Proceedings of the 10th World Congress on New Technologies (NewTech'24)
Barcelona, Spain - August 25-27, 2024
Paper No. ICCEIA 113
DOI: 10.11159/icceia24.113

Digital Twin Applications in Construction: Creation and Interaction of Virtual and Data Layers

Yingpei Sun[1], Dr. Qing Wang[1], Prof. David Toll [1] Dr. Stefan Szyniszewski[1]
[1]Durham University
Department of Engineering, Durham City, United Kingdom
First. yingpei.sun@durham.ac.uk; Second. qing.wang@durham.ac.uk
Third. d.g.toll@durham.ac.uk; Fourth. stefan.t.szyniszewski@durham.ac.uk

Abstract **-** This paper provides an overview of the establishment of a digital twin framework for building construction. The research focuses on the construction of the virtual layer and the data layer and the connection between the two. The article discusses the choice of all technology stacks and discusses the reasons and advantages of the choice. Three.js was used to create the virtual layer in the digital twin framework, using AWS as the data processing and storage platform. Data communication between the front-end application and back-end service was achieved through HTTP protocol. HTTP as the data transfer method has comprehensive performance and low development cost, but still has technical limitations so in future we will consider adopting high-performance tools based on the UDP protocol such as Aspera. The aim of this paper is to promote the application and development of digital twins in building construction through the study of digital twins in the construction industry.

Keywords**:** Digital twin, Virtual layer, Data layer, Connectivity

1. Introduction

The construction industry is one of the main pillars of the global economy, accounting for 11% -13% of global GDP and providing 7% of employment [1]. Despite the impact of COVID-19, the construction industry has shown great resilience; according to the Oxford Economics forecast the global construction industry output value will achieve significant growth by 2030 [2]. In the Asia-Pacific region, Western Europe, and North America, it is expected to grow by 50%, 23%, and 32% respectively. However, the construction industry has an average profit margin of only 5%. It faces many challenges, such as participation of an informal labour force, the shortage of skilled workers, project delivery delays, and quality and safety [3-4]. In addition, resistance to change within the industry has made the construction industry one of the least digitized globally [5], which has led to inefficiencies, delays, and cost overruns.

With this background, digital twin (DT) has the ability to provide dynamic digital reflection of the behavior or function of physical entities, showing its revolutionary potential. DT not only improves decision-making, optimization, and management processes but also has the possibility to completely change the overall mode of construction project management by realizing real-time simulation and visualization. Our research aims to improve the overall performance and user experience of construction DT, by building and optimizing the interaction between the virtual layer and the data layer to improve data integration and real-time interaction. In the current research, the researchers aim to develop a comprehensive DT framework that includes a physical layer, virtual layer, data layer, and service layer. This work focuses on the development of the virtual layer and implementation of the data layer, as well as the interaction between these two layers. In the virtual layer, Three.js was used, we can build an intuitive virtual environment, allowing users to view building data more conveniently and enhance the sense of interaction. Amazon Web Services (AWS) was chosen as the data layer for the framework. AWS has a wide range of services and tools that can be used to collect, store, process, and analyse large amounts of data from construction projects. As for the connection between the virtual layer and the data layer, an interaction process was designed based on HTTP requests. The process starts with an HTTP request initiated by the front-end component, and after the request reaches the back-end controller, it interacts with the database to retrieve and return data. Once the data is returned, it is dynamically displayed on the front-end interface, realizing the connection of the data layer and the flow of data

through the complete full-stack application. This cycle embodies the foundation of full-stack application operation, that is the dynamic data exchange between the underlying architecture of data storage and the user-oriented Web technology.

In summary, our research work not only provides an innovative way to process and display construction data but also promotes the transformation of the construction industry to digitalization and automation.

2. Literature review

DT originated from NASA's Apollo program in the 1960s [6]. Until 2002, Dr. Michael Grieves named this concept digital twins [7]. The definition of DT has not yet been unified in academia and industry. Different industries and fields have their own definitions of DT, such as aerospace: Digital twin is an integrated multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system [8]. Manufacturing: Digital Twin is a highly realistic model that mirrors the current state of a process and its behavior in interaction with its real-world environment [9]. Construction: Digital twins are a method for assessing the performance and energy efficiency of buildings by combining real-time data from physical systems with digital models [10]. In short, the core concept of DT is Mirroring and Optimization. The DT framework needs to encompass the mapping of physical entities to virtual models and the optimization of projects. Researchers will also modify the DT framework according to different projects. For example, Han et al.'s DT framework for smart hospitals contains an acquisition level (responsible for collecting data), modelling level (responsible for building models), and service level (responsible for optimizing services) [11]. Tao et al. built a five-dimensional DT model containing physical shop floor (PS), virtual shop floor (VS), shop floor service system (SS), and shop floor digital twin data (DD), as well as the connection of the components [12].

