Available online at www.sciencedirect.com

Manufacturing Letters

Manufacturing Letters 44 (2025) 1486–1497

53rd SME North American Manufacturing Research Conference (NAMRC 53, 2025)

A Real-Time VR-Enabled Digital Twin Framework for Multi-User Interaction in Industry 4.0

Nicholas K. Dewberry[a], Issa AlHmoud[a], Kevin Benton[a], Derick Suarez[b], Yi-Ping Chen[b], Vispi Karkaria[b], Ying-Kuan Tsai[b], Meccaya Brock[a], Nooralhuda Alazzawi[a], Shuva Chowdhury[a], Wei Chen[b], Jian Cao[b], Balakrishna Gokaraju[a]

[a]*Department of Computational Data Science & Engineering, North Carolina Agricultural and Technical State University, 1601 E Market St, Greensboro 27411, United States*

[b]*Department of Mechanical Engineering, Northwestern University, 633 Clark St, Evanston 60208, United States*

Abstract

Digital Twins (DTs) have become pivotal in Industry 4.0, providing a dynamic, real-time virtual representation of physical systems that enables advanced monitoring, predictive analysis, and operational optimization. Despite their widespread application in fields like manufacturing and robotics, integrating DTs with virtual reality (VR) for enhanced user interaction and multi-dimensional visualization remains challenging. This paper presents a flexible framework that bridges this gap by deploying a DT within a VR environment, specifically designed to enable real-time interaction and visualization of industrial processes. Focusing on the operation of a robotic arm manipulating a metal sheet passed through an English wheel (a specialized tool for sheet metal shaping), our framework demonstrates an efficient and scalable backend architecture that supports seamless multi-user interaction connected to a multi-user VR application. The system enables real-time data synchronization between the physical and virtual environments, enhancing the fidelity of the DT model for both visualization and control purposes. Through streamlined robust data flow management, automated processing and continuous integration and deployment, this framework contributes a versatile solution and novel methods for VR-based DTs, reinforcing their potential to drive predictive maintenance, operator training, and decision-making in Industry 4.0 applications.

Keywords: Digital Twins; Industry 4.0; Virtual Reality (VR); Real-Time Data Synchronization; Multi-User Interaction; Predictive Maintenance; Unity3D; Backend System; Data Flow Management; Cyber-Physical Systems; Immersive Visualization.

1. Introduction

The advent of Industry 4.0 marks the fourth industrial revolution, characterized by the integration of advanced technologies such as the Internet of Things (IoT), artificial intelligence (AI), and cloud computing into manufacturing and industrial processes. This paradigm shift enables real-time data exchange, automation, and data-driven decision-making across interconnected systems, enhancing efficiency, productivity, and flexibility in industrial operations [1]. Industry 4.0 emphasizes ”smart” environments, where cyber-physical systems communicate seamlessly, allowing for predictive maintenance [2], remote monitoring [3], and process optimization.

Among the foundational technologies in Industry 4.0, Digital Twins (DTs) have emerged as transformative tools, providing real-time, dynamic virtual representations of physical assets, systems, or processes [4]. DTs are continuously synchronized with their real-world counterparts through bidirectional data exchange, allowing for enhanced monitoring, predictive maintenance, and operational optimization. Originally conceptualized in 2003 by Grieves [5] [6], DTs have since been integrated into robust tools in fields such as manufacturing, robotics and smart systems, driving the progression of Industry 4.0 tech-

* Corresponding author. Tel.:+1-336-285-3210 ; fax: +1-336-256-1247.
E-mail address: bgokaraju@ncat.edu (Balakrishna Gokaraju).

nologies. Unlike traditional simulations, DTs can adapt dynamically to real-time data, providing continuous feedback and allowing for simulation-based adjustments to real-world operations [7, 8, 9].

In this paper, we embody DT as interactive, real-time virtual models that reflect the current state of physical assets through live data integration. This capability is particularly valuable in manufacturing, where DTs help anticipate failures, streamline operations, and improve decision making by enabling scenario testing in a virtual environment [10] [11]. The integration of DTs with virtual reality (VR) adds an immersive dimension to industrial monitoring and interaction. In VR settings, DTs enable users to interact with complex systems in a highly visual and interactive format, improving understanding and facilitating hands-on training without affecting actual equipment. This immersive capacity has proven beneficial in operational safety and training, as well as in making complex systems more accessible for education [12] [13]. However, existing frameworks for VR-enabled DTs often lack seamless multi-user functionality and automated data processing features in manufacturing.

To address these gaps, this paper introduces multiple novel methods for developing a VR-based DT framework designed for real-time visualization and multi-user interaction in a virtual environment. This paper focuses on a robotic arm manipulating an English wheel—a traditional tool for sheet metal deformation. This framework incorporates a robust backend architecture to support efficient data flow and enhance interaction fidelity. By advancing the integration of DTs within VR, this work contributes to a practical, scalable solution that strengthens predictive maintenance, operational training, and decision-making capabilities within Industry 4.0 applications. The main contributions of this work are as follows:

A Real-Time VR-Enabled DT Framework for Manufacturing: We propose a flexible DT framework that enables real-time synchronization between physical manufacturing processes and their digital representations in a VR environment. The system allows manufacturers to monitor process variables, material deformation, and system performance in a virtual space, improving real-time decision-making.

Multi-User Collaboration for Data-Driven Manufacturing: The framework supports multiple users interacting simultaneously within the DT. A secure backend architecture, developed in Golang, ensures efficient data handling, automated processing of manufacturing data, and secure user authentication using Two-Factor Authentication (TOTP).

Scalable and Extensible System for Industry 4.0 Applications: The proposed system is designed to integrate seamlessly with various manufacturing processes, allowing expansion beyond sheet metal deformation to other industrial applications.

Intuitive visualization techniques: This framework adds a unique way of displaying time-series information, using sheet metal forming as an example.

The remainder of this paper is organized as follows. Section 2 discusses the background and literature review of the state-of-the-art VR DT technologies. Section 3 details the methodology and development of the proposed DT framework, outlining the problem we seek to address and its intended applications. Section 4 delves into the detailed methodology for the backend system and network communication. Section 5 discusses the process of building the VR application in Unity3D, with multi-user functionality. Section 6 details the challenges and solutions to implement the plan. Section 7 analyzes performance, results, and benchmarks. Section 8 concludes the paper by addressing current limitations, and future work.

2. Background and Literature Review

Since their initial definition, DTs have demonstrated substantial benefits in real-time monitoring and predictive maintenance, aiding in quality control and operational optimization. Integrating DTs with VR is a burgeoning area that promises enhanced interactivity, visualization, and training capabilities.

Recent research has shown the potential of integrating DTs and robotic systems to advance metal-forming techniques and improve manufacturing accuracy and efficiency. Suarez *et al.* [14] explored an integrated English wheeling system that combined robotic control, force monitoring, metrology, and computational modeling to achieve high flexibility and repeatability in sheet metal forming. Their work demonstrates the benefits of real-time tracking and control in metal forming, highlighting the adaptability and precision achievable with robotic support for traditional metalworking tools. Similarly, Benton *et al.* [15] focused on VR and DT applications to enable more compelling, collaborative training environments for operators in industrial settings. Their approach addressed the challenges of interoperability and visualization in virtual environments, aiming to replicate complex manufacturing processes and enhance operator interaction with digital twin models.

Our work extends these efforts by adding multi-user VR functionality, automated data processing, user authentication, and advanced animation features. The inclusion of these features aims to enhance both the collaborative and interactive aspects of DTs in a VR environment, creating a scalable backend that can be extended for further usage. By developing a framework that combines these elements, our project builds upon the foundations laid by previous studies and addresses the challenges of real-time, secure, and immersive interactions with DTs in industrial applications.

Several tools and platforms have been developed to support the visualization and interactivity of DTs, including software such as Unity3D [16], Unreal Engine [17] [18], and Gazebo [19]. These tools allow developers to create detailed and customized virtual models that integrate data and behavioral logic through scripts. This project adds to the literature by using Unity3D, alongside a server system that is able to process data for the VR application.

