follows:

1. Camera Calibration: The intrinsic and extrinsic parameters of both cameras are obtained using the same Matlab toolbox. Calibration is performed using a patterned platform for both cameras.
2. Stereo Calibration: The transformation between the left and right cameras, known as stereo calibration, is calculated using 3D triangulation of points in space. To execute the calculations in 3D, the width, length, and height of the points in the world frame are determined based on pixel coordinates from the stereo camera images.
3. Radial Distortion Correction: By removing radial distortion from the pixel values, the accuracy is significantly improved, enhancing the precision of the 3D location data.

The remaining steps are the same as those used in the calibration with a single camera.

4.2.5 Path Correction Programming

The final and most crucial stage is developing path correction software using ABB robot's APIs, which represents a significant advancement in adaptive manufacturing. This path correction software allows for real-time adjustments to the path plan, addressing any discrepancies that may occur during the deposition process. By enabling on-the-fly corrections, the software can ensure that each layer is accurately deposited, enhancing the overall quality and reliability of the manufactured products. This capability offers a new perspective and tool on producing complex geometries and large-scale components, particularly where even minor imperfections and deviations can have a substantial impact on the final product.

To assess the overlap between the beads, an experiment was designed to understand how this overlap affects the integrity of the deposited part and to what extent optimal conditions can be found to meet desirable dimensions and integration. In this experiment, the deposition of five, six, and seven beads was assessed (Figure 4-1). The characterization metrics were determined as follows: top bead, penetration, top-bottom, cavity, bead width, max top bead, bead spacing, max penetration, and welding area width. The definitions of these key metrics are presented below:

i) Top-Bottom: Measurement of the total height from the top of the top bead to the bottom of the bottom bead.

ii) Cavity: Size of any cavities present within the deposition.

iii) Bead Width: Measurement of the width of individual beads deposited.

iv) Max Top Bead: Maximum height of the top bead.

v) Bead Spacing: Distance between adjacent beads.

vi) Max Penetration: Maximum depth of penetration.

vii) Welding Area Width: Width of the welding area.

In this chapter, the results of microstructure tests on the deposited material are presented, highlighting the impact of bead size on penetration into the base material and its role in the fabrication process. These findings were crucial in optimizing the welding process parameters for our programming.

As illustrated in **Figure 4-1**, the bead width for five beads ranges from 4.4 to 5.8 mm. This variability in penetration and bead height suggests potential inconsistencies in the deposition process. Additionally, the presence of cavities indicates the areas with poor material fusion.

In contrast, the deposition of six beads shows more consistency with a slightly lower average bead width. The assessments of penetration and top bead height demonstrate better control throughout the deposition process and significantly fewer cavities.

For the deposition of seven beads, while the bead width and penetration variability increase, the presence of cavities is noticeably reduced. This suggests that using a higher number of beads may effectively minimize void formation.

To summarize, there is a trade-off with increased variability in bead width and penetration when using seven beads. Optimizing the deposition pattern and the number of beads is crucial for achieving consistent results and superior quality.

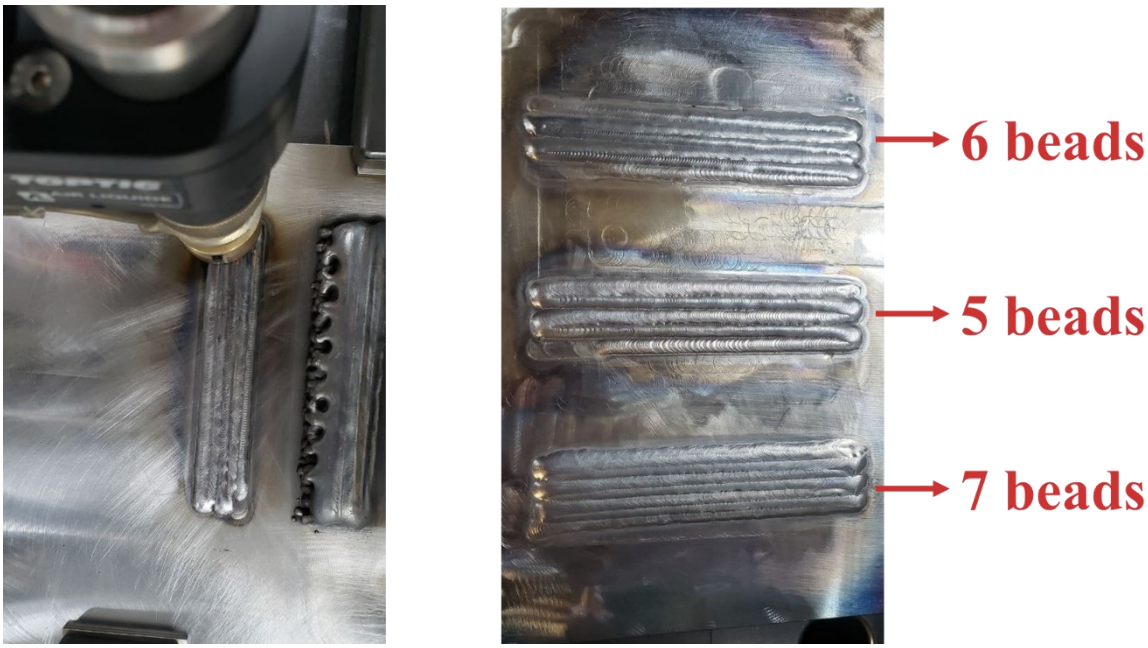


Figure 4-1 Deposition process, focusing on the number of beads deposited. Left: illustration of six-bead multiline multi-layer deposition. Right: comparison of number of beads.

The effect of different slicers was studied for varying numbers of beads (Table 4-1), using UtilMaker Cura and Simplify 3D software. For five beads, it was observed that the sliced width and the deposited width increased when using the Cura slicer, indicating better integration for this study. The number of preheat passes refers to the number of times the base material is heated without feeding wire into the melting pool. Additionally, the temperature values in Tables 4-1 to 4-5 represent the temperature of the base material at the start of the object's deposition.

Table 4-1 Effect of slicer on bead width and penetration.

Slicer	Number of Beads	One bead width (mm)	Cad Sample Width(mm)	Sliced Width (mm)	Deposited Width (mm)	Number of Preheat	Temp (C°)
Simplify3D	5	5-6	28	15.36	22.5	2	200
Cura	5	5-6	28	20	25-27	2	200
Cura	6	5-6	28	20	26-27	2	240
Cura	7	5-6	28	21	27-28	2	300

Figures 4-2 and **4-3** highlight the importance of consistency in bead width for achieving uniform material deposition. The quality of layer bonding is indicated by the height of the top bead and the depth of penetration. In the case of five beads, cavities are present, suggesting that this configuration may not effectively fill all voids. **Figure 4-3**, which illustrates the depth and width of penetration, shows that in some areas, the material may not have fully bonded with the neighboring layers. Additionally, the characteristics of each bead concerning key metrics for the five-bead configuration are presented in **Table 4-3.**

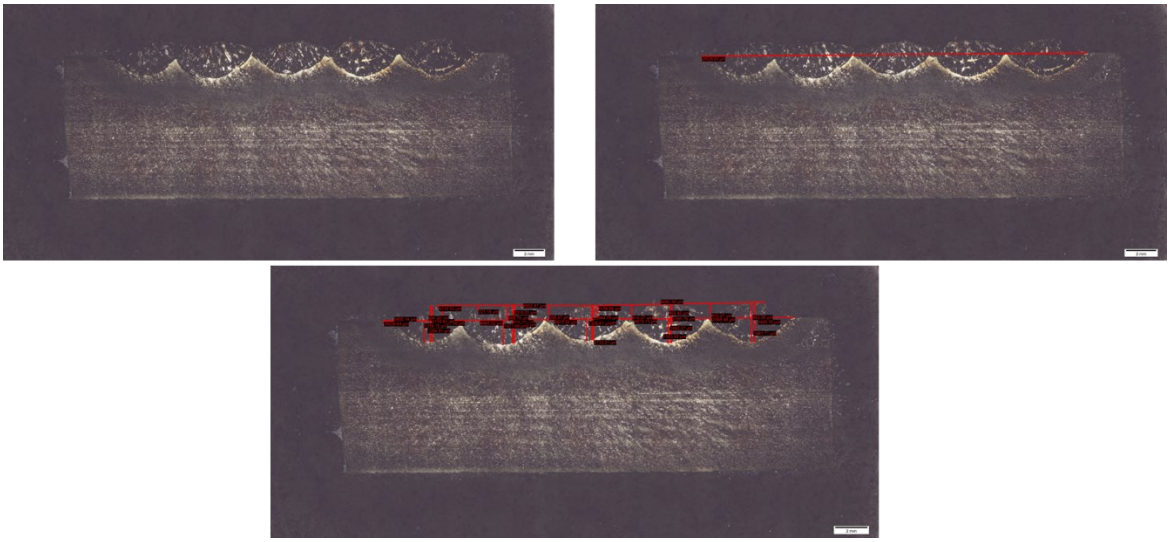


Figure 4-2 Microscopic image of five-bead microstructure. Beads can be counted from left to right in the bottom figure. Refer to Table 4-3 for dimensions.

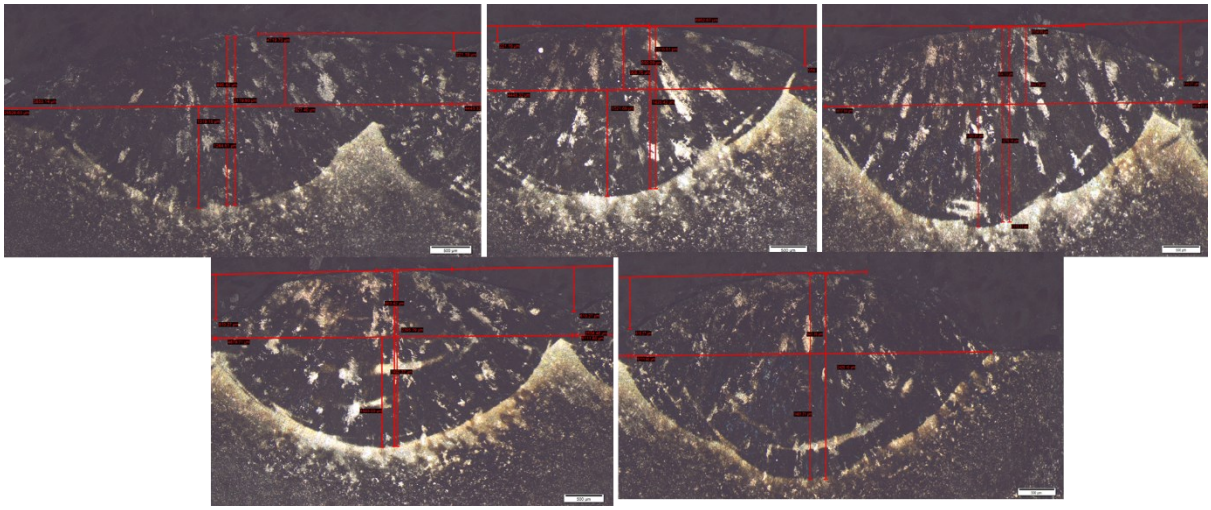


Figure 4-3 Depth and width of penetration of five-bead microstructure (please see Table 4-3 for the dimensions).

Figures 4-4 and **4-5** prove the findings presented in Figure 4-1 regarding the six-bead deposition pattern. These figures provide both quantitative (Table 4-4) and qualitative evidence that the top bead height and penetration are significantly improved compared to the five-bead microstructure. As shown in Table 4-2, the average bead width decreased from 5.04 mm for the five-bead structure to 4.42 mm for the six-bead structure. This improvement is accompanied by a substantial enhancement in the bonding between layers, leading to a more cohesive structure.

Table 4-2 Comparison of key parameters across bead configurations.

Bead Configuration	Avg Bead Width (mm)	Max Bead Width (mm)	Min Bead Width (mm)	Std Dev Bead Width (mm)	Average Penetration (mm)	Max Penetration (mm)	Min Penetration (mm)	Std Dev Penetration (mm)
5 Beads	5.04	5.85	4.44	0.46	1.39	1.52	1.28	0.09
6 Beads	4.42	6.75	3.70	1.07	1.00	1.43	0.67	0.10
7 Beads	3.72	6.26	3.20	1.01	1.17	1.70	0.93	0.08

The analysis reveals a significant decrease in the presence of cavities, which indicates that the six-bead deposition pattern is more effective in achieving a uniform fill. This pattern ensures that the material is more evenly distributed and better bonded, reducing the likelihood of weak points

within the structure. The six-bead approach demonstrates a superior balance between deposition efficiency and quality, highlighting its effectiveness in producing a more robust and reliable material deposition.

These results suggest that the six-bead pattern is optimal for achieving high-quality, consistent results, making it a preferable choice over the five-bead configuration. The improved bonding and reduced cavities are crucial for applications requiring strong, durable materials, underscoring the advantages of the six-bead deposition pattern in practical use cases.

Table 4-3 The characteristics of five-bead microstructure for each bead

5 beads	Bead width (mm)	Top Bead (mm)	Penetration (mm)	Top-Bottom (mm)	Cavity (mm)	Max Top Bead (mm)	Max Penetration (mm)	Welding Area Width (mm)	Bead Spacing (mm)
Bead 1	5.85	0.89	1.28	2.18	0.22	0.93	1.32	25.53	0.05
Bead 2	4.44	0.89	1.42	2.31	0.56	0.91	1.52		
Bead 3	4.92	0.83	1.28	2.11	0.61	0.84	1.33		
Bead 4	4.88	0.90	1.49	2.39	0.61	0.90	1.50		
Bead 5	5.11	0.94	1.48	2.42	-	0.94	1.48		

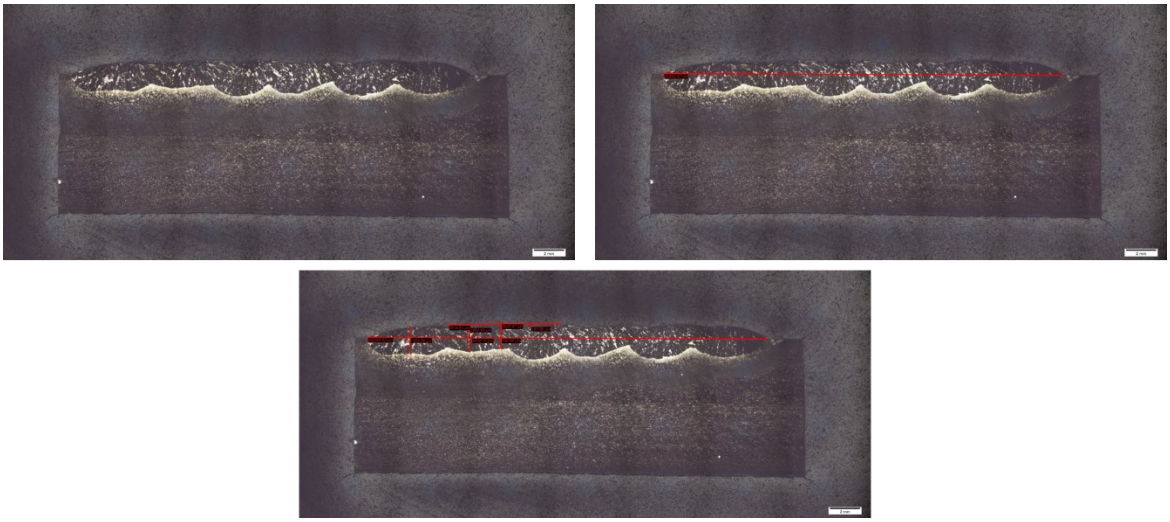


Figure 4-4 Microscopic image of six-bead microstructure. Beads can be counted from left to right in the bottom figure. Refer to Table 4-4 for dimensions.

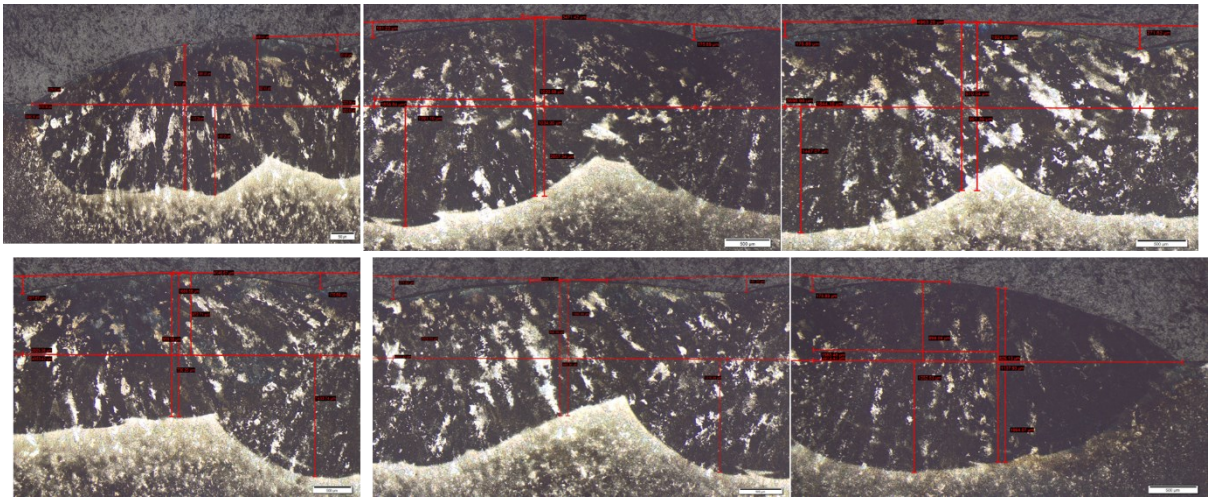


Figure 4-5 Depth and width of penetration of six-bead microstructure (please see Table 4-4 for the dimensions).