Creating a virtual layer is a good start for building a DT framework. The virtual layer is a copy of a physical entity that represents the entity in digital form [14]. According to Tao [13], a virtual layer should include geometric modelling, physical modelling, behavioural modelling, and rule modelling. They correspond to the geometric parameters, physical properties, evolutionary behaviour, and historical data changes of physical entities respectively. Liu et al [15] summarised the virtual layer establishment method as follows: firstly, the modelling paradigm is selected. Then the physical and geometric models are developed using specialised software (ANSYS, ABAQUS, etc.). Then the Behavioural model is developed using programming languages and algorithms (C++, Neural Networks, Markov Chains, etc.). All Modelling paradigms are then integrated to ensure that they can work together. Finally, the accuracy and reliability of the model is verified and optimised by comparing it with the physical data. In Forest DT studied by Qiu et al [16], in order to build the virtual layer, they firstly constructed the Forest DT terrain using texture images, remote sensing techniques and Cesium Earth Engine. Then an interactive forest digital twin visualisation scene was constructed using Unreal Engine 4 and 3D tree species database. Finally Forest DT imports spatial location information, field measurements from the physical world to update in real time. The study by Wang et al [17] is interesting, as they combined DT with gaming. Geographic and architectural data were first collected for a specific area, then a virtual model of the city was built in Minecraft, and then participants were called to interact and modify it. In the transport DT model constructed by Kušić et al [18], actual traffic data was first collected and then pre-processed with noise reduction and normalisation. Then a virtual model was constructed using the Simulation of Urban Mobility (SOM) tool. The virtual model was then updated using real-time data streams. Finally the model was compared with the actual traffic data to verify its accuracy.

In the data layer, DT-related data are multi-sourced and massive, requiring big data storage techniques [14]. The selection criteria for storing databases are accessibility, scalability, high performance, and management capabilities [19]. Although there are many types of databases, researchers all have very different approaches to building data layers [15, 18, 20]. This requires the use of sensors to collect data from the physical entities; then pre-processing the data (filtering, normalisation, etc.); then storing the data in a database; and finally, using data exchange protocols (MQTT, AMQP, etc.) in order to achieve the transfer of the data between the physical entities and the virtual model. There are also a few ways to connect the virtual and data layers. Tao et al. [21] used JDBC (Java database connectivity) and ODBC (Open Database Connectivity). On the one hand, the relevant data generated by the virtual layer can be stored in the data layer in real-time, and on the other hand, the relevant data of the data layer can be read in real-time to drive dynamic emulation. The construction site management research by Pauwels et al. [22] both use JDBC as the API of the database.

3. Methodology

Based on the above mentioned, we propose a DT core concept and a high-dimensional DT framework. Figure 1 shows shows the DT framework for building construction management. This framework consists of the physical layer, virtual layer, layer, data layer and service layer.

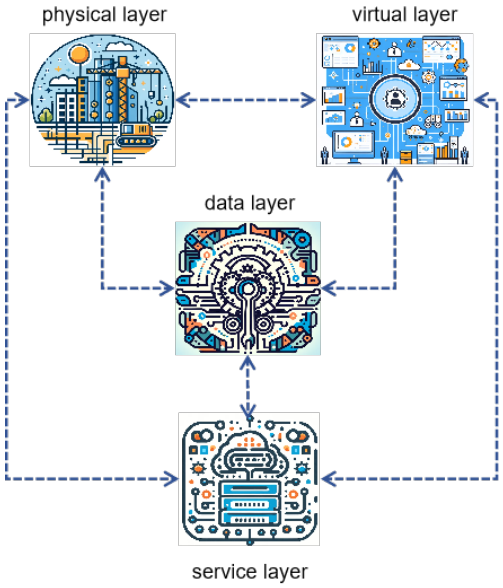

Fig. 1: Digital twin framework.

This research focuses on the building of the virtual and data layers, and their interconnections. In terms of the virtual layer, our work focuses on the development of an intuitive interface that presents the building model and provides relevant information about the construction. For the data layer, this research will establish a centralized data management system. In terms of connecting the virtual and data layers, our work focuses on seamless integration and efficient data exchange between the two. Data will be transferred from the back-end to the front-end via APIs, enabling full-stack application operations.

