DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Frontend development of edge-based monitoring systems for additive manufacturing

Sumit Bhirud

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Frontend development of edge-based monitoring systems for additive manufacturing

Frontend-Entwicklung von Edge-basierte Überwachungssysteme für die additive Fertigung

Author:	Sumit Bhirud
Supervisor:	Prof. Dr. Pramod Bhatotia
Advisor:	Dr. Philip Howell
Submission Date:	29.04.2022

I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 29.04.2022 Sumit Bhirud

Acknowledgments

I would first like to thank my advisor Dr. Philip Howell for his guidance and constructive criticism that has helped me understand the research field better. He has been a true mentor and his high standards of quality were really inspiring and continuously motivating. I have learned many things from him including the basics of framing a research idea to tackling the disruptive research questions that come along.

I would also like to thank my supervisor Prof. Dr. Pramod Bhatotia for allowing me to start this research for the chair of Decentralized Systems Engineering in the Department of Informatics at the Technical University of Munich (TUM).

A special thank you goes to my colleagues at Siemens specifically Tobias Kamps, Ingmar Hofmann, and Raven Reisch for welcoming me into their team, supporting my work, and most importantly building an idea into reality together as a team. I would like to recognize them for their invaluable feedback and would also like to thank them for their guidance throughout all the phases of this thesis.

Abstract

Additive manufacturing (AM) is an upcoming technology through which objects with complex structures and forms can be constructed as one single unit. Compared to traditional manufacturing methods, it provides better accuracy for complicated specimens which opens up the possibility of rapid prototyping. Direct Energy Deposition (DED) based Wire Arc Additive Manufacturing (WAAM) is the most cost-effective metal additive manufacturing process. Being a complicated process, it involves the use of various sensors and equipment to track the manufacturing process which in return generates a vast amount of data for respective parameters such as voltage, current, gas flow, temperature, velocity, wire feed, acoustic, etc. By gaining insights into this real-time data, users can perform optimization for the ongoing WAAM process or make adjustments for future manufacturing processes. To facilitate such optimizations, an easy-to-use and functional monitoring dashboard is needed.

In this work, the proposed monitoring dashboard is developed as a web-based application using Vue.js. The dashboard displays time-series and 3D visualization of the selected sensors. It also allows monitoring of multiple sensors at a time and provides a user-friendly interface to manage the timeline of the visualizations. Various algorithms are integrated to improve dashboard performance allowing the user to track a vast amount of data from the WAAM process with ease.

Kurzfassung

Die additive Fertigung (AM) ist eine aufstrebende Technologie, durch die Objekte mit komplexen Strukturen und Formen als eine einzige Einheit konstruiert werden können. Im Vergleich zu herkömmlichen Herstellungsverfahren bietet es eine bessere Genauigkeit für komplizierte Proben, was die Möglichkeit des Rapid Prototyping eröffnet. Die auf Direct Energy Deposition (DED) basierende Wire Arc Additive Manufacturing (WAAM) ist das kostengünstigste additive Metallfertigungsverfahren. Da es sich um einen komplizierten Prozess handelt, müssen verschiedene Sensoren und Geräte verwendet werden, um den Herstellungsprozess zu verfolgen, was wiederum eine große Menge an Daten für entsprechende Parameter wie Spannung, Strom, Gasfluss, Temperatur, Geschwindigkeit, Drahtvorschub, Akustik usw. generiert Durch Einblicke in diese Echtzeitdaten können Benutzer Optimierungen für den laufenden WAAM-Prozess durchführen oder Anpassungen für zukünftige Fertigungsprozesse vornehmen. Um solche Optimierungen zu erleichtern, wird ein einfach zu bedienendes und funktionales Überwachungs-Dashboard benötigt.

In dieser Arbeit wird das vorgeschlagene Monitoring-Dashboard als webbasierte Anwendung mit Vue.js entwickelt. Das Dashboard zeigt Zeitreihen und 3D-Visualisierung der ausgewählten Sensoren an. Es ermöglicht auch die gleichzeitige Überwachung mehrerer Sensoren und bietet eine benutzerfreundliche Oberfläche zur Verwaltung der Zeitachse der Visualisierungen. Verschiedene Algorithmen sind integriert, um die Leistung des Dashboards zu verbessern, sodass der Benutzer eine große Datenmenge aus dem WAAM-Prozess problemlos verfolgen kann.

Contents

Acknowledgments **iii**

Abstract **iv**

Kurzfassung **v**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Objectives . . . 3
1.3 Thesis Structure . . . 4

2 Background **5**
2.1 Front-end Frameworks . . . 5
2.1.1 React . . . 5
2.1.2 Angular . . . 5
2.1.3 Vue.js . . . 6
2.2 Build Tools . . . 6
2.2.1 Vue CLI . . . 6
2.2.2 Bili . . . 6
2.2.3 Vite . . . 7
2.3 Bundlers . . . 7
2.3.1 Webpack . . . 7
2.3.2 Rollup . . . 7
2.3.3 Parcel . . . 7
2.3.4 Esbuild . . . 8
2.4 Software Development Models . . . 8
2.4.1 Waterfall . . . 8
2.4.2 Iterative . . . 9
2.4.3 Spiral . . . 10
2.4.4 Scrum . . . 10
2.5 Third-party Libraries Integration . . . 11
2.5.1 Plotly.js . . . 11
2.5.2 Babylon.js . . . 12
2.5.3 Element-plus . . . 12

3 Overview **13**
3.1 Suitable Framework . . . 13
3.1.1 Comparison of top three front-end frameworks . . . 14
3.2 Suitable Build Tool and Bundler . . . 15
3.3 Collaboration . . . 15
3.3.1 Stand-up . . . 16
3.3.2 hackathon . . . 16

4 Design **17**
4.1 Design Overview . . . 17
4.2 Architecture . . . 18
4.3 Data Overview . . . 18
4.4 View Hierarchy . . . 21

5 Implementation **22**
5.1 Login Page . . . 22
5.2 Print Jobs View . . . 22
5.3 Monitoring View . . . 24
5.3.1 Time Series Visualisation . . . 24
5.3.2 3D Visualisation . . . 29
5.3.3 Global Slider . . . 30
5.3.4 Global Navigator . . . 32
5.4 Performance Optimisation . . . 33
5.4.1 Aggregation Level . . . 33
5.4.2 Dynamic manipulation of Aggregation Level . . . 34
5.4.3 Vuex - Global State Management . . . 34

6 Evaluation **35**
6.1 Qualitative Analysis . . . 35
6.2 Quantitative Analysis . . . 36
6.2.1 Scaffolding . . . 37
6.2.2 Dev server . . . 37
6.2.3 Building for production . . . 38
6.2.4 Results . . . 38

7 Related Work **39**
7.1 MATLAB GUI . . . 39
7.2 In situ monitoring system . . . 40
7.3 Arc analyzer . . . 41
7.4 Comparison of related work . . . 42
7.4.1 MATLAB GUI . . . 42
7.4.2 In situ monitoring system . . . 43
7.4.3 Arc Analyzer . . . 43

8 Summary and Conclusion **45**
8.1 Summary 45
8.2 Conclusion 46

9 Future Work **47**
9.1 CSV Imports 47
9.2 Global Navigator Selector 47
9.3 Parametrization 47
9.4 Transparent Anomaly Indicator 48

List of Figures **49**

List of Tables **51**

Bibliography **52**

1 Introduction

1.1 Motivation

Additive manufacturing (AM), also known as 3D printing, is a transformative approach to industrial production that enables the creation of lighter, stronger parts and systems [1]. It is a layer-based manufacturing process aimed at producing parts with complex geometric shapes and reducing material use and weight [2]. By depositing the material layer by layer under the control of a computer, AM offers significant advantages in saving raw materials and reducing the processing cycle.
Metal additive manufacturing is considered to be the most difficult cutting-edge technology among all AM technologies that most directly serve the equipment manufacturing industry. Metal powder and metal wire are usually used as the deposited materials in metal additive manufacturing. Direct Energy Deposition (DED) based Wire arc approach reduces raw material cost as shown in Table 1.1, and also avoids the use of powder protection and recycling system.
DED uses a laser, electron beam, or electric arc as the heat source to quickly melt metal wire

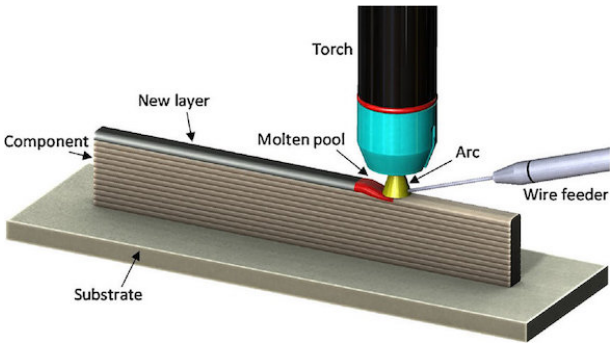

Figure 1.1: Working of WAAM system [3].

or metal powder, forming molten droplets which accumulate on the substrate layer by layer and finally become additive product [5]. DED also has a much higher material deposition rate compared to Powder Bed Fusion (PBF) and is more suitable for the economic production of large metallic components. Hence, wire arc additive manufacturing (WAAM) is gaining popularity due to its efficiency and cost advantage in large-scale industrial production. As seen in Fig. 1.1, the WAAM system is mainly composed of an arc welding machine responsible for providing the heat source to melt the metal wire and an execution unit like an industrial robot or Computer Numerical Control (CNC) machines that are responsible to manage the deposition process by controlling the movement of the welding torch. These components work

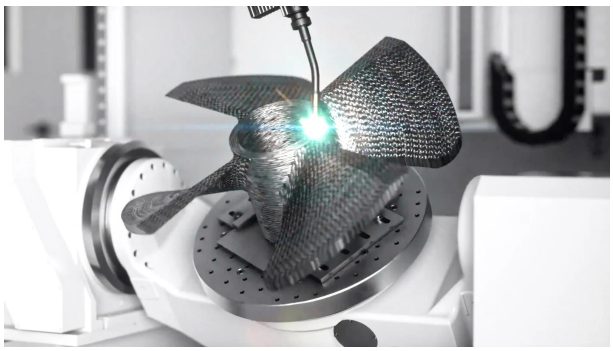

Figure 1.2: WAAM example [4].

together along a defined path by melting the wire and depositing it layer by layer to finally manufacture a product as seen in Fig. 1.2. WAAM is an emerging AM technology with many

Table 1.1: Approximate cost of different metal materials in wire and in powder [6].

Feedstock	Ti–6Al–4V	Inconel 718	Inconel 625	Stainless Steel 316L
Wire (£/kg)	120	58	49	12
Powder (£/kg)	280	80	80	40

technical challenges like high levels of inbuilt residual stress, porosity, deformation, crack, and delamination [5]. Effective monitoring and control of the WAAM process is important for enabling consistent and accurate component generation. Several sensors are used during the WAAM process to monitor different parameters like gas flow, voltage, current, acoustic, temperature, velocity, and many more [7]. These sensors generate a vast amount of data, for example, 44100 data points per second for acoustic sensor, 4000 data points per second for voltage sensor, and 10 data points per second for gas flow.

Data visualisation applications like Tableau [8] are popular for analysing a vast amount of data as they allow users to transform and visualize data from heterogeneous sources without programming but just analysis of data is not enough for AM processes. Since the post-process quality assurance of large parts is costly and time-consuming, process monitoring is inevitable [9] and to facilitate this the sensor data should be monitored in real-time with the help of 2D and 3D visualisation to understand and debug problems with the ongoing production job.

The Web-based application allows the creation of a user-friendly monitoring system with different types of visualisation like time-series and 3D views in real-time. The web application can also include various algorithms and functions which would help make a flexible monitoring system and which would also allow the possibility of improving the performance of the system. With the responsive design of the web application, it also allows users to access the dashboard from multiple platforms and with various screen sizes so that monitoring of the WAAM process won't be limited to users with specific screen size hardware.

1.2 Objectives

The following section presents the research questions that are based on the motivation of this thesis.
RQ1: How can the additive manufacturing data visualisation be combined with different aspects of the monitoring system for creating a user-friendly and easy to analyze dashboard?

This research question will be answered through the use of different components and how their co-dependency works together to provide a categorized and structured visualisation of additive manufacturing data.

RQ2: How can the monitoring system be made more flexible and modularised to support various types of print jobs?

This research question aims to provide a process for dynamically customizing the monitoring dashboard based on the requirements of various facilities and different types of print jobs.

RQ3: How can monitoring visualisation performance be improved for the dashboard?

This research question will be answered through the implementation of various concepts for improving dashboard performance. The outcome of this research question aims to provide more insights into how a vast amount of data is processed in monitoring systems.

1.3 Thesis Structure

This section gives short descriptions for the chapters of this thesis.

Chapter 1: Introduction provides the motivation of the thesis and the research questions to be answered. The approach to answering these questions are also explained.

Chapter 2: Background introduces the theoretical concepts that are required for a better understanding of this thesis. It describes the front-end frameworks, build tools, bundlers, software development models, and third-party libraries involved in the development.

Chapter 3: Overview compares tools and concepts defined in the background chapter and selects a suitable framework, build tool, and bundler for the monitoring system. It also explains the software development model and collaboration efforts carried out by the team.

Chapter 4: Design shows requirements and features from the monitoring system set forth by the product owner.

Chapter 5: Implementation presents the architecture and design principles of the implementation. It also shows different components or screens involved in the dashboard. From there, it continues by explaining the technical development process of each component. Finally, it also talks about the performance optimization carried out for a better user experience.

Chapter 6: Evaluation provides the findings of this thesis and gives insights into the qualitative and quantitative analysis of the dashboard.

Chapter 7: Related Work details similar implementations performed to monitor the WAAM process.

Chapter 8: Summary and Conclusion wraps up the work done during this research in a detailed summary and briefly concludes the research.

Chapter 9: Future Work provides improvement possibilities for the implemented monitoring dashboard.

2 Background

The following sections are dedicated to theoretical foundations that are essential for understanding the topics that are going to be discussed in this thesis. The concepts that will be explained here are front-end frameworks, build tools, bundlers, software development models, and third-party libraries. These concepts are essential for understanding the software stack chosen to develop the monitoring dashboard for the WAAM process.

2.1 Front-end Frameworks

Front-end frameworks allow the creation of web applications with the use of HTML, CSS, and Javascript. There are many such frameworks available for building web applications. The Top three frameworks as discussed in section 3.1. are described below.