[19] compares features and developments for Unity3d and Gazebo, and identifies multiple gaps in DT visualization software. One of which is the relative complexity of integrating unsupported technologies (such as ROS into Unity3d) and unsupported file formats (such as URDF in Gazebo) into DT systems. This paper seeks to provide a solution to these problems by having a flexible backend server handle the transfer of information and automatically process information into compatible file formats. The backend described in this paper focuses on integrating into a Unity3d environment, however, it is funda-

mentally software-agnostic, and is able to be modified to work into other technologies such as Gazebo or Unreal Engine.

[16] provides a detailed use case of a VR DT for manufacturing. This paper recognizes the importance of this contribution, and Table 1 offers a detailed, itemized overview of the comments and issues identified in the paper, along with an explanation of the approaches used to address each point.

Table 1: Concerns and Solutions

Concerns and comments	The solutions this paper provides
Interdisciplinary – stakeholders and developers for DTs will need to be intimately familiar with the technologies and how they interact with the entire system.	This paper includes a CI/CD pipeline that accounts for remote collaboration and uses HTTP requests to add information to the VR application. This means that individuals have a simplified way of contributing to the development of the DT with minimal coding experience, allowing for further flexibility in DT development.
Computational resources – initial system uses cloud services and a single laptop. Scaling up would require further resources for storage and for computationally expensive features.	This paper decouples the VR application from the data processing backend, allowing for scalability and customization. The VR applications can focus solely on rendering, and the other components can focus on other tasks.
Multi-User Environments – most DTs are oriented toward single users, and enabling multiple users in a virtual space allows for remote collaboration and troubleshooting.	This paper builds upon previous work and integrates a multi-user environment within Unity3D.
Cybersecurity risks – cybersecurity is a large concern. DTs include sensitive information and control over physical assets.	This paper solves this issue through the creation of user accounts (username and password) protected by TOTP-based two-factor authentication. By default, the VR application prevents connection and visualization if there are no authorized users logged in, and logins time out requiring re-authorization.
Networking risks – latency and interruptions can give inaccurate pictures of the physical system, leading to inefficient decision-making.	This paper includes a section on benchmarking and a discussion pertaining to performance.
Applications beyond the mining industry	This paper implements many novel features from [16] and integrates them into an industrial manufacturing use case, demonstrating the scalability, importance, and feasibility of VR DTs.

Manual methods to update and integrate 3D models and data are often time-consuming and require high attention to detail [20]. This paper implements multiple automated methods to integrate 3D models into DT screens through the usage of a robust networking backend. The automated pipeline allows multiple users to send data without worrying about pre-processing or size constraints. This paper expands upon this previous work by not just integrating 3D models, but also time series data (particularly in the form of xyz points), and animating it in an easily understood manner.

By enhancing the interactivity and scalability of Digital Twins in Virtual Reality, this research seeks to contribute to the growing body of knowledge on DT with a practical and extensible architecture that can be utilized in Industry 4.0 applications.

3. Technical Methodology

3.1. Framework Overview

The DT framework developed in this project is designed to enable real-time, immersive interaction with a virtual model of an English wheel and robotic arm within a VR environment. This framework supports multi-user interaction and provides a responsive visualization of sheet metal deformation, accurately mirroring the real-world operations of the robotic arm and English wheel. The core components include a backend server developed in Golang (Also known as Go), and a VR visualization built using Unity3D, Together, these elements form an extensible DT framework optimized for real-time performance, multi-user interaction, and high visual fidelity. The DT framework also includes a read-only web interface that is used for authentication and data downloading. This can be seen in Figure 1, which represents a simplified framework of the DT. Notably, the backend serves as an intermediary for the other systems present in the DT, such as the VR application and web front end. These components communicate indirectly (the backend handles data transfer) using HTTP requests.

In contrast to monolithic DT architectures, this framework's novel implementation of responsibility separation offers numerous advantages, including improved computational efficiency via specialized tasks, enhanced modularity and customization, increased accessibility for non-technical stakeholders (e.g., through simple code snippets or web interfaces), and robust security measures (e.g., two-factor authentication) facilitated by a dedicated back end.

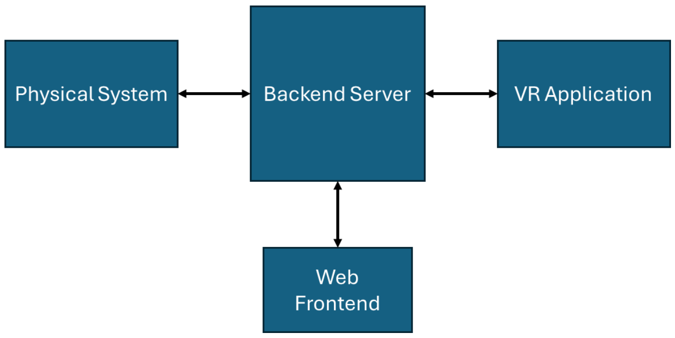


Figure 1: DT system components.

3.2. Goals and Design Principles

The development of this VR-based DT framework was guided by several key goals. (1) Real-time data synchronization is fundamental to the framework, aiming to minimize latency and maintain the connection between the physical robotic arm and its virtual counterpart. This is crucial in VR to preserve user engagement and prevent cybersickness, i.e. a condition in which users of VR devices feel nauseous after and during usage [21]. (2) The system is designed to allow multiple users to interact simultaneously within the VR environment. Having multiple users in a single DT environment has been previously identified as an important addition to the DT space [16]. (3) Data processing must be automated. This is done in two methods: using POST requests - an HTTP request method that is commonly used for sending data to a server - alongside parameters, and by having the backend server able to recognize file extensions, and process data accordingly.

Several key design principles were followed in this work. The first being security. Two factor authentication is implemented through the usage of time-based one-time passwords (TOTP), and HTTP session management is used to protect sensitive data. The second being high-fidelity visualization in the virtual environment. High-fidelity visualization in VR is achieved through detailed CAD modeling, real-time data display, and by customizing graphics in Unity3D. This is essential for accurately replicating the processes of the English wheel, metal sheet, and robotic arm. Lastly, the scalability is achieved with the backend's modular architecture and Golang's efficient data handling facilitating the potential expansion of the system to incorporate additional machines or processes. By using switch statements and automated data detection functions, developers are able to add new processing functions and customize current ones for each use case.

3.3. Overview of Server System

The server backend, built with Golang [22], is designed to handle the data processing, synchronization, and storage requirements of a real-time VR-based DT framework. Golang was selected for its performance in managing real-time data streams and supporting the rapid data exchange necessary for networking applications. The backend is responsible for parsing incoming data, typically in .csv or .txt formats, extracting x, y, and z coordinates, and processing them for compatibility with the VR environment. Once parsed, the data is stored locally in UTF-8 .csv files with timestamped filenames, facilitating tracking and historical analysis. For 3D model data, the backend utilizes an auxiliary Python function with the Open3D API to convert .stl or .gltf files into .obj format, enabling real-time display and manipulation within Unity3D. The system also incorporates a Continuous Integration and Continuous Delivery (CI/CD) pipeline using GitHub Actions to automate testing and deployment, ensuring the backend remains stable and performs reliably across updates.

Real-time synchronization between the physical robotic arm and its virtual representation in VR is achieved through a TCP/IP connection. This setup ensures that the VR model updates in real time, accurately reflecting the movements of the physical system. The network layer is configured to maintain this real-time data flow, with Unity3D scripts updating the UR3 robotic model based on incoming data, allowing the VR environment to mirror the physical movements seamlessly. To optimize network latency, the system prioritizes efficient packet transmission and processing. For instance, we can employ techniques such as optimization before sending data to Unity to ensure responsive and low-latency data transfer. Data consistency is ensured through error-checking routines that manage smooth updates to the VR environment. Fallback protocols are in place to handle any data inconsistencies or network disruptions.

3.4. VR Integration and Visualization in Unity3D

Unity3D was selected for its flexibility, widespread adoption, and extensive customization capabilities. The VR environment is constructed to accurately represent the English wheel and robotic arm, visualizing the metal deformation process with high fidelity. CAD models, created in Rhino3D, are imported into Unity3D to form the basis of the virtual environment, allowing for precise scaling and high-detail visualization. Real-time animation and data interpolation techniques are employed to visualize the deformation of sheet metal smoothly. A linear interpolation algorithm calculates the transitions between time steps, ensuring continuous and realistic animations within the VR space. Performance optimizations, such as mesh reduction and real-time adjustments, are implemented to maintain high performance and visual quality, even under heavy data loads. The VR application was tested on desktop, on an Oculus Rift system, and on a HTC Vive system.