3.1. Virtual layer development

HyperText Markup Language (HTML), has been widely used to visualize models and data on Web pages, usually supplemented by Cascading Style Sheets (CSS) to define the structure and style of web pages. Three.js, a WebGL-based JavaScript library that encapsulates WebGL, makes it easier to create and display 3D graphics in a web browserJavaScript frameworks are designed to help developers quickly complete common web development tasks, improve development efficiency, with Vue being a widely used progressive JavaScript framework for its simplicity, flexibility, and ease of use.

The above describes the technology stack used to build the virtual layer. The following section describes how to build a virtual layer. Developing a virtual layer using Three.js usually includes a few key functions: creating the scene, setting up the camera, creating the renderer, adding light sources, loading the model, user interaction, and other functions. The code shown in Figure 2 first creates a Three.js scene using 'const scene = new THREE.Scene()'. This scene is the container for all the 3D objects and light sources. Next the perspective camera is created and its position is set. The third step is to create the WebGL renderer, where { alpha: true } means rendering the background transparent. Next the ambient and directional light is added, the ambient light provides uniform light and the directional light simulates a light source like the sun. Next comes 'OrbitControls'. Creating OrbitControls allows the user to interact with the 3D scene by rotating, zooming, and panning the scene with the mouse. Lastly the loading of the model requires converting the building model to gltf format and then creating the 'GLTFLoader',. This is used to load the 3D model in gltf format under the specified path. In the creation of 'GLTFLoader',

we also set up a callback function. The model is loaded and the callback function adjusts its position and rotation appropriately to ensure that the model is displayed properly in the Three.js scene.

```
// construction scene
const scene = new THREE.Scene();
// construction camera
const camera = new THREE.PerspectiveCamera(
  50,
  container.clientWidth / container.clientHeight,
  0.1, 1000,);
camera.position.set(50, 70, 200);
// construction renderer
const renderer = new THREE.WebGLRenderer({ alpha: true });
renderer.setSize(container.clientWidth, container.clientHeight);
container.appendChild(renderer.domElement);
// construction ambientLight
const ambientLight = new THREE.AmbientLight(0xffffff, 2);
const directionalLight = new THREE.DirectionalLight(0xffffff, 3);
directionalLight.position.set(50, 150, 50);
scene.add(ambientLight);
scene.add(directionalLight);
 // construction OrbitControl
const controls = new OrbitControls(camera, renderer.domElement);
// GLTFLoader
const loader = new GLTFLoader();
let model: THREE.Object3D<THREE.Event> | THREE.Group;
loader.load("model/modelHouse.gltf", (gltf) => {
  model = gltf.scene;
  model.rotateX(-Math.PI / 2);
  model.rotateZ(-Math.PI / 2);
  model.translateX(-160);
  scene.add(model);
});
```

Fig. 2: 3D Scene Construction Code Snippet.

In the virtual layer, in addition to the need to display the model, it is also necessary to display building-related information, such as floor plans or building components. Therefore, relevant triggers need to be set in the virtual layer interface so that users can quickly query building information. Figure 3 shows the code that triggers the display of floor plans. Firstly, the ‘ref()’ function from Vue.js 3's Composition API was utilised to declare a reactive reference named ‘floorPlanDialog’. Secondly, in event handling and interaction design, the 'Model' button on the page was directly bound to the ‘floorPlanDialog.showModal()’ method using Vue.js's ‘@click’ directive. This means that when the user clicks the button, Vue.js's event handling system will invoke the ‘showModal()’ method of the ‘floorPlanDialog’ reference without the need for additional methods or complex event handling logic. Thirdly, in DOM manipulation, ‘showModal()’ is a native DOM method of the ‘<dialog>’ element, which is used to display the mod-al dialogue. In the Vue template, a reference to the ‘<dialog>’ element was defined using the ‘ref’ attribute and associated with the button's click event using Vue.js's binding mechanism. Lastly, within the ‘<dialog>’ element, the building's floor plan PDF file was embedded using an ‘<iframe>’ tag. The size of the ‘<iframe>’ was set to ensure it adapts to the modal dialogue's size and the path to the PDF file was specified using the ‘src’ attribute. In this way, a responsive interface is achieved, allowing users to preview the building's relevant information with a single click without being navigated away from the current page or interrupting the user experience.

```
import { ref } from "vue";
const floorPlanDialog = ref();
function showFloorPlan() {
    floorPlanDialog.value.showModal();
  }
  <template>
  <div class="dashboard">
    <div class="item" @click="showFloorPlan">
      <div class="title">Model</div>
    </div>
    <dialog ref="floorPlanDialog" class="modal floor-plan">
      <iframe
        height="90%"
        src="file/building.pdf"
        width="98%"
      ></iframe>
      <button class="btn" @click="floorPlanDialog.value.close()">Close</button>
    </dialog>
  </div>
</template>
```

Fig. 3: Vue Floor Plan Dialog Code Example.