2.1.1 React

React is an open-source framework developed and created by Facebook. It stands apart because of its virtual Document Object Model (DOM), which presents its excellent functionality. A React application is a collection of discrete components, each representing a single view [10]. Having individual view components allows making changes to a single view or component without considering the entire system. React makes it painless to create an interactive user interface (UI). React enables the creation of simple views for each state in the application and React will efficiently update and render just the right components when particular data changes update the state of the application [11]. In react, code for a single component is structured in two main files, one is Cascading Style Sheets (CSS) file which holds styling rules and configuration for each class, and the Javascript file which holds logic as well as JSX which is a javascript extension for writing HTML. JSX produces React elements for the UI.

2.1.2 Angular

Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps [12]. Angular was developed and created by Google. Angular supplies developers with an efficient way to create single-page web applications and it also provides the developers with an end-to-end tool that will help to develop both front-end and back-end tools [13]. Angular efficiently creates the front-end by using the Model View Controller design pattern. In Angular, code for a single component is structured in 3 different files like a traditional front-end project. One is an HTML file to design the view, a CSS file to style the view, and a javascript file for logic.

2.1.3 Vue.js

Vue is an approachable, performant, and versatile framework for building web user interfaces. The major benefits of vue.js are visual DOM, component-based and it also employs two-way data binding. Vue.js can be used for constructing web applications, mobile applications, and progressive web apps (PWA). In Vue.js, any type of application interface can be abstracted into a component tree [14]. If the component is abstracted reasonably, several small components can be reused to build a large system, so that the repetitive development shall be reduced [15]. In Vue.js, code for a single component is structured in just one file. This one file includes 3 subsections for script i.e. logic, template i.e. the design of the view, and style i.e. styling the view elements.

2.2 Build Tools

Modern web applications are often a mix of custom-written modules and third-party packages, written in various versions of languages [16]. Various tools like Syntactically Awesome Style Sheets (SASS), Babel, Typescript, etc are to transform applications to be suitable for running in multiple browsers and sometimes in the server context. Such tools need to be configured to work properly across the platforms and this configuration is managed by Build Tools. Build tools help in making them work across operating systems, and more importantly between each other. Section 3.2. explains how one of the following popular build tools was chosen for monitoring dashboard application. These are in popular among the Vue community for building both web applications and libraries.

2.2.1 Vue CLI

Vue CLI is the official Vue.js build tool, which allows developers to focus on writing application code instead of configuring complex build processes. Projects created with Vue CLI have a certain ready-made initial structure, an integrated configuration, a set of basic files, as well as a configuration for Webpack [17]. Vue CLI being built on top of Webpack makes it ideal for large and complex web applications. It comes with a rich and extensive API and plugins which allows developers to leverage features like Babel, Unit and End-to-end Testing, PWA support, Instant Prototyping, Hot Module Replacement (HMR), etc.

2.2.2 Bili

Bili is a build tool meant to make bundling JavaScript libraries easier. It is built on top of Rollup bundler and its zero-config strategy allows library developers to focus on the development of the packages instead of focusing on the build process. It offers both a command-line interface and Node.js API. It also features various advantages like built-in ES2015+ transpilation via Buble and Babel, Typescript support, style extraction with pre and post-processing for Sass, Less, and Stylus along with support for CSS Modules.

2.2.3 Vite

Vite is a build tool that aims at providing a faster and leaner development experience for modern web projects. It first divides the modules in an application into two categories i.e. dependencies and source code which improves the development server start time. Vite pre-bundles dependencies using esbuild bundler which is 10-100x faster than other JavaScript-based bundlers. Vite serves source code over native ESM which is the official standard format to package JavaScript which is essentially letting the browser take over part of the job of a bundler. Vite transforms and serves source code on demand, as the browser requests it.

2.3 Bundlers

Bundlers are complex tools that take both small and large applications, often made out of multiple packages and sub-modules, transform them into a format, that is ready to work in the Browser, Node.js or both [16]. Section 3.2. explains how one of the following popular bundlers was chosen for monitoring dashboard application.

2.3.1 Webpack

Webpack is a static module bundler and is the most popular tool for bundling complex web applications. Webpack begins processing web applications firstly by building an internal dependency graph from one or more entry points and then combines every module the application needs into one or more bundles. It provides a widely configurable API and has an enormous plugin ecosystem. It can import JavaScript files as well as static assets like images and styles, and build a dependency graph that maps modules between each other, generating one or more application bundles. Webpack also provides capabilities for bundle splitting and lazy module loading with built-in highly customizable plugins as well as supporting sharing modules between different applications. [18].

2.3.2 Rollup

Rollup is a module bundler for JavaScript which compiles small pieces of code into something larger and more complex, such as a library or application [19]. A new standardized format for code modules included in the ES6 revision of JavaScript is used by Rollup and hence it works better with the ES6 version of JavaScript modules, as it offers a superior API. It also allows transforming older javascript formats like CommonJS and AMD via plugins. The bundler offers both a command-line interface and a Node.js based API, Babel, and Buble support, Tree-Shaking (removes unused code), etc. The major difference to Webpack is that Rollup is much more lightweight having fewer dependencies and fewer out-of-the-box features [18].

2.3.3 Parcel

Parcel is a fast, zero-configuration web application bundler that aims to bundle the most complex applications without any configuration from the developer's side [17]. Like other

bundlers, Parcel enables code splitting and code transformation and also has a development server. Another great feature of Parcel is its parallelly performed multicore processing when doing the tree shaking and caching of the file system, which keeps the build times extremely fast [18].

2.3.4 Esbuild

Esbuild is an open-source JavaScript bundler written solely by Evan Wallace in Go language focusing mainly on the speed aspect of the bundling process. It is extremely fast and way better than other popular bundlers like Rollup, Webpack, and Parcel. Instead of following sequential processes, ESbuild fully parallelizes parsing, printing, and source map generation.

2.4 Software Development Models

Software development life cycle (SDLC) models show the ways to navigate through the complex and demanding process of software development [20]. Choosing the right model for development is important to ensure project's quality, timeframe, budget, and ability to meet the stakeholders' expectations. Each model has its favorable aspects and disadvantages and none of them is perfect for general software development projects. Section 3.3. explains the type of model used during the development of the monitoring dashboard.

2.4.1 Waterfall

The waterfall is one of the primitive models for software development. Through all development stages requirement analysis, design, implementation, testing, deployment, and maintenance, the process moves in a cascade mode. It starts with stage one and flows like a waterfall means never going back or giving feedback. The main disadvantage is that there is no feedback from one stage to another stage, means bugs cannot be detected until and unless whole stages are completed [21]. Each stage has defined deliverables and the next stage cannot start before the previous one is fully completed.

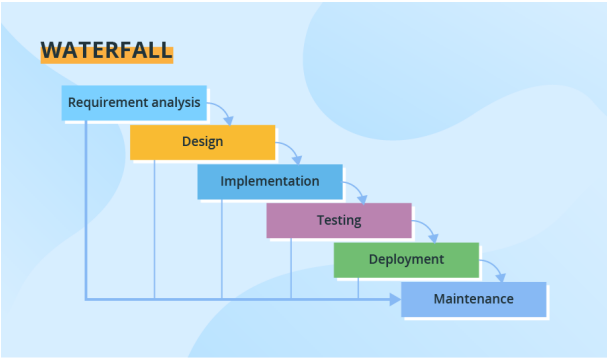

Figure 2.1: Waterfall Model [20]

Advantages:

- Easy to understand and follow.
- Simple implementation.

Disadvantages:

- Time consuming.
- No feedback facility.

2.4.2 Iterative

The iterative model is a modified version of the waterfall model. It has facility of giving feedback to the previous stage, with the help of this facility it is easy to rectify the error before going to the next stage [21]. As each iteration builds on the previous one, overall software design remains consistent and since the software is delivered in parts, there is no need for a full specification about the project from the beginning and some small changes to requirements are possible in the course of the development process.

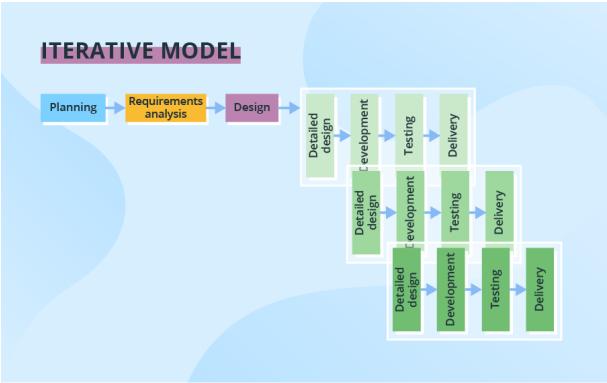

Figure 2.2: Iterative Model [20]

Advantages:

- Allows feedback and error rectification.
- Stages are well defined.

Disadvantages:

- Time consuming.
- No iteration after implementation.

2.4.3 Spiral

The spiral model was defined by Barry Boehm, based on experience with various refinements of the waterfall model. This model puts focus on risk assessment and hence to benefit from using such a model ability to evaluate risks is very important. A typical Spiral iteration has 4 main phases - planning, risk analysis, prototypes creation, and evaluation of the previously delivered product. Customer involvement appears in excess in this model and they can be involved in the exploration and review stages of each cycle. At the development stage, the customer's alterations are not acceptable.

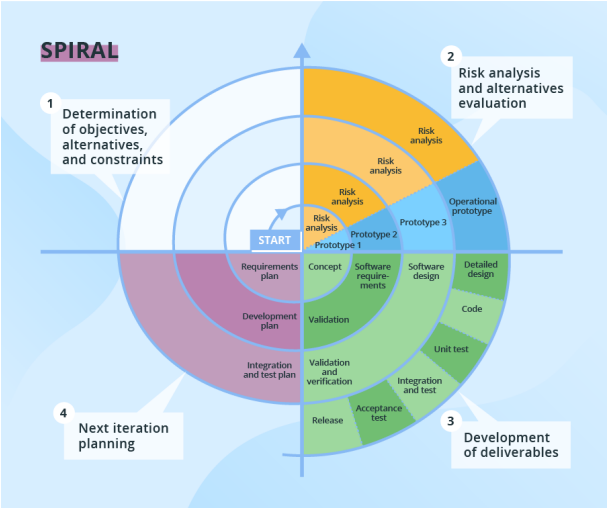

Figure 2.3: Spiral Model [20]

Advantages:

- Changing requirements can be accepted.
- Development stages are divided into smaller parts.

Disadvantages:

- It is complicated.
- Goals need to be well understood before development begins.

2.4.4 Scrum

Scrum is an iterative and incremental process for developing, delivering, and sustaining products in a complex environment. Being Agile, it is implemented by close collaboration both across the team and with the customers. Initially, the product owner creates a product backlog which is essentially a list of tasks that needs to be prioritized. Scrum team performs sprint planning where the product backlog list is broken down into smaller and more manageable tasks. Time duration is decided for this sprint and the scrum master is tasked with guiding

the team and ensuring the sprint is successful. The stakeholders and product owners conduct reviews at the end of each sprint to stay on track to the product backlogs generated in the beginning.

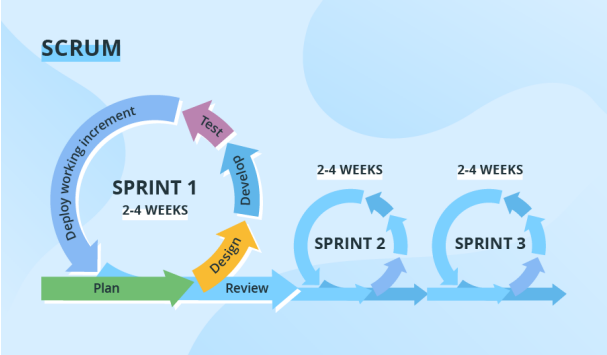

Figure 2.4: Scrum Model [20]

Advantages:

- Efficient use of time and money.
- Allows feedback from customers and stakeholders.

Disadvantages:

- Required experienced team members.
- Aggressive testing process needed to ensure quality.

2.5 Third-party Libraries Integration

Important aspects of the monitoring dashboard are the time-series and 3D visualisations and the general user interface of the front-end. To increase functionality, it is better to use third-party libraries that are available pre-configured for various use cases. Following are the most popular libraries for time-series visualisation, 3D visualisation, and design components of the web application.

2.5.1 Plotly.js

Plotly.js is the best and the most popular visualisation library [22]. Initially developed for python language is now extended to other languages like javascript and dart. It is built on top of d3.js and stack.gl and it allows plotting with over 40 chart types, including 3D charts, statistical graphs, and SVG maps. It abstracts the types of statistical and scientific charts that are available in packages like matplotlib, ggplot2, or MATLAB. Every parameter of a chart in plotly.js such as colors, gridlines, and the legend, has a corresponding set of JSON attributes that can be customized. Most Plotly graphs are drawn with SVG which offers great

compatibility across browsers and publication-quality vector image export. This library is highly used to showcase trends in data example time-series data.

2.5.2 Babylon.js

Babylon.js is a javascript open-source framework that is used to develop 3D applications/ video games for the web. It allows the creation of 3D visualisation based on any type of data.

2.5.3 Element-plus

Element-plus provides design components for Vue framework. It includes components and design principles for improving the user interface.

Components used in monitoring dashboard are:

- Menu - High level navigation menu across different pages.
- Dropdown - Drops a list of items for particular parameter.
- Slider - Continuous slider to allow sliding through values and setting a value for certain parameter.
- Switch - Toggle button to switch between two variables of the application.

3 Overview

This chapter focuses on explaining the chosen software stack as well as the development processes followed for the monitoring system. The reasons and concepts of this chapter will work as a base for further implementation.

3.1 Suitable Framework

Popularity is an important factor when choosing a suitable framework for web applications as the frameworks, in general, are dependent on the community for the creation of libraries which makes other developer's life easier. According to StackOverflow insights trends for various frameworks, as seen in Fig. 3.1, the top three frameworks for web application development as of 2022 are React, Angular, and Vue.js. Many frameworks failed to form a community and hence were out-performed by others that were developed by companies like Google and Facebook which had a tech community from the beginning. Among the top three, the Vue.js framework is the most optimized framework for the development of web applications [23].