Game engines have become essential tools in DT development due to their advanced capabilities in real-time 3D rendering, interactivity, and simulation. Originally designed for creating gaming experiences, engines like Unity3D and Unreal Engine are now widely adopted in industrial and research settings for DT applications. These engines excel at rendering high-fidelity 3D environments, making them ideal for visualizing complex physical systems within DTs. Game engines also provide built-in physics simulation, lighting, and animation frameworks, which allow developers to replicate real-world behaviors and interactions with impressive accuracy. Furthermore, their support for multi-platform deployment enables DTs to be accessed across devices, from VR headsets to mobile applications, enhancing accessibility and collaboration. The adaptability of game engines, along with their robust development ecosystems, make them powerful tools for creating interactive scalable DT applications. Because the backend server works with HTTP requests, which are present in many visualization software suites and game engines, connectivity to the backend is software-agnostic.

4. Backend System, Network Protocol, and Data Flow Management

The backend system of this DT framework is designed to support real-time data processing, synchronization, and secure communication between the physical and virtual environments. Built-in Golang, the backend is designed to handle the intensive data flow and computational demands required for the VR-based DT. The VR application integrates into the backend through continuous HTTP requests. Golang was chosen over Python due to performance concerns, particularly in read/write and data processing operations. Despite Python being a high-quality programming language that's often used for data science, potential speed concerns exist due to the global interpreter lock, which may cause issues in the user experience.

To accommodate a range of data types, the backend has the capacity to autonomously convert information it has been sent into consistent formats. When incoming text files, such as .csv or .txt coordinates relevant to the VR environment are detected, the backend is able to convert them into consistent UTF-8 .csv files with no extra information. It does this by looking for columns that correspond to an "(x, y, z)" format and then strips away all other metadata. The parsed files are then stored in .csv format with timestamps, facilitating tracking, historical analysis, and compatibility with Unity3D's rendering needs. For more complex 3D model data, such as .stl or .gltf files, the backend leverages an auxiliary Python script using the Open3D library [23] to convert these files into .obj format. This conversion process enables Unity3D to efficiently display and manipulate 3D models in real-time, mirroring the physical robotic arm and other components in the virtual environment. Figure 2 describes this process in further detail.

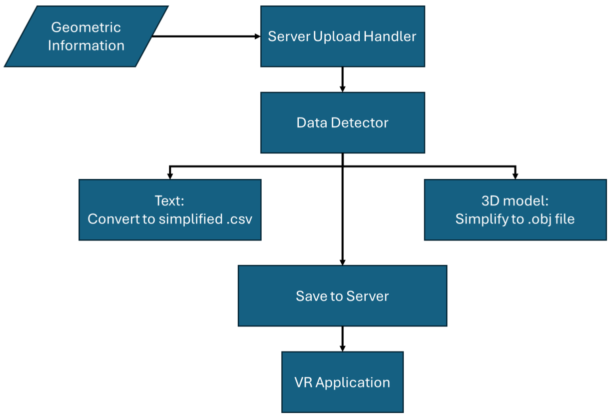


Figure 2: Data processing from raw sheet geometric information to the VR application.

The backend relies on TCP/IP protocols to facilitate continuous, low-latency communication between the physical system and VR application. Custom scripts in Unity3D handle the reception and application of this data, dynamically updating the VR environment to mirror changes in the physical system. These scripts ensure that each movement of the robotic arm and each deformation of the sheet through the English wheel is displayed instantaneously, enabling accurate interaction for users and parity with the physical system.

The backend system also includes error handling and data validation routines that continuously monitor data consistency and integrity. For example, if a file fails to fully be uploaded, or if an upload is taking too long, the DT system is able to drop the connection and return to a stable state. These routines detect discrepancies or interruptions in data transmission, initiating fallback protocols if needed. An example of a fallback protocol would be when a user uploads invalid data, such as a .csv file that has non-utf-8 characters, the server is able to abort reading further.

Redundancy mechanisms are also integrated into the system to further safeguard against data loss, with automatic retransmission requests for packets that fail to arrive on time, ensuring that critical updates are processed accurately and consistently.

For security, the backend implemented the usage the Pquerna Go TOTP library in order to make sure that data could not be taken off the server when no authorized users were present. The backend has a token that tracks if a user is authenticated and logged into the DT. If the token is off, then the DT cannot be read from or written to. The token resets every 10 minutes, requiring a user to constantly log in using TOTP. In effect, this mandates the presence of an authorized user.

To activate the TOTP token and unlock the DT, the user needs to go to the login page and submit a correct login. After the login is correct, the user will then be prompted to scan a QR code. Afterward, they will press a button and be brought to a page where they validate their TOTP code. If it is correct, they will be brought to the main dashboard. Figure 3 shows a flowchart describing how users are able to unlock the system using TOTP.

To streamline deployment and maintain backend stability, the system incorporates a (CI/CD) pipeline using GitHub Actions. This pipeline automates testing, building, and deployment processes, providing consistent environments and minimizing potential discrepancies during execution. The CI/CD pipeline also supports rapid development cycles, ensuring that the backend remains reliable and capable of meeting real-time demands, even as updates or modifications are introduced. The CI/CD pipeline is implemented by including a .yaml file in the GitHub repository. This file allows a consistent set of commands to be run and automatically deployed in an environment.

The pipeline of CI/CD is broken into two steps. The CI step involves automated code testing in which developer-submitted code is tested for validity. This test involves making sure the code compiles and doesn't use any dependencies that have security vulnerabilities. If the code is valid, then the CD step begins. Firstly, the program is containerized into a consistent environment. Containerization is the process of creating a lightweight package for the program to run in. After containerization, the CD step can result in three different outcomes. Firstly, the container could be sent to a Google cloud server for hosting. Secondly, the container could be executed locally, with a set host machine automatically running the code. Finally, it is possible for both options to be chosen at once, with two separate hosting instances. This flexibility is an intended part of the DT system, allowing users to choose their own trade-offs. Figure 4 displays this CI/CD pipeline.

A frontend website is also included, allowing users to upload and download data from the server. The buttons on the website allow users to upload data to the backend, allowing for direct data processing. This only appears if an authenticated user is logged in. If no authenticated user is present, an error message will be displayed.

For data that corresponds to the geometry of the metal sheet, the server has a downsampling parameter that is able to either save only a certain amount of rows, save only a certain percentage of rows, or remove rows randomly until a desired filesize is reached. This is done by the user using a POST request with a parameter of size, as well as the type of downsampling that they would like. The server is able to read the POST URL and query the necessary parameters. By default, no downsampling is performed. If processing type 1 is applied alongside a decimal fraction (a number like .1 or .03) that represents a percentage, then the server will process the data down to the specified percent threshold. It does this by reading in the amount of rows,