3.2. Data Layer Development

Spring Boot is an open-source framework for Java applications, inheriting Spring Framework's features and streamlining project development with simplified configurations. It seamlessly integrates web services and database operations through Spring Boot starters, simplifies MySQL database connectivity and queries, and offers a built-in server for efficient deployment.Amazon EC2 offers a flexible cloud computing environment for deploying and running Spring Boot applications and MySQL databases in virtualized instances.

The above is about the technology stack used when setting up the data layer. When deploying the data layer, there are mainly the following two steps. The first step is to build the backend service. In this step, a Spring Boot project was first created using IntelliJ IDEA and the required dependencies were selected (e.g. Web, JPA, MySQL). Next the data model was defined using @Entity annotation and the database tables and fields were mapped using @Id, etc. annotations. Next the database connection was created and the database connection information were set including URL, username and password, etc. Finally the API endpoints were written; Taking data retrieval as an example, the endpoint for GET request was mapped to the root path/power-consumption of the controller by @GetMapping annotation. The client can send a GET request to this path and perform data retrieval as shown in Figure 4. The second step was to configure the database. First the database was installed, using the command line interface provided by Amazon EC2, executing the command to install the MySQL database. Next the network was configured, making sure that the database listens to the appropriate ports and allows access from specific IPs by modifying the database configuration file. The last step was to create the database and users and set the appropriate permissions.

```
// API endpoint
@GetMapping
public Map<String, List<BigDecimal>> list(@RequestParam String date) {
    List<Map<String, Object>> mapList = jdbcTemplate.queryForList(
                "select * from power_consumption where column_b = ?", date);
    Map<String, List<BigDecimal>> result = new HashMap<>();
    mapList.forEach(map -> {
        List<BigDecimal> list = new ArrayList<>();
        map.keySet().forEach(key -> {
            if (!notColumns.contains(key)) {
                Object value = map.get(key);
                list.add(value != null ? new BigDecimal(value.toString()) : null);
            }
        });
        result.put(map.get("column_a").toString(), list);
    });
    return result;
}
```

Fig. 4: API endpoints for data retrieval.

3.3. Virtual layer and data layer connectivity

During the interaction between the virtual layer and the data layer, the data displayed in the virtual layer needs to uploaded first. To implement the data upload process, the data needs to be formatted first. Then the upload logic is using the HTTP library in the front-end code to send the data to the back-end API endpoint. The next step is the of the front-end interface. Although the front-end code was completed during the deployment of the virtual layer, in this it is necessary to continue writing code to call the back-end API, get the data and save it in the state of the front-end application. This is followed by data visualisation, using a charting library to render the data to the user interface, ensuring that the data is real-time and accurate. The code snippet shown in Figure 5 is integrating data calls and data visualisation. The next step is the front-end and back-end integration testing. First, the front-end and back-end are run in the local environment to ensure that they can interact correctly and that the front-end can receive data from the back-end. Then the front-end and back-end were deployed in the development environment or pre-production environment to perform integration testing and to check the functionality and performance of the whole system. Finally, the Spring Boot application was packaged into a JAR, uploading it to the EC2 instance, and starting it as a service.

```
function refresh() {
  axios
    .get(`http://35.178.189.142:8080/power-consumption?date=${date}`)
    .then(({ data }) => {
      option.value = {
        title: {
          text: date,
          left: "center",
          top: "5%",
        },
        tooltip: {
          trigger: "axis",
          axisPointer: {
            type: "shadow",
          },
        },
        xAxis: {
          type: "category",
        },
```

Fig. 5: Data calling and visualisation code snippets.