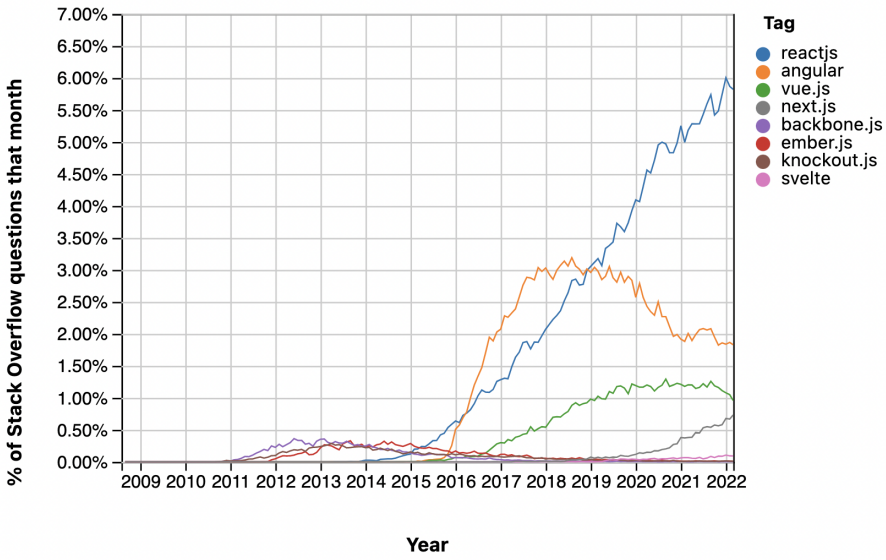

Figure 3.1: Stackoverflow Insight trends

3.1.1 Comparison of top three front-end frameworks

Factors	React	Angular	Vue.js
Readability	Yes	No	Yes
High Performance	Yes	No	Yes
Flexibility	No	No	Yes
Lighweight	No	No	Yes
Crisp documentation	No	No	Yes
Strong community	Yes	Yes	No
Long learning curve	Yes	Yes	No

Table 3.1: Comparison of front-end frameworks.

Comparison in above table shows Vue.js is the most suited framework for developing monitoring dashboard for additive manufacturing process.

The factors responsible for the selection of Vue.js are described in detail below:

Readability And Single-file Components:
Webpage in Vue.js can be divided into multiple components. These components signify snapshots or small pieces of the entire interface. In this framework, components can be written in CSS, JavaScript, and HTML without separating them into separate files as compared to Angular in which each component includes three different files for HTML, CSS, and Javascript code.

Lightweight:
The downloaded zip of the Vue.js framework weighs just 20 KB while Angular and React weigh 143 KB and 32 KB respectively. Being lightweight allows the download and installation of the library quick and also influences User Experience and Search Engine Optimization(SEO) positively.

Flexibility And Integration Capabilities:
emerging technologies need to be able to integrate with prevailing applications. With Vue, it becomes extremely easy as it depends only on JS frameworks and does not need anything new to operate on. While development in React and Angular is majorly dependent on libraries that work only with these particular frameworks.

High Performance:
Vue because of its simplicity and lightweight nature is extremely fast [24]. The advantages of high performing framework can be seen in user experience. Vue.js is also capable of launching mobile apps faster than Angular and React.

Crisp Documentation:

One of the most essential factors in front-end framework is its learning curve and that it exhibits all the choices offered by the framework and matches best practice instances. Vue.js documentation is constantly being updated and improved and is very detailed compared to React and Angular documentation. It also includes an overview of the API and a simple introductory guide. It is well-structured and contains all the probable topics that a developer might need, specifically unfolding everything from installation to in-depth aspects of the framework like scaling and reactivity of the app.

3.2 Suitable Build Tool and Bundler

Table 3.2 approximates a large JavaScript codebase by duplicating the three.js library 10 times and building a single bundle from scratch, without any caches. Hence for having an optimum performance for monitoring the dashboard ESbuild bundler with Vue.js is the best combination available. Vite build mentioned above uses ESBuild and hence Vite + Vue.js will be used for creating the dashboard.

Bundler	Time	Relative slowdown	Absolute speed	Output size
esbuild	0.33s	1x	1658.9 kloc/s	5.80mb
parcel	32.48s	98x	16.9 kloc/s	5.87mb
rollup	34.95s	106x	15.7 kloc/s	5.81mb
webpack	41.53s	126x	13.2 kloc/s	5.84mb

Table 3.2: Bundler Benchmark [25].

3.3 Collaboration

Development of the monitoring dashboard followed a hybrid model of software development, mainly a hybrid between waterfall, scrum, and spiral models due to knowledge gaps with new members and limited feedback from customers. Initially, it began as an individual work, understanding the concepts and focusing on the implementation of initial requirements like in the waterfall model. After delivery of the minimal product, collaboration among the team began as a scrum model. The Product owner decided on the product backlog and the team members along with the scrum master started planning high-level sprints to meet the product owner's requirements. Within these sprints, each individual followed a spiral model by analyzing the risks and developing in iterations for their respective areas involved in dashboard development. Stand-up meetings and hackathons were organized by the scrum master to stay on track with the sprints and allow the opportunity to perform tasks that require team involvement.

3.3.1 Stand-up

Stand-ups also known as the daily scrum, are one of the fundamental parts of agile development. Agile methods introduced the practice of daily stand-ups to improve communication in software development teams and projects [26]. A stand-up is a daily meeting that involves the core team: developers, and the scrum master. In these meetings, team members share their progress about their respective tasks, and this way everyone in the team becomes aware of what other members are doing. The intention is that team members will then align their actions to fit the actions of the others and proceed together to target the sprint deliverables. The daily reinforcement of sharing individual successes and plans keeps team members excited about the team's overall contribution and advancement toward product development.

3.3.2 hackathon

A hackathon is a meeting that spans around 4 - 5 hours and has defined start and end times. Within this period, team members are tasked with managing their backlog, planning out work, and breaking them down into further pieces. Hackathons set individual goals as well as collective goals to be reached within that particular time. Hackathons are used majorly for collaboration work like merging branches and performing collective work from different aspects of software development like frontend and backend.

4 Design

This chapter details the requirements set forth by the product owners for the development of a monitoring dashboard. Design screenshots and feature requirements are discussed along with high-level software architecture, view hierarchy, and data models overview.

4.1 Design Overview

Preliminary requirements for the monitoring dashboard were designed and described by the product owner. The following figures 4.1 and 4.2 shows the designs provided. The designs were well thought out according to the application and use case of the monitoring dashboard and also featured various features along with the designs.

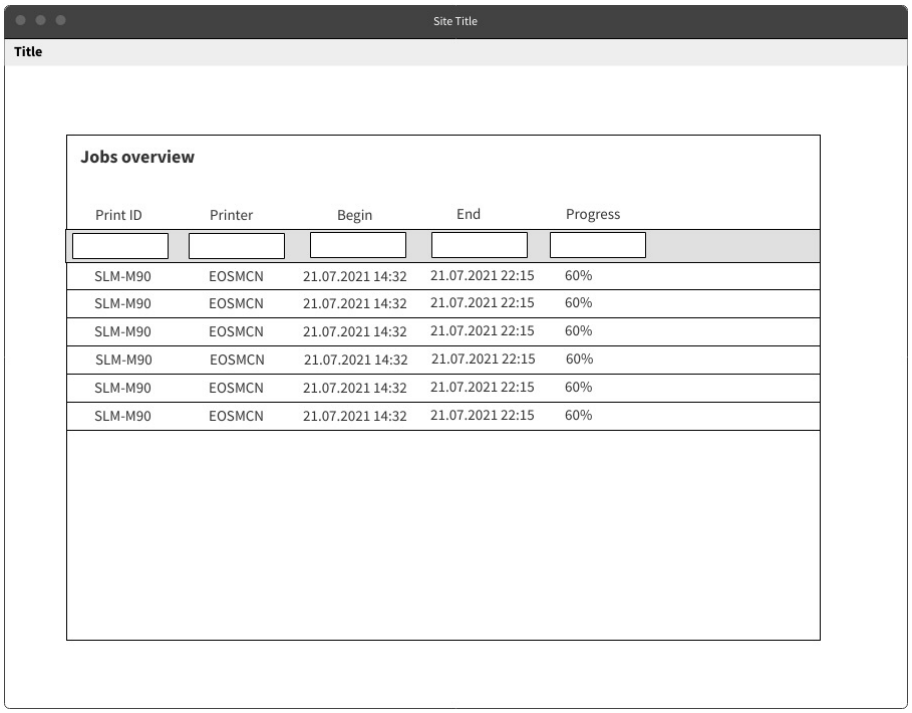

Figure 4.1: Print jobs design

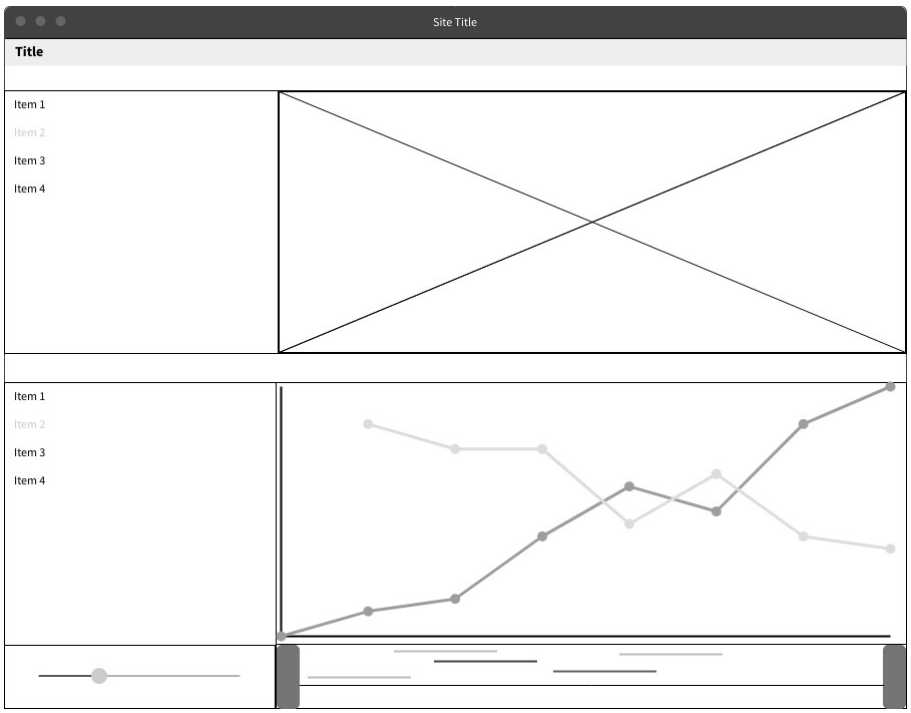

Figure 4.2: Monitoring design

4.2 Architecture

The monitoring dashboard is a web-based application that is split into a views layer and a data layer. The views layer includes the login view, jobs view, and monitoring view. The data layer consists of InfluxDB for time series data and SpatialDB for high-level data about jobs and their constituents. As seen in Fig. 4.3, the application is based on a parent-child relationship between components where Login, Print jobs, and Monitoring are parents at the same level and then there exist child components like 3D visualisation and time series in monitoring view. App data and configuration data are not passed from parent to child as in the traditional way but rather a more robust global state management is used such that all the components whether a parent or a child can access similar data.

4.3 Data Overview

AM dashboard requires different types of data to demonstrate the proper working of the monitoring system. Data is served via endpoints and different databases are used according to data type. Section 5.1. explains endpoints used to query print jobs list and general details about each print job. For monitoring purposes, two important areas are time series and 3D visualisation. SpatialDB serves Octrees and InfluxDB serves the time series data, both of which are explained below.

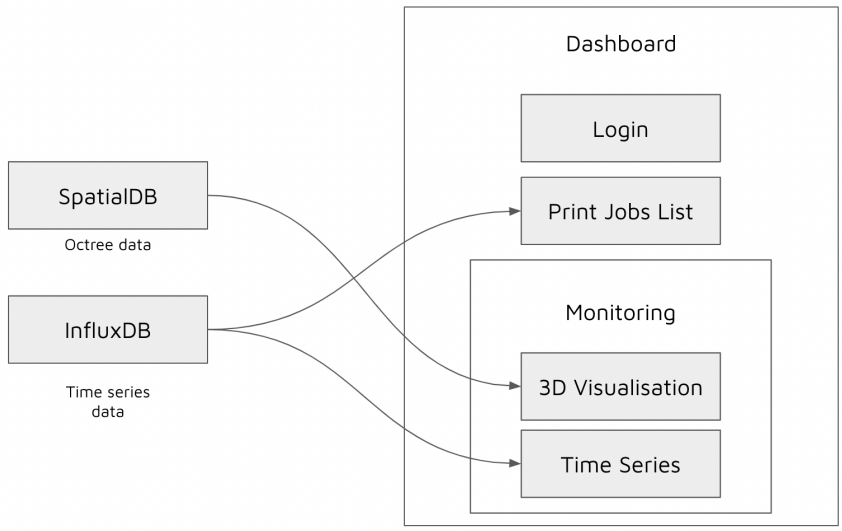

Figure 4.3: Dashboard Architecture

SpatialDB

Spatial data is represented by octrees which are based on nodes that efficiently enable spatial analysis [7]. Gathering 3D data in the monitoring dashboard occurs with a nested search in spatial DB. First search, location code based on the depth-first search gives access to the quadrant of the octree. This search is facilitated through the use of the following parameters:

- **Starting Octant** - location code of the octant.
- **Search Depth** - specifies buckets for points to split into.

A detailed second search is performed using location codes of these buckets to get corresponding sensor readings with parameters like:

- **Sensor** - list of sensors.
- **Aggregation type** - type of aggregation on the sensor values.
- **Start and End Time** - to limit the search further into specific time duration.

InfluxDB

InfluxDB manages the time series data which is gathered from the sensors. Time series data has 2 components, Timestamp, and value. Each sensor has different frequencies with which it captures data and hence every sensor will have a different number of data points per second in InfluxDB. SQL type commands can be used to query data from InfluxDB. Following are some example queries.