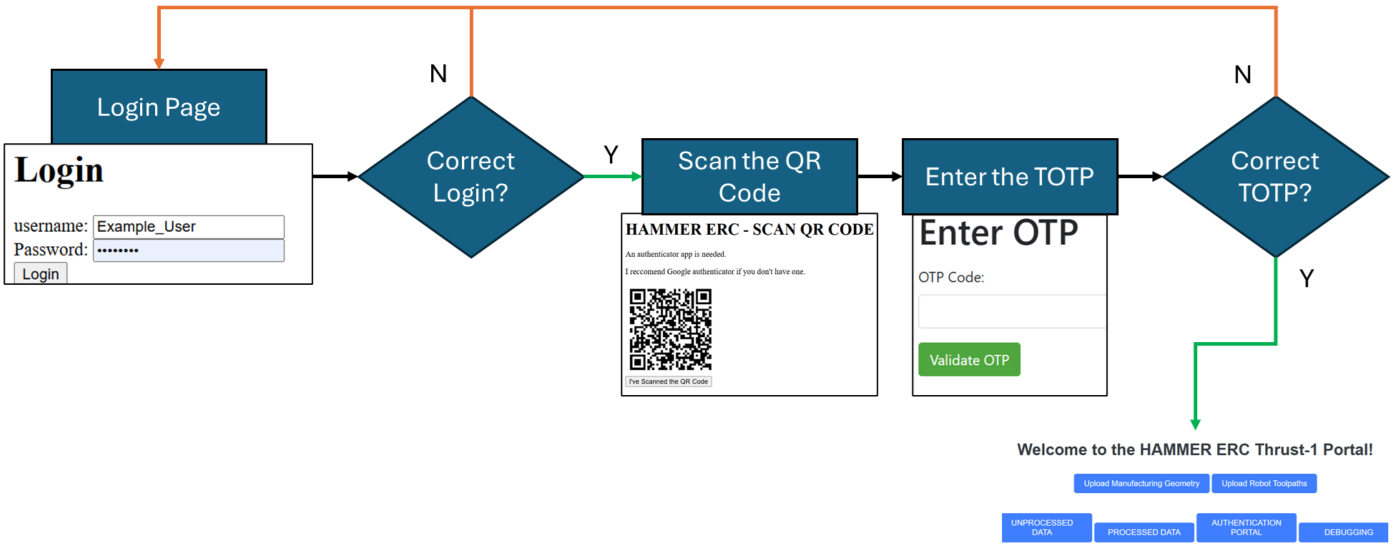


Figure 3: A user experience flowchart for using TOTP.

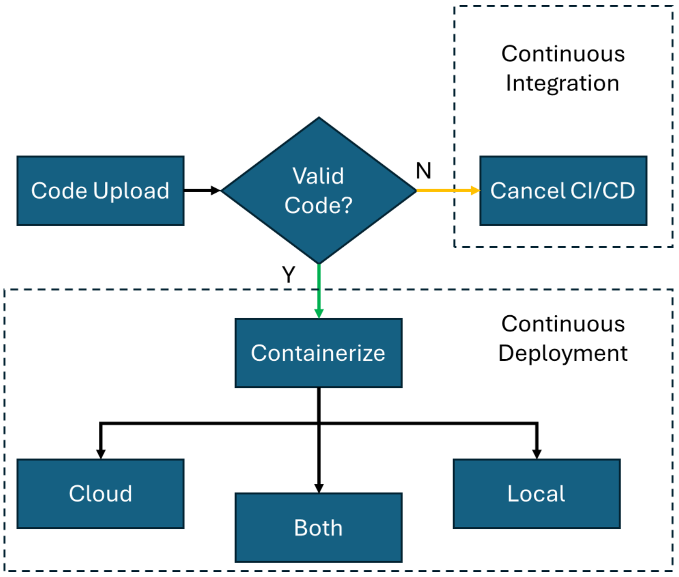


Figure 4: CI/CD pipeline.

then sampling until there is only a percent left (rounded down). Processing type 2 requires a user to put a size threshold, after which the backend will take the file and randomly remove rows until the size threshold is met. Figure 5 shows the process selection mechanism.

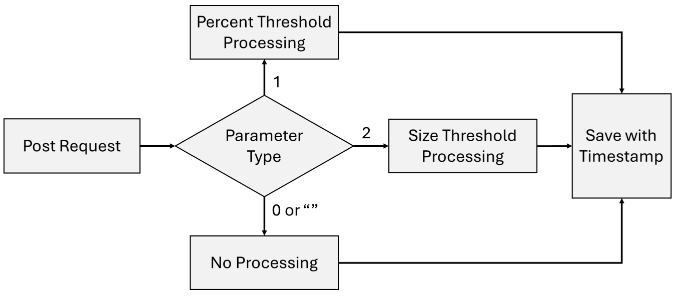


Figure 5: Selection mechanism for automated data processing using POST request and URL parameters.

In summary, the combined backend system, network protocol, and data flow management processes create a robust infrastructure that supports real-time synchronization, data integrity, and low-latency interactions in the VR-based DT framework. By leveraging Golang’s data processing capabilities, TCP/IP networking, and HTTP protocols, the system provides a responsive and stable environment for users, enabling immersive interactions with the virtual representation of industrial processes.

5. VR Application Development in Unity3D with Multi-User Functionality

To integrate VR into this project, the application uses the XR interaction toolkit provided by Unity3D without any additional packages. This was done to simplify code, allow for custom modification without breaking interconnected scripts and for ease of use. The VR application also supports desktop mode - if no VR device is detected, users will automatically be put in the desktop version.

To create an accurate virtual environment, CAD models of the English wheel and robotic arm are first imported into Unity3D. These models, developed in Rhino3D, are scaled and adjusted to match the dimensions and specifications of the physical systems, ensuring that the virtual representation aligns closely with real-world measurements. High-precision CAD modeling allows for realistic and detailed visualization, which is crucial for the immersive experience. The models are further optimized within Unity3D to maintain efficient rendering speeds, particularly important in VR where real-time performance directly impacts user engagement and immersion. Techniques such as mesh optimization, where unnecessary polygons are reduced, ensure the VR environment runs smoothly even with detailed models and high data loads. Figure 6 shows an example diagram that was used to create the English Wheel system, alongside its virtual representation. **A** corresponds to a front view of the English wheel system, **B** corresponds to a side view, **C** corresponds to a side view in Rhino3D and **D** corresponds to a side view in Unity3D. Figure 7 displays a desktop (non-VR) representation of the English wheel model, robotic arm, and metal sheet made in Unity3D. Shadows are enabled, and the lighting is improved.

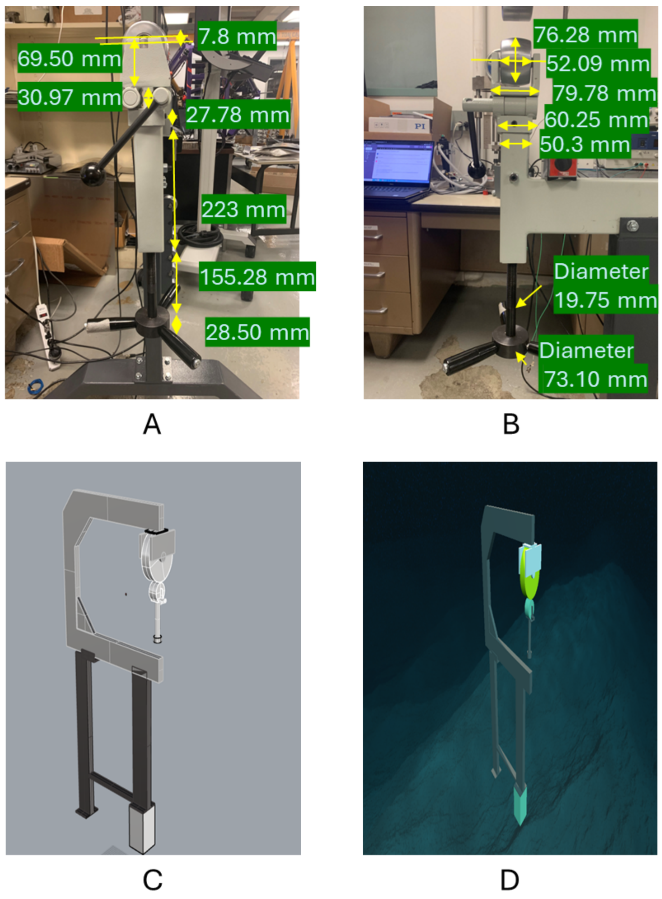


Figure 6: Physical and virtual representation of the English wheel. A) front view of the physical system, B) Side view of the physical system, C) Digital model of the English wheel created in Rhino3D, D) DT of the English wheel with colorization imported from Rhino3D to Unity3D.

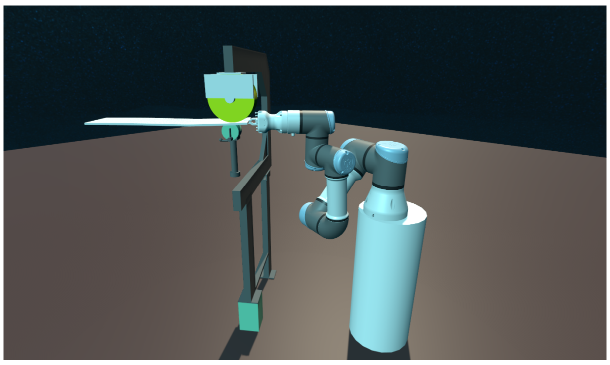


Figure 7: Visualization of the English wheel VR system, a Universal Robot 3 (UR3) robot arm, gripper and a metal sheet.