Figures 4-6 and **4-7** illustrate that the seven-bead microstructure introduces increased variability in bead width, which can significantly affect the overall uniformity of the deposition. This increased variability in bead width can lead to inconsistencies in the final structure, which may impact its mechanical properties and performance. Additionally, the bead penetration and height exhibit more variability compared to the five- and six-bead configurations (see Table 4-2). This

inconsistency in bead penetration and height raises concerns about maintaining consistent layer bonding, which is crucial for the structural integrity and durability of the material.

Despite these challenges, there are notable benefits to the seven-bead deposition pattern. The presence of cavities is markedly reduced, indicating improved material fusion and better filling of voids. This reduction in cavities suggests that the seven-bead configuration can achieve a more thorough and complete material deposition, potentially leading to fewer weak points and a stronger overall structure. This enhanced material fusion is a significant advantage, as it can improve the performance and longevity of the deposited material.

The key metrics of the seven-bead deposition pattern, as detailed in **Table 4-5**, support these microscopic observations. These metrics provide quantitative evidence of the increased variability in bead width, penetration, and height, as well as the reduced presence of cavities. Together, these findings highlight both the benefits and challenges of the seven-bead configuration. While it offers improved material fusion and reduced cavities, the increased variability in bead dimensions underscores the need for careful optimization and control to ensure consistent and high-quality deposition.

Table 4-4 The characteristics of six-bead microstructure for each bead.

6 beads	Bead width (mm)	Top Bead (mm)	Penetration (mm)	Top-Bottom (mm)	Cavity (mm)	Max Top Bead (mm)	Max Penetration (mm)	Welding Area Width (mm)	Bead Spacing (mm)
Bead 1	6.75	0.78	1.10	1.88	0.21	0.89	1.17	26.49	0.04
Bead 2	4.05	0.96	0.73	1.69	0.17	0.97	1.43		
Bead 3	3.70	0.97	0.95	1.92	0.27	0.97	1.45		
Bead 4	4.05	0.90	0.67	1.57	0.18	0.90	1.32		
Bead 5	3.71	1.02	1.03	2.05	0.17	1.02	1.39		
Bead 6	3.97	0.83	1.14	1.96	-	0.90	1.25		

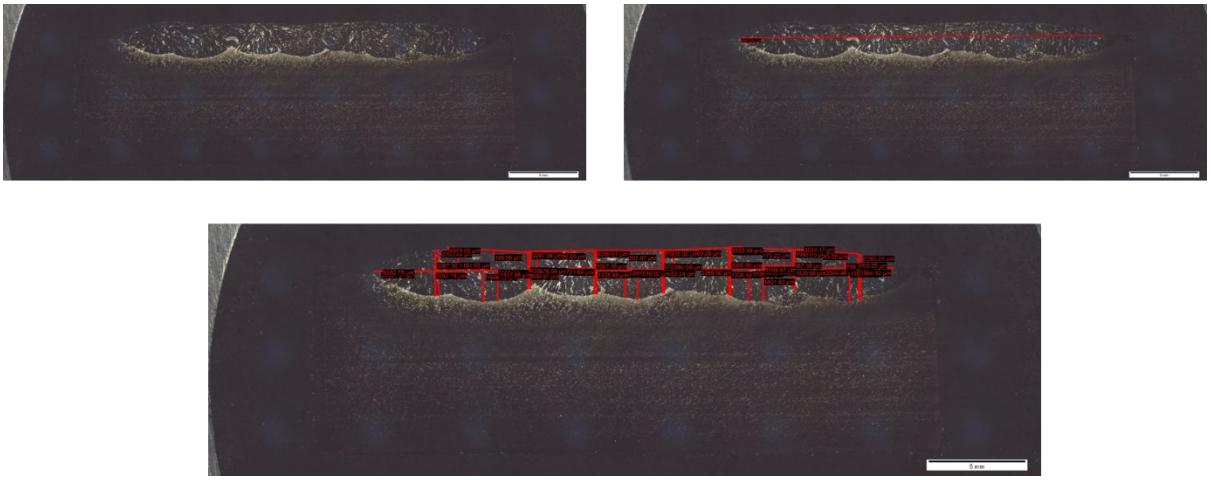


Figure 4-6 Microscopic image of the seven-bead microstructure. Beads can be counted from left to right in the bottom figure. Refer to Table 4-5 for dimensions

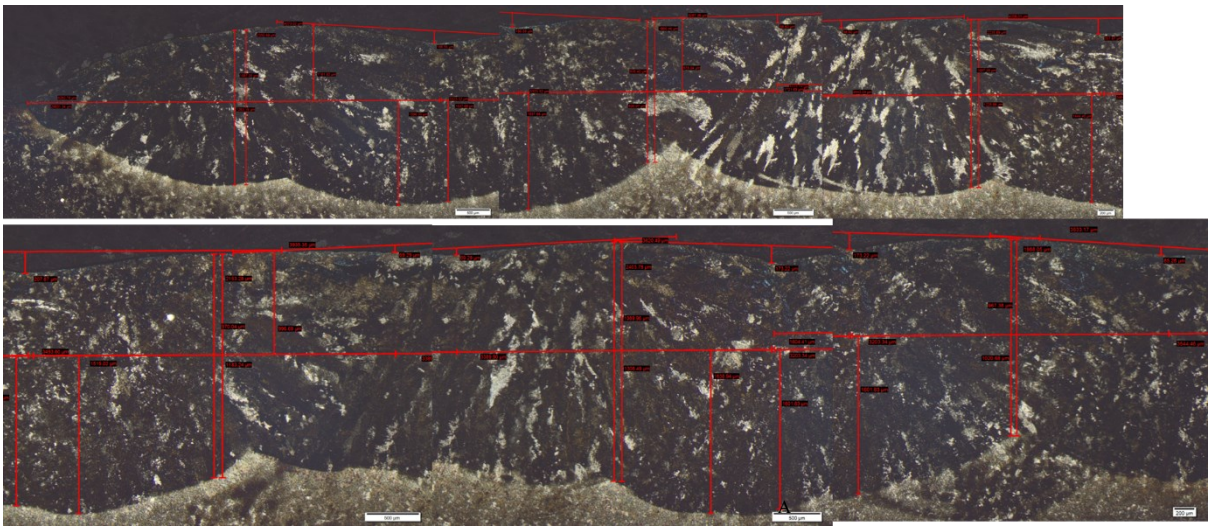


Figure 4-7 Depth and width of penetration of seven bead microstructure (please see Table 4-5 for the dimensions).

Table 4-5 The characteristics of seven-bead microstructure for each bead.

7 beads	Bead width (mm)	Top Bead (mm)	Penetration (mm)	Top-Bottom (mm)	Cavity (mm)	Max Top Bead (mm)	Max Penetration (mm)	Welding Area Width (mm)	Bead Spacing (mm)
Bead 1	6.26	1.09	1.26	2.35	0.19	1.16	1.60	26.83	0.03
Bead 2	3.32	0.93	0.93	1.86	0.10	0.98	1.54		
Bead 3	3.44	1.01	1.22	1.01	0.21	1.01	1.44		
Bead 4	3.49	0.97	1.183	2.15	0.07	0.99	1.52		
Bead 5	3.39	1.09	1.31	2.40	0.17	1.09	1.64		
Bead 6	3.20	0.97	1.02	1.99	0.08	0.97	1.60		
Bead 7	3.54	0.70	1.56	2.26	-	0.79	1.70		

4.3 On-Line Height Adjustment and Mesh Analysis

In this section we discuss the steps that are necessary to take for analyzing each deposited layer in order to modify the next layer path plan if necessary. For this purpose, we used the API of RoboDK and the program was developed in C#.

4.3.1 Flowchart of Programming

Flowchart of Programming

The flowchart of the programming and its components is illustrated in **Figures 4-8** to **4-10**. Pressing the start button initiates the network search block, which begins searching for the robot controller. Once the controller is located and a connection is established, the uploaded task is executed. In the layer manager block, the number of layers to be deposited can be controlled. The Cognex scanner block accesses the point cloud data, and by executing the extracting points block, the mesh processing is managed. The cluster block identifies cavities, and this data is sent through the path correction block to manually modify the next layer's data.

Convex Hull

A convex hull is the smallest convex polygon that can enclose a set of points in a plane. If you imagine stretching a rubber band around a set of points, the shape that the band takes is the convex hull. Mathematically, it is defined as the minimal convex set that contains all the points.

Finding the borders of a point cloud using the concept of a convex hull is a common technique in computational geometry. A point cloud is a set of data points in a coordinate system, typically representing the external surface of an object. The convex hull is used to identify the outermost boundary that encloses all the points in the point cloud.

Characteristics of a Convex Hull

1. Convexity: A shape is convex if, for any two points within the shape, the line segment connecting them lies entirely within the shape. The convex hull of a set of points is always convex by definition.

2. Minimality: The convex hull is the smallest convex shape that can contain all the points in the set. No other convex shape that contains the points can have a smaller perimeter or area.

Applications of Convex Hull

Convex hulls are used in various fields and for numerous applications:

1. Computer Graphics: Used for object recognition and collision detection.

2. Geographic Information Systems (GIS): Used for finding the boundary of a geographic region.

3. Robotics: Helps in path planning and motion planning for robots.
4. Statistics: Used in data analysis and pattern recognition.
5. Computational Geometry: Fundamental for solving many problems related to geometry.

Using this algorithm, we were able to draw polygons with arbitrary shapes, allowing us to remove extra points and noise from the display window.

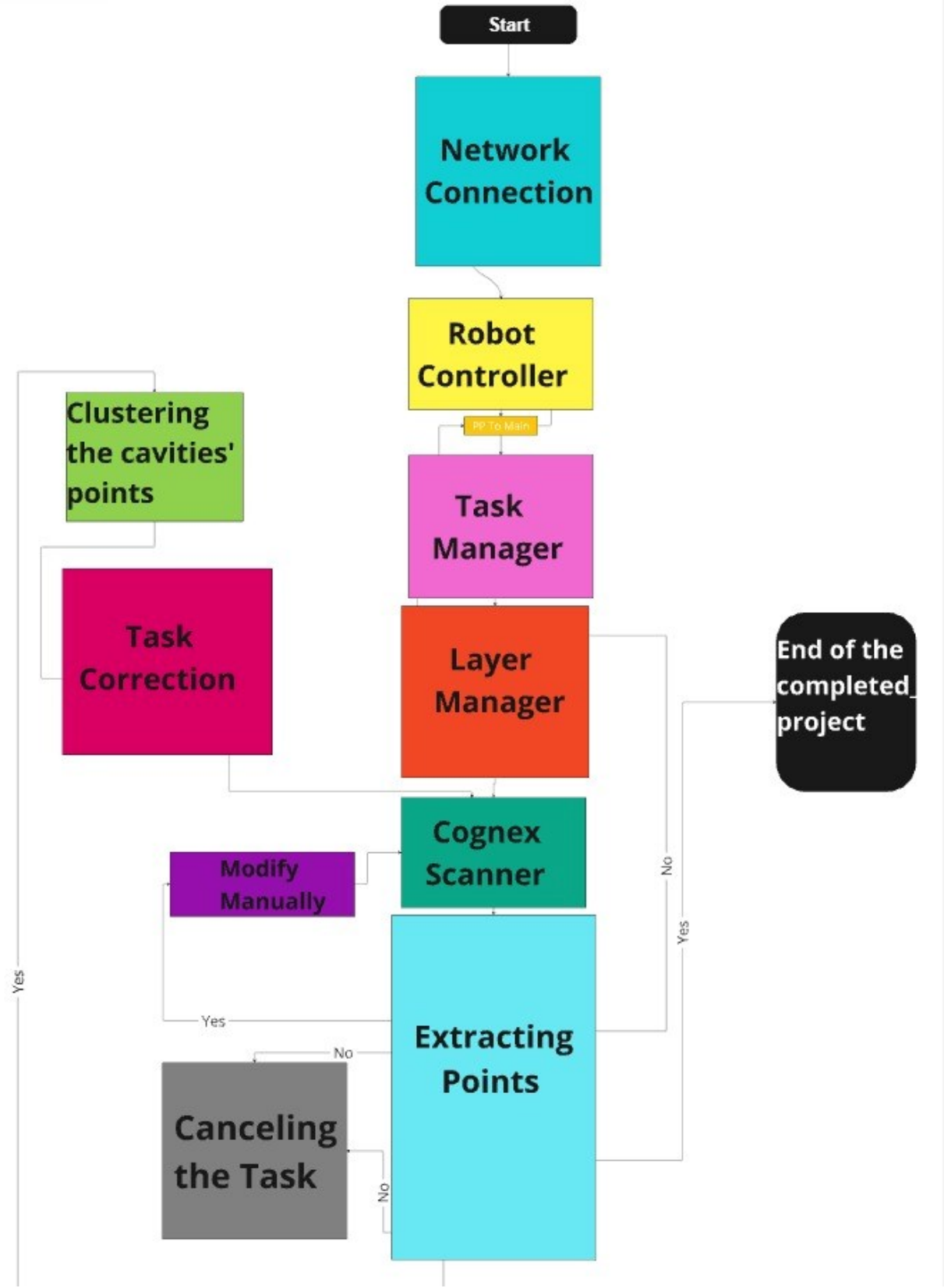


Figure 4-8 Flowchart of programming

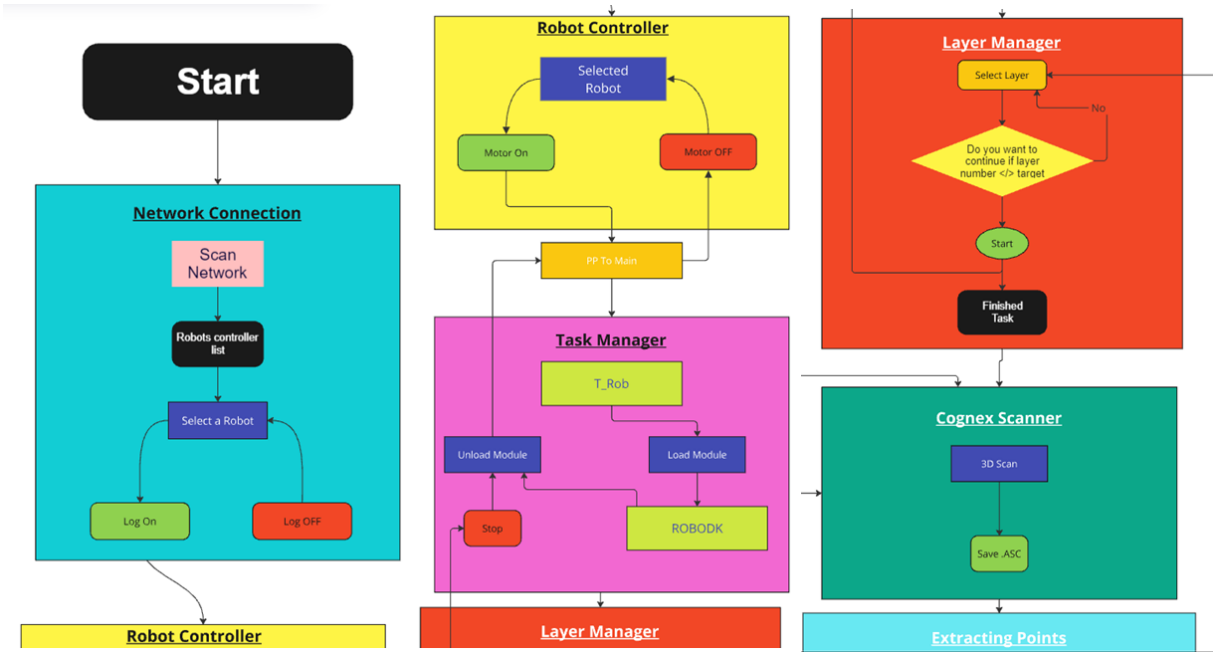


Figure 4-9 Zoom in flowchart of programming part 1.