- Acoustic sensor data query which gives around 44100 data points per second.

  ```
  select acoustic from print_id
  ```

- Acoustic sensor data query using grouping to get only one value for 1 second.

  ```
  select mean(acoustic) from print_id group by time(1s)
  ```

- Acoustic sensor data query using grouping to get only one value for 1 second with fill operation to remove null entries.

  ```
  select mean(acoustic) from print_id group by time(1s) fill(none)
  ```

- Gathering "acoustic" sensor data using grouping to get only one value for 1 second.

  ```
  select mean(acoustic) from print_id group by time(1s)
  ```

- Multiple sensors Acoustic, Current and Gasflow data using grouping to get only one value for 1 second.

  ```
  select mean(/acoustic/current/gasflow/) from print_id group by time(1s)
  ```

4.4 View Hierarchy

The view hierarchy follows sequential structure. The user flow begins at login page as shown in Fig. 4.4. After authentication, user is directed to print jobs overview page where all the ongoing as well as completed jobs are listed. Print jobs view is explained in section 5.2. User gets redirected to monitoring view by clicking on a particular list item in print jobs page. Monitoring view contains the main solution part which solves WAAM process monitoring problem. Monitoring view is described in section 5.3.

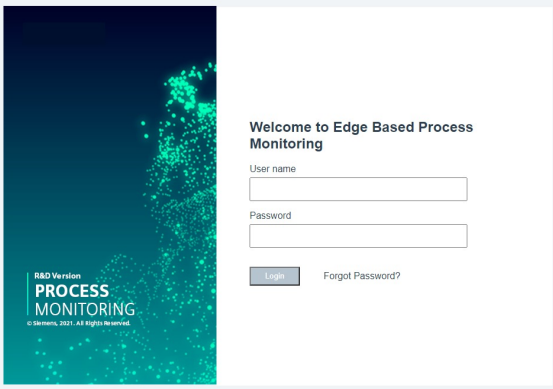

Figure 4.4: Login page

5 Implementation

This chapter describes the view-wise implementation details of the monitoring dashboard. Section 5.1. provides details on the login page. Section 5.2. shows the implementation of the print jobs view while section 5.3. explains the implementation of the monitoring view. Performance optimisation techniques used to improve and manage the vast amount of data can also be seen in section 5.4. In order to maintain confidentiality of the work done during this thesis, wireframes have been displayed instead of the actual monitoring system.

5.1 Login Page

The implementation of the dashboard started with preparing the login page for the application. Authentication is important to segregate access rights and to provide data access only to legitimate users. The implemented login page can be seen in Fig. 4.4. It allows users to authenticate themselves via username and password. Currently, only the user interface of the page is implemented and the actual authentication process with distinguished access rights will be implemented during later stages of the development of this monitoring dashboard.

5.2 Print Jobs View

Print Job Overview

Job	Printer	Begin	End	Progress	
print-257b6217	WAAM Robot Cell	05-03-2021 12:06:04		1.816	save CSV
print-420a993e	WAAM Robot Cell	05-03-2021 12:06:04		0.557	save CSV
print-43a2c9a9	WAAM Robot Cell	05-03-2021 12:06:04		0.775	save CSV
print-43e8ee6b	WAAM Robot Cell	05-03-2021 12:06:04		0	save CSV
print-5f103704	WAAM Robot Cell	05-03-2021 12:06:04		0.532	save CSV
print-93c191be	WAAM Robot Cell	05-03-2021 12:06:04		9.121	save CSV
print-99e6ed8b	WAAM Robot Cell	05-03-2021 12:06:04		1.007	save CSV
print-a5219983	WAAM Robot Cell	05-03-2021 12:06:04		0.569	save CSV
print-acd3086d	WAAM Robot Cell	05-03-2021 12:06:04		0.899	save CSV
print-b0d8ce2f	WAAM Robot Cell	05-03-2021 12:06:04		4.813	save CSV

Figure 5.1: Print jobs list

This view lists all the print jobs in a defined table of various columns. Users can find details like job ID, printer type, start time, end time, and progress of the jobs. The job ID is a unique identifier associated with every manufacturing process. The printer type shows the

type of printer used which will be fixed to "WAAM Robot Cell" as this monitoring dashboard is built for the WAAM process but can be extended further to accommodate other printer types and processes as well. Start and end time indicates the duration of the particular job while progress shows time in seconds to get progress status when the job is ongoing. The "Save CSV" button can be used to export the entire data available for a completed print job for long-term storage or sharing between users which can later be used to import into the dashboard to view that particular CSV file data.
To make it more convenient for the user each column consists of an input field to filter the tabular data as required. Users can also sort the tabular data alphabetically in ascending and descending order.
A local webserver hosts an Application programming interface (API) service for the front end that provides delete and query data endpoints. Following endpoints are used here:

- **GET /prints** - retrieves all prints in database and additional information.
- **POST /print/delete** - deletes print data based on print ID parameter.
- **POST /print/savecsv** - exports influxDB data as CSV to disk also based on print ID parameter.

The print ID or job ID mentioned before can be used to query data in monitoring view for that particular job. Once a user selects a print job, a state "print_id" in global state management is set which can be used in all other views and components of the dashboard.

5.3 Monitoring View

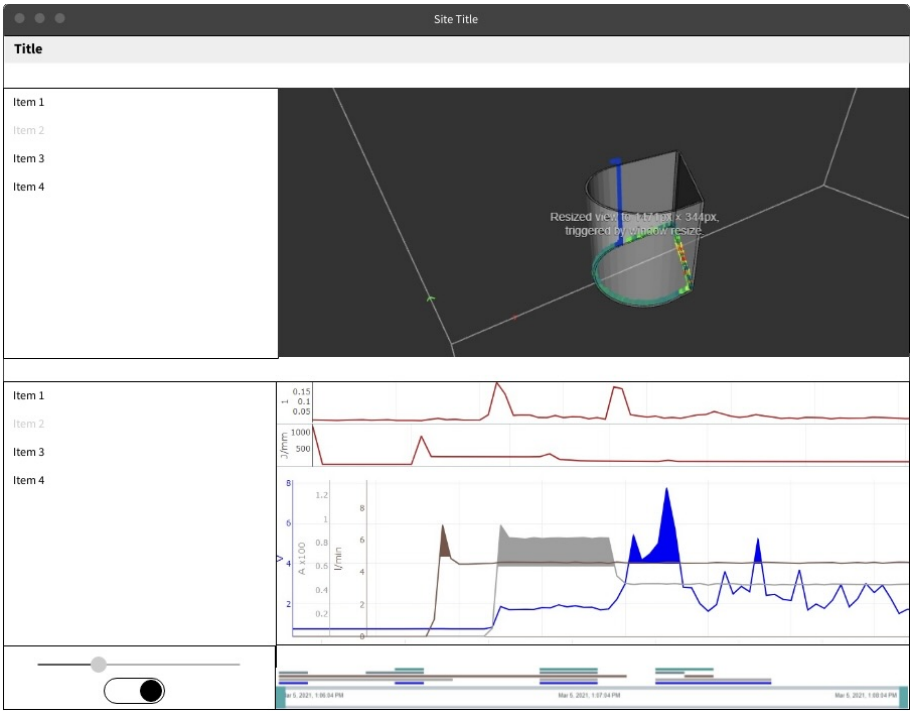

Figure 5.2: Monitoring View

As mentioned in chapter 1 about the vast amount of data produced by sensor monitoring of the WAAM process, it is not enough to just plot the charts for this data to make it more understandable and more debug friendly the charts need to be grouped with complementary components. The monitoring view consists of 4 child components that conclusively target the user-friendliness of the monitoring dashboard. Fig. 5.2 shows the overview of the monitoring page marked with 4 child components that will be explained in detail in further sections.

5.3.1 Time Series Visualisation

A time-series visualisation also called a time series graph or time series plot is a data visualization tool that illustrates data points at successive intervals of time. Each point on the chart corresponds to both a time and a quantity that is being measured. In a classic x-y graph, the horizontal axis of the chart is used to plot increments of time while the vertical axis pinpoints values of the sensor readings being measured. Plotly.js graphing library is discussed in section 2.5.1. is used for time series visualisation in this monitoring dashboard. There are many ways to visualise time-series data and plotly.js alone offers around 20 different charting types such as scatter plots, histograms, line charts, bar charts, bubble charts, contour plots, violin plots, etc. Line charts emphasize the overall trend of the data points and also line charts are better to have when trends of multiple sensors need to be analyzed at a time.

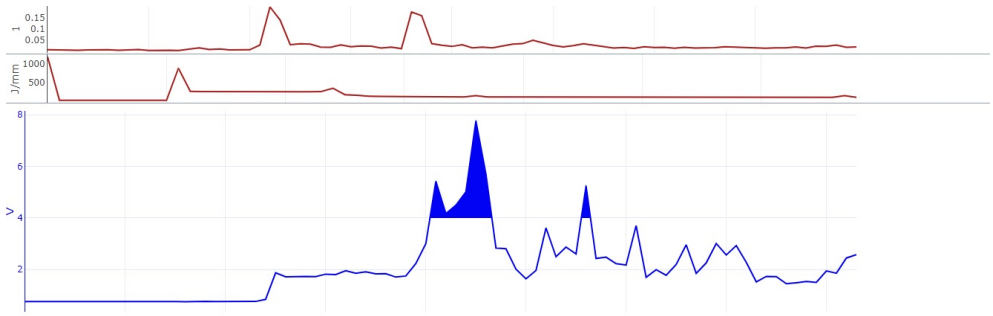

Figure 5.3: Time series visualisation

For those reasons line chart was selected for plotting time series data from InfluxDB.
A line graph is the simplest way to represent time series data. It helps the viewer get a quick sense of how something has changed over time. A line graph uses points connected by lines (also called trend lines) to show how a dependent variable and independent variable changed. Time is always the independent variable, which is plotted on the horizontal axis while the dependent variable or sensor value is plotted on the vertical axis. As seen in Fig. 5.3, the selected sensor is highlighted in the sensor list on the left side panel and this selected sensor data is plotted in blue color. Time series visualisation in the monitoring dashboard is not just a plot but a combination of different aspects that complement the main time series plot. More details about those complementary concepts can be seen below.

Anomaly Voltage and Power Over Path

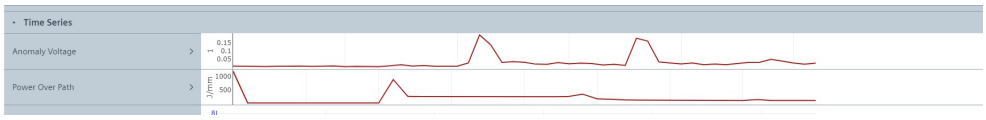

Figure 5.4: Anomaly voltage and power over path charts.

Anomaly voltage and power over path are the two most important Key Performance Indicators (KPIs) and hence are plotted separately above sensor plots. Anomaly voltage is a plot of anomaly scores calculated from voltage sensor data and power over the path is a combined reading that accumulates current, voltage, wire feed speed, and robot travel speed. All the plots of sensor data, anomaly voltage, and power over path share the same X-axis and have a similar timeline.

Critical readings indicator

Trend lines give insights into how the sensor value changes with time but this visualisation can be improved further for easy debugging or easy understanding of anomaly values of that particular sensor. Based on preliminary evaluation and planning of the print job, the user can set upper and lower limits of all the sensors. These limits indicate the anomaly behavior of

that sensor. Sensor values between those limits indicate that the job is being carried out as expected without any errors and when values go beyond those upper and lower limits then it indicates an anomaly and this specific time duration of anomaly needs to be checked to rectify ongoing job or even to avoid those anomalies in future print jobs.
Fig. 5.5. and Fig. 5.5. show trend lines of the same print job of the same sensor. Considering

Figure 5.5: Without critical indicator

Figure 5.6: With critical indicator

Fig. 5.5., extracting anomaly values is not difficult, the user can just look for y-axis values and then trace along the x-axis to know when the sensor crosses that line. Although this is not impossible or difficult, it is time-consuming. The WAAM process is a time-sensitive job and just one second can change the entire output of the print job. Hence, it is important to have a dashboard that reduces the time to debug issues allowing users to fix them much earlier compared to the traditional way of tracing values. As seen in figure 5.6., the filled area indicate anomaly duration and values of that sensor plot. Users can easily get the timestamp of those values and can look at all other factors to try to fix and understand the anomaly. Configuration of upper and lower limits of the particular sensor is defined in global state variables like shown in Fig. 5.7.

Multiple sensor plots

As seen in Fig. 5.3, the acoustic sensor was selected and a plot for the same was drawn in blue color. There are cases where it's important to group or see the plots of multiple sensors at the same time to deduce more information. To facilitate this instead of just allowing the selection of one sensor from the left side list, the dashboard provides multiple sensors selection. There is no limit to the number of sensors selected. The list of sensors is configured in the global store of the application in arrays and JSON format as seen in Fig. 5.9. All the selected sensor data will be visualised according to their respective color definitions. Fig. 5.8 shows the plot of 5 sensors namely, acoustic, current, gas flow, voltage, and wire feed.
As seen in Fig. 5.8, the X-axis is common for all the plots which facilitate viewing sensor values at one particular timestamp and the figure also shows multiple Y-axis. Each sensor having its Y-axis is better than having one single Y-axis. This helps in tackling the sensor range problem, since different sensors will have different ranges as well as different units, it gets difficult to differentiate between them with just one Y-axis. The range for each Y-axis is based on the minimum and maximum value of that particular sensor data, which means the

```
{
  upperLimitMapping: {
    acoustic: 4,
    current: 0.6,
    gasflow: 5,
    wirefeed: 400,
    voltage: 1.2,
    velocity: 2000
  },
  lowerLimitMapping: {
    acoustic: -4,
    current: -0.6,
    gasflow: -5,
    wirefeed: -400,
    voltage: -1.2,
    velocity: -4
  }
}
```

Figure 5.7: Upper and lower limit configuration of sensors.