The metal sheet visualization in Unity3D is achieved by reading x, y, and z coordinates from a .csv file and rendering them as 3D objects in the scene. This process begins by parsing the .csv file with a reader function that extracts the coordinate values from each row. The parser looks for a specific pattern where each row contains three values, separated by new lines, which ensures data consistency and validity as it's loaded into Unity3D. For each row, a cube object is instantiated in the scene, and its position is assigned based on the corresponding x, y, and z coordinates from the file. This process repeats for each row until all data from the file has been rendered in the scene. The finalized metal sheet has its end placed directly at the robotic arm. The metal sheet in the virtual environment has the same proportions as the physical metal sheet (alternatively, can take data, including dimensions, from metal-forming finite element simulations). For reading in live data from the server, the VR application reads in once every X seconds, where X is a number that is able to be changed during runtime. This is so that the VR application could move with less latency when working with smaller types of data, but still allow advanced processing when working with larger types of data. The VR application also supports reading in cached time-series data.

The VR representation of the metal sheet in the DT is derived directly from its physical counterpart, which employs scanning devices to measure thousands of points on the metal sheet. A Python script consolidates this data into a CSV file and uploads it to the backend. Each measurement includes x, y, and z coordinates, where the z-value represents the depth of deformation at a particular point. Further details on the physical system are provided in [14].

Real-time data animation and interpolation methods are used to enhance the fluidity of movements and transformations within the VR environment. Unity3D scripts dynamically adjust the position, orientation, and state of the virtual robotic arm based on incoming data from the physical system, allowing the virtual model to mirror the exact movements of its real-world counterpart.

To ensure smooth visual transitions of the metal sheet between different time steps, a linear interpolation algorithm is applied, calculating incremental position and orientation changes between time steps. Every single individual point in the metal sheet has a corresponding point in future timesteps, and animation is done by moving the points along a line to their future position. This interpolation process eliminates abrupt transitions and provides a more seamless, continuous animation, preserving immersion for users and enabling precise observation of the metal deformation process. To clarify where the stresses on the sheet are greatest, a colorization function is used to display the most significant changes between timesteps. Figure 8 shows this process. Screenshots were taken over the course of 15 seconds as the metal sheet was animated. For these screenshots, the robotic arm and English wheel model were removed from the scene. This animation has been uploaded as a supplement to this manuscript.

Other types of interpolation methods exist, such as quadratic interpolation or cubic interpolation. Linear interpolation was chosen over cubic or quadratic interpolation for its simplicity and efficiency in the VR application. A core design principle is to allow for constant and consistent model updates for the robotic arm and metal sheet updates. As a result, an advanced interpolation algorithm with higher-quality fitting is deemed to be unnecessary for this specific use case. Other DT systems

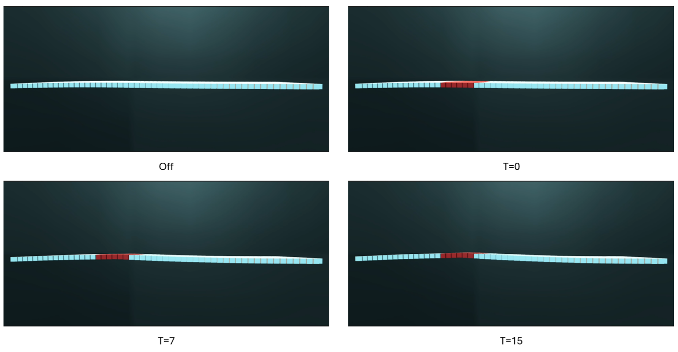


Figure 8: Sheet deformations at different time periods. A) Normal sheet and animation is off, B) Animation is started at T=0, C) Deformation at T=7, D) Deformation at T=15.

have been proven to use more advanced and accurate interpolation methods, such as inverse distance weighted interpolation [24]. [25] describes a system that uses Lagrange interpolation algorithms alongside higher-order full-discretization methods in a predictive system. Future work that involves predicting future forms of the sheet rather than strictly analyzing current ones could involve one of the previous algorithms.

There are two methods to connect the movement of the virtual robotic arm to the movement of the physical system. Movement of the robotic arm in the Unity3D scene was handled through integrating the Unity3D Robotics UR project [26] into the Unity3D environment. This integration allows a direct link from a Universal Robot TCP/IP connection into the DT. This project uses this integration by having a physical arm move, and then a virtual arm move alongside it in real time. The robotic arm modeled in the Virtual environment is a UR3 robot, however the physical robotic arm it is connected to is a UR10 robot. The robot models are interchangeable, as the TCP/IP connection is not model-specific. This method works by having the virtual and physical robotic arms communicate with cartesian coordinates, alongside direct movement commands. Notably, when connecting using the libraries involved in this method, the backend server is not involved.

Alternatively, to integrate the backend server, the usage of a TCP/IP connection is also integrated into the DT system. Users in the virtual environment are able to choose between selecting set toolpaths or specific coordinates through a UI. The server recognizes when inputs are made, and sends them to the UR robot. A key benefit of this is allowing for dynamically choosing multiple toolpaths, however further work is needed to provide unique benefits from backend server integration. Implementing an AI system that runs on the server and is able to give insight between the physical and virtual system would be an ideal addition.

The multi-user functionality of the VR application is a core feature, enabling multiple users to access and interact with the DT simultaneously. The application is designed to manage and synchronize multiple user inputs, ensuring that each user experiences consistent interactions and views the same virtual environment state. The package used for multi-user networking is known as FishNet. The Unity3D application used the default server authoritative functions involved within the package. This approach allows only the host machine to connect to the robotic arm, and have all users having an accurate representation of the robotic arm. In particular, the VR application uses TCP for connection to the physical UR machine and back-end, while using UDP for multiuser interaction. Although UDP is superior in speed and bandwidth, TCP is still used due to its superiority in maintaining connections and guaranteeing files being reliably delivered.

To enhance the VR experience, custom VR controls and user interface elements are integrated within Unity3D. These controls allow users to interact with the virtual robotic arm, adjust viewpoints, and explore the VR environment with intuitive navigation options. Unity3D's VR capabilities are customized to allow users to engage with the environment in a way that closely resembles real-life interactions, such as adjusting the position of the English wheel or observing the deformation process from different angles. The user interface is optimized for VR, ensuring that buttons, display elements, and interaction cues are easily accessible and do not obstruct the experience.

Performance optimization is a priority in developing the VR application to ensure smooth multi-user functionality and responsive interactions. Real-time data updates and multi-user interactions place a significant load on system resources, especially within VR. To address this, Unity3D's rendering settings and VR parameters are fine-tuned to balance visual quality and processing demands. These optimizations allow the application to handle high data loads, maintaining a stable frame rate and minimizing latency, which is essential for preserving the immersive experience, particularly in a multi-user setup. In particular, there is an option for shadows alongside all sources of lighting to be removed. Instead, color lighting is used, and objects are lit up by a default color (white) - removing the need for lighting calculations.

In summary, the VR application in Unity3D combines detailed modeling, real-time animation, multiple users, and optimizations to create an immersive and responsive DT environment. By utilizing custom VR controls that have the same functionality as controller or keyboard input [27], secure multi-user management, and performance-enhancing techniques, the application provides a collaborative platform where users can explore and interact with manufacturing processes in a virtual setting. This Unity3D-based VR application is central to the DT framework, enabling practical, immersive interactions that support Industry 4.0's goals of enhanced visualization, training, and operational insight.

6. Implementation Challenges and Solutions

The development of the framework described in this paper involved numerous technical challenges. Each of these challenges required specific solutions to ensure the framework's responsiveness, accuracy, and security, enabling it to meet the rigorous demands of industrial applications in Industry 4.0.