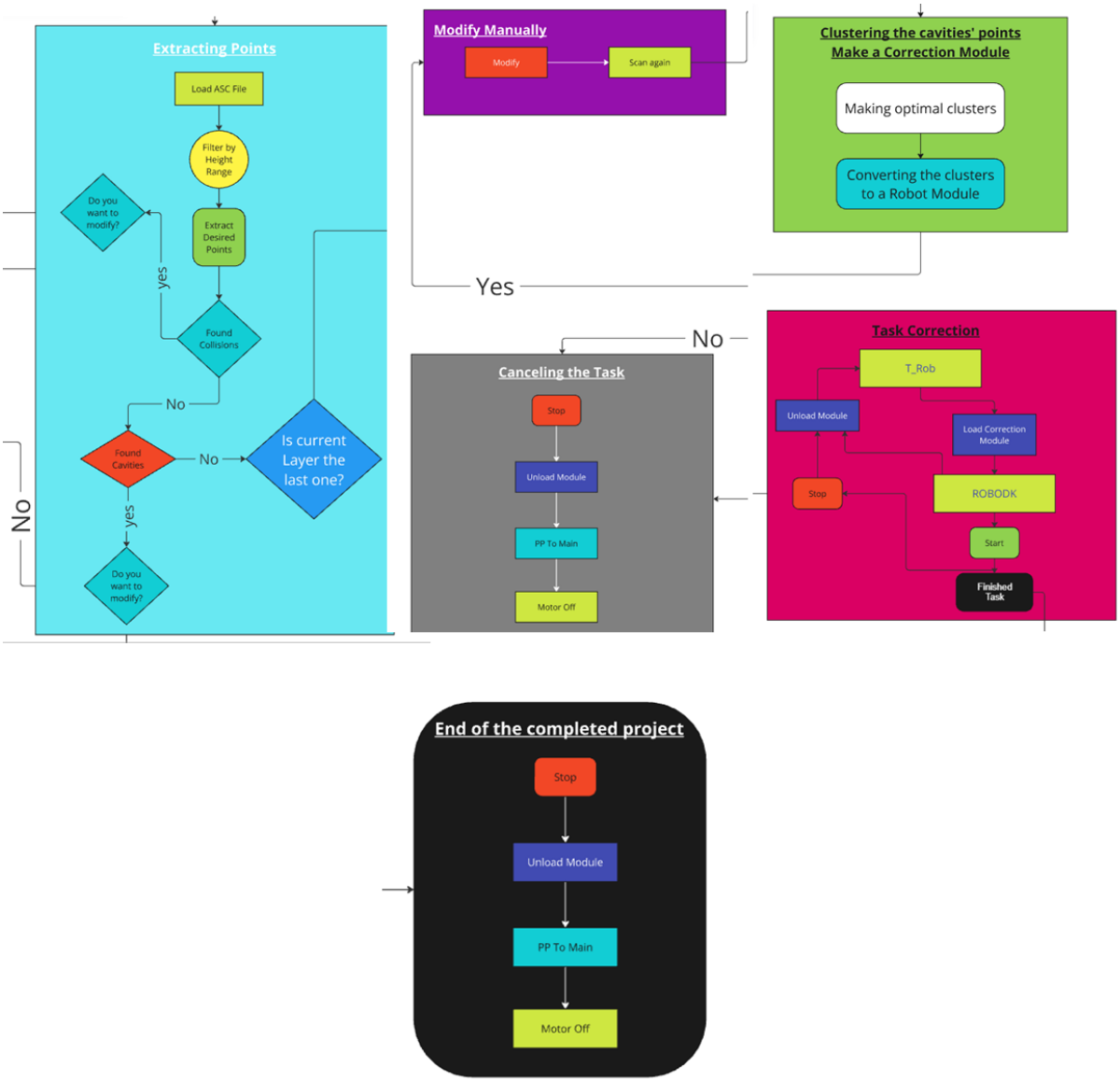


Figure 4-10 Zoom in flowchart of programming part 2.

We start to read the points coordinates from an .asci file and save them in two point data. The backup one is to be used as a reference for translating the points position after the mesh analysis has been completed.

Picturewidth-$Y_{\text{of point clouds}}$

PictureHeight-$X_{\text{of point clouds}}$

Picture width and height refers to the visualization screen of the C# program

Finding the width and length of the point cloud

Variables and Initialization

- **SumX**: Sum of all x-coordinates of the points.
- **SumY**: Sum of all y-coordinates of the points.
- **MinX**: Minimum x-coordinate value among the points.
- **MaxX**: Maximum x-coordinate value among the points.
- **MinY**: Minimum y-coordinate value among the points.
- **MaxY**: Maximum y-coordinate value among the points.
- **lengthX**: Length of the bounding box in the x-direction.
- **lengthY**: Length of the bounding box in the y-direction.

Loop to Process Points

The code iterates through each point in the list and updates the variables:

1. **Sum of Coordinates**:

$$SumX = \sum_{i=1}^{n} x_i \tag{4.2}$$

$$SumY = \sum_{i=1}^{n} y_i \tag{4.3}$$

where n is the number of points, x_i and y_i are the x and y coordinates of the i-th point, respectively.

2. **Minimum and Maximum Coordinates**:

$$\text{Min } X = \min(x_1, x_2, \dots, x_n) \tag{4.4}$$

$$Max\ X = \max(x_1, x_2, \dots, x_n) \tag{4.5}$$

$$\text{Min } Y = \min(y_1, y_2, \dots, y_n) \tag{4.6}$$

$$Max\ Y = \max(y_1, y_2, \dots, y_n) \tag{4.7}$$

Length Calculation

The length of the bounding box in the x and y directions is calculated using the distance formula:

1. **Length in the x-direction**:

$$lengthX = \sqrt{(MaxX - MinX)^2} \tag{4.8}$$

Simplifying this, since squaring and then taking the square root are inverse operations:

$$lengthX = |MaxX - MinX| \tag{4.9}$$

2. **Length in the y-direction**:

$$lengthY = \sqrt{(MaxY - MinY)^2} \tag{4.10}$$

Similarly:

$$lengthY = |MaxY - MinY| \tag{4.11}$$

Transformation and Scaling of Points Based on Bounding Box Coordinates

We have to scale and transform the positions of points within a bounding box, so that the scan of any objects with any sizes would fit the visualization screen and would be placed in the middle of the screen. This transformation is based on the distances of the points from the minimum x and y coordinates in the dataset. The process ensures that the points are appropriately adjusted relative to a defined picture or drawing area. Below, we describe the mathematical derivation and the reasoning behind each step in the code.

Variables and Initialization

Before diving into the loop that processes each point, several key variables are initialized:

- **MinX, MinY**: These represent the minimum x and y coordinates among all points in the dataset. These values serve as reference points for calculating distances.
- **lengthX, lengthY**: These denote the lengths of the bounding box in the x and y directions, respectively. These lengths are critical for scaling the points proportionally.
- **PicItemSizeWithd, PicItemSizeHeight**: These are the dimensions of the picture or drawing area where the transformed points will be plotted.

Processing Points

The core of the transformation process occurs within a loop that iterates over each point in the dataset. For each point, the following steps are executed:

1. **Transform Coordinates**: The initial coordinates of the point are transformed relative to the picture size. This step ensures that the points are scaled correctly within the new coordinate system.

$$LengthPointX = \sqrt{(MinX - p_x)^2} = |MinX - p_x| \tag{4.12}$$

where p_x is the x-coordinate of point $\mathcal{P}$.

2. **Calculate X Ratio**:

$$XRatio = \frac{lengthX \times 1.1}{lengthX - LengthPointX} \tag{4.13}$$

This ratio scales the x-coordinate based on its distance from the minimum x-coordinate. The reason for 1.1 multiplication is that, so the points will not attach to the edge of the screen, and they would be drawn by a distance.

3. **Calculate Length Point Y**:

$$LengthPointY = \sqrt{(MinY - p_y)^2} = \left|MinY - p_y\right| \tag{4.14}$$

where p_y is the y-coordinate of point $\mathcal{P}$.

4. **Calculate Y Ratio**:

$$YRatio = \frac{lengthY \times 1.1}{lengthY - LengthPointY} \tag{4.15}$$

This ratio scales the y-coordinate based on its distance from the minimum y-coordinate.

5. **Transform Point Coordinates**:

$$p'_x = \frac{PicItemSizeWithd}{XRatio} + \frac{lengthX}{10} \tag{4.16}$$

$$p'_y = \frac{PicItemSizeHeight}{YRatio} + \frac{lengthY}{10} \tag{4.17}$$

where p'_x and p'_y are the new coordinates of the point $\mathcal{P}$.

Transformation and Scaling of Points Based on Bounding Box Coordinates

The Points from IntersectionFinder.NewLines, which contains Vector4 elements representing coordinates, are transformed into a new coordinate system. The goal is to transform these points

into a new coordinate system by scaling and translating them based on specified dimensions (PicItemSizeWithd, PicItemSizeHeight, lengthX, and lengthY) and minimum coordinate values (MinX, MinY). Below, we detail the mathematical derivations and the logic behind each step in the code, providing a comprehensive understanding suitable for inclusion in a thesis.

Variables and Initialization

Before processing each point, several key variables are initialized:

- **MinX, MinY**: These variables represent the minimum x and y coordinates among all the points in the dataset. These values serve as reference points for calculating distances from the boundaries of the bounding box.
- **lengthX, lengthY**: These denote the lengths of the bounding box in the x and y directions, respectively. These lengths are crucial for proportional scaling of the points.
- **PicItemSizeWithd, PicItemSizeHeight**: These are the dimensions of the picture or drawing area where the transformed points will be plotted. They determine the new coordinate system's boundaries.

Loop to Process Points

The core of the transformation process occurs within a loop that iterates over each point in the dataset. For each Vector4 point $\mathcal{P}$ in IntersectionFinder.NewLines, the following steps are executed:

1. **Transform Coordinates**: The initial coordinates of the point are transformed relative to the picture size. This step ensures that the points are correctly positioned within the new coordinate system.

$$p_x = PicItemSizeWithd - p_y \tag{4.18}$$

$$p_y = PicItemSizeHeight - p_x \tag{4.19}$$

$$p_z = PicItemSizeWithd - p_w \tag{4.20}$$

$$p_w = PicItemSizeHeight - p_z \quad (4.21)$$

Where p_x, p_y, p_z are the coordinates of the points transformed from the original Vector4 point. This transformation flips the coordinates based on the dimensions of the picture area.

The remaining calculations are as same as in the previous section.

4.3.2 Mesh Analysis Results

This subsection outlines the steps for identifying irregularities and cavities in the deposited layer using a developed program. The program preprocesses scanned images to diagnose surface features and applies a surface analysis algorithm to detect deviations, marking peaks as irregularities and troughs as cavities. It measures their dimensions, ignoring cavities smaller than 0.5 mm² while locating irregularities of any size. The program then compares the findings with welding reference standards to ensure accuracy. Finally, it generates a detailed report, including visual representations, statistical summaries, and recommendations for improving deposition quality.

Figure 4-11 Illustrates a graphical user interface defined by the cavity clustering software used in the WAAM process. The interface highlights points in red that are below a predefined height and were sliced for further processing. Slicing is crucial for segmenting the part into manageable layers that can be processed individually later on. The software also allows users to manually draw polygons, and through a sequence of clustering, creating centroids, clearing centroids, and forming clusters, it enables effective slicing. Consequently, identifying and isolating

these points helps maintain the quality of the AM process. Any irregularities and defects can be diagnosed and corrected early.

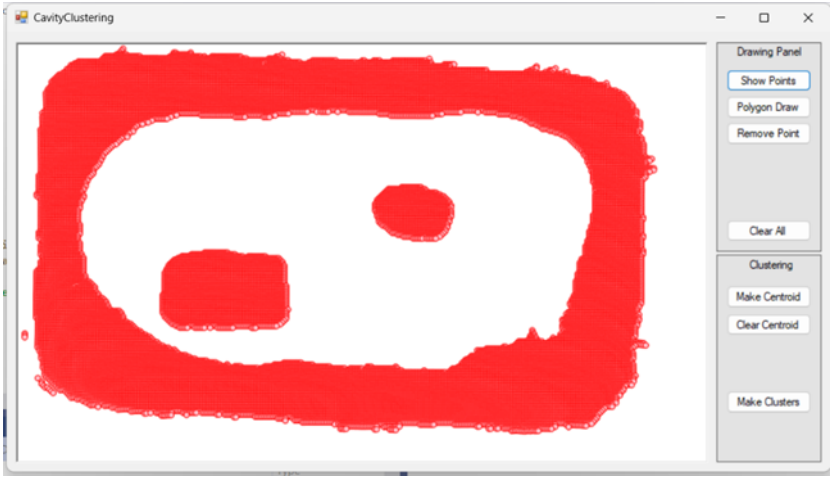


Figure 4-11 Slicing points below a predefined height for further processing.

Another step is identifying areas for refinement by drawing polygons around them. This functionality allows for the elimination of unnecessary points through a selective process, removing parts that could cause noise in the data (Figure 4-12). This process can be applied multiple times, iteratively removing undesired points. Such features enable users to retain only the relevant points for subsequent processing, thereby improving the quality and precision of the WAAM process. Equally important, the added precision can prevent significant defects in the final project, where even small inaccuracies may dramatically affect the final product.

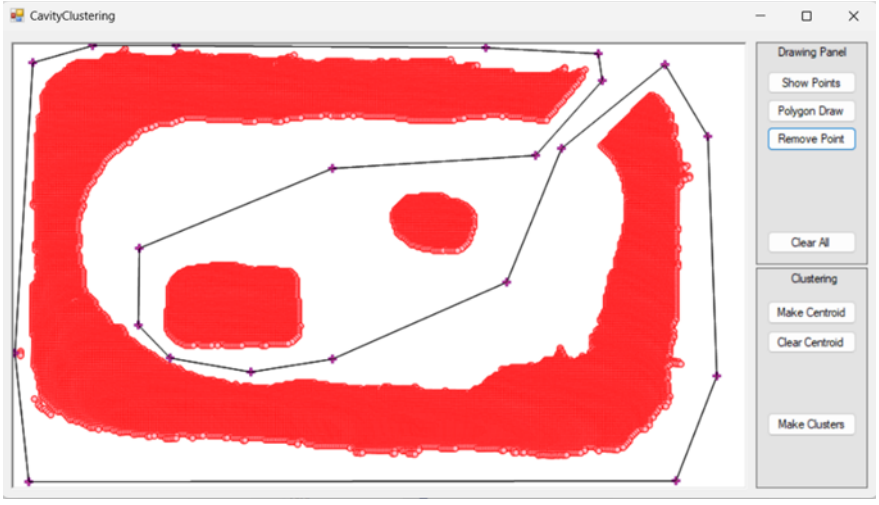


Figure 4-12 Interface allowing multiple uses of a polygon tool to remove unnecessary points.

Cavity clustering is identified in **Figure 4-13**, resulting from the removal of extraneous points using the “Remove Point” button. This action leaves behind only the necessary cavity clusters, eliminating points that are not required for further processing. The elimination of unnecessary points significantly reduces computational load and substantially enhances the efficiency of the path planning process.

Figure 4-13 Extra points can be removed by clicking the ‘Remove Point’.

In the next step, the center of each cluster is approximately chosen by the user, and the program refines this centroid. These markers serve as initial guesses for the center of each cluster, combining human intuition with computational accuracy. This step helps ensure that the centroids are properly located for the subsequent processing stages (Figure 4-14). By leveraging both user input and automated refinement, the process maintains a balance between flexibility and precision, ultimately enhancing the accuracy and quality of the clustering results.

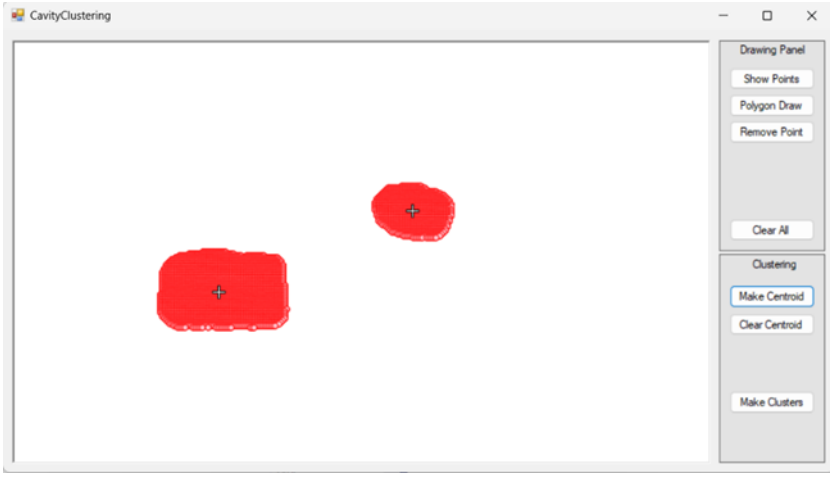


Figure 4-14 The center of the cluster is approximately chosen by the user and then corrected by the program.

Following this, the identification of cluster centers and the number of cavities within the clusters can be carried out by clicking the "Make Clusters" button, as depicted in **Figure 4-15**. Each cluster is identified by a distinct color (e.g., green and red), making them visually distinguishable. The number of cavities and temporary points is vital for understanding the internal structure of clusters and recognizing any potential voids or gaps that need to be addressed.

Then, the cavity clustering software tests the cloud of points for each layer for cavities and collisions. The interface (Figure 4-16) displays a warning if collisions are detected. This step is essential to ensure that the integrity of the manufactured part is not compromised due to collisions and to prevent material waste. Additionally, it can detect cavities within the layer, which is crucial for the structural soundness of the final product. Effective troubleshooting and process control are facilitated by halting the WAAM process when a warning message appears, allowing for corrective actions before proceeding.

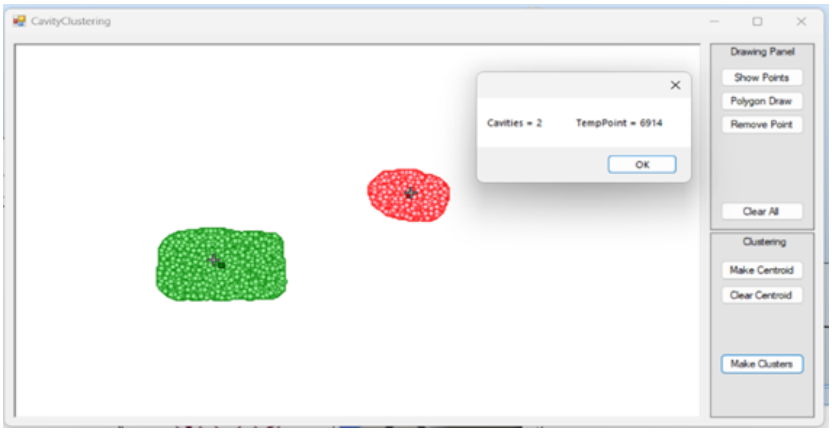


Figure 4-15 By clicking "Make Clusters," the program accurately identifies the exact locations of the cluster centers.