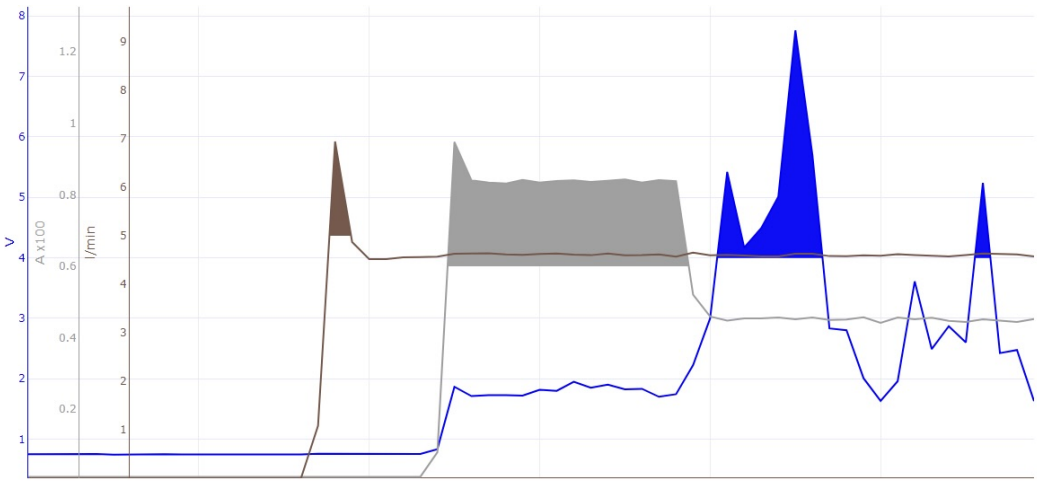

Figure 5.8: Multiple Sensor Plots

```
column_data: ["anomaly_voltage", "acoustic", "current", "gasflow", "voltage", "wirefeed", "temperature", "velocity"],

columnUnitsMapping: {
    anomaly_voltage: 'V',
    acoustic: 'V',
    current: 'A x100',
    gasflow: 'l/min',
    voltage: 'V x10',
    wirefeed: 'm/min',
    velocity: 'mm/min',
    temperature: 'Ã‚Â°C',
    power_over_path: 'J/mm'
},

columnColorMapping: {
    anomaly_voltage: '#3f51b5',
    acoustic: '#0000FF',
    current: '#9e9e9e',
    gasflow: '#795548',
    voltage: '#607d8b',
    wirefeed: '#009688',
    velocity: '#3f51b5',
    temperature: null,
}
```

Figure 5.9: JSON configuration of sensor list.

range is dynamic and keeps on updating as new data points are queried from the database. Multiple Y-axis also fixes outliers problem where if one sensor value goes beyond a certain extent then other sensor readings seem as a flat line. Each trend line also shows critical anomaly values area filled with respective sensor color.

Dynamic time slots

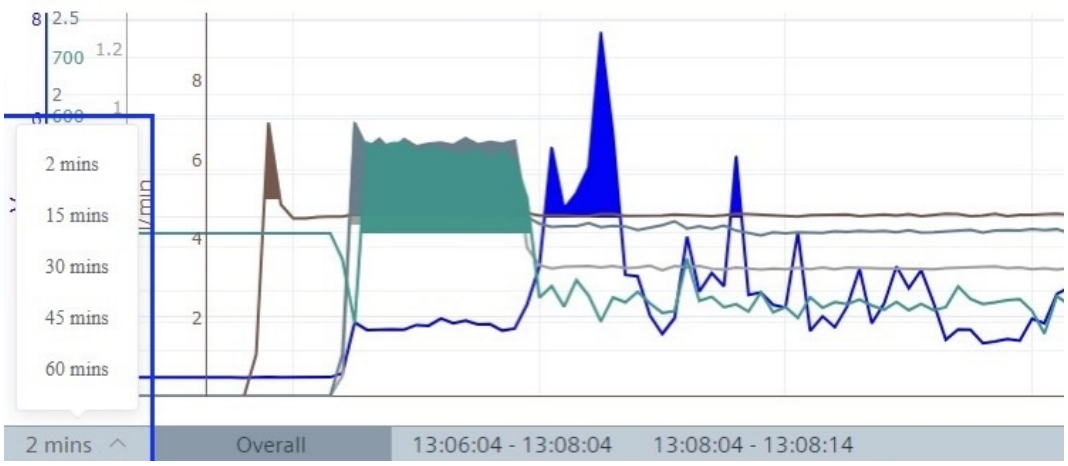

Figure 5.10: Time slot selection dropdown

A particular print job can last for around 1-2 hrs to 2-3 days. Assuming data availability of 2 days i.e 48 hrs of job, it is very difficult to zoom into 1 min of the job using the global slider. So even though with the help of a critical readings indicator it is easier to differentiate anomaly values, it is not easier to get details of timestamp and value readings if the job has been running for a long time. To fix this issue, dynamic time slot generation was introduced. In live mode, the slider spans across the entire lifespan of the job and hence is marked under

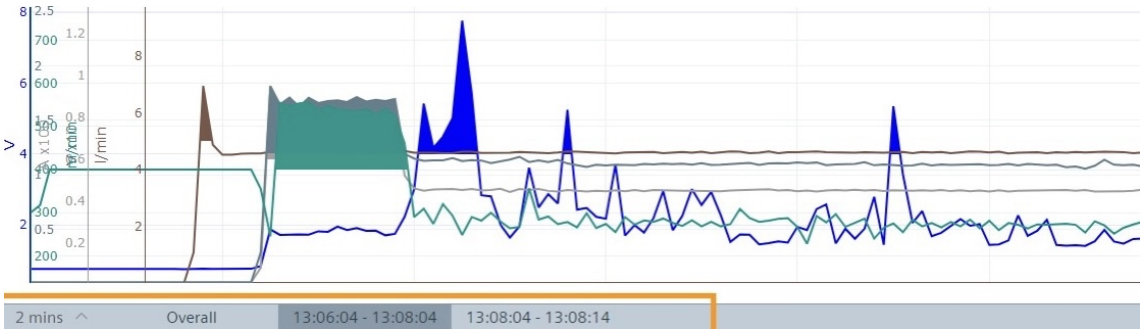

Figure 5.11: Selected time slot

the "Overall" time slot. Users can switch the mode to analyze and then jump to any time slot as shown on the right side of the "Overall" tab. These time slots are generated dynamically as time increases based on the slot duration set by the user. The dashboard allows user to set slot duration according to their needs. The acceptable values are shown in dropdown menu as shown in Fig 5.10. Once the slot is selected, all the time slots are adjusted according to the duration of the selected slot.

5.3.2 3D Visualisation

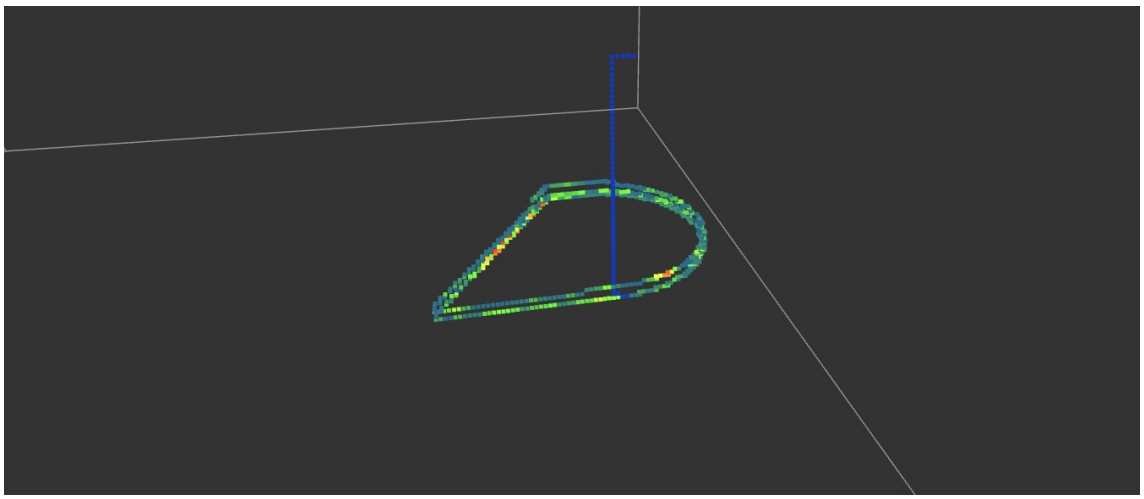

Figure 5.12: 3D Visualisation

As discussed in the above sections, each child component contributes to improving the user experience and functionality of monitoring a WAAM process. 3D visualisation is one of those child components that allow users to view the progress in 3D while tracking the plots in time series. 3D visualisation is implemented using Babylon.js and the data responsible for this visualisation is provided by spatial DB as described in section 4.3. 3D view also follows a similar timeline as the time series and selected sensor data can be seen in 3D as

multiple color-coded octant-shaped data points. Color codes in 3D visualisation indicate the critical nature of that particular data point. Fig. 5.12 shows a visualisation of spatial data

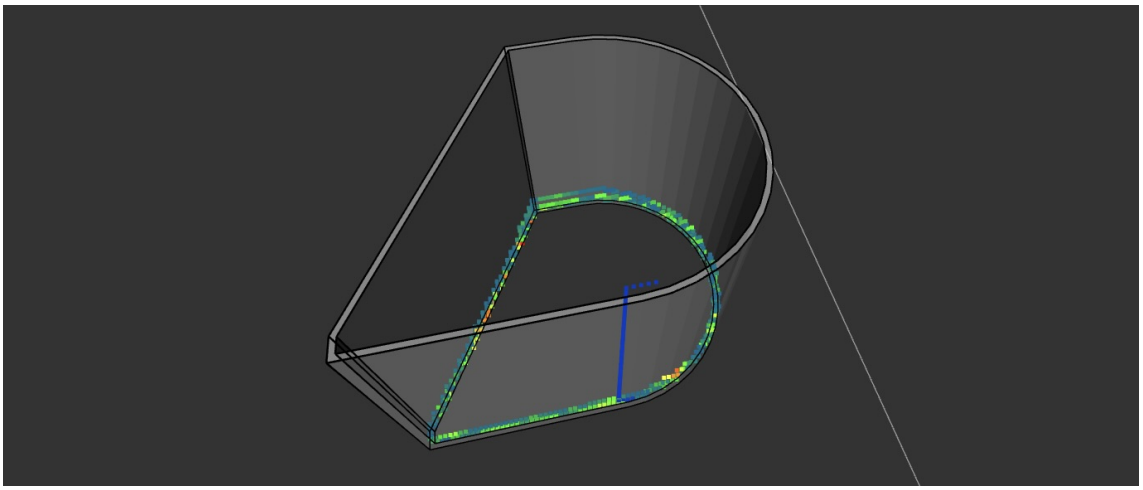

Figure 5.13: 3D visualisation with STL file

with maximum aggregation level as seen in the bottom right aggregation level slider which points to level 10. 3D view also allows users to get a better view as required by providing zoom, pan, and rotate functionality.
3D view adjusts its behavior according to the current mode of the job. In "Analyze" mode, a user can stop real-time query of the data and view current data points collected till this time. In "Live" mode, a new data point is added every 5 seconds and the user can see the model being built in 3D space like a simulation. The final product requirement can be seen by toggling the keyboard button "T". With this user can show/hide the final step file as seen in Fig. 5.13 in 3D space to compare and interpret the progress of the ongoing print job.

5.3.3 Global Slider

Time series and 3D visualisation have already made the dashboard more user-friendly and have allowed easy debugging and interpretation of data. This can be further improved by allowing users to get a more detailed view or more generalized view in a certain time frame. To facilitate users to control both the visualisation time interval, a global slider component is used.
The global slider can be used in **Analyze** mode and can allow users to manipulate visualisation based on the time interval set in the slider. The global slider has two ends with movable bars. Left end indicates **start time** and right end indicates **end time** of the visualisation. This extreme interval is set by selecting a time slot duration as shown in section 5.3.1. In the "Overall" time slot, the ends indicate the entire print job while other time slots indicate a particular interval as a part of the overall print job.
Both the ends of the slider can be moved and adjusted independently as well as the interval set by the ends can be fixed and this interval can be moved. To avoid duplicate queries to the

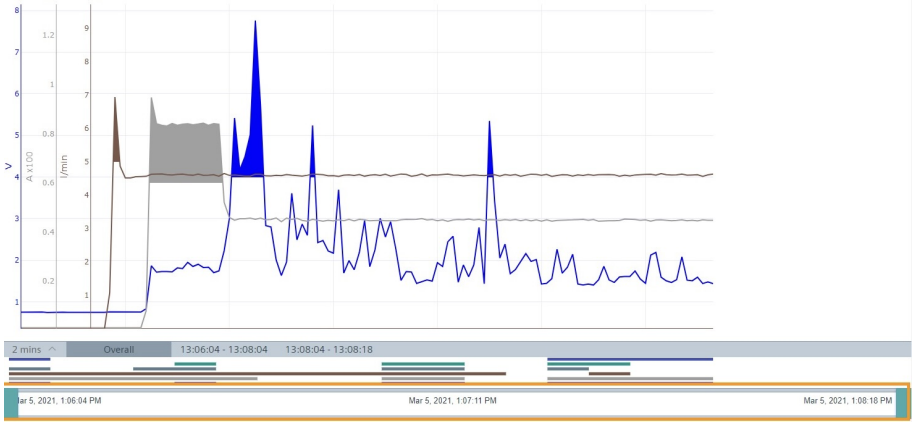

Figure 5.14: Global slider with maximum values

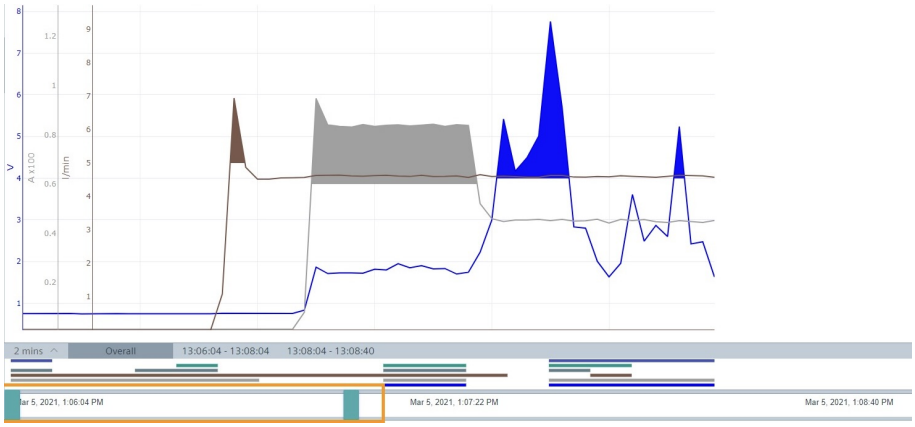

Figure 5.15: Zoomed using global slider

database and hinder the efficiency and performance, queries to the database is only made after releasing the ends and not while the ends are being moved.