A major challenge was achieving real-time data synchronization between the physical robotic arm and its DT within the VR environment. Maintaining precise, low-latency communication was essential to ensure that the virtual model responded accurately and instantly to changes in the physical system, allowing users to interact seamlessly with the DT. To overcome this, the backend was optimized to prioritize efficient data flow, implementing the Real-Time Data Exchange (RTDE) mechanism. Additionally, Unity3D scripts were refined to handle incoming data more effectively, ensuring that updates to the virtual model occurred in real-time. These optimizations enabled the system to achieve the level of synchronization necessary for an immersive and interactive user experience.

Another significant challenge lay in optimizing performance within the VR environment, which needed to accommodate high-fidelity CAD models and process continuous real-time data updates for multiple users. The computational demands of rendering detailed 3D models, particularly when multiple users interact simultaneously, posed a risk to the system's frame rate and overall responsiveness. To address this, several optimization techniques were employed in Unity3D, including mesh simplification and data downsampling. As discussed in Section 4, the amount of data points used to represent the metal sheet was reduced significantly. This allows for better performance, without impairing visual fidelity.

Ensuring secure multi-user functionality presented another challenge, as the system had to support multiple simultaneous users while maintaining data integrity and preventing unauthorized access. To manage this, a two-factor authentication (TOTP) system was implemented, requiring an authenticated user to be logged in for the backend to allow sending and receiving data. HTTP session management was also added to track user activity securely, allowing only authenticated users to interact with the DT. These security measures prevented unauthorized access and preserved the integrity of user interactions within the shared VR space. By managing multi-user access securely, the framework supports collaborative experiences, enabling multiple users to observe and interact with the DT in a controlled, protected environment.

Deploying the DT system to a public IP address was difficult and remains a potential problem for future cybersecurity concerns. Initially, the CI/CD pipeline uploaded everything into Google Cloud where it would be hosted on servers provided by Google. This allowed for quick and easy iteration, as authorized users from any location would be able to upload and download data without the need for self-hosting. The system ran into issues when it came to performance and storage. This system uses the free tier provided by Google Cloud, and as a result, HTTP requests were relatively slow. Many cloud providers, such as Google, offer better connectivity and computing resources for higher paid tiers. Future work could analyze self-hosting an efficient, private DT server.

In summary, the development of the DT framework in Unity3D required addressing multiple implementation challenges to ensure real-time performance, secure multi-user access, and reliable synchronization. By implementing direct RTDE, optimizing VR rendering, establishing robust security protocols, and incorporating error-handling mechanisms, the framework achieved the high level of stability, accuracy, and user experience necessary for engaging, industrial-scale applications in Industry 4.0.

7. Performance, Results and Benchmarks

This section outlines the results of testing the DT framework, with a focus on performance metrics crucial to its operation as a real-time, multi-user VR system. In the context of this paper, performance refers to the time taken to execute commands (such as processing) and network latency (measured in milliseconds). It does not refer to the accuracy of the DT system.

7.1. The importance of consistent performance benchmarking

Benchmarking provides a standardized method to evaluate performance across key metrics such as latency, throughput, frame rate, and resource utilization. Without rigorous benchmarking, there may be issues with visualization, accuracy, and performance. Establishing robust benchmarking frameworks is essential for assessing the trade-offs between model precision and computational overhead, enabling engineers to optimize performance while maintaining the reliability of the DT.

However, benchmarking DT systems presents unique challenges due to their highly dynamic and complex nature. Unlike traditional software applications, DTs often integrate diverse components, including IoT devices, cloud infrastructure, AI-driven analytics, and physics-based simulations, each with distinct performance characteristics and requirements. The variability in data sources, update frequencies, and computational demands complicates the creation of standardized benchmarks. Additionally, DTs often operate in distributed environments where network latency and bandwidth constraints impact overall system performance. In practice, this makes identifying performance bottlenecks difficult.

The lack of universally accepted benchmarking methodologies further exacerbates the difficulty, making it challenging to compare different implementations and ensure consistent performance evaluations. These factors necessitate a more nuanced approach to benchmarking that accounts for the unique constraints and complexities of DT ecosystems, as well as the benefits and deficits of individual additions.

There's also an issue with the lack of a clear indicator of what constitutes an acceptable level of performance in DT systems. This is due to the unique requirements of each DT system, as well as a lack of unified standards. For example, when it comes to frames per second (fps) measurements for the VR application, 30 or 60 fps is the common industry standard across interactive media such as video games. However, according to [28] 120 fps seems to be an important threshold for VR applications for comfort and user experience. When it comes to latency, there are many different requirements for systems made by developers. When analyzing the impact of latency in the context of searching, [29] found that 1000ms can be a noticeable limit for users. Thus, this is an area where user testing and qualitative data gathering (such as surveys) would be able to help create the shape of a high-quality set of requirements.

With these limitations in mind, the upcoming sections describe different methods of processing information and sharing files. It highlights the benefits and downsides of different approaches and comments on avenues of future work. Section 7.2 analyzes the networking latencies for data transfer and processing. Section 7.3 will discuss methods of benchmarking within

Table 2: The first two columns represent how long a specific server solution took to take uploaded data and download it to storage. The last column involves how long a solution took to take uploaded data, process it, and download it to storage.

	No processing (320.36 KB file)	No processing (2 MB file)	Processing (2 MB → 512 KB)
Local Server	13 ms upload, 6 ms download	20 ms upload, 58 ms download	20 ms upload, 454 ms processing, 6 ms download
Google Cloud Server	361 ms upload, 111 ms download	484 ms upload, 446 ms download	484 ms upload, 500 ms processing, 471 ms download

the Unity3d engine, performance gaps in this solution and time taken to display data sets.

7.2. *Benchmarking backend HTTP requests*

To benchmark the speed of HTTP requests from the server to clients, Postman is used to measure the speed of requests. This will compare the uploads and downloads speeds. The local server as well as the Google cloud server will be tested. Three different files will be used. One 320.36kb .csv file that doesn't need processing, a 2mb .csv file, and a 2mb file that is processed down to 512kb.

The end benchmark will be a set of numbers representing how long it took for a particular server instance to finish uploading, downloading, and processing. Averages were taken from 5 POST requests of each type, and rounded up. For this benchmark, the hardware running the local server is an Alienware X14 gaming laptop operating on Windows 11. Another device connects to the laptop through an Ethernet/USB C interface. Table 2 compares processing speeds between methods and hardware.

Even though a Google Cloud solution allows for greater flexibility as well as easier setup, the performance it provides is lacking. In the context of the VR application, this means that there is an additional amount of latency that is added and true real-time synchronization will be harder to achieve. This limitation may be acceptable, as the system could still provide high-quality visualization for multiple users. This issue also extends to processing a 2mb file to 512kb. Currently, the local server is the fastest, however it has downsides. The biggest one, by far, is the custom networking involved in making it work. To make remote connections this paper implements SSH tunneling, which can be a point of friction, and a point of insecurity. There is also the fact that many factories and Universities have their own rules around networking and IT infrastructure, requiring more security protocols compared to a simple Google Cloud instance. Even in the free tier, Google Cloud supports authentication methods that offer solutions to these problems.

7.3. *Benchmarking Sheet Visualization in VR.*

Unity3D has a profiler for benchmarking. Developers are able to use a GUI and analyze specific function calls and rendering jobs, and see how long they take. By clicking a specific time slice, every computational process is listed, alongside how long they took. The screenshots alongside the function diagnostics are listed below. This feature exists in other game Engines, such as Unreal Engine and Gazebo. The logic discussed here applies to those. **A** corresponds to reading in a processed and non-sampled sheet (9630 rows), **B** represents another processed and half-sampled sheet (4815 rows), and textbfC corresponds to reading in a processed and quarter-sampled sheet (2409 rows). The data from this benchmark comes from a set of time series data collected from the physical system [14].

Screenshots of the sheet with different sampling methods are displayed in Figure 9. A screenshot of the profiler is displayed in Figure 10.