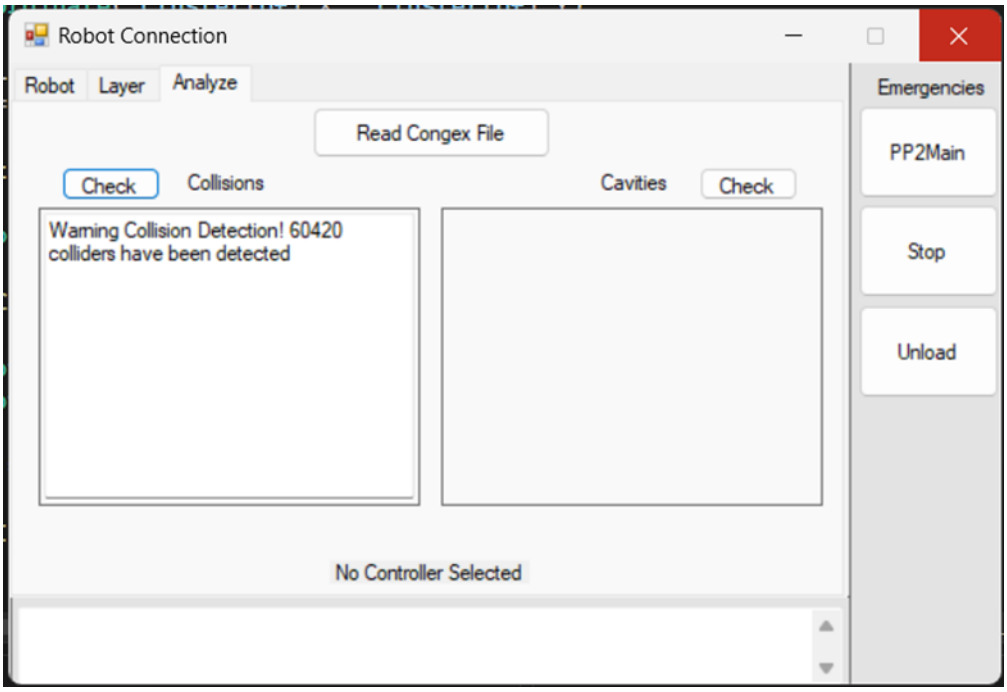


Figure 4-16 In this step, the point cloud of each layer is tested for collisions and cavities. If a collision is detected, the process stops; if cavities are found, the subsequent steps are followed.

Figure 4-17 illustrates the generation of a new path plan for corrections made by the cavity clustering software. The cluster borders are aligned with the path plan, shown as blue lines. This path plan is created based on the alignment between the cluster borders and the original path plan. By generating correction paths, the need for manual adjustments is reduced, process efficiency is improved, and the likelihood of errors is decreased. This proactive approach significantly enhances the overall quality of the manufactured parts. Following this step, the program can distinguish the borders of each cluster, automatically identifying and drawing them to show the extent of each cluster's separation (Figure 4-18). This visual clarity is essential for understanding the spatial distribution and organization of the points within each cluster, helping isolate regions that require specific attention throughout the process.

Figure 4-17 The program distinguishes the border of each cluster.

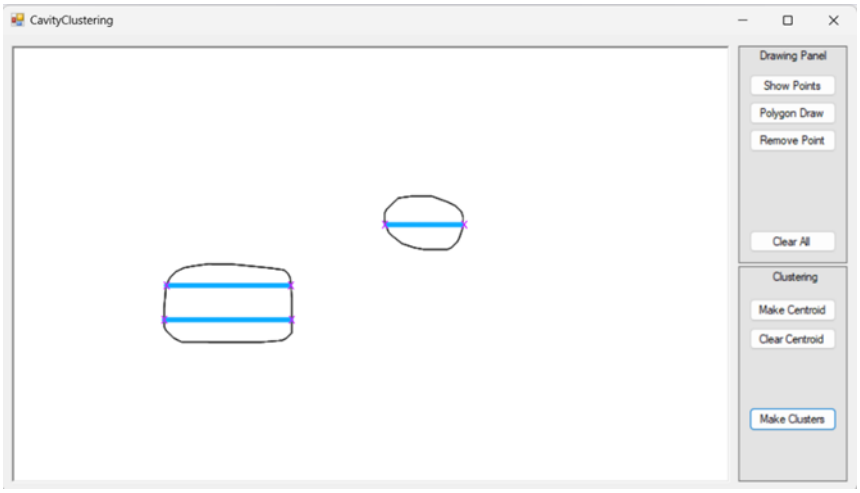


Figure 4-18 The new path plan for corrections has been generated in correspondence to the original path. The blue lines represent the alignment of the cluster borders with the new path plan.

In the following updated example of mesh analysis, after capturing the image using the camera, a slice of the mesh at a predetermined height is extracted for detailed examination. The process begins by removing noise and unwanted data points, ensuring that only relevant information remains. The centroids of each cluster are now automatically defined to account for more complex deposition patterns, improving the accuracy and efficiency of the analysis. Any sections below a specified area threshold are filtered out using software, reducing computational load and focusing on significant regions. Once the data is filtered, the precise centroids of the

remaining clusters are calculated, and their boundaries are clearly defined. Finally, the overlap between the correction robot's path and the identified clusters is mapped, demonstrating the system's ability to handle intricate patterns and ensure accurate adjustments in these more complex settings.

Figure 4-19 Slicing points below a predefined height for further processing.

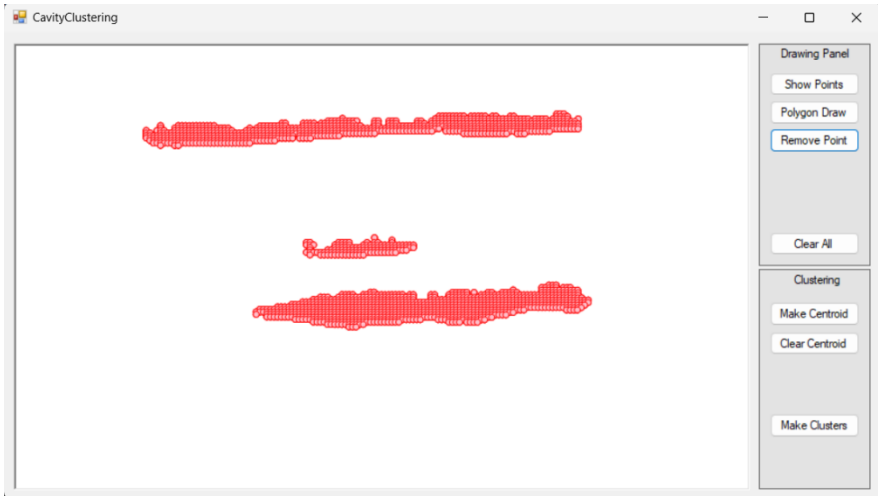


Figure 4-20 Removing extra points

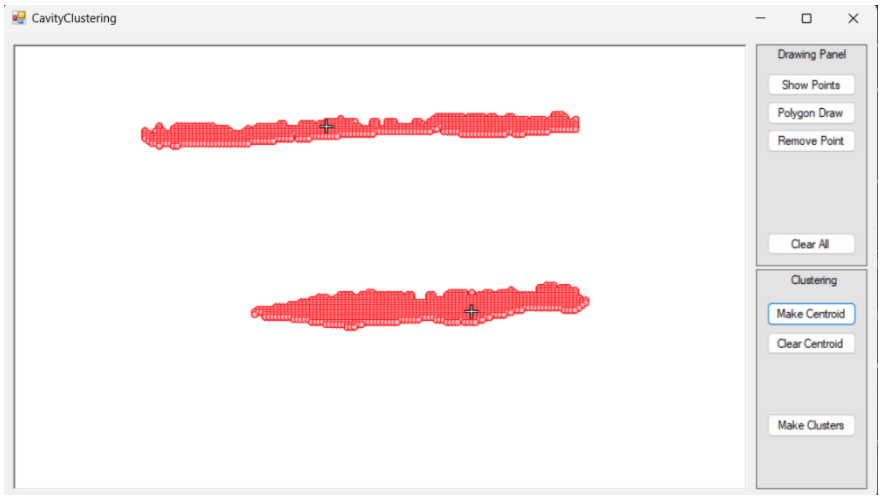


Figure 4-21 The center of the cluster is approximately chosen by the user and then corrected by the program.

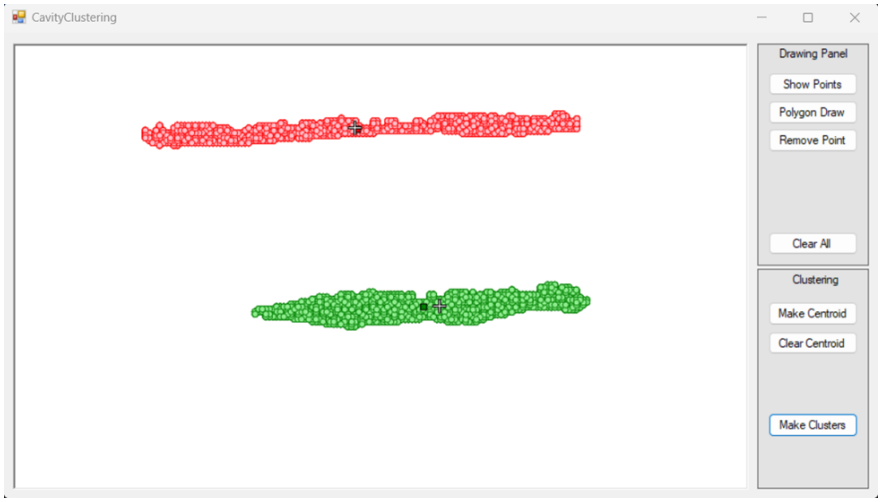


Figure 4-22 By clicking "Make Clusters," the program accurately identifies the exact locations of the cluster centers.

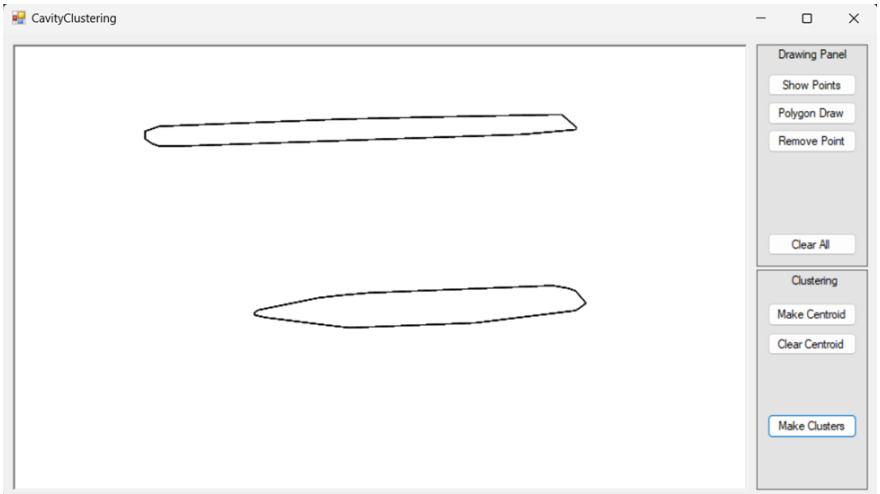


Figure 4-23 The program distinguishes the border of each cluster.

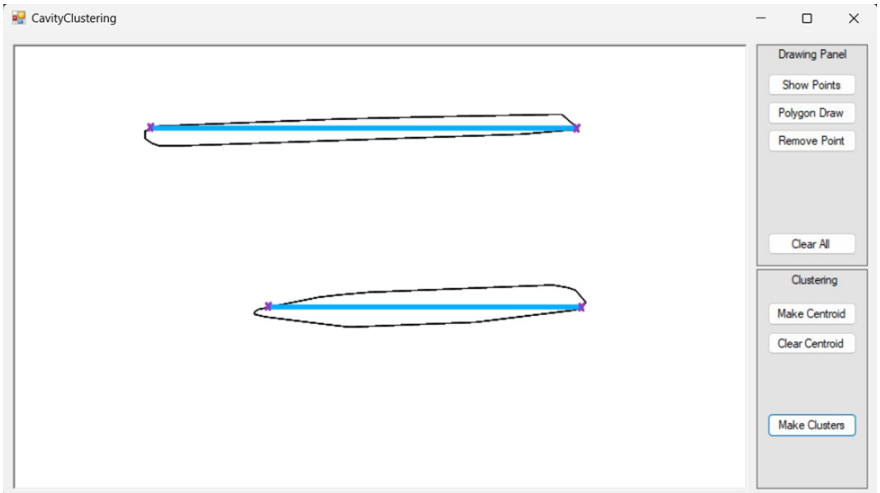


Figure 4-24 The new path plan for corrections has been generated in correspondence to the original path. The blue lines represent the alignment of the cluster borders with the new path plan.

4.4 Concluding Remarks

The developed methodologies provide a comprehensive approach to improving the WAAM process through adaptive correction strategies, microstructure analysis, and optimized welding device settings. The integration of a vision system enables real-time identification and correction of defects such as cavities and irregularities, ensuring that each subsequent layer is accurately deposited. The calibration of the 3D camera with the robot, combined with the development of path correction software, allows for precise and reliable material deposition. These advancements not only enhance the overall quality and consistency of the manufactured components but also contribute significantly to the efficiency and adaptability of the WAAM process.

Chapter 5

5 Conclusion and Recommendations for future works:

5.1 Conclusion

AM technology has become prevalent in various industries, including aerospace, marine and shipbuilding, power generation and oil and gas. These industries have adopted AM due to its capability to produce complex geometries, reduce material waste, and shorten production times. To further improve efficiency, the automation of processes stands as a paramount demand. Particularly in aerospace, minimizing operator error is crucial due to the high standards and precision required. To meet these requirements, the application of a vision system comes into play. The incorporation of a vision system into the AM process addresses inherent barriers such as cavities and geometrical irregularities.

In this thesis, we propose a closed-loop vision guided robotic WAAM system. This system is capable of diagnosing the inconsistencies on the deposited surface and adapting subsequent layer deposition based on detected imperfections. It is noteworthy that this program is expected to be implemented in an online control of the robot, ensuring real-time adjustments and monitoring.

In the presence of issues such as irregularities and cavities, which are common and significant challenges in WAAM, our algorithm can detect these problems and apply adaptive correction

strategies by adjusting the deposition path for the next layer. First, the existing limitations in literature and the contributions to the anticipated knowledge made by this thesis will be addressed. Each contribution will then be discussed in detail.

- **Limitation in Offline Programming and Pre-Determined Path Planning:**

Limitation: Previous WAAM systems often relied on offline robot programming and pre-determined algorithms for collision-free path planning but lacked real-time control mechanisms [48]. This limitation restricted their ability to adjust dynamically during the deposition process, leading to inconsistent layer quality.

Contribution: This research introduces a real-time, closed-loop control system using the Cognex A5000 3D camera, which enables dynamic adjustments to the deposition path. By monitoring surface irregularities in real time, the system can correct defects immediately, offering a significant improvement over pre-determined static approaches.

- **Lack of Real-Time Adaptive Layer Adjustments:**

Limitation: A research conducted by Reisch et al. [49] utilized multivariate sensor frameworks to gather process data such as gas flow, voltage, and thermal imaging, but lacked real-time adaptive layer adjustments. These systems also did not focus on material stability or proper clamping techniques, which can lead to warping or distortions.

Contribution: In contrast, this work employs a vision-based system to monitor and adapt to surface defects during deposition. It also addresses material stability by introducing effective clamping techniques that prevent warping, ensuring a higher degree of dimensional accuracy and structural integrity in the final product.

- **Predefined Trajectories Limiting Adaptability:**

Limitation: In another research conducted by Coutinho et al. [50] rely on predefined trajectories and sensor-based feedback systems, limiting the ability to adapt to surface irregularities in real time. These methods do not support real-time detection and correction of imperfections during the deposition process.

Contribution: This research integrates real-time vision-based monitoring, allowing the system to dynamically correct surface imperfections as they arise. By offering real-time surface correction,

the system surpasses the limitations of predefined trajectory methods, enhancing process adaptability.

- **Time-Consuming Optimization with Iterative Learning Models:**

Limitation: Lizarralde et al. [52] implemented reinforcement learning models to correct geometric errors, but these approaches required lengthy training phases and iterative learning, which often led to suboptimal performance during the early stages of deposition.

Contribution: This research avoids the time-consuming learning phase by utilizing a real-time vision system. This approach provides immediate error correction and faster, more accurate adjustments during the deposition process, significantly improving the efficiency of multi-layer, multi-bead deposition.

The research work yielded several important developments:

- **Optimal Welding Path for Small-Scale Objects**: For the geometry in this study, the optimal number of beads was determined through experimental methods and slicing software. This is essential for optimizing material usage and maintaining structural integrity in similar geometries. In the case of small-scale objects, a six-bead coverage for a rectangular shape is determined to be the most effective. This finding is crucial for optimizing material usage and ensuring structural integrity.

- **Management of Heat Accumulation**: Continuous process operation is not feasible due to heat accumulation. After three layers are deposited, a pause is necessary until the object's temperature returns to an acceptable range. This step is vital to prevent overheating and material degradation.

- **Clamping and Securing Base Material**: During the experimental procedure, it was noted that appropriate clamping and securing the movement of the base material are essential to prevent warping of the base plate during the additive manufacturing process. This measure ensures dimensional accuracy and stability of the final product.

- **Vision System Selection**: The Cognex A5000 camera was utilized for the vision system in this study. It functions both as a camera and a scanner, providing high-resolution imaging

and accurate 3D scanning capabilities that helped finding the geometrical irregularities within the deposited material.