5.3.4 Global Navigator

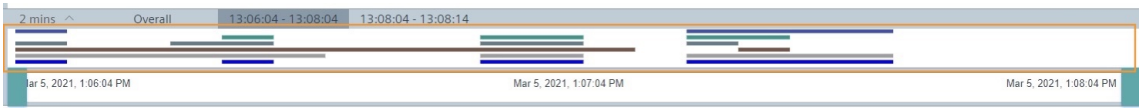

Figure 5.16: Global Navigator

Some time-series charts will have certain critical readings either upper critical or lower critical. These readings can be seen as filled colors in the chart for the respective sensor as shown in fig. 5.8, but it gets difficult to go into detail when the job has been running for a long time or the interval is too big to easily differentiate between filled areas. To counter this issue, a global navigator is used. The global navigator includes all 6 sensors charts in a constrained manner. The constraints are such that it only plots critical readings in limited height. This global navigator stays in correlation with the global slider i.e. it uses the start and end time as set by the global slider and is also in sync with the time series live mode. Such type of critical reading plot of all the sensors makes it easier for users to jump into any critical section to get more details about it. Clicking on any subplot in the global navigator will give information like reading value, the time when this anomaly started, and also the time when such type of anomaly ends. This feature is an improvement for multiple plots and critical readings indicator discussed in the above sections.

5.4 Performance Optimisation

5.4.1 Aggregation Level

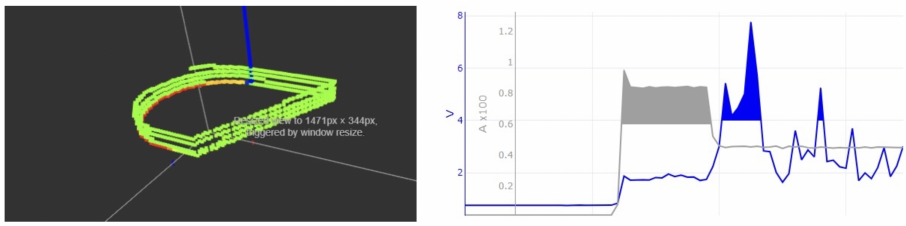

Figure 5.17: Time series and 3D visualisation with aggregation level 10.

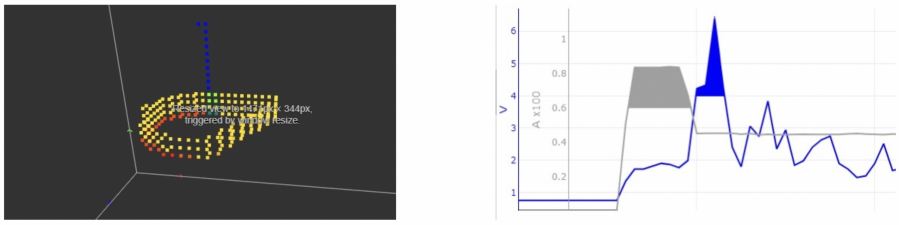

Figure 5.18: Time series and 3D visualisation with aggregation level 7.

Sensor readings in the WAAM process can reach up to 44000 data points per second as discussed in section 1.1. Considering the elapsed time of just 1 hour for a print job, the number of data points for one sensor is 158 million. This vast amount of data points is multiplied if more than one sensor is selected resulting in negative effects on the performance of the monitoring dashboard such as delay in the rendering of time series and 3D visualisation in live mode. The reason for this adverse effect on performance delays in getting this data from influx DB and spatial DB. A solution to fix this issue is the use of a new parameter called "Aggregation Level" which groups certain data points and gives the mean value of the group. The aggregation level slider has multiple steps, and this, in turn, changes the grouping interval for data queries. Query statements for influx DB while using group are described in detail in section 4.3. Fig. 5.17 and fig. 5.18 shows time series and 3D visualisation for higher aggregation level and lower aggregation level respectively. If the lowest value for aggregation level is used, the maximum amount of grouping takes place and their mean value is returned. In this case, approximately 2 million data points are reduced to just 1 data point. As the aggregation level increases the grouping interval decreases. At the highest aggregation level, a minimum grouping of seconds occurs. The introduction of such parameter has thereby reduced the query time by 3 seconds and resulted in better performance of the dashboard.

5.4.2 Dynamic manipulation of Aggregation Level

The aggregation level slider allowed the user to manage latency issues and get a better-performing dashboard during run-time. Instead of fixing the vast data points issue after it occurs during monitoring, it's better to have a pre-existing algorithm that manages the aggregation level value according to the elapsed time of the job. This works by using a switch case and defining aggregation levels for intervals of elapsed time.

5.4.3 Vuex - Global State Management

Vuex is a state management library that allows vue.js applications to store information globally at one place called the store. Good state management is needed when the application gets complex and includes many variables or states being passed from one component to another. This store contains various states. Considering monitoring dashboard, the global store contains states like startTime, endTime, aggregation, print ID, color mapping, etc. Vuex empowers Vue's reactivity system and creates easily accessible data between independent components. A single state tree makes it straightforward to locate a specific piece of state and get the current updated state via getter functions defined in the global store. The only way to change the state in a Vuex store is by committing a mutation. Vuex mutations are handler functions that are also defined in the global store for each state. This allows the components to update the state when required and at the same time provide all other dependent components with the updated value of that state.

6 Evaluation

Following the design and implementation of the monitoring dashboard from chapters 4 and 5, this chapter aims to shed light on the viability of the discussed concepts. Section 6.1 and 6.2, display qualitative and quantitative analysis of the dashboard.

6.1 Qualitative Analysis

This section focuses on whether the requirements set froth in chapter 4 have been met with and what other improvements have been implemented which were not the part of those requirements.

Print jobs page

Print Job Overview

Job	Printer	Begin	End	Progress	
print-257b6217	WAAM Robot Cell	05-03-2021 12:06:04		1.816	save CSV
print-420a993e	WAAM Robot Cell	05-03-2021 12:06:04		0.557	save CSV
print-43a2c9a9	WAAM Robot Cell	05-03-2021 12:06:04		0.775	save CSV
print-43e8ee6b	WAAM Robot Cell	05-03-2021 12:06:04		0	save CSV
print-5f103704	WAAM Robot Cell	05-03-2021 12:06:04		0.532	save CSV
print-93c191be	WAAM Robot Cell	05-03-2021 12:06:04		9.121	save CSV
print-99e6ed8b	WAAM Robot Cell	05-03-2021 12:06:04		1.007	save CSV
print-a5219983	WAAM Robot Cell	05-03-2021 12:06:04		0.569	save CSV
print-acd3086d	WAAM Robot Cell	05-03-2021 12:06:04		0.899	save CSV
print-b0d8cac2f	WAAM Robot Cell	05-03-2021 12:06:04		4.813	save CSV

Figure 6.1: Implemented print job page

Figure 4.1 shows print job overview page design and indicate a sorting and search feature for tabular data. Here progress column indicate percentage of the job being done which was not feasible since every job can last of different time range and it is difficult to predict the exact time it would take to complete that job. Due to this getting progress in percent was not implemented but rather a progress column shows elapsed time of that particular job now as seen in figure 6.1. To improve the user experience further, a "save CSV" button was added to each row of the table with which user can easily export data from all the sensors until the elapsed time in one single click.

Monitoring page

Figure 4.2 shows monitoring page design and differentiates 3D and time series visualisation along with having global slider at the bottom. The design didn't consider a print job can be

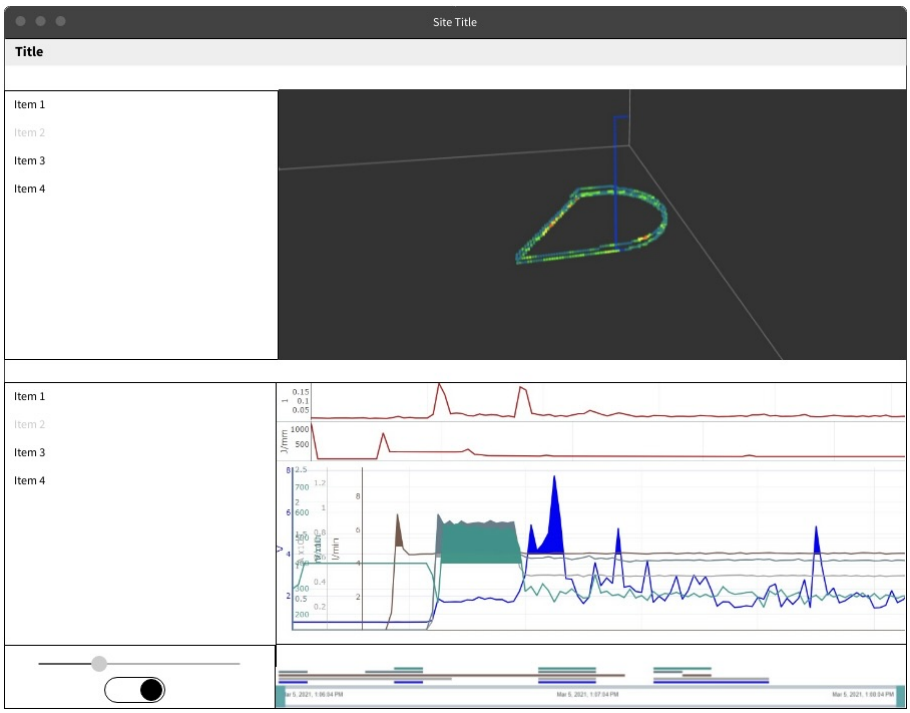

Figure 6.2: Implemented monitoring page

in progress as well. For fixing this issue a switch was added below aggregation level slider which toggled between "Live mode" and "Analyze Mode" of the job which can be seen in figure 6.2 The difference between these two modes is that in Live mode new data is queried every 5 seconds which gives real-time updates of the ongoing job while in analyze mode already queried data of the incomplete job or entire data of completed job can be viewed and studied. Another improvement to the provided designs was addition of time slots and time slot range picker above global slider. Figure 4.10. shows how time slot range picker is a dropdown and the range selected can be seen further in time slots. This features gives user easy access to certain time interval either out of interest or to debug and understand certain anomaly in sensor readings. Anomaly indicators at the bottom namely "Global Navigator" were separated from global slider in order to avoid interference during click events. One more major improvement was addition of "Power Over Path" plot between "Anomaly Voltage" and time-series plots. Power over path is an interesting reading which accumulates current, voltage, wire feed speed, robot travel speed and gives a combined reading.

6.2 Quantitative Analysis

This section details difference in performance when using vite build tool instead of vue cli build tool that is the default build tool of any vue project. The monitoring dashboard was built using vite due to the performance difference seen below during scaffolding i.e. creating

a new project, running a dev server and building the project for production. Following subsection provide insights into metrics related to a new vue project. Similar benchmarks can be considered for a complex project like monitoring dashboard.

6.2.1 Scaffolding

```
sumitbhirud@Sumits-MacBook-Pro Desktop % vue create test-app

Vue CLI v4.5.15

  New version available 4.5.15 → 5.0.4
    Run npm i -g @vue/cli to update!

? Please pick a preset: Default (Vue 3) ([Vue 3] babel, eslint)

Vue CLI v4.5.15
✨  Creating project in /Users/sumitbhirud/Desktop/test-app.
🗃  Initializing git repository...
⚙  Installing CLI plugins. This might take a while...

added 1307 packages in 1m
🚀  Invoking generators...
📦  Installing additional dependencies...

added 73 packages in 7s
⚓  Running completion hooks...

📄  Generating README.md...

🎉  Successfully created project test-app.
👉  Get started with the following commands:

 $ cd test-app
 $ npm run serve
```

Figure 6.3: Create with Vue cli.

```
sumitbhirud@Sumits-MacBook-Pro Desktop % yarn create vite vite-test-app
yarn create v1.22.17
[1/4] 🔍  Resolving packages...
[2/4] 🚚  Fetching packages...
[3/4] 🔗  Linking dependencies...
[4/4] 🔨  Building fresh packages...

warning Your current version of Yarn is out of date. The latest version is "1.22.18",
info To upgrade, run the following command:
$ curl --compressed -o- -L https://yarnpkg.com/install.sh | bash
success Installed "create-vite@2.9.2" with binaries:
      - create-vite
      - cva
✔ Select a framework: › vue
✔ Select a variant: › vue

Scaffolding project in /Users/sumitbhirud/Desktop/vite-test-app...

Done. Now run:

  cd vite-test-app
  yarn
  yarn dev

✨  Done in 7.24s.
```

Figure 6.4: Create with Vite.

As seen in Fig. 6.3 and Fig. 6.4, creating new vue project with command "vue create test-app" took 67s while creating new vite project with command "yarn create vite vite-test-app" took 7.24s.

6.2.2 Dev server

```
sumitbhirud@Sumits-MacBook-Pro test-app % npm run serve

> test-app@0.1.0 serve
> vue-cli-service serve

 INFO  Starting development server...
98% after emitting CopyPlugin

 DONE  Compiled successfully in 1140ms


  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.0.192:8080/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

Figure 6.5: Dev server with Vue cli.

```
sumitbhirud@Sumits-MacBook-Pro vite-test-app % npm run dev

> vite-test-app@0.0.0 dev
> vite


  vite v2.9.5 dev server running at:

  > Local: http://localhost:3000/
  > Network: use `--host` to expose

  ready in 154ms.
```

Figure 6.6: Dev server with Vite.

As seen in Fig. 6.5 and Fig. 6.6, running dev server of this new vue project with command "npm run serve" took 1140ms to compile successfully, while running dev server of this new vite project with command "npm run dev" took 154ms.

6.2.3 Building for production

```
sumitbhirud@Sumits-MacBook-Pro test-app % npm run build

> test-app@0.1.0 build
> vue-cli-service build


  Building for production...

 DONE  Compiled successfully in 2512ms

  File                                  Size                    Gzipped

  dist/js/chunk-vendors.9ae1b50e.js     91.41 KiB               34.10 KiB
  dist/js/app.e985efca.js               4.48 KiB                1.63 KiB
  dist/css/app.fb0c6e1c.css             0.33 KiB                0.23 KiB

  Images and other types of assets omitted.

 DONE  Build complete. The dist directory is ready to be deployed.
 INFO  Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html
```

Figure 6.7: Build with Vue cli.