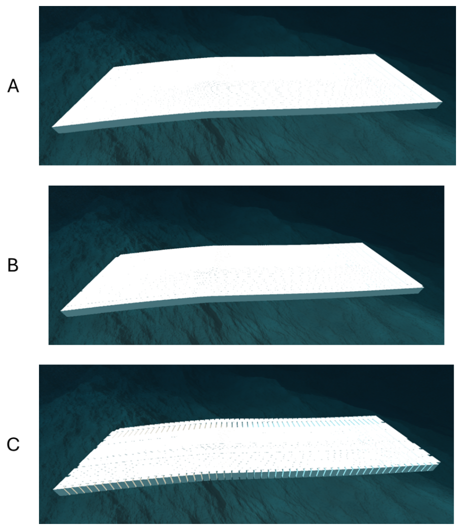
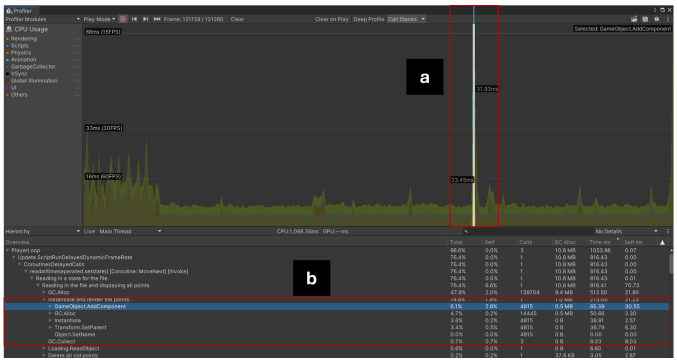


Figure 9: Base sheet with different processing applied: A) Unprocessed sheet, B) Half-sampled sheet, C) Quarter-sampled sheet.



Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms
PlayerLoop	98.6%	0.0%	3	10.8 MB	1053.98	0.07
Update.ScriptRunDelayedDynamicFrameRate	76.4%	0.0%	1	10.8 MB	816.43	0.00
CoroutinesDelayedCalls	76.4%	0.0%	1	10.8 MB	816.43	0.00
readalltimeseperated.setstate() [Coroutine: MoveNext] [Invoke]	76.4%	0.0%	1	10.8 MB	816.43	0.00
Reading in a state for the file.	76.4%	0.0%	1	10.8 MB	816.43	0.01
Reading in the file and displaying all points.	76.4%	6.6%	1	10.8 MB	816.41	70.73
GC.Alloc	47.9%	2.0%	139704	9.4 MB	512.50	21.80
Instantiate and render the points.	19.9%	1.9%	1	1.0 MB	213.00	21.23
GameObject.AddComponent	6.1%	2.8%	4815	0.5 MB	65.39	30.55
GC.Alloc	4.7%	0.2%	14445	0.5 MB	50.66	2.30
Instantiate	3.6%	0.2%	4815	0 B	38.91	2.57
Transform.SetParent	3.4%	0.5%	4815	0 B	36.79	6.30
Object.SetName	0.0%	0.0%	4815	0 B	0.00	0.00
GC.Collect	0.7%	0.7%	3	0 B	8.03	8.03
Loading.ReadObject	0.6%	0.0%	1	0 B	6.60	0.01
Delete all old points.	0.2%	0.2%	1	37.6 KB	3.05	2.87

Figure 10: Screenshot of the profiler. a) Sheet data read in with a spike corresponding to the data read, b) Duration taken to run the function.

This is useful for benchmarking short bursts of code, however, it suffers when trying to analyze longer periods of operation. By default, Unity3D stores the statistics of 300 frames (5 seconds). The amount of frames stored could be increased, but this creates a performance overhead as more frames have their information stored. As a result, this paper also implements custom stopwatch time tracking to track longer-term performance. This is done by setting a timer whenever a function begins, ending it when the function is finished, and sending the information to a log file. Despite the fact that this method has more performance overhead, it allows for easier storage and understanding of long-term performance.

For future work, the instantiation script used to visualize the points from a .csv file requires a rewrite. The profiler displays the garbage collector called a significant amount of times (over 69,000 times when trying to read in 2408 rows). This is seen in Figure 10, where users are able to Other optimization methods, such as OBJECT POOLING and HEAP MANAGEMENT under Unity could prove useful for the application.

To increase the speed of visualizing large datasets, a potential solution is the addition of pausing functions and changing thread priority. By pausing all functions of the DT, and then having all computing power solely dedicated to the initial population of metal sheet data, perhaps the results could be faster.

8. Conclusion and Future Work

In conclusion, this paper presents a flexible DT framework designed to integrate real-time data from physical industrial processes into a multi-user VR environment. By combining a high-performance backend system, optimized data flow management, and advanced VR rendering in Unity3D, this framework addresses the unique challenges of developing an immersive and interactive DT for Industry 4.0 applications.

As DT technology continues to evolve, its role in Industry 4.0 will be emphasized, encompassing broader applications and more sophisticated simulations. Ultimately, DTs provide a scalable, adaptable opportunity that embodies the core goals of Industry 4.0. This paper adds to the literature by creating a DT server that is able to automatically test and deploy itself, process data, and host multiple users in a secure environment. While this is a strong start and furthers the work of DTs in the realm of software, future work remains to be done.

Benchmarking and performance comparison remains an area that is not adequately addressed in common discussions of distributed task systems. Due to the unique goals and requirements of each DT system, the creation of a consistent set of standards will be exceptionally challenging. To address this issue, a potential solution involves authors discussing trade-offs and limitations with their proposed solutions as well as discuss their performance requirements. By doing so, future systems can identify performance gaps and establish standards based on shared objectives and program architectures. Automated performance metric collection can also support maintenance efforts, enabling the identification and correction of measurable drops in performance.

In the realm of cybersecurity, authentication and error handling have the potential to be significantly enhanced. Currently, only TOTP and a basic username/password system are used for login. Future work could include packet analysis could be used to analyze and confirm data integrity, alongside making sure no data is modified in transit. The login system could be revamped with high-quality security questions and other authentication methods besides TOTP. Encryption could also be implemented for communication of data. End-to-end encryption could also be used for secure communications, especially for transferring sensitive information.

One promising area for future work is the further integration of bidirectional data flow between the virtual and physical systems. Currently, the DT only has bidirectional operation for movement of the robotic arm and unidirectional visualization of the metal sheet. Future work would involve further English wheel integration (such as being able to synchronize upper and lower wheel positions) and the integration of artificial intelligence (AI). To solve the English Wheel integration issue, the usage of IoT sensors could be used to measure specific English wheel physical features and map them into the DT. AI integration would take much more time. AI and further time series data analysis provide a suite of benefits, with one of the most important being the ability to identify performance reductions, manufacturing uncertainties and smaller errors that would be difficult to catch otherwise.

The current web frontend offers basic security and file transfer, allowing limited DT interaction for non-programmers. This allows subject matter experts—who may not have deep programming skills—to interact with the DT system in a limited manner. Future work would include bringing the VR application to mobile devices, web browsers - alongside a ”viewport” function that allows unskilled users to see through the eyes of technical users. There also needs to be a qualitative analysis done on how beneficial the web frontend is for low-code development. Ideally, stakeholders and developers can use the web application in lieu of code, but an actual detailed case study on how it helps is lacking. Because DT systems are so technical, there exists a barrier to usage. Future work could find ways to bridge this gap and allow more people to interact with these systems.

In terms of qualitative and qualitative analysis, the usage of software-agnostic frameworks such as this allow for the same backend to be used in multiple programs. Future work can find ways to visualize the same data in different settings, and measure performance and stakeholder/developer preferences. [19] provides a framework for comparing two separate DT solution software suites, and future work could expand upon this with more data and in a more specific use case.

The framework separates VR visualization from data storage and processing, allowing dedicated resources and modular extensions—like isolated storage, security, and AI processing—to scale horizontally and reduce resource contention. For example, future work could implement machines dedicated solely to running AI models and separate other computationally-intensive tasks from visualization or interaction logic.

Enhancing the system’s multi-user capabilities is also a key area for development. Currently, the DT supports multi-user interaction, but scaling this functionality to support larger teams across distributed locations would make it suitable for broader collaborative applications. Future work includes exploring cloud-based architectures or distributed network systems. There also could be further qualitative analysis on use cases, such as utilizing this DT for training, remote troubleshooting, and collaborative planning. Improved user management features, such as role-based access control, could also be implemented to ensure data security and provide tailored experiences for different types of users.