4. Discussion

Figure 6 shows the virtual layer interface developed for this research development. As shown in the figure, the interface contains a 3D model, and several data graphs and information buttons.The 3D model can be scaled, rotated, and other interactions just like a Revit model. In order for the Echart to dynamically display relevant data, a timer was added to the front-end Echart component, which ensures that the Echart is automatically refreshed every 30 seconds. Click on the information button, the interface will smoothly display relevant information, such as building components and floor plans. The virtual layer is deployed online, and users can access the virtual layer directly through the browser without the need for specialised software, which is a good way to avoid information silos. During front-end testing, Mock API Development was adopted. Its fundamental purpose is to emulate the responses and behaviours of actual servers, enabling developers to continue with front-end development and testing even without the completion of back-end services. The basic principle of Mock service is to intercept HTTP requests from front-end applications, match the requests and return pre-defined mock response to simulate the behaviour of the back-end service.

As the front-end framework is used to develop the virtual layer, once the virtual layer is running, we can analyse the performance and results of the virtual layer through the developer tools of the web page to determine whether the development is successful or not. Through the developer tool we found that the web request time was 57 milliseconds, which is a good indicator of the corresponding speed, showing that the back-end server responds quickly to requests.The FCP (First Contentful Paint) of 1.5 seconds is a reasonable value, but it can be improved through optimisation of the resources, reducing the response time of the server or asynchronous loading of non-critical resources. LCP (Largest Contentful Paint) is also 1.5 seconds. This indicates good page performance, meaning that most of the content can be rendered to the user quickly. In terms of execution efficiency, 'Expensive function call' appears in the performance panel of the developer tool, which means that there is a performance bottleneck during JavaScript execution. The functions

take a long time to execute and consume a lot of resources, causing the main thread to block, which in turn affects the page's responsiveness and the user's interactive experience.Although there are some flaws in the development results, in general, the performance of the virtual layer and data layer of this development is good, and it can be seen as a successful case implementation.

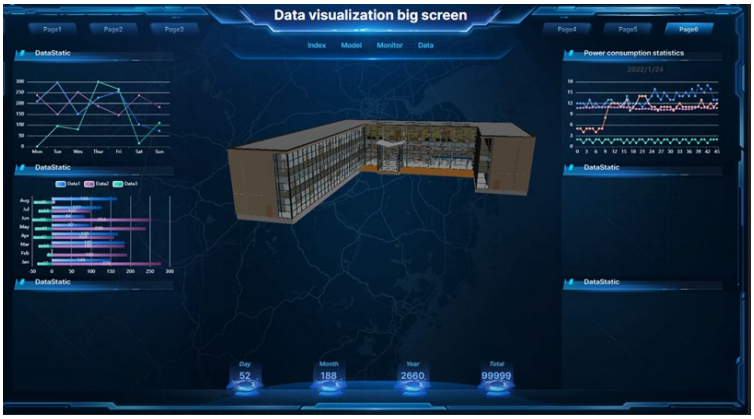

Fig. 6: Virtual Layer Interface.

5. Conclusion

During our research on building construction based DT, the DT framework was divided into a physical layer, virtual layer data layer and service layer. We worked on developing a good data layer, virtual layer, and connecting the two together for data and model visualisation. Three.js was chosen as the technology stack for developing the virtual layer and the development process of the virtual layer is detailed. Next, a powerful backend service was built using Spring Boot as a data layer for processing and storing data. Finally, the data exchange was implemented between the front and back end, i.e. the connection between the virtual and data layers, through a HTTP. We made sure that the front-end was developed correctly by testing through a Mock service. Then the whole framework was analysed based on the performance tests in the developer tools of the web page. We found that the JavaScript performance bottleneck in this framework required optimisation of the relevant functions, but the overall performance was good and it was a successful development.

Our current technology uses HTTP as the data transfer method, which has comprehensive performance and low development cost, but still has technical limitations. In the future, we can consider adopting high-performance tools based on the UDP protocol such as Aspera, in order to provide higher data transfer speeds and support for cross-platform and cloud storage. In future work, our research will also focus on further refining the application of digital twin technology in construction. Building the physical layer, and the connections between the physical and virtual layers, and the data layer. This will include the use of advanced sensor technologies and communication protocols (e.g. MQTT) to improve the efficiency and accuracy of data collection. In terms of service layer development, our research will leverage machine learning and artificial intelligence algorithms to enhance data analysis and processing. The goal is to provide in-depth and actionable analyses for project management through these automated data processing tools. Additionally, the research will focus on building efficient data interaction systems that ensure bi-directional data flow between the service and data layers. This will involve optimising data processing processes to improve processing efficiency and data accuracy.

References

[1] M. Buckley, A. Zendel, J. Biggar, L. Frederiksen, and J. Wells, "*Migrant Work and Employment in the Construction Sector*," Geneva, ILO, 2016.