```
sumitbhirud@Sumits-MacBook-Pro vite-test-app % npm run build

> vite-test-app@0.0.0 build
> vite build

vite v2.9.5 building for production...
✓ 14 modules transformed.
dist/assets/logo.03d6d6da.png      6.69 KiB
dist/index.html                    0.42 KiB
dist/assets/index.c8be967f.css     0.20 KiB / gzip: 0.17 KiB
dist/assets/index.83d1ee42.js      52.26 KiB / gzip: 21.09 KiB
```

Figure 6.8: Build with Vite.

As seen in Fig. 6.7, building vue project code for production took 2.5s to compile successfully and gzipped production files created had size of 34 KB for JS file and 1.63 KB for CSS file. while building vite project code for production as seen in Fig. 6.8, took 1.15s to compile successfully and gzipped production files created had size of 21 KB for JS file and 0.17 KB for CSS file.

6.2.4 Results

Table 6.1 provides an overview of discussions and metrics seen in above sections. As seen in the table, vite build is an extremely fast build tool that can provide significant results in the development experience of vue project. Using vite build for monitoring dashboard has allowed to expedite development time frame and has contributed positively towards building of a fast and efficient monitoring system for WAAM.

Scenario	Vue-cli	Vite
Scaffolding	67s	7.24s
Dev server	1140ms	154ms
Build for production	2.5s	1.15s

Table 6.1: Performance metrics for scaffolding, dev server and building scenario.

7 Related Work

Despite the fact that WAAM technology has wide range of potential applications across a number of manufacturing sectors, there remains set of issues that hinder widespread integration of WAAM in manufacturing industries [27]. One of the greatest challenges of WAAM systems is the control algorithms needed to assure that the deposited geometry fits the required geometry with the required structural integrity [28]. Thus, in order to avoid defects during processing, specific monitoring systems are required. This chapter discusses previous attempts at implementation of a monitoring system for WAAM and it also compares work done in this thesis to those implementations.

7.1 MATLAB GUI

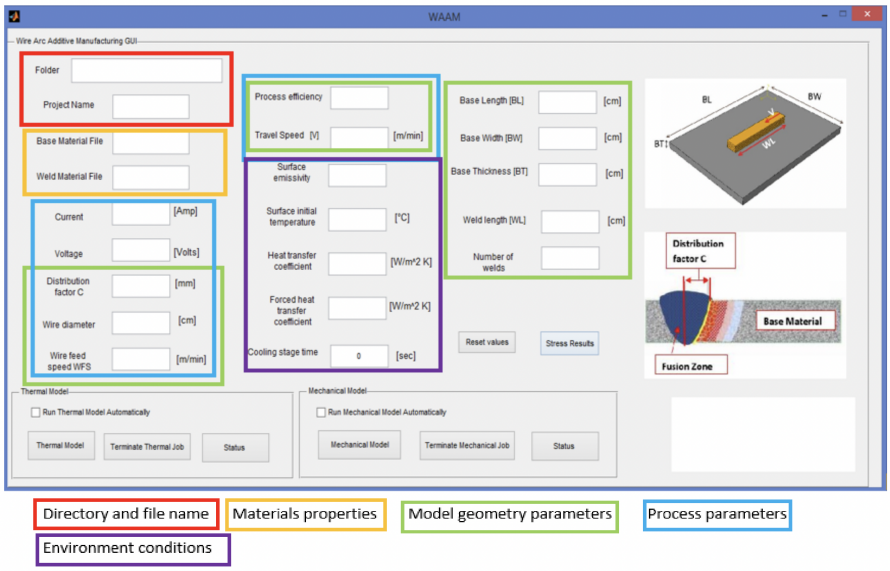

Figure 7.1: Wire Arc Additive Manufacturing GUI [29].

The author of this paper emphasises that limitations or defects caused in WAAM process are mainly caused by thermal cycles that the workpiece is subjected to. The author bases this claims based on the argument that residual thermal stress caused by non-uniform temperature changes during the process can exceed the yield strength of the material, causing geometric distortions in the final product. The author proposes a graphical user interface as seen in Fig. 7.1, which would allow users to enter values for the parameters such as voltage,

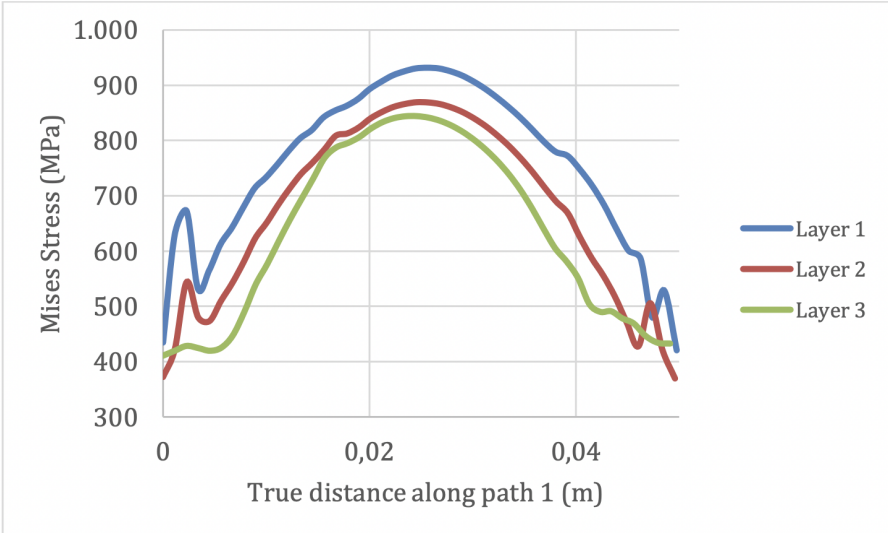

Figure 7.2: Mises stresses calculated in three different layers [29].

electric current, heat source speed, yield strength, thermal conductivity, coefficient of thermal expansion, material used, which will be used to run a thermo-mechanical simulation. After simulation is complete, user will be able to observe a plot of mises stresses similar to Fig. 7.2, which will help to determine whether geometric distortions will occur during the actual process or not.

7.2 In situ monitoring system

The author of this paper highlights that the greatest challenges during WAAM process is of control algorithms that ensures the deposited geometry fits the required geometry with

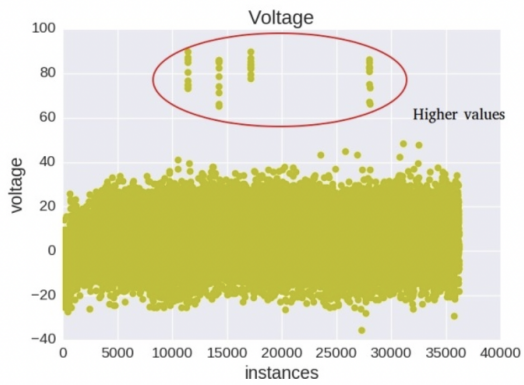

Figure 7.3: Voltage with higher values circled in red [28].

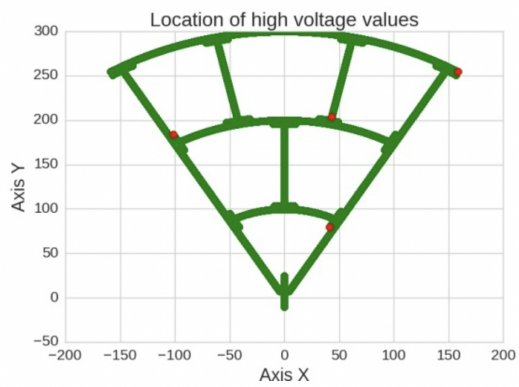

Figure 7.4: Location of higher voltage values [28].

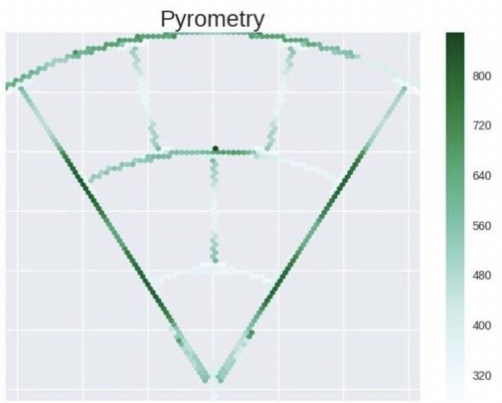

Figure 7.5: Reached temperature and its location measured by pyrometer [28].

the required structural integrity. The proposed solution is to have a monitoring system to control both process parameters and deposited geometry to avoid defects. A closed loop control system is developed which ensures constant distance between the plasma torch and the subtrate [28]. The monitoring of the process is facilitated through the use of external sensors like pyrometer, laser scanner, welding camera, PC application and oxygen meter. Several internal signals such as voltage, intensity, feed rate and machine positions have been monitored as well. The paper does not display the entire monitoring system but rather explains few plots based on signals acquired from external and internal devices. Fig. 7.3 and Fig. 7.4 indicates high voltages values during the process and the reason is difference in distance between the torch and the part at joints compared to rest of the bead. Similarly Fig. 7.5 displays difference in temperature in large beads compared to shorter beads. The author discussed how information from these three figures demonstrates the utility of monitoring the process.

7.3 Arc analyzer

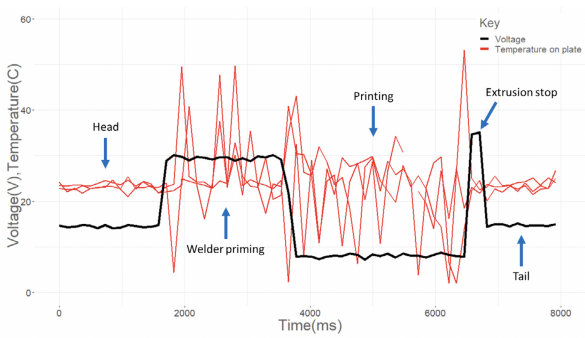

Figure 7.6: Thermistor sensor values [30]

The author of this paper points out the importance of control and monitoring of the arc due to the inconsistencies and weld quality problems that may arise from arc instability or noise factors such as contamination or environmental conditions. The paper proposes arc analyzer, which is a multi-sensor monitoring system which includes parameters such as voltage, current, sound, light intensity, radio frequency, and temperature. Monitoring differences in sensor data can be used to improve the WAAM process. This solution focuses more on hardware side of monitoring like setting up of these sensors to collect respective data. Gathered data is plotted in individual charts or combined charts as seen in Fig. 7.6, to track the process and analyze them in order to improve the WAAM process.

7.4 Comparison of related work

All three implementations as seen above are not generic in nature and target only a part of the monitoring problem related to WAAM process. Each of them focuses on one aspect of the potential cause of issues with WAAM process. None of them provide an inclusive user interface like the one introduced in this thesis. Following sections details the differences in these implementations and the monitoring dashboard introduced in this thesis.

7.4.1 MATLAB GUI

Table 7.1 shows a comparison between the introduced monitoring system and MATLAB GUI mentioned in section 7.1. The most important difference is live monitoring of the WAAM process. Uncertainties during the WAAM process is the reason why monitoring of the process is needed, to tackle and fix issues that arise.

Factors	MATLAB GUI	Introduced Monitoring Dashboard
Time series visualisation	No	Yes
3D visualisation	No	Yes
Data extraction	No	Yes
Live monitoring	No	Yes
Anomaly Detection	Manual tracking of plot in simulation	Managed by the dashboard by displaying anomalies in visualisations and global navigator
Interactive UI	No	Yes
Multi sensor monitoring	No	Yes
Analysis based on	Plot of Mises stresses	Multi sensor time series and 3D visualisation

Table 7.1: Comparison of MATLAB GUI with the introduced monitoring dashboard.

7.4.2 In situ monitoring system

Table 7.2 shows a comparison between the introduced monitoring system and in situ monitoring system mentioned in section 7.2. In situ monitoring system tracks limited number of sensor data specifically voltage, current and temperature compared to almost double the sensors in the introduced monitoring dashboard. An inclusive user interface is not provided as well but rather just plots of the sensor data.

Factors	In Situ Monitoring System	Introduced Monitoring Dashboard
Time series visualisation	Yes	Yes
3D visualisation	No	Yes
Data extraction	No	Yes
Live monitoring	Yes	Yes
Anomaly Detection	Manual tracking of plots	Managed by the dashboard by displaying anomalies in visualisations and global navigator
Interactive UI	No	Yes
Multi sensor monitoring	Voltage, Current and Temperature	Acoustic, Voltage, Current, Temperature, Wire feed, Velocity, Gas flow, Anomaly Voltage, Power Over Path
Analysis based on	Multi sensor plots	Multi sensor time series and 3D visualisation

Table 7.2: Comparison of in situ monitoring system with the introduced monitoring dashboard.

7.4.3 Arc Analyzer

Table 7.3 shows a comparison between the introduced monitoring system and arc analyzer mentioned in section 7.3. Arc analyzer implementation is the most suitable monitoring system among the three mentioned above. It provides live monitoring of multiple sensors and allows users to group different sensors plots together to extract some more details. Although having multiple advantages, it does not provide an inclusive user interface, that hinders control from the user. 3D visualisation can be very helpful to track the actual movement of the arc but that is also not provided in this arc analyzer.

Factors	Arc Analyzer	Introduced Monitoring Dashboard
Time series visualisation	Yes	Yes
3D visualisation	No	Yes
Data extraction	No	Yes
Live monitoring	Yes	Yes
Anomaly Detection	Manual tracking of plots	Managed by the dashboard by displaying anomalies in visualisations and global navigator
Interactive UI	No	Yes
Multi sensor monitoring	Voltage, Current, Sound, Light intensity, Radio frequency, Temperature	Acoustic, Voltage, Current, Temperature, Wire feed, Velocity, Gas flow, Anomaly Voltage, Power Over Path
Analysis based on	Multi sensor plots	Multi sensor time series and 3D visualisation

Table 7.3: Comparison of arc analyzer with the introduced monitoring dashboard.

8 Summary and Conclusion

8.1 Summary

This chapter summarizes the work done in this thesis. It introduces a web application for monitoring WAAM processes. It first examines with additive manufacturing concept and explains how WAAM is a type of AM process which is cost-effective due to its distinguish deposition method and use of metal wire instead of metal powder. The thesis then presents reasons for the use of a tool or a dashboard that allows monitoring of the ongoing WAAM process and also compares different solutions to finally consider a web application for the development of a monitoring dashboard.

The thesis continues by detailing the reasons and choices made in the pre-development phase to finalize the front-end framework, build tool, bundler, software development model and libraries to be used for the development of the dashboard. It also describes the collaboration efforts performed among the team members to support a more streamlined and targeted development. The requirements for the user interface, high-level architecture and data types and databases are shown and implementation details are described in later sections of the thesis.

During implementation we see sections about each page of the dashboard, beginning with the print jobs view which lists all the jobs, ongoing or completed by the WAAM system. The thesis continues with the main monitoring view of the application which includes descriptions of different child components within this view. Child components like time-series visualisation, 3D visualisation, global slider, and global navigator are focused on in this thesis.

The Time series visualisation section describes features introduced in the dashboard such as critical readings (anomaly) indicator, multiple sensor plots in one chart, and time slots generation for providing an analytical advantage to the users. Global slider and global navigator focus on showcasing the connectivity of components and provides a single entry point for the users to control the entire monitoring view. The global slider allows manipulation of the time duration of the monitoring view while the global navigator allows the interface to spot and navigate anomalies occurring in sensor values without the need for the user to look into the plots.

Further, we see discussions on techniques used to optimize front-end performance by introducing an aggregation level slider which allows manipulation of the number of data points to be displayed or plotted in visualisations. It continues by showing further optimization

potential by introducing an algorithm that manipulates aggregation level without any inputs from the user but solely based on available data points in selected time duration for the visualisations. It also describes the use of Vuex, a global state management library that allows storing variables or states in a global store that is accessible to all the components in the entire application irrespective of their relationship whether it's a child component or a parent component.

After that, qualitative and quantitative analysis is used to evaluate the dashboard. The evaluation indicates whether the requirements outlined at the beginning of this thesis are satisfied and it also displays metrics of the performance of the application. Finally, we see previous work related to the monitoring of WAAM process and also a comparison of the current implemented monitoring system with the previous related works.

8.2 Conclusion

The purpose of this research aimed to identify methods that can be used to efficiently monitor additive manufacturing processes and bring these multiple components together in a user-friendly and easy-to-understand monitoring dashboard. The combination of data visualisation with components like sensor selector, global slider and global navigator not only fits the required description but also provides a flexible interface for the monitoring system. Concepts introduced like dynamic aggregation level manipulation, monitoring mode changes, anomaly detector and use of global state management system describe various performance enhancing techniques for the monitoring dashboard.

9 Future Work

Although, the goal of the thesis has been accomplished, there are plenty of improvements that could be done in order to achieve a better performing WAAM monitoring dashboard. Some of the ideas are described in the following sections.

9.1 CSV Imports

As discussed in section 5.2., users are allowed to export the completed job data from print job view page as a CSV file. This file contains all the data about that particular job like print ID, start time of the job, end time of the job, duration of the job, sensor data for the entire duration, anomaly data for the entire duration, spatial data for the entire duration etc. Exporting data as a CSV file benefits users for having a long term storage of data outside of the dashboard and also benefits sharing of data easily if required. The monitoring dashboard should also allow importing of the same CSV file. Importing and viewing the data in dashboard without running any print job allows user to analyse any previous job without connecting to WAAM system and can deduce some insights for future manufacturing processes.

9.2 Global Navigator Selector

Currently global navigator as seen in Fig. 5.15, displays anomalies in sensor values and is synchronised with X axis of time series plots. The navigator is dynamic and also moves according to sensor data in live mode. In order to debug such anomalies it is important to get details like timestamp of start time and end time of the anomaly. The idea here would be a small popup on top of the navigator item. So when an item i.e. anomaly is clicked inside the global navigator, it should display a popup which gives details like start time and end time of the anomaly and also the value of the sensor during this anomaly to focus on the severity of the same.

9.3 Parametrization

As seen in Fig. 5.3 and Fig. 5.4, although the monitoring system is dynamic and show real-time tracking of the printing process, elements of the user interface are fixed and hard-coded. Different manufacturers could have different requirements from the user interface they need for monitoring their WAAM process. For example, in Fig. 5.3, along with details like Job ID, printer, start time, end time and progress, some manufacturers would like to have

"heat dissipated" details as well in the table or considering Fig. 5.4, in Time series section instead of "Anomaly Voltage" some might like to have "Temperature" data. Such type of modularity is not offered by the current version of the monitoring dashboard. This type of modularization can be accessible by parametrizing the components view defining them based on some configuration file. Following is a sample JSON configuration file for print jobs page. Using such type of file to create the elements of the component allows manufacturers to have their own set of JSON configuration files according to their needs and requirements.