- **Mesh Analysis Program**: Developing an in-house mesh analysis program is necessary due to the limitations of the camera software, which is not an open access system and does not provide full API access. The existing software of the camera could not deliver the required output for our specific needs.
- **Temperature Control**: Controlling the temperature of the object is crucial for maintaining process stability and ensuring the quality of the deposited layers.
- **Real-Time and Online Process Control**: To achieve full automation, it is essential to control the process in real-time and online. This capability enables immediate adjustments and corrections during the manufacturing process. This project took the initial steps for addressing this need for a successful WAAM process.
- **Software for Robot Path Transfer**: Cura and RoboDK are identified as the appropriate software of choice for transferring the .cad file to the robot path. These tools facilitate the conversion and optimization of design files for robotic execution.
- **RobotStudio Externally Guided Motion (EGM)**: The RobotStudio EGM add-on has limitations in path planning corrections since the external device must be mounted on the robot. Correction is only applied in the path coordinate system, and only position corrections in the y and z axes can be performed.

The research work proposes online control of a robotic welding process using a 3D scanner. This unified approach involved optimizing welding parameters, developing offline robot programming, and establishing an effective path planning strategy. Each component of the configuration was successfully tested offline, and the entire system was virtually validated through online configuration. However, when transitioning to the physical apparatus, the Human-Machine Interface (HMI) prevented the configuration from functioning as intended. Despite this challenge, the work laid a strong foundation for future integration and testing, highlighting the potential for precise and adaptive control in enhancing the efficiency and reliability of the welding process.

5.2 Future Works

Further investigations are recommended in the following areas:

- **Transparent Shield for Nozzle:** Producing the shield of the nozzle from transparent materials would allow users to monitor the tip of the nozzle or wire during the process using a proper camera capable of monitoring the melt pool. This transparency would enhance the ability to monitor and troubleshoot the welding process in real-time.
- **Cooling Methods for Base Material and Object:** Developing a cooling method for the object is essential to build up and preventing meltdown. Additionally, finding a cooling method for the base material is necessary since clamping alone cannot completely prevent the warping of the base material.
- **Automatic Clustering:** Currently, the number of clusters is defined by the operator based on the chosen cluster centers during mesh processing. Automating this process is a priority to reduce operator dependency and improve efficiency.
- **Solving HMI Issues:** The HMI is defined as a safety device in the PLC program of the robot, giving it priority for system Mastership. We need to find a method to overcome the issue where the HMI prevents our C# program from gaining Mastership. This solution is crucial for ensuring seamless integration and control of the manufacturing process.
- **Adding a Second Robot for Machining:** Presently, irregularities are manually ground by an operator. In the near future, a second robot could be linked to cooperate, using the same mesh process data to remove irregularities automatically. This addition would enhance process automation and reduce manual intervention.
- **Adding a Thermal Camera:** Incorporating a thermal camera would automate the control of object temperature and pause times between layers. This enhancement would improve the consistency and quality of the manufacturing process by providing real-time temperature monitoring and control.

These future works will significantly enhance the automation, efficiency, and reliability of the additive manufacturing process, making it more adaptable and capable of meeting the high standards required by various industries.

References

[1] S.A.M. Tofail, E.P. Koumoulos, A. Bandyopadhyay, S. Bose, L. O'Donoghue, C. Charitidis, Additive manufacturing: scientific and technological challenges, market uptake and opportunities, Mater. Today. 21 (2018) 22–37. https://doi.org/10.1016/J.MATTOD.2017.07.001.

[2] A. Bandyopadhyay, S. Bose, Additive Manufacturing, Second Edition, CRC Press, 2019. https://books.google.ca/books?id=70W4DwAAQBAJ.

[3] I. Gibson, D.W. Rosen, B. Stucker, M. Khorasani, D. Rosen, B. Stucker, M. Khorasani, Additive manufacturing technologies, Springer, 2021.

[4] M. Baumers, P. Dickens, C. Tuck, R. Hague, The cost of additive manufacturing: machine productivity, economies of scale and technology-push, Technol. Forecast. Soc. Change. 102 (2016) 193–201. https://doi.org/10.1016/J.TECHFORE.2015.02.015.

[5] D.G. Ahn, Directed Energy Deposition (DED) Process: State of the Art, Int. J. Precis. Eng. Manuf. Technol. 2021 82. 8 (2021) 703–742. https://doi.org/10.1007/S40684-020-00302-7.

[6] A. Dass, A. Moridi, State of the Art in Directed Energy Deposition: From Additive Manufacturing to Materials Design, Coatings 2019, Vol. 9, Page 418. 9 (2019) 418. https://doi.org/10.3390/COATINGS9070418.

[7] K. Treutler, V. Wesling, A.M. Camacho, R.(Chunhui,) Yang, The Current State of Research of Wire Arc Additive Manufacturing (WAAM): A Review, Appl. Sci. 2021, Vol. 11, Page 8619. 11 (2021) 8619. https://doi.org/10.3390/APP11188619.

[8] S.W. Williams, F. Martina, A.C. Addison, J. Ding, G. Pardal, P. Colegrove, Wire + Arc Additive Manufacturing, Mater. Sci. Technol. 32 (2016) 641–647. https://doi.org/10.1179/1743284715Y.0000000073.

[9] Y. Li, C. Su, J. Zhu, Comprehensive review of wire arc additive manufacturing: Hardware system, physical process, monitoring, property characterization, application and future

prospects, Results Eng. 13 (2022) 100330. https://doi.org/10.1016/J.RINENG.2021.100330.

[10] S.R. Singh, P. Khanna, Wire arc additive manufacturing (WAAM): A new process to shape engineering materials, Mater. Today Proc. 44 (2021) 118–128. https://doi.org/10.1016/J.MATPR.2020.08.030.

[11] A. Shah, R. Aliyev, H. Zeidler, S. Krinke, A Review of the Recent Developments and Challenges in Wire Arc Additive Manufacturing (WAAM) Process, J. Manuf. Mater. Process. 2023, Vol. 7, Page 97. 7 (2023) 97. https://doi.org/10.3390/JMMP7030097.

[12] P. Kazanas, P. Deherkar, P. Almeida, H. Lockett, S. Williams, Fabrication of geometrical features using wire and arc additive manufacture, Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. 226 (2012) 1042–1051. https://doi.org/10.1177/0954405412437126.

[13] R. Samson, In-situ Automated Scan-assisted Repair using Hybrid Manufacturing, University of Alberta, 2022. https://doi.org/10.7939/R3-G871-7S04.

[14] Y. Xu, G. Fang, N. Lv, S. Chen, J. Jia Zou, Computer vision technology for seam tracking in robotic GTAW and GMAW, Robot. Comput. Integr. Manuf. 32 (2015) 25–36. https://doi.org/10.1016/J.RCIM.2014.09.002.

[15] M. Dinham, G. Fang, Weld seam detection using computer vision for robotic Arc Welding, IEEE Int. Conf. Autom. Sci. Eng. (2012) 771–776. https://doi.org/10.1109/COASE.2012.6386339.

[16] G. Tucci, G. Guidi, D. Ostuni, F. Costantino, M. Pieraccini, J.-A. Beraldin, Photogrammetry and 3D Scanning: Assessment of Metric Accuracy for the Digital Model of Danatello's Maddalena, (2001).

[17] W. Lin, H. Shen, J. Fu, S. Wu, Online quality monitoring in material extrusion additive manufacturing processes based on laser scanning technology, Precis. Eng. 60 (2019) 76–84. https://doi.org/10.1016/J.PRECISIONENG.2019.06.004.

[18] Z. Wang, R. Liu, T. Sparks, H. Liu, F. Liou, Stereo vision based hybrid manufacturing process for precision metal parts, Precis. Eng. 42 (2015) 1–5. https://doi.org/10.1016/J.PRECISIONENG.2014.11.012.

[19] G.F. Marshall, G.E. Stutz, Handbook of Optical and Laser Scanning, CRC Press, 2012. https://doi.org/https://doi.org/10.1201/9781315218243.

[20] L. He-xi, S. Yong-hua, W. Guo-rong, Z. Xiao-xi, Automatic Teaching of Welding Robot for 3-Dimensional Seam Based on Ant Colony Optimization Algorithm, in: 2009 Second Int. Conf. Intell. Comput. Technol. Autom., 2009: pp. 398–402. https://doi.org/10.1109/ICICTA.2009.562.

[21] O.H. Karatas, E. Toy, Three-dimensional imaging techniques: A literature review., Eur. J. Dent. 8 (2014) 132–140. https://doi.org/10.4103/1305-7456.126269.

[22] J. Huang, S. You, Detecting Objects in Scene Point Cloud: A Combinational Approach, in: 2013 Int. Conf. 3D Vis. - 3DV 2013, 2013: pp. 175–182. https://doi.org/10.1109/3DV.2013.31.

[23] C. Kohrt, R. Stamp, A.G. Pipe, J. Kiely, G. Schiedermeier, An online robot trajectory planning and programming support system for industrial use, Robot. Comput. Integr. Manuf. 29 (2013) 71–79. https://doi.org/https://doi.org/10.1016/j.rcim.2012.07.010.

[24] D. Nguyen-Tuong, J. Peters, Local Gaussian process regression for real-time model-based robot control, in: 2008 IEEE/RSJ Int. Conf. Intell. Robot. Syst., 2008: pp. 380–385. https://doi.org/10.1109/IROS.2008.4650850.

[25] B. Chen, C. Hua, B. Dai, Y. He, J. Han, Online control programming algorithm for human–robot interaction system with a novel real-time human gesture recognition method, Int. J. Adv. Robot. Syst. 16 (2019) 1729881419861764. https://doi.org/10.1177/1729881419861764.

[26] K. Ishii, T. Fujii, T. Ura, Neural network system for online controller adaptation and its application to underwater robot, in: Proceedings. 1998 IEEE Int. Conf. Robot. Autom. (Cat. No.98CH36146), 1998: pp. 756–761 vol.1. https://doi.org/10.1109/ROBOT.1998.677067.

[27] W. Kaczmarek, B. Lotys, S. Borys, D. Laskowski, P. Lubkowski, Controlling an Industrial Robot Using a Graphic Tablet in Offline and Online Mode, Sensors. 21 (2021). https://doi.org/10.3390/s21072439.

[28] J.L. Prado-Cerqueira, A.M. Camacho, J.L. Diéguez, Á. Rodríguez-Prieto, A.M. Aragón, C. Lorenzo-Martín, Á. Yanguas-Gil, Analysis of Favorable Process Conditions for the Manufacturing of Thin-Wall Pieces of Mild Steel Obtained by Wire and Arc Additive Manufacturing (WAAM), Materials (Basel). 11 (2018). https://doi.org/10.3390/ma11081449.

[29] M. Awd, J. Tenkamp, M. Hirtler, S. Siddique, M. Bambach, F. Walther, Comparison of Microstructure and Mechanical Properties of Scalmalloy® Produced by Selective Laser Melting and Laser Metal Deposition, Materials (Basel). 11 (2018). https://doi.org/10.3390/ma11010017.

[30] I. Sizova, M. Hirtler, M. Günther, M. Bambach, Wire-arc additive manufacturing of pre-forms for forging of a Ti–6Al–4V turbine blade, AIP Conf. Proc. 2113 (2019) 150017. https://doi.org/10.1063/1.5112693.

[31] X. Xu, S. Ganguly, J. Ding, S. Guo, S. Williams, F. Martina, Microstructural evolution and mechanical properties of maraging steel produced by wire+arc additive manufacture process, Mater. Charact. 143 (2018) 152–162. https://doi.org/https://doi.org/10.1016/j.matchar.2017.12.002.

[32] F. Wang, S. Williams, P. Colegrove, A.A. Antonysamy, Microstructure and Mechanical Properties of Wire and Arc Additive Manufactured Ti-6Al-4V, Metall. Mater. Trans. A. 44 (2013) 968–977. https://doi.org/10.1007/s11661-012-1444-6.

[33] A. Caballero, J. Ding, S. Ganguly, S. Williams, Wire + Arc Additive Manufacture of 17-4 PH stainless steel: Effect of different processing conditions on microstructure, hardness, and tensile strength, J. Mater. Process. Technol. 268 (2019) 54–62. https://doi.org/https://doi.org/10.1016/j.jmatprotec.2019.01.007.

[34] S. Suryakumar, K.P. Karunakaran, U. Chandrasekhar, M.A. Somashekara, A study of the mechanical properties of objects built through weld-deposition, Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. 227 (2013) 1138–1147. https://doi.org/10.1177/0954405413482122.

[35] D. Ding, Z. (Stephen) Pan, D. Cuiuri, H. Li, A tool-path generation strategy for wire and arc additive manufacturing, Int. J. Adv. Manuf. Technol. 73 (2014) 173–183.

https://doi.org/10.1007/s00170-014-5808-5.

[36] Y.M. Zhang, P. Li, Y. Chen, A.T. Male, Automated system for welding-based rapid prototyping, Mechatronics. 12 (2002) 37–53. https://doi.org/https://doi.org/10.1016/S0957-4158(00)00064-7.

[37] Y. Li, Q. Han, G. Zhang, I. Horváth, A layers-overlapping strategy for robotic wire and arc additive manufacturing of multi-layer multi-bead components with homogeneous layers, Int. J. Adv. Manuf. Technol. 96 (2018) 3331–3344. https://doi.org/10.1007/s00170-018-1786-3.

[38] L. Nguyen, Tool path planning for wire-arc additive manufacturing processes, 2022. https://doi.org/10.26127/BTUOpen-5982.

[39] C.F. Cheung, L.T. Ho, P. Charlton, L.B. Kong, S. To, W.B. Lee, Analysis of surface generation in the ultraprecision polishing of freeform surfaces, Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. 224 (2009) 59–73. https://doi.org/10.1243/09544054JEM1563.

[40] B.H. Kim, B.K. Choi, Machining efficiency comparison direction-parallel tool path with contour-parallel tool path, Comput. Des. 34 (2002) 89–95. https://doi.org/https://doi.org/10.1016/S0010-4485(00)00139-1.

[41] S. Routhu, D. Kanakanala, J. Ruan, X. Liu, F. Liou, 2-D Path Planning for Direct Laser Deposition Process, 2010. https://doi.org/10.1115/DETC2010-28440.

[42] G. Zhao, G. Ma, W. Xiao, Y. Tian, Feature-based five-axis path planning method for robotic additive manufacturing, Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. 233 (2018) 1412–1424. https://doi.org/10.1177/0954405417752508.

[43] R. French, H. Marin-Reye, G. Kapellmann-Zafra, S. Abrego-Hernandez, Robotic additive manufacturing system featuring wire deposition by electric arc for high-value manufacturing, in: 2019 IEEE 15th Int. Conf. Autom. Sci. Eng., IEEE Press, 2019: pp. 1748–1754. https://doi.org/10.1109/COASE.2019.8842997.

[44] P.F. Bariani, G. Berti, L. D'Angelo, Tool Cost Estimating at the Early Stages of Cold Forging Process Design, CIRP Ann. 42 (1993) 279–282. https://doi.org/https://doi.org/10.1016/S0007-8506(07)62443-3.

[45] S. Chen, J.T. Wen, Industrial Robot Trajectory Tracking Control Using Multi-Layer Neural Networks Trained by Iterative Learning Control, Robotics. 10 (2021). https://doi.org/10.3390/robotics10010050.

[46] M. Bi, L. Xia, P. Tran, Z. Li, Q. Wan, L. Wang, W. Shen, G. Ma, Y.M. Xie, Continuous contour-zigzag hybrid toolpath for large format additive manufacturing, Addit. Manuf. 55 (2022) 102822. https://doi.org/https://doi.org/10.1016/j.addma.2022.102822.

[47] A.M. Ikotun, A.E. Ezugwu, L. Abualigah, B. Abuhaija, J. Heming, K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data, Inf. Sci. (Ny). 622 (2023) 178–210. https://doi.org/10.1016/J.INS.2022.11.139.

[48] L. Yuan, Z. Pan, J. Polden, D. Ding, S. van Duin, H. Li, Integration of a multi-directional wire arc additive manufacturing system with an automated process planning algorithm, J. Ind. Inf. Integr. 26 (2022) 100265. https://doi.org/10.1016/J.JII.2021.100265.

[49] R. Reisch, T. Hauser, T. Kamps, A. Knoll, Robot Based Wire Arc Additive Manufacturing System with Context-Sensitive Multivariate Monitoring Framework, Procedia Manuf. 51 (2020) 732–739. https://doi.org/10.1016/J.PROMFG.2020.10.103.

[50] F. Coutinho, N. Lizarralde, M. Mendes, R. Bostrom, T. Silva, M. Couto, F. Lizarralde, iWAAM: An automated system for monitoring and control of wire-arc additive manufacturing, IFAC-PapersOnLine. 56 (2023) 6576–6581. https://doi.org/10.1016/J.IFACOL.2023.10.309.

[51] A.G. Dharmawan, Y. Xiong, S. Foong, G. Song Soh, A Model-Based Reinforcement Learning and Correction Framework for Process Control of Robotic Wire Arc Additive Manufacturing, Proc. - IEEE Int. Conf. Robot. Autom. (2020) 4030–4036. https://doi.org/10.1109/ICRA40945.2020.9197222.

[52] N. Lizarralde, F. Coutinho, F. Lizarralde, Online Coordinated Motion Control of a Redundant Robotic Wire Arc Additive Manufacturing System, IEEE Robot. Autom. Lett. 7 (2022) 9675–9682. https://doi.org/10.1109/LRA.2022.3191741.

[53] Y.C. Shiu, S. Ahmad, Calibration of Wrist-Mounted Robotic Sensors by Solving Homogeneous Transform Equations of the Form AX = XB, IEEE Trans. Robot. Autom. 5

(1989) 16–29. https://doi.org/10.1109/70.88014.