[2] Marsh, The Future of Construction, Marsh, 2021. [Online]. Available: https://www.marsh.com/uk/industries/construction/insights/the-future-of-construction.html.

[3] M.J. Ribeirinho, J. Mischke, G. Strube, E. Sjödin, J.L. Blanco, R. Palter, J. Biörck, D. Rockhill, T. Andersson, J.L. Blanco, R. Plater, and D. Rockhill, 'The next normal in construction: How disruption is reshaping the world's largest

ecosystem', McKinsey & Company, June 4, 2020. [Online]. Available: https://www.mckinsey.com/capabilities/operations/our-insights/the-next-normal-in-construction-how-disruption-is-reshaping-the-worlds-largest-ecosystem (accessed Mar. 13, 2024).
[4] D. Oswald, F. Sherratt, and S. Smith, "Problems with safety observation reporting: A Construction Industry Case Study," *Safety Science*, vol. 107, pp. 35–45.
[5] D. Young, K. Panthi, and O. Noor, "Challenges involved in adopting Bim on the construction jobsite," EPiC Series in *Built Environment*.
[6] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications," IEEE Access, vol. 7, pp. 167653–167671, 2019.
[7] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," *Transdisci-plinary Perspectives on Complex Systems*, pp. 85–113, 2016.
[8] E. Glaessgen and D. Stargel, "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles," 53rd AI-AA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference
20th AI-AA/ASME/AHS Adaptive Structures Confer-ence
14th AIAA, 2012.
[9] R. Rosen, G. von Wichert, G. Lo, and K. D. Bettenhausen, "About The Importance of Autonomy and Digital Twins for the Future of Manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567–572, 2015.
[10] S. Kaewunruen, P. Rungskunroch, and J. Welsh, "A Digital-Twin Evaluation of Net Zero Energy Building for Existing Buildings," *Sustainability*, vol. 11, no. 1, p. 159, 2018.
[11] Y. Han, Y. Li, Y. Li, B. Yang, and L. Cao, "Digital Twinning for Smart Hospital Operations: Framework and proof of concept," *Technology in Society*, vol. 74, p. 102317, 2023.
[12] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F, Sui, "Digital twin-driven product design, manufacturing and service with Big Data," *The International Journal of Advanced Manufacturing Technology*, vol. 94, no. 9–12, pp. 3563–3576, 2017.
[13] F. Tao and M. Zhang, "Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing," in IEEE Access, vol. 5, pp. 20418-20427, 2017.
[14] Q. Qi, F. Tao, T. Hu, N. Anwer, A. Liu, Y. Wei, L. Wang, and A.Y.C. Nee, "Enabling technologies and tools for digital twin," *J. Manuf. Syst.*, vol. 58, pt. B, pp. 3–21, 2021,
[15] X. Liu, D. Jiang, B. Tao, F. Xiang, G. Jiang, Y. Sun, J. Kong, and G. Li, "A systematic review of digital twin about physical entities, virtual models, twin data, and applications," *Adv. Eng. Inform.*, vol. 55, Art. no. 101876, 2023.
[16] H. Qiu, H. Zhang, K. Lei, H. Zhang, and 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,* vol. 215, p. 108416, 2023, ISSN 0168-1699.
[17] S. Wang and L. H. Vu, "The integration of digital twin and serious game framework for new normal virtual urban exploration and social interaction," *Journal of Urban Management*, vol. 12, no. 2, pp. 168-181, 2023, ISSN 2226-5856.
[18] K. Kušić, R. Schumann, and E. Ivanjko, "A digital twin in transportation: Real-time synergy of traffic data streams and simulation for virtualizing motorway dynamics," *Advanced Engineering Informatics*, vol. 55, p. 101858, 2023, ISSN 1474-0346.
[19] V. V. Tuhaise, J. H. M. Tah, and F. H. Abanda, "Technologies for digital twin applications in construction," *Autom. Constr.*, vol. 152, Art. no. 104931, 2023.
[20] J. Song, S. Liu, T. Ma, Y. Sun, F. Tao, and J. Bao, "Resilient digital twin modeling: A transferable approach," *Advanced Engineering Informatics*, vol. 58, p. 102148, 2023, ISSN 1474-0346.
[21] F. Tao and W. Liu, "Five-dimension digital twin model and its ten applications," *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*, vol. 25, no. 1, January 2019, pp. 1-18.
[22] P. Pauwels, R. de Koning, B. Hendrikx, and E. Torta, "Live semantic data from building digital twins for robot navigation: Overview of data