For the VR application, asset localization (The process of precisely placing models into virtual environments where they correspond to physical environments) [20] remains an important next step of DT realism. Currently, 3D models are loaded into preset areas rather than dynamically placed into the world. Future work could analyze metadata during upload, allowing for accurate placement. Further development of application physics would greatly enhance the fidelity and immersion of the VR experience. While the current system accurately visualizes movement and basic deformations, incorporating advanced physics modeling, such as material properties, and advanced stress analysis, would create a more accurate representation of complex processes. For example, material deformation could be simulated with greater precision, making the DT more suitable for process testing and validation. The current DT system does not implement physics or collision detection, and this addition would assist in its versatility.

To address these limitations, future work will focus on gathering qualitative feedback from users and developers to analyze non-functional requirements and impacts of performance degradation in terms of both latency and FPS. This feedback will also find areas where further system flexibility is needed, allowing for further customization. The addition of AI to help decision-making will be an important milestone from the DT, and will be done through analyzing the underlying mathematical functions behind sheet deformation, as well as integrating historical time series data.

Acknowledgements

The authors appreciate the funding support from the United States Department of Commerce (USDOC), Economic Development Administration Good Jobs Challenge Awardee STEPs4GROWTH (ED22HDQ3070099), National Centers of Academic Excellence in Cybersecurity Grant (H98230-21-1-0326), and the NSF Engineering Research Center for Hybrid Autonomous Manufacturing: Moving from Evolution to Revolution (ERC-HAMMER) under Award Number EEC-2133630.

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, M. Hoffmann, Industry 4.0, Business & information systems engineering 6 (2014) 239–242.

[2] W. Hu, Y. He, Z. Liu, J. Tan, M. Yang, J. Chen, Toward a digital twin: Time series prediction based on a hybrid ensemble empirical mode decomposition and bo-lstm neural networks, Journal of Mechanical Design 143 (5) (2020) 051705.

[3] Y. He, Y. Ma, K. Huang, L. Wang, J. Zhang, Digital twin bayesian entropy framework for corrosion fatigue life prediction and calibration of bridge suspender, Reliability Engineering System Safety 252 (2024) 110456.

[4] M. Javaid, A. Haleem, R. Suman, Digital twin applications toward industry 4.0: A review, Cognitive Robotics 3 (2023) 71–92.

[5] M. Grieves, Digital twin: Manufacturing excellence through virtual factory replication (03 2015).

[6] M. Grieves, J. Vickers, Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems, Springer International Publishing, Cham, 2017, pp. 85–113.

[7] V. Karkaria, Y.-K. Tsai, Y.-P. Chen, W. Chen, An optimization-centric review on integrating artificial intelligence and digital twin technologies in manufacturing, Engineering Optimization (2024) 1–47.

[8] Y.-P. Chen, V. Karkaria, Y.-K. Tsai, F. Rolark, D. Quispe, R. X. Gao, J. Cao, W. Chen, Real-time decision-making for digital twin in additive manufacturing with model predictive control using time-series deep neural networks, arXiv preprint arXiv:2501.07601 (2025).

[9] V. Karkaria, A. Goeckner, R. Zha, J. Chen, J. Zhang, Q. Zhu, J. Cao, R. X. Gao, W. Chen, Towards a digital twin framework in additive manufacturing: Machine learning and bayesian optimization for time series process optimization, Journal of Manufacturing Systems (2024).

[10] H. Wang, L. Lv, X. Li, H. Li, J. Leng, Y. Zhang, V. Thomson, G. Liu, X. Wen, C. Sun, G. Luo, A safety management approach for industry 5.0s human-centered manufacturing based on digital twin, Journal of Manufacturing Systems 66 (2023) 1–12.

[11] L. Jin, X. Zhai, K. Wang, K. Zhang, D. Wu, A. Nazir, J. Jiang, W.-H. Liao, Big data, machine learning, and digital twin assisted additive manufacturing: A review, Materials Design 244 (2024) 113086.

[12] F. Longo, L. Nicoletti, A. Padovano, Smart operators in industry 4.0: A human-centered approach to enhance operators' capabilities and competencies within the new smart factory context, Computers Industrial Engineering 113 (2017) 144–159.

[13] A. Cimino, F. Longo, G. Mirabelli, V. Solina, S. Verteramo, An ontology-based, general-purpose and industry 4.0-ready architecture for supporting the smart operator (part ii – virtual reality case), Journal of Manufacturing Systems 73 (2024) 52–64.

[14] D. Suarez, F. Chen, P. Kang, B. Forbes, M. Gao, O. Ineza, K. Benton, N. Dewberry, C. Jaiswal, B. Gokaraju, K. Ehmann, J. Cao, On the feasibility of an integrated english wheel system, Journal of Manufacturing Systems 74 (2024) 665–675.

[15] K. Benton Jr, N. Dewberry, C. Jaiswal, S. Chowdhury, I. AlHmoud, D. Suarez, K. Ehmann, J. Cao, B. Gokaraju, Initial framework design of a digital twin mixed-reality-application on human-robot bi-directional collaboration for forming double curvature plate, Manufacturing Letters 41 (2024) 1476–1487, 52nd SME North American Manufacturing Research Conference (NAMRC 52).

[16] J. Plavšić, I. Mišković, Vr-based digital twin for remote monitoring of mining equipment: Architecture and a case study, Virtual Reality & Intelligent Hardware 6 (2) (2024) 100–112.

[17] H. Qiu, H. Zhang, K. Lei, H. Zhang, X. Hu, Forest digital twin: A new tool for forest management practices based on spatio-temporal data, 3d simulation engine, and intelligent interactive environment, Computers and Electronics in Agriculture 215 (2023) 108416.

[18] B. Starly, P. Koprov, A. Bharadwaj, T. Batchelder, B. Breitenbach, "unreal" factories: Next generation of digital twins of machines and factories in the industrial metaverse, Manufacturing Letters 37 (2023) 50–52.

[19] G. D. Wijaya, W. Caesarendra, M. I. Petra, G. Królczyk, A. Glowacz, Comparative study of gazebo and unity 3d in performing a virtual pick and place of universal robot ur3 for assembly process in manufacturing, Simulation Modelling Practice and Theory 132 (2024) 102895.

[20] M. Kamali, B. Atazadeh, A. Rajabifard, Y. Chen, Advancements in 3d digital model generation for digital twins in industrial environments: Knowledge gaps and future directions, Advanced Engineering Informatics 62 (2024) 102929.

[21] S. Davis, K. Nesbitt, E. Nalivaiko, A systematic review of cybersickness, in: Proceedings of the 2014 conference on interactive entertainment, 2014, pp. 1–9.

[22] R. Cox, R. Griesemer, R. Pike, I. L. Taylor, K. Thompson, The go programming language and environment, Communications of the ACM 65 (5) (2022) 70–78.

[23] Q.-Y. Zhou, J. Park, V. Koltun, Open3D: A modern library for 3D data processing, arXiv:1801.09847 (2018).

[24] Y. Yang, X. Zhao, Q. Cheng, R. Guo, M. Li, J. Zhou, Pod–ann as digital twins for surge line thermal stratification, Nuclear Engineering and Design 428 (2024) 113487.

[25] Y. Sun, Z. Xiong, High-order full-discretization method using lagrange interpolation for stability analysis of turning processes with stiffness variation, Journal of Sound and Vibration 386 (2017) 50–64.

[26] R. Parak, A digital-twins in the field of industrial robotics integrated into the unity3d development platform (2020-2021).

[27] N. Dewberry, I. W. AlHmoud, S. Chowdhury, B. Gokaraju, Problems and solutions of point cloud mapping for vr and cave environments for data visualization and physics simulation, in: 2023 IEEE Applied Imagery Pattern Recognition Workshop (AIPR).

[28] J. Wang, R. Shi, W. Zheng, W. Xie, D. Kao, H.-N. Liang, Effect of frame rate on user experience, performance, and simulator sickness in virtual reality`doi:10.1109/TVCG.2023.3247057`.

[29] Impact of response latency on user behavior in web search. `doi:10.1145/2600428.2609627`.