[54] M. Dinham, G. Fang, A low cost hand-eye calibration method for arc welding robots, 2009 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2009. (2009) 1889–1893. https://doi.org/10.1109/ROBIO.2009.5420552.

[55] M. Dinham, G. Fang, Low cost simultaneous calibration of a stereo vision system and a welding robot, 2010 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2010. (2010) 1452–1456. https://doi.org/10.1109/ROBIO.2010.5723543.

Appendix A

```
!CYLINDER OFFLINE RAPID PROGRAM
PROC CylinderWelding()
wobTable:= wobj0;
MoveJ ClctWeldAprch, v5, z10, tProd\WObj:= wobTable;
MoveL reltool(ClctWeldStart, 0, 0, -5), v5, z10, tProd\WObj:= wobTable;
WaitTime 5;
numLayer:= 0;
FOR numLayer FROM 0 TO nLayer DO
TPWrite "numLayer before layer loop =" \num:= numLayer;
numLine:= 0;
FOR numLine FROM 0 TO nLine DO
TPWrite "numLayer inside line loop=" \num:= numLayer;
SlicingCal numLine, numLayer;
Set pdoTigWeldingStart;
MoveL ClctWeldStart, v5, fine, tProd\WObj:= wobTable;
IF numLayer = 0 OR arret=TRUE THEN
    WaitTime 3;
    arret:= FALSE;
ENDIF

Movec ClctScndpCrcl,Clctthrdpcrcl, v5, z20, tProd\WObj:=wobTable;
Movec Clctfrthpcrcl, ClctWeldStart, v5, fine, tProd\WObj:=wobTable;
reset pdoTigWeldingStart;
ENDFOR

IF cooling=4 AND numLayer>=1 AND (numLayer mod 2)=1 THEN
arret:=TRUE;
!stop;
!WaitTime Attente;
!Set pdoTigWeldingStart;
MoveL ClctWeldStart, v5, fine, tProd\WObj:= wobTable;
Movec ClctScndpCrcl,Clctthrdpcrcl, v5, z20, tProd\WObj:=wobTable;
Movec Clctfrthpcrcl, ClctWeldStart, v5, fine, tProd\WObj:=wobTable;
!reset pdoTigWeldingStart;
!Stop;
ENDIF

ENDFOR
MoveL ClctWeldRtrct, v5, z50, tProd\WObj:= wobTable;
prMoveHome;
WaitRob \InPos;
    ENDPROC

!MULTI LINE MULTI LAYER OFFLINE RAPID PROGRAMMING
PROC prMultiSoudage()
    !MoveJ rtHome,vHome,z100,tool0\WObj:=wobj0;

    wobTable:=wobj0;
    !tProd:=tTopTig; ! The tool is loaded and offsetd for angles in the routine prcMultiInitCalc

    MoveJ pProgM230WeldAppr,vTrans,z10,tProd\WObj:=wobTable; ! Rough Approach

    ! Loop for the multiple layer

    FOR nIndexLayer FROM 0 TO nLayer DO
        TPWrite "nIndexLayer before layer loop= " \num:=nIndexLayer;
        ! If this is turned on, the approach and the retract will be inverted in the alculation
routine prCalcMulti
        IF bEtageInv THEN
        IF (nIndexLayer MOD 2)=1 THEN
          bAlt:=FALSE;
```

```
        ELSE
          bAlt:=TRUE;
        ENDIF
      ELSE
        bAlt:=TRUE;
      ENDIF

      ! Loop for the multiple lines
      nIndexLine:=0;
      FOR nIndexLine FROM 0 TO nLine DO
          TPWrite "nIndexLayer inside line loop= " \num:=nIndexLayer;
        prCalcMulti nIndexLine,nIndexLayer;

        MoveL reltool(pProgM230WeldStart,0,0,-5),vTrans,z10,tProd\WObj:=wobTable; ! Final
Approach

        Set pdoTigWeldingStart;

        MoveL pProgM230WeldStart, v5, fine, tProd\WObj:=wobTable;

        WaitTime 3;
        MoveL pProgM230WeldEnd, v5, fine, tProd\WObj:=wobTable; ! Weld end

        reset pdoTigWeldingStart;
        WaitTime 3;

        MoveL RelTool(pProgM230WeldEnd,0,0,-5), v5, fine, tProd\WObj:=wobTable;

      ENDFOR

    ENDFOR

    MoveL pProgM230WeldRet, v5, z50, tProd\WObj:=wobTable;

    prMoveHome;

    WaitRob \InPos;

       ENDPROC
```

Appendix B

```
     ' ************************************************************************
     ' C# Online Program, Robot Connections
     ' ************************************************************************

// EmergencyPanel
            //
            this.EmergencyPanel.Anchor =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.EmergencyPanel.AutoSize = true;
            this.EmergencyPanel.BackColor = System.Drawing.SystemColors.ControlLight;
            this.EmergencyPanel.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.EmergencyPanel.Controls.Add(this.Unload_BTN);
            this.EmergencyPanel.Controls.Add(this.Stop_BTN);
            this.EmergencyPanel.Controls.Add(this.PPToMain_BTN);
            this.EmergencyPanel.Controls.Add(this.Emergency_Panel);
            this.EmergencyPanel.Location = new System.Drawing.Point(634, -6);
            this.EmergencyPanel.Name = "EmergencyPanel";
            this.EmergencyPanel.Size = new System.Drawing.Size(122, 444);
            this.EmergencyPanel.TabIndex = 0;
            //
            // Unload_BTN
            //
            this.Unload_BTN.Location = new System.Drawing.Point(6, 174);
            this.Unload_BTN.Name = "Unload_BTN";
            this.Unload_BTN.Size = new System.Drawing.Size(106, 65);
            this.Unload_BTN.TabIndex = 3;
            this.Unload_BTN.Text = "Unload";
            this.Unload_BTN.UseVisualStyleBackColor = true;
            this.Unload_BTN.Click += new System.EventHandler(this.Unload_BTN_Click);
            //
            // Stop_BTN
            //
            this.Stop_BTN.Location = new System.Drawing.Point(6, 104);
            this.Stop_BTN.Name = "Stop_BTN";
            this.Stop_BTN.Size = new System.Drawing.Size(106, 65);
            this.Stop_BTN.TabIndex = 2;
            this.Stop_BTN.Text = "Stop";
            this.Stop_BTN.UseVisualStyleBackColor = true;
            this.Stop_BTN.Click += new System.EventHandler(this.Stop_BTN_Click);
            //
            // PPToMain_BTN
            //
            this.PPToMain_BTN.Location = new System.Drawing.Point(6, 34);
            this.PPToMain_BTN.Name = "PPToMain_BTN";
            this.PPToMain_BTN.Size = new System.Drawing.Size(106, 65);
            this.PPToMain_BTN.TabIndex = 1;
            this.PPToMain_BTN.Text = "PP2Main";
            this.PPToMain_BTN.UseVisualStyleBackColor = true;
            this.PPToMain_BTN.Click += new System.EventHandler(this.PPToMain_BTN_Click);
            //
            // Emergency_Panel
            //
            this.Emergency_Panel.AutoSize = true;
            this.Emergency_Panel.Location = new System.Drawing.Point(16, 14);
            this.Emergency_Panel.Name = "Emergency_Panel";
            this.Emergency_Panel.Size = new System.Drawing.Size(87, 16);
            this.Emergency_Panel.TabIndex = 0;
            this.Emergency_Panel.Text = "Emergencies";
            //
            // LogPanel
            //
```

```
            this.LogPanel.Anchor =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.LogPanel.BackColor = System.Drawing.SystemColors.ControlLight;
            this.LogPanel.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.LogPanel.Controls.Add(this.LogTextBox);
            this.LogPanel.Location = new System.Drawing.Point(1, 372);
            this.LogPanel.Name = "LogPanel";
            this.LogPanel.Size = new System.Drawing.Size(636, 66);
            this.LogPanel.TabIndex = 1;
            //
            // LogTextBox
            //
            this.LogTextBox.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
            | System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.LogTextBox.BackColor = System.Drawing.SystemColors.ControlLightLight;
            this.LogTextBox.HideSelection = false;
            this.LogTextBox.Location = new System.Drawing.Point(4, 6);
            this.LogTextBox.Multiline = true;
            this.LogTextBox.Name = "LogTextBox";
            this.LogTextBox.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.LogTextBox.ShortcutsEnabled = false;
            this.LogTextBox.Size = new System.Drawing.Size(623, 55);
            this.LogTextBox.TabIndex = 0;
            this.LogTextBox.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.LogTextBox_TextChanged);
            //
            // MainTabControl
            //
            this.MainTabControl.Controls.Add(this.RobotTab);
            this.MainTabControl.Controls.Add(this.LayerTab);
            this.MainTabControl.Controls.Add(this.AnalyzeTab);
            this.MainTabControl.Location = new System.Drawing.Point(1, 1);
            this.MainTabControl.Name = "MainTabControl";
            this.MainTabControl.SelectedIndex = 0;
            this.MainTabControl.Size = new System.Drawing.Size(632, 372);
            this.MainTabControl.TabIndex = 2;
            this.MainTabControl.TabStop = false;
            this.MainTabControl.SelectedIndexChanged += new
System.EventHandler(this.MainTabControl_SelectedIndexChanged);
            //
            // RobotTab
            //
            this.RobotTab.Controls.Add(this.MotorOn_BTN);
            this.RobotTab.Controls.Add(this.Scan_BTN);
            this.RobotTab.Controls.Add(this.RobotList);
            this.RobotTab.Location = new System.Drawing.Point(4, 25);
            this.RobotTab.Name = "RobotTab";
            this.RobotTab.Padding = new System.Windows.Forms.Padding(3);
            this.RobotTab.Size = new System.Drawing.Size(624, 343);
            this.RobotTab.TabIndex = 0;
            this.RobotTab.Text = "Robot";
            this.RobotTab.UseVisualStyleBackColor = true;

            // RobotList
            //
            this.RobotList.AutoArrange = false;
            this.RobotList.BackColor = System.Drawing.SystemColors.Window;
            this.RobotList.Columns.AddRange(new System.Windows.Forms.ColumnHeader[] {
            this.IP_Field,
            this.Virtual_Field,
            this.SystemName_Field,
            this.Mode_Field,
            this.Motor_Status_Field});
            this.RobotList.ForeColor = System.Drawing.SystemColors.InfoText;
            this.RobotList.GridLines = true;
            this.RobotList.HideSelection = false;
```

```
            listViewItem1.StateImageIndex = 0;
            listViewItem1.UseItemStyleForSubItems = false;
            this.RobotList.Items.AddRange(new System.Windows.Forms.ListViewItem[] {
            listViewItem1});
            this.RobotList.Location = new System.Drawing.Point(4, 62);
            this.RobotList.Margin = new System.Windows.Forms.Padding(5, 6, 5, 6);
            this.RobotList.MultiSelect = false;
            this.RobotList.Name = "RobotList";
            this.RobotList.Size = new System.Drawing.Size(624, 213);
            this.RobotList.TabIndex = 1;
            this.RobotList.UseCompatibleStateImageBehavior = false;
            this.RobotList.View = System.Windows.Forms.View.Details;
            this.RobotList.SelectedIndexChanged += new
System.EventHandler(this.RobotList_SelectedIndexChanged);
            //
            // IP_Field
            //
            this.IP_Field.Text = "IP Address";
            this.IP_Field.Width = 100;
            //
            // Virtual_Field
            //
            this.Virtual_Field.Text = "Virtual";
            this.Virtual_Field.Width = 70;
            //
            // SystemName_Field
            //
            this.SystemName_Field.Text = "System name";
            this.SystemName_Field.Width = 110;
            //
            // Mode_Field
            //
            this.Mode_Field.Text = "Mode";
            this.Mode_Field.Width = 70;

             Motor_Status_Field

            this.Motor_Status_Field.Text = "Motor Status";
            this.Motor_Status_Field.Width = 100;

             LayerTab

            this.LayerTab.BackColor = System.Drawing.Color.Transparent;
            this.LayerTab.Controls.Add(this.LayerPanel2);
            this.LayerTab.Controls.Add(this.LayerPanel1);
            this.LayerTab.ForeColor = System.Drawing.SystemColors.ActiveCaptionText;
            this.LayerTab.Location = new System.Drawing.Point(4, 25);
            this.LayerTab.Name = "LayerTab";
            this.LayerTab.Padding = new System.Windows.Forms.Padding(3);
            this.LayerTab.Size = new System.Drawing.Size(624, 343);
            this.LayerTab.TabIndex = 1;
            this.LayerTab.Text = "Layer";

             LayerPanel2


            //
            // TableHeightField
            //
            this.TableHeightField.Location = new System.Drawing.Point(98, 29);
            this.TableHeightField.Name = "TableHeightField";
            this.TableHeightField.Size = new System.Drawing.Size(77, 22);
            this.TableHeightField.TabIndex = 6;
            this.TableHeightField.Text = "1";
            this.TableHeightField.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;
            //
            // BaseHeightField
            //
            this.BaseHeightField.Location = new System.Drawing.Point(98, 57);
            this.BaseHeightField.Name = "BaseHeightField";
            this.BaseHeightField.Size = new System.Drawing.Size(77, 22);
```

```
this.BaseHeightField.TabIndex = 5;
this.BaseHeightField.Text = "1";
this.BaseHeightField.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;
//
// TableHeightLable
//
this.TableHeightLable.AutoSize = true;
this.TableHeightLable.Location = new System.Drawing.Point(3, 32);
this.TableHeightLable.Name = "TableHeightLable";
this.TableHeightLable.Size = new System.Drawing.Size(85, 16);
this.TableHeightLable.TabIndex = 4;
this.TableHeightLable.Text = "Table Height";
//
// BaseHeightLable
//
this.BaseHeightLable.AutoSize = true;
this.BaseHeightLable.Location = new System.Drawing.Point(3, 63);
this.BaseHeightLable.Name = "BaseHeightLable";
this.BaseHeightLable.Size = new System.Drawing.Size(81, 16);
this.BaseHeightLable.TabIndex = 3;
this.BaseHeightLable.Text = "Base Height";
//
// LayerNumTextBox
//
this.LayerNumTextBox.Location = new System.Drawing.Point(98, 88);
this.LayerNumTextBox.Name = "LayerNumTextBox";
this.LayerNumTextBox.Size = new System.Drawing.Size(77, 22);
this.LayerNumTextBox.TabIndex = 2;
this.LayerNumTextBox.Text = "1";
this.LayerNumTextBox.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;
//
// LayerNumLabel
//
this.LayerNumLabel.AutoSize = true;
this.LayerNumLabel.Location = new System.Drawing.Point(3, 91);
this.LayerNumLabel.Name = "LayerNumLabel";
this.LayerNumLabel.Size = new System.Drawing.Size(48, 16);
this.LayerNumLabel.TabIndex = 1;
this.LayerNumLabel.Text = "Layer#";
//
// StartTask_BTN
//
this.StartTask_BTN.Location = new System.Drawing.Point(236, 50);
this.StartTask_BTN.Name = "StartTask_BTN";
this.StartTask_BTN.Size = new System.Drawing.Size(135, 43);
this.StartTask_BTN.TabIndex = 0;
this.StartTask_BTN.Text = "Start Robot Task";
this.StartTask_BTN.UseVisualStyleBackColor = true;
this.StartTask_BTN.Click += new System.EventHandler(this.StartTask_BTN_Click);
//
// LayerPanel1
//
this.LayerPanel1.BackColor = System.Drawing.Color.WhiteSmoke;
this.LayerPanel1.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
this.LayerPanel1.Controls.Add(this.TaskLable);
this.LayerPanel1.Controls.Add(this.CorrectModul);
this.LayerPanel1.Controls.Add(this.LoadRobotFile_BTN);
this.LayerPanel1.Location = new System.Drawing.Point(7, 6);
this.LayerPanel1.Name = "LayerPanel1";
this.LayerPanel1.Size = new System.Drawing.Size(611, 121);
this.LayerPanel1.TabIndex = 1;
//
// LoadRobotFile_BTN
//
this.LoadRobotFile_BTN.Location = new System.Drawing.Point(41, 37);
this.LoadRobotFile_BTN.Name = "LoadRobotFile_BTN";
this.LoadRobotFile_BTN.Size = new System.Drawing.Size(224, 44);
this.LoadRobotFile_BTN.TabIndex = 0;
this.LoadRobotFile_BTN.Text = "Load Robot File";
this.LoadRobotFile_BTN.UseVisualStyleBackColor = true;
```

```
            this.LoadRobotFile_BTN.Click += new
System.EventHandler(this.LoadRobotFile_BTN_Click);
            //
            // AnalyzeTab
            //
            this.AnalyzeTab.Controls.Add(this.CheckCavity_BTN);
            this.AnalyzeTab.Controls.Add(this.CheckCollision_BTN);
            this.AnalyzeTab.Controls.Add(this.CollisionLabel);
            this.AnalyzeTab.Controls.Add(this.CavityLabel);
            this.AnalyzeTab.Controls.Add(this.CavityPanel);
            this.AnalyzeTab.Controls.Add(this.CollisionPanel);
            this.AnalyzeTab.Controls.Add(this.ReadCognexFile_BTN);
            this.AnalyzeTab.Location = new System.Drawing.Point(4, 25);
            this.AnalyzeTab.Name = "AnalyzeTab";
            this.AnalyzeTab.Size = new System.Drawing.Size(624, 343);
            this.AnalyzeTab.TabIndex = 2;
            this.AnalyzeTab.Text = "Analyze";
            this.AnalyzeTab.UseVisualStyleBackColor = true;
            //
            // CheckCavity_BTN
            //
            this.CheckCavity_BTN.Location = new System.Drawing.Point(514, 44);
            this.CheckCavity_BTN.Name = "CheckCavity_BTN";
            this.CheckCavity_BTN.Size = new System.Drawing.Size(75, 23);
            this.CheckCavity_BTN.TabIndex = 5;
            this.CheckCavity_BTN.Text = "Check";
            this.CheckCavity_BTN.UseVisualStyleBackColor = true;
            this.CheckCavity_BTN.Visible = false;
            this.CheckCavity_BTN.Click += new System.EventHandler(this.CheckCavity_BTN_Click);
            //
            // CheckCollision_BTN
            //
            this.CheckCollision_BTN.Location = new System.Drawing.Point(33, 44);
            this.CheckCollision_BTN.Name = "CheckCollision_BTN";
            this.CheckCollision_BTN.Size = new System.Drawing.Size(75, 23);
            this.CheckCollision_BTN.TabIndex = 4;
            this.CheckCollision_BTN.Text = "Check";
            this.CheckCollision_BTN.UseVisualStyleBackColor = true;
            this.CheckCollision_BTN.Visible = false;
            this.CheckCollision_BTN.Click += new
System.EventHandler(this.CheckCollision_BTN_Click);
            //
            // CollisionLabel
            //
            this.CollisionLabel.AutoSize = true;
            this.CollisionLabel.Location = new System.Drawing.Point(126, 47);
            this.CollisionLabel.Name = "CollisionLabel";
            this.CollisionLabel.Size = new System.Drawing.Size(65, 16);
            this.CollisionLabel.TabIndex = 3;
            this.CollisionLabel.Text = "Collisions";
            //
            // CavityLabel
            //
            this.CavityLabel.AutoSize = true;
            this.CavityLabel.Location = new System.Drawing.Point(438, 47);
            this.CavityLabel.Name = "CavityLabel";
            this.CavityLabel.Size = new System.Drawing.Size(55, 16);
            this.CavityLabel.TabIndex = 3;
            this.CavityLabel.Text = "Cavities";
            //
            // CavityPanel
            //
            this.CavityPanel.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.CavityPanel.Location = new System.Drawing.Point(320, 73);
            this.CavityPanel.Name = "CavityPanel";
            this.CavityPanel.Size = new System.Drawing.Size(287, 207);
            this.CavityPanel.TabIndex = 2;
            //
            // CollisionPanel
            //
            this.CollisionPanel.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
```

```
            this.CollisionPanel.Controls.Add(this.CollisionReportField);
            this.CollisionPanel.Location = new System.Drawing.Point(16, 73);
            this.CollisionPanel.Name = "CollisionPanel";
            this.CollisionPanel.Size = new System.Drawing.Size(287, 207);
            this.CollisionPanel.TabIndex = 1;
            //
            // CollisionReportField
            //
            this.CollisionReportField.Location = new System.Drawing.Point(3, 3);
            this.CollisionReportField.Multiline = true;
            this.CollisionReportField.Name = "CollisionReportField";
            this.CollisionReportField.Size = new System.Drawing.Size(279, 199);
            this.CollisionReportField.TabIndex = 0;
            //
            // ReadCognexFile_BTN
            //
            this.ReadCognexFile_BTN.Location = new System.Drawing.Point(223, 3);
            this.ReadCognexFile_BTN.Name = "ReadCognexFile_BTN";
            this.ReadCognexFile_BTN.Size = new System.Drawing.Size(180, 36);
            this.ReadCognexFile_BTN.TabIndex = 0;
            this.ReadCognexFile_BTN.Text = "Read Congex File";
            this.ReadCognexFile_BTN.UseVisualStyleBackColor = true;
            this.ReadCognexFile_BTN.Click += new
System.EventHandler(this.ReadCognexFile_BTN_Click);
            //
            // LogMenuStrip
            //
            this.LogMenuStrip.ImageScalingSize = new System.Drawing.Size(20, 20);
            this.LogMenuStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.ClearLog,
            this.CopyLog});
            this.LogMenuStrip.Name = "LogMenuStrip";
            this.LogMenuStrip.Size = new System.Drawing.Size(111, 52);
            //
            // ClearLog
            //
            this.ClearLog.Name = "ClearLog";
            this.ClearLog.Size = new System.Drawing.Size(110, 24);
            this.ClearLog.Text = "clear";
            this.ClearLog.Click += new System.EventHandler(this.ClearLog_Click);
            //
            // CopyLog
            //
            this.CopyLog.Name = "CopyLog";
            this.CopyLog.Size = new System.Drawing.Size(110, 24);
            this.CopyLog.Text = "copy";
            this.CopyLog.Click += new System.EventHandler(this.CopyLog_Click);
            //
            // labelSelectedController
            //
            this.labelSelectedController.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
            | System.Windows.Forms.AnchorStyles.Left)
            | System.Windows.Forms.AnchorStyles.Right)));
            this.labelSelectedController.AutoSize = true;
            this.labelSelectedController.Location = new System.Drawing.Point(243, 347);
            this.labelSelectedController.Name = "labelSelectedController";
            this.labelSelectedController.Size = new System.Drawing.Size(142, 16);
            this.labelSelectedController.TabIndex = 5;
            this.labelSelectedController.Text = "No Controller Selected";
            //
            // CorrectModul
            //
            this.CorrectModul.Location = new System.Drawing.Point(349, 37);
            this.CorrectModul.Name = "CorrectModul";
            this.CorrectModul.Size = new System.Drawing.Size(224, 44);
            this.CorrectModul.TabIndex = 1;
            this.CorrectModul.Text = "Load Correction File";
            this.CorrectModul.UseVisualStyleBackColor = true;
            //
```

```
            // TaskLable
            //
            this.TaskLable.AutoSize = true;
            this.TaskLable.Location = new System.Drawing.Point(266, 20);
            this.TaskLable.Name = "TaskLable";
            this.TaskLable.Size = new System.Drawing.Size(84, 16);
            this.TaskLable.TabIndex = 2;
            this.TaskLable.Text = "Select a task";
            //
            // RobotConnection
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.AutoSize = true;
            this.ClientSize = new System.Drawing.Size(750, 433);
            this.Controls.Add(this.labelSelectedController);
            this.Controls.Add(this.MainTabControl);
            this.Controls.Add(this.EmergencyPanel);
            this.Controls.Add(this.LogPanel);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.MaximizeBox = false;
            this.Name = "RobotConnection";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Robot Connection";
            this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.RobotConnection_FormClosing);
            this.Load += new System.EventHandler(this.RobotConnection_Load);
            this.EmergencyPanel.ResumeLayout(false);
            this.EmergencyPanel.PerformLayout();
            this.LogPanel.ResumeLayout(false);
            this.LogPanel.PerformLayout();
            this.MainTabControl.ResumeLayout(false);
            this.RobotTab.ResumeLayout(false);
            this.LayerTab.ResumeLayout(false);
            this.LayerPanel2.ResumeLayout(false);
            this.LayerPanel2.PerformLayout();
            this.LayerPanel1.ResumeLayout(false);
            this.LayerPanel1.PerformLayout();
            this.AnalyzeTab.ResumeLayout(false);
            this.AnalyzeTab.PerformLayout();
            this.CollisionPanel.ResumeLayout(false);
            this.CollisionPanel.PerformLayout();
            this.LogMenuStrip.ResumeLayout(false);
            this.ResumeLayout(false);
                 this.PerformLayout();
```