```
{
  "order_list": {
    "display_name": "Order list",
    "display_type": "table",
    "order_properties": {
      "job_ID": {
        "display_name": "Job ID",
        "is_visible": true,
        "type": "string",
        "description": ""
      },
      "start_time": {
        "display_name": "Start time",
        "is_visible": true,
        "type": "time",
        "description": ""
      },
      "end_time": {
        "display_name": "End time",
        "is_visible": true,
        "type": "time",
        "description": ""
      },
      "status": {
        "display_name": "Status",
        "is_visible": true,
        "type": "string",
        "description": ""
      }
    },
    "order_filename": "path/to/file"
  }
}
```

Figure 9.1: Print jobs View JSON configuration

9.4 Transparent Anomaly Indicator

Fig. 5.8 displays time series plot of multiple sensors. As discussed in section 5.3.1, each sensor has predefined upper and lower critical values which defines an anomaly if sensor value goes above or below that value respectively. Currently, if two sensors are plotted and their anomalies coincide with each other then only one of them is shown clearly. As the figure shows "Wire feed" sensor anomaly trumps over "Current" anomaly. A better solution would be to have transparent anomaly indicators which would not hinder any interpretation about the anomaly and users get clear understanding of such occurrences.

List of Figures

1.1 Working of WAAM system [3]. 1
1.2 WAAM example [4]. 2

2.1 Waterfall Model [20] 8
2.2 Iterative Model [20] 9
2.3 Spiral Model [20] 10
2.4 Scrum Model [20] 11

3.1 Stackoverflow Insight trends 13

4.1 Print jobs design 17
4.2 Monitoring design 18
4.3 Dashboard Architecture 19
4.4 Login page 21

5.1 Print jobs list 22
5.2 Monitoring View 24
5.3 Time series visualisation 25
5.4 Anomaly voltage and power over path charts. 25
5.5 Without critical indicator 26
5.6 With critical indicator 26
5.7 Upper and lower limit configuration of sensors. 27
5.8 Multiple Sensor Plots 27
5.9 JSON configuration of sensor list. 28
5.10 Time slot selection dropdown 28
5.11 Selected time slot 29
5.12 3D Visualisation 29
5.13 3D visualisation with STL file 30
5.14 Global slider with maximum values 31
5.15 Zoomed using global slider 31
5.16 Global Navigator 32
5.17 Time series and 3D visualisation with aggregation level 10. 33
5.18 Time series and 3D visualisation with aggregation level 7. 33

6.1 Implemented print job page 35
6.2 Implemented monitoring page 36
6.3 Create with Vue cli. 37

6.4 Create with Vite. 37
6.5 Dev server with Vue cli. 37
6.6 Dev server with Vite. 37
6.7 Build with Vue cli. 38
6.8 Build with Vite. 38

7.1 Wire Arc Additive Manufacturing GUI [29]. 39
7.2 Mises stresses calculated in three different layers [29]. 40
7.3 Voltage with higher values circled in red [28]. 40
7.4 Location of higher voltage values [28]. 40
7.5 Reached temperature and its location measured by pyrometer [28]. 41
7.6 Thermistor sensor values [30] 41

9.1 Print jobs View JSON configuration 48

List of Tables

1.1 Approximate cost of different metal materials in wire and in powder [6]. . . . 2

3.1 Comparison of front-end frameworks. 14
3.2 Bundler Benchmark [25]. 15

6.1 Performance metrics for scaffolding, dev server and building scenario. 38

7.1 Comparison of MATLAB GUI with the introduced monitoring dashboard. . . 42
7.2 Comparison of in situ monitoring system with the introduced monitoring dashboard. 43
7.3 Comparison of arc analyzer with the introduced monitoring dashboard. 44

Bibliography

[1] *What is Additive Manufacturing | GE Additive*. URL: https://www.ge.com/additive/additive-manufacturing.

[2] M. Yakout, M. Elbestawi, and S. Veldhuis. "A Review of Metal Additive Manufacturing Technologies". In: *Solid State Phenomena* 278 (July 2018), pp. 1–14. DOI: 10.4028/www.scientific.net/SSP.278.1.

[3] A. R. McAndrew, M. Alvarez Rosales, P. A. Colegrove, J. R. Hönnige, A. Ho, R. Fayolle, K. Eyitayo, I. Stan, P. Sukrongpang, A. Crochemore, and Z. Pinter. "Interpass rolling of Ti-6Al-4V wire+arc additively manufactured features for microstructural refinement". In: *Additive Manufacturing* 21 (2018), pp. 340–349. ISSN: 2214-8604. DOI: https://doi.org/10.1016/j.addma.2018.03.006. URL: https://www.sciencedirect.com/science/article/pii/S2214860417306085.

[4] *WAAM! What Is Wire Arc Additive Manufacturing?* 2022. URL: https://all3dp.com/1/waam-what-is-wire-arc-additive-manufacturing/.

[5] Y. Li, C. Su, and J. Zhu. "Comprehensive review of wire arc additive manufacturing: Hardware system, physical process, monitoring, property characterization, application and future prospects". In: 13 (2022). ISSN: 2590-1230. DOI: https://doi.org/10.1016/j.rineng.2021.100330. URL: https://www.sciencedirect.com/science/article/pii/S2590123021001316.

[6] J. Xiong, H. Geng, X. Lin, D. Huang, J. Li, and F. Zhang. "Research Status of Arc Additive Manufacturing and Application Prospects in Aerospace Manufacturing". In: (2015), pp. 80–85. DOI: 10.16080/j.issn1671-833x.2015.23/24.080.

[7] R. Reisch, T. Hauser, T. Kamps, and A. Knoll. "Robot Based Wire Arc Additive Manufacturing System with Context-Sensitive Multivariate Monitoring Framework". In: *Procedia Manufacturing* 51 (2020). 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021), pp. 732–739. ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2020.10.103. URL: https://www.sciencedirect.com/science/article/pii/S2351978920319600.

[8] N. Akhtar, N. Tabassum, D. Perwej, and D. Y. Perwej. "Data analytics and visualization using Tableau utilitarian for COVID-19 (Coronavirus)". In: *Global Journal of Engineering and Technology Advances* Volume 3 (May 2020), Page 28–50. DOI: 10.30574/gjeta.2020.3.2.0029.

[9] R. Reisch, T. Hauser, and B. Lutz. "Context awareness in process monitoring of additive manufacturing using a digital twin". In: *The International Journal of Advanced Manufacturing Technology* 119 (2022), pp. 3483–3500. DOI: https://doi.org/10.1007/s00170-021-08636-5.

[10] N. Naim. "ReactJS: An Open Source JavaScript Library for Front-end Developement". In: (May 2017).

[11] *React - A JavaScript library for building user interfaces*. URL: https://reactjs.org/.

[12] *Introduction to the Angular Docs*. URL: https://angular.io/docs.

[13] E. Thorén and F. Stål. "Usage of Angular from developer's perspective". In: (2017).

[14] J. Song, M. Zhang, and H. Xie. "Design and Implementation of a Vue.js-Based College Teaching System". In: *International Journal of Emerging Technologies in Learning (iJET)* 14 (July 2019), p. 59. DOI: 10.3991/ijet.v14i13.10709.

[15] N. Linge and D. Parsons. "Problem-based learning as an effective tool for teaching computer network design". In: *IEEE Transactions on Education* 49.1 (2006), pp. 5–10. DOI: 10.1109/TE.2005.852600.

[16] *Build Tools*. URL: https://vue-community.org/guide/ecosystem/build-tools.html.

[17] A. Tikhonova. "Design and Implementation of Reusable Component for Vue.js". In: (2021). URL: https://urn.fi/URN:NBN:fi:amk-202105067554.

[18] S. Laurila. "Comparison of JavaScript Bundlers". In: (2020). URL: https://urn.fi/URN:NBN:fi:amk-2020100621034.

[19] *rollup.js*. URL: https://rollupjs.org/guide/en/.

[20] *8 Software Development Models: Sliced, Diced and Organized in Charts*. URL: https://www.scnsoft.com/blog/software-development-models.

[21] V. Chandra. "Comparison between Various Software Development Methodologies". In: *International Journal of Computer Applications* 131 (2015). URL: https://www.academia.edu/29931163/Comparison_between_Various_Software_Development_Methodologies?auto=citations&from=cover_page.

[22] *4 Reasons Why Plotly Is The Best Visualization Library*. URL: https://towardsdatascience.com/4-reasons-why-plotly-is-the-best-visualization-library-18c27de05b95.

[23] M. Kaluža and B. Vukelic. "Comparison of front-end frameworks for web applications development". In: *Zbornik Veleučilišta u Rijeci* 6 (Jan. 2018), pp. 261–282. DOI: 10.31784/zvr.6.1.19.

[24] *js-framework-benchmark*. URL: https://rawgit.com/krausest/js-framework-benchmark/master/webdriver-ts-results/table.html.

[25] *Benchmark Details*. URL: https://esbuild.github.io/faq/#benchmark-details.

[26] V. Stray, N. B. Moe, and D. I. K. Sjøberg. "Daily Stand-Up Meetings: Start Breaking the Rules". In: *CoRR* abs/1808.07650 (2018). arXiv: 1808.07650. URL: http://arxiv.org/abs/1808.07650.

[27] X. Chunyang, P. Zengxi, P. Joseph, and L. Huijun. "A review on wire arc additive manufacturing: Monitoring, control and a framework of automated system". In: *Journal of Manufacturing Systems* 57 (2020), pp. 31–45. DOI: 10.1016/j.jmsy.2020.08.008.

[28] T. Artaza, A. Alberdi, M. Murua, J. Gorrotxategi, J. Frías, G. Puertas, M. Melchor, D. Mugica, and A. Suárez. "Design and integration of WAAM technology and in situ monitoring system in a gantry machine". In: *Procedia Manufacturing* 13 (2017), pp. 778–785. DOI: 10.1016/j.promfg.2017.09.184.

[29] V. Palomeque and S. Juan. "Development of a MatLab Graphical User Interface to Simulate the Thermo-Mechanical Model of Wire + Arc Additive Manufacturing Process". In: (2019). URL: http://repositorio.usfq.edu.ec/handle/23000/8195.

[30] P. Joshua. "Open Source Arc Analyzer: Multi-Sensor Monitoring of Wire Arc Additive Manufacturing". In: (2020).