Appendix C

```
' ************************************************************************
' C# Online Program, Design
' ************************************************************************
InitializeComponent()
        {

            this.button1 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.button3 = new System.Windows.Forms.Button();
            this.button4 = new System.Windows.Forms.Button();
            this.button5 = new System.Windows.Forms.Button();
            this.button6 = new System.Windows.Forms.Button();
            this.richTextBox1 = new System.Windows.Forms.RichTextBox();
            this.LoadFile = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.ConvertCord_BTN = new System.Windows.Forms.Button();
            this.FindCollisionBTN = new System.Windows.Forms.Button();
            this.FindCavityBTN = new System.Windows.Forms.Button();
            this.LayerNumberText = new System.Windows.Forms.TextBox();
            this.label3 = new System.Windows.Forms.Label();
            this.UnloadBTN = new System.Windows.Forms.Button();
            this.panel1 = new System.Windows.Forms.Panel();
            this.SuspendLayout();
            //
            // listView1
            //
            this.listView1.AutoArrange = false;
            this.listView1.BackColor = System.Drawing.SystemColors.Window;
            this.listView1.Columns.AddRange(new System.Windows.Forms.ColumnHeader[] {
            this.columnHeader25,
            this.columnHeader26,
            this.columnHeader27,
            this.columnHeader28,
            this.columnHeader29,
            this.columnHeader30,
            this.columnHeader31,
            this.columnHeader32});
            this.listView1.ForeColor = System.Drawing.SystemColors.InfoText;
            this.listView1.FullRowSelect = true;
            this.listView1.GridLines = true;
            this.listView1.HideSelection = false;
            listViewItem1.StateImageIndex = 0;
            this.listView1.Items.AddRange(new System.Windows.Forms.ListViewItem[] {
            listViewItem1});
            this.listView1.Location = new System.Drawing.Point(16, 23);
            this.listView1.Margin = new System.Windows.Forms.Padding(5, 6, 5, 6);
            this.listView1.MultiSelect = false;
            this.listView1.Name = "listView1";
            this.listView1.Size = new System.Drawing.Size(732, 236);
            this.listView1.TabIndex = 0;
            this.listView1.UseCompatibleStateImageBehavior = false;
            this.listView1.View = System.Windows.Forms.View.Details;
            //
            // columnHeader25
            //
            this.columnHeader25.Text = "IP Address";
            this.columnHeader25.Width = 150;
            //
            // columnHeader26
            //
            this.columnHeader26.Text = "ID";
            this.columnHeader26.Width = 47;
            //
            // columnHeader27
```

```
            //
            this.columnHeader27.Text = "Availability";
            this.columnHeader27.Width = 100;
            //
            // columnHeader28
            //
            this.columnHeader28.Text = "Virtual";
            this.columnHeader28.Width = 70;
            //
            // columnHeader29
            //
            this.columnHeader29.Text = "System name";
            this.columnHeader29.Width = 120;
            //
            // columnHeader30
            //
            this.columnHeader30.Text = "Version";
            this.columnHeader30.Width = 80;
            //
            // columnHeader31
            //
            this.columnHeader31.Text = "Name";
            this.columnHeader31.Width = 88;
            //
            // columnHeader32
            //
            this.columnHeader32.Text = "Mode";
            this.columnHeader32.Width = 70;
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.SystemColors.ActiveCaption;
            this.button1.Font = new System.Drawing.Font("Microsoft YaHei UI", 16.2F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(134)));
            this.button1.Location = new System.Drawing.Point(827, 61);
            this.button1.Margin = new System.Windows.Forms.Padding(5, 6, 5, 6);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(144, 50);
            this.button1.TabIndex = 1;
            this.button1.Text = "Scan Network";
            this.button1.UseVisualStyleBackColor = false;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.BackColor = System.Drawing.SystemColors.Highlight;
            this.button2.Font = new System.Drawing.Font("Microsoft YaHei UI", 16.2F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(134)));
            this.button2.Location = new System.Drawing.Point(768, 196);
            this.button2.Margin = new System.Windows.Forms.Padding(5, 6, 5, 6);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(250, 60);
            this.button2.TabIndex = 2;
            this.button2.Text = "PP to Main";
            this.button2.UseVisualStyleBackColor = false;
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // button3
            //
            this.button3.BackColor = System.Drawing.SystemColors.Highlight;
            this.button3.Font = new System.Drawing.Font("Microsoft YaHei UI", 16.2F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button3.Location = new System.Drawing.Point(768, 300);
            this.button3.Margin = new System.Windows.Forms.Padding(5, 6, 5, 6);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(103, 60);
            this.button3.TabIndex = 3;
            this.button3.Text = "Start";
            this.button3.UseVisualStyleBackColor = false;
            this.button3.Click += new System.EventHandler(this.button3_Click);
            //
```

```
            // button4
            //
            this.button4.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(255)))),
((int)(((byte)(128)))), ((int)(((byte)(128)))));
            this.button4.Font = new System.Drawing.Font("Microsoft YaHei UI", 16.2F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button4.Location = new System.Drawing.Point(768, 372);
            this.button4.Margin = new System.Windows.Forms.Padding(5, 6, 5, 6);
            this.button4.Name = "button4";
            this.button4.Size = new System.Drawing.Size(103, 55);
            this.button4.TabIndex = 4;
            this.button4.Text = "Stop";
            this.button4.UseVisualStyleBackColor = false;
            this.button4.Click += new System.EventHandler(this.button4_Click);
            //
            // button5
            //
            this.button5.BackColor = System.Drawing.SystemColors.Highlight;
            this.button5.Font = new System.Drawing.Font("Microsoft YaHei UI", 13F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(134)));
            this.button5.Location = new System.Drawing.Point(768, 114);
            this.button5.Margin = new System.Windows.Forms.Padding(5, 6, 5, 6);
            this.button5.Name = "button5";
            this.button5.Size = new System.Drawing.Size(120, 70);
            this.button5.TabIndex = 5;
            this.button5.Text = "Motors ON";
            this.button5.UseVisualStyleBackColor = false;
            this.button5.Click += new System.EventHandler(this.button5_Click);
            //
            // button6
            //
            this.button6.BackColor = System.Drawing.Color.FromArgb(((int)(((byte)(255)))),
((int)(((byte)(128)))), ((int)(((byte)(128)))));
            this.button6.Font = new System.Drawing.Font("Microsoft YaHei UI", 13F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.button6.Location = new System.Drawing.Point(898, 114);
            this.button6.Margin = new System.Windows.Forms.Padding(5, 6, 5, 6);
            this.button6.Name = "button6";
            this.button6.Size = new System.Drawing.Size(120, 70);
            this.button6.TabIndex = 6;
            this.button6.Text = "Motors OFF";
            this.button6.UseVisualStyleBackColor = false;
            this.button6.Click += new System.EventHandler(this.button6_Click);
            //
            this.Controls.Add(this.panel1);
            this.Controls.Add(this.UnloadBTN);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.LayerNumberText);
            this.Controls.Add(this.FindCavityBTN);
            this.Controls.Add(this.FindCollisionBTN);
            this.Controls.Add(this.ConvertCord_BTN);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.LoadFile);
            this.Controls.Add(this.richTextBox1);
            this.Controls.Add(this.button6);
            this.Controls.Add(this.button5);
            this.Controls.Add(this.button4);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.listView1);
            this.Font = new System.Drawing.Font("Microsoft YaHei UI Light", 10.2F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.Margin = new System.Windows.Forms.Padding(5, 6, 5, 6);
            this.Name = "RobotArmController";
            this.Text = "RobotArmController";
            this.ResumeLayout(false);
            this.PerformLayout();

        }
```

```
#endregion
private System.Windows.Forms.ColumnHeader columnHeader25;
private System.Windows.Forms.ColumnHeader columnHeader26;
private System.Windows.Forms.ColumnHeader columnHeader27;
private System.Windows.Forms.ColumnHeader columnHeader28;
private System.Windows.Forms.ColumnHeader columnHeader29;
private System.Windows.Forms.ColumnHeader columnHeader30;
private System.Windows.Forms.ColumnHeader columnHeader31;
private System.Windows.Forms.ColumnHeader columnHeader32;
private System.Windows.Forms.Button button1;
private System.Windows.Forms.Button button2;
private System.Windows.Forms.Button button3;
private System.Windows.Forms.Button button4;
private System.Windows.Forms.Button button5;
private System.Windows.Forms.Button button6;
private System.Windows.Forms.RichTextBox richTextBox1;
private System.Windows.Forms.ListView listView1;
private System.Windows.Forms.Button LoadFile;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Button ConvertCord_BTN;
private System.Windows.Forms.Button FindCollisionBTN;
private System.Windows.Forms.Button FindCavityBTN;
private System.Windows.Forms.TextBox LayerNumberText;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Button UnloadBTN;
     private System.Windows.Forms.Panel panel1;
```

Appendix D

```
' ****************************************************************************
' C# Online Program, Extension
' ****************************************************************************
public static class Extensions
    {
        private static Random Rand = new Random();

        public static void DrawPoints(this Graphics gr,
            List<PointF> points, Brush brush, Pen pen, float radius)
        {
            if (points == null) return;
            foreach (PointF point in points)
                gr.DrawPoint(point, brush, pen, radius);
        }

        public static void DrawRectangle(this Graphics gr,
            Pen pen, RectangleF rect)
        {
            gr.DrawRectangle(pen,rect.X, rect.Y, rect.Width, rect.Height);
        }
        public static void DrawRect(this Graphics gr, PointF point,
            Brush brush, Pen pen, float radius)
        {
            RectangleF rect = new RectangleF(
                point.X - radius, point.Y - radius,
                2 * radius, 2 * radius);
            gr.FillRectangle(brush, rect);
            gr.DrawRectangle(pen, rect);
        }
        public static void DrawPoint(this Graphics gr, PointF point,
            Brush brush, Pen pen, float radius)
        {
            RectangleF rect = new RectangleF(
                point.X - radius, point.Y - radius,
                2 * radius, 2 * radius);
            gr.FillEllipse(brush, rect);
            gr.DrawEllipse(pen, rect);
        }
        public static void DrawCross(this Graphics gr,
            Pen pen, PointF point, float radius)
        {
            gr.DrawLine(pen, point.X - radius, point.Y, point.X + radius, point.Y);
            gr.DrawLine(pen, point.X, point.Y - radius, point.X, point.Y + radius);
        }
        public static void DrawCross(this Graphics gr,
            Color outer_color, Color inner_color, PointF point, float radius)
        {
            using (Pen pen = new Pen(outer_color, 3))
            {
                gr.DrawLine(pen, point.X - radius - 1, point.Y, point.X + radius + 1, point.Y);
                gr.DrawLine(pen, point.X, point.Y - radius - 1, point.X, point.Y + radius + 1);
            }
            using (Pen pen = new Pen(inner_color, 1))
            {
                gr.DrawLine(pen, point.X - radius, point.Y, point.X + radius, point.Y);
                gr.DrawLine(pen, point.X, point.Y - radius, point.X, point.Y + radius);
            }
        }

        // Pick a random item from the list.
        public static T Random<T>(List<T> items)
        {
            return items[Rand.Next(items.Count)];
        }
```

```
    // Randomize an array.
    public static void Randomize<T>(this T[] items)
    {
        // For each spot in the array, pick
        // a random item to swap into that spot.
        for (int i = 0; i < items.Length - 1; i++)
        {
            int j = Rand.Next(i, items.Length);
            T temp = items[i];
            items[i] = items[j];
            items[j] = temp;
        }
    }

    // Randomize a list.
    public static void Randomize<T>(this List<T> items)
    {
        // Convert into an array.
        T[] item_array = items.ToArray();

        // Randomize.
        item_array.Randomize();

        // Copy the items back into the list.
        items.Clear();
        items.AddRange(item_array);
    }
}
 }
```

Appendix E

```
' ******************************************************************************
' C# Online Program, Cavities
' ******************************************************************************
        private Brush[] PointBrushes =
        {
            Brushes.Pink, Brushes.LightGreen, Brushes.LightBlue, Brushes.Yellow,
            Brushes.Orange, Brushes.Lime, Brushes.Cyan, Brushes.White,
        };
        private Pen[] PointPens =
        {
            Pens.Red, Pens.Green, Pens.Blue, Pens.Black,
            Pens.Red, Pens.Green, Pens.Blue, Pens.Black,
        };
        private Brush[] CentroidBrushes =
        {
            Brushes.Red, Brushes.Green, Brushes.Blue, Brushes.Yellow,
            Brushes.Orange, Brushes.Lime, Brushes.Cyan, Brushes.White,
        };



        private void CavityClustering_Load(object sender, EventArgs e)
        {
            MaxClusters = PointBrushes.Length;
        }

        private void picItems_Paint(object sender, PaintEventArgs e)
        {

            const float RADIUS = 3;
            e.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
            //e.Graphics.DrawPoint(new PointF(50, 50), PointBrushes[5],PointPens [5], 5);
            // Draw the points.
            foreach (PointData point_data in Points)
            {
                e.Graphics.DrawPoint(new PointF(point_data.Location.X , point_data.Location.Y),
                    PointBrushes[point_data.ClusterNum % MaxClusters],
                    PointPens[point_data.ClusterNum % MaxClusters], RADIUS);
            }

            // Draw the centroids.
            for (int i = 0; i < Centroids.Count; i++)
            {
                e.Graphics.DrawRect(Centroids[i],
                    CentroidBrushes[i % MaxClusters], Pens.Black, RADIUS);
            }
```

```
            //PointsBackUp = Points;
        }
        //float AvrageX = 0.0f;
        //float AvrageY = 0.0f;
        SumX = 0.0f;
        SumY = 0.0f;
        MinX = Points[0].Location.X;
        MaxX = Points[0].Location.X;
        MinY = Points[0].Location.Y;
        MaxY = Points[0].Location.Y;
        foreach (PointData p in Points)
        {
            SumX += p.Location.X;
            SumY += p.Location.Y;
            if(p.Location.X < MinX)
            {
                MinX = p.Location.X;
            }
            else if (p.Location.X > MaxX)
            {
                MaxX = p.Location.X;
            }
            if(p.Location.Y < MinY)
            {
                MinY = p.Location.Y;
            }
            else if(p.Location.Y > MaxY)
            {
                MaxY = p.Location.Y;
            }
        }
        //MessageBox.Show((MinX).ToString() + " " + (MaxX).ToString());
        //MessageBox.Show((MinY).ToString() + " " + (MaxY).ToString());

        lengthX = Math.Sqrt(Math.Pow(Math.Abs(MinX - MaxX), 2));
        lengthY = Math.Sqrt(Math.Pow(Math.Abs(MinY - MaxY), 2));
        //MessageBox.Show("Length X = "+lengthX.ToString()+" "+"Length Y = "+
lengthY.ToString());


        //MessageBox.Show("Length X = "+lengthX.ToString(), "Length Y = " +
lengthY.ToString());
        //AvrageX = SumX / Points.Count;
        //AvrageY = SumY / Points.Count;
        //float tempX = -(AvrageX - (picItems.Size.Width / 2));
```

```
            //float tempY = -(AvrageY - (picItems.Size.Height / 2));        public static void
MakePath()
        {
            Paths = new List<PointF>();
            //MessageBox.Show();
            foreach (Vector4 p in IntersectionFinder.NewLines)
            {
                //MessageBox.Show("MinX = " + MinX + " lengthX = "+ lengthX + " point.X = "+p.X +
                    //"Math.Sqrt(Math.Pow(Math.Abs(MinX - (lengthX-p.X)), 2))");
                float px = PicItemSizeWithd - p.Y;
                float py = PicItemSizeHeight - p.X;
                float pz = PicItemSizeWithd - p.W;
                float pw = PicItemSizeHeight - p.Z;
                double LengthPointX1 = Math.Sqrt(Math.Pow(Math.Abs(MinX - (px)), 2));
                //MessageBox.Show("LengthPointX1 = " + LengthPointX1 + " MinX = " + MinX + "
lengthX = "+ lengthX+ " px = "+ px);

                double XRatio1 = (lengthX * 1.1f) / ((lengthX - LengthPointX1));

                double LengthPointY1 = Math.Sqrt(Math.Pow(Math.Abs(MinY - (py)), 2));
                double YRatio1 = (lengthY * 1.1f) / (lengthY - LengthPointY1);

                double LengthPointX2 = Math.Sqrt(Math.Pow(Math.Abs(MinX - (pz)), 2));
                double XRatio2 = (lengthX * 1.1f) / ((lengthX - LengthPointX2));

                double LengthPointY2 = Math.Sqrt(Math.Pow(Math.Abs(MinY - (pw)), 2));
                double YRatio2 = (lengthY * 1.1f) / (lengthY - LengthPointY2);

                PointF p1 = new PointF(PicItemSizeWithd / (float)XRatio1 + (float)(lengthX / 10),
PicItemSizeHeight / (float)YRatio1 + (float)(lengthY / 10));
                Paths.Add(p1);
                //MessageBox.Show(p1.X + ","+p1.Y);
                PointF p2 = new PointF(PicItemSizeWithd / (float)XRatio2 + (float)(lengthX / 10),
PicItemSizeHeight / (float)YRatio2 + (float)(lengthY / 10));
                //MessageBox.Show(p2.X + "," + p2.Y);
                Paths.Add(p2);
            }
        }
        private void MakePoints()
        {
            // Make sure there is at least one seed.
            if (Seeds.Count < 1)
            {
                //MessageBox.Show("Please define at least one seed first.");
                return;
            }

            // Clear the Centroids list.
```

```
    Centroids.Clear();

    // Make the points.
    Random rand = new Random();
    double max_r = Math.Min(
        PicItemSizeWithd,
        PicItemSizeHeight)/* / 6*/;

    picItems.Refresh();
}

private double Score(List<PointF> centroids, List<PointData> points)
{
    double total = 0;
    foreach (PointData point_data in points)
    {
        total += Distance2(point_data.Location,
            centroids[point_data.ClusterNum]);
    }
    return total;
}

private double Distance2(PointF point1, PointF point2)
{
    float dx = point1.X - point2.X;
    float dy = point1.Y - point2.Y;
    return dx * dx + dy * dy;
}
private double Distance(PointF point1, PointF point2)
{
    float dx = point1.X - point2.X;
    float dy = point1.Y - point2.Y;
    return Math.Sqrt(dx * dx + dy * dy);
}

private void UpdateSolution()
{
    // Find new centroids.
    int num_clusters = Centroids.Count;
    PointF[] new_centers = new PointF[num_clusters];
    int[] num_points = new int[num_clusters];
    //textBoxLog.Text += "Update";
    foreach (PointData point in Points)
    {
        double best_dist = Distance(point.Location, Centroids[0]);
        int best_cluster = 0;
        for (int i = 1; i < num_clusters; i++)
        {
```

```
            double test_dist =
                Distance(point.Location, Centroids[i]);
            if (test_dist < best_dist)
            {
                best_dist = test_dist;
                best_cluster = i;
                point.ClusterCount = i;
            }
        }
        point.ClusterNum = best_cluster;

        new_centers[best_cluster].X += point.Location.X;
        new_centers[best_cluster].Y += point.Location.Y;
        num_points[best_cluster]++;
    }

    // Calculate the new centroids.
    List<PointF> new_centroids = new List<PointF>();
    for (int i = 0; i < num_clusters; i++)
    {
        new_centroids.Add(new PointF(
            new_centers[i].X / num_points[i],
            new_centers[i].Y / num_points[i]));
    }

    // See if the centroids have moved.
    bool centroids_changed = false;
    for (int i = 0; i < num_clusters; i++)
    {
        const float min_change = 0.5f;
        if ((Math.Abs(Centroids[i].X - new_centroids[i].X) > min_change) ||
            (Math.Abs(Centroids[i].Y - new_centroids[i].Y) > min_change))
        {
            centroids_changed = true;
            break;
        }
    }
    if (!centroids_changed)
    {
        tmrUpdate.Enabled = false;
        FinalizedCluster();
        Cursor = Cursors.Default;
        return;
    }
    // Update the centroids.
    Centroids = new_centroids;
}
private void btnMakeClusters_Click(object sender, EventArgs e)
```

```
{
    int num_clusters = Seeds.Count;
    MaxClusters = Seeds.Count;
    if (Points.Count < num_clusters)
    {
        //MessageBox.Show("No Centroid selected");
        return;
    }
    // Reset the data.
    // Pick random centroids.
    Centroids = new List<PointF>();
    Points.Randomize();
    for (int i = 0; i < num_clusters; i++)
        Centroids.Add(Seeds[i]);
    //foreach (PointData point_data in Points)
    //    point_data.ClusterNum = 0;
    //FinalizedCluster();
    NumSteps = 0;
    picItems.Refresh();
    //lblScore.Text = "";
    Cursor = Cursors.WaitCursor;
    tmrUpdate.Enabled = true;
}
private void tmrUpdate_Tick(object sender, EventArgs e)
{
    NumSteps++;
    UpdateSolution();
    picItems.Refresh();
}
private int Score()
{
    float score = 0;
    foreach (PointData point in Points)
    {
        float dx = Centroids[point.ClusterNum].X - point.Location.X;
        float dy = Centroids[point.ClusterNum].Y - point.Location.Y;
        score += dx * dx + dy * dy;
    }
    return (int)score;
}
private void picItems_MouseClick(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left && isCentroidEnabled)
    {
        Seeds.Add(e.Location);
        Centroids.Clear();
        picItems.Refresh();
    }
```

```
            else if (e.Button == MouseButtons.Left && isPolygonalActive)
            {
                Vertis.Add(e.Location);
                picItems.Refresh();
            }
        }
        private void btnClear_Click(object sender, EventArgs e)
        {
            ClearAll();
        }
        private void CentroidActivator_Click(object sender, EventArgs e)
        {
            isCentroidEnabled = !isCentroidEnabled;
            isPolygonalActive = false;
        }
        private void btnCleanCentroid_Click(object sender, EventArgs e)
        {
            Seeds.Clear();
            Centroids.Clear();
            picItems.Refresh();
        }
        private void PolyDraw_Click(object sender, EventArgs e)
        {
            isPolygonalActive = !isPolygonalActive;
            isCentroidEnabled = false;
        }
        bool IsPointInPolygon(PointF[] polygon, PointF testPoint)
        {
            int count = polygon.Length;
            bool inside = false;

            for (int i = 0, j = count - 1; i < count; j = i++)
            {
                if (((polygon[i].Y > testPoint.Y) != (polygon[j].Y > testPoint.Y)) &&
                    (testPoint.X < (polygon[j].X - polygon[i].X) * (testPoint.Y - polygon[i].Y) /
(polygon[j].Y - polygon[i].Y) + polygon[i].X))
                {
                    inside = !inside;
                }
            }

            return inside;
        }
        private void RemovePoints_Click(object sender, EventArgs e)
        {
            TempPoints = new List<PointData>();
            //int counter = 0;
            for (int i = 0; i < Points.Count; i++)
```

```
            {
                if (!IsPointInPolygon(_points, new PointF(Points[i].Location.X,
Points[i].Location.Y)))
                {
                    TempPoints.Add(new PointData(Points[i].Location.X, Points[i].Location.Y,
Points[i].ClusterNum, Points[i].ID, Points[i].ClusterCount));
                    //counter++;
                }
            }
            //textBoxLog.Text = "Counter: "+counter.ToString();
            Points.Clear();
            Points = TempPoints;
            Seeds.Clear();
            Vertis.Clear();
            Centroids.Clear();
            //Points.Clear();

            picItems.Paint += new System.Windows.Forms.PaintEventHandler(this.picItems_Paint);
            picItems.Refresh();
        }
        void FinalizedCluster()
        {
            StreamWriter writer = new StreamWriter(Directory.GetCurrentDirectory() +
"/Clusters.asc");
            TempPoints = new List<PointData>();
            foreach (PointData p in Points)
            {
                foreach (PointData pd in PointsBackUp)
                {
                }

            }
            writer.Close();
            MyClustersPoint = new List<List<PointF>>();

            for (int i = 0; i < Seeds.Count; i++)
            {
                MyClustersPoint.Add(new List<PointF>());
                borderLinePoints.Add(new List<PointF>());
                for (int j = 0; j < Points.Count; j++)
                {
                    if (Points[j].ClusterNum == i)
                    {
                        MyClustersPoint[i].Add(Points[j].Location);
                    }
                }
                borderLinePoints[i] = MakeBorderLine(MyClustersPoint[i]);
            }
```

```
            IntersectionFinder.Convert2Coordinate();
            StreamWriter LineWriter = new StreamWriter(Directory.GetCurrentDirectory() +
"/NewLines.asc");
            for(int l = 0 ; l < IntersectionFinder.NewLines.Count; l++)
            {
                LineWriter.WriteLine(IntersectionFinder.NewLines[l].ToString());
            }
            LineWriter.Close();
            Seeds.Clear();
            Vertis.Clear();
            Centroids.Clear();
            Points.Clear();
            isCentroidEnabled = false;
            isPolygonalActive = false;
        }
        public static List<PointF> MakeBorderLine(List<PointF> _MyClustersPoint)
        {
            List<PointF> convexHull = new List<PointF>(ConvexHull(_MyClustersPoint));
            List<PointF> borderPoints = new List<PointF>();
            for (int k = 0; k < convexHull.Count; k++)
            {
                PointF current = convexHull[k];
                PointF next = convexHull[(k + 1) % convexHull.Count];
                PointF prev = convexHull[(k - 1 + convexHull.Count) % convexHull.Count];

                if (current == prev || current == next)
                {
                    // Skip points that are adjacent to the same point
                    continue;
                }

                if (!IsCollinear(current, next, prev))
                {
                    borderPoints.Add(current);
                    // Add the current point to the border line points
                    //borderLinePoints[i].Add(current);
                    //MyClustersPoint[i].Add(ClusterPoints[k].Location);
                }
            }
            return borderPoints;
        }
            foreach (PointF p in sortedPoints)
            {
                while (upperHull.Count >= 2 &&
                       CrossProduct(upperHull[upperHull.Count - 2], upperHull[upperHull.Count -
1], p) <= 0)
                {
                    upperHull.RemoveAt(upperHull.Count - 1);
```

```
                }
                upperHull.Add(p);
            }

            // Find the lower hull
            List<PointF> lowerHull = new List<PointF>();
            for (int i = sortedPoints.Count - 1; i >= 0; i--)
            {
                PointF p = sortedPoints[i];
                while (lowerHull.Count >= 2 &&
                       CrossProduct(lowerHull[lowerHull.Count - 2], lowerHull[lowerHull.Count -
1], p) <= 0)
                {
                    lowerHull.RemoveAt(lowerHull.Count - 1);
                }
                lowerHull.Add(p);
            }

            // Combine the upper and lower hulls
            lowerHull.RemoveAt(lowerHull.Count - 1);
            upperHull.AddRange(lowerHull);
            return upperHull;
        }
        static float CrossProduct(PointF A, PointF B, PointF C)
        {
            float crossP = (B.X - A.X) * (C.Y - A.Y) - (B.Y - A.Y) * (C.X - A.X);
            //Console.WriteLine(crossP);
            return crossP;
        }
        // Collinear function
        static bool IsCollinear(PointF A, PointF B, PointF C)
        {
            return Math.Abs(CrossProduct(A, B, C)) < 0.000000000001;
        }

        private void CavityClustering_Load_1(object sender, EventArgs e)
        {
            PicItemSizeWithd = picItems.Width;
            PicItemSizeHeight = picItems.Height;
            //MessageBox.Show(picItems.Width + "," + picItems.Height);
            GetPoints();
        }

        void ClearAll()
        {
            Seeds.Clear();
            Vertis.Clear();
            Centroids.Clear();
```

```
            Points.Clear();
            picItems.Refresh();
        }
        private void CavityClustering_FormClosing(object sender, FormClosingEventArgs e)
        {
            ClearAll();
            e.Cancel = true;
            this.Hide();
        }
    }
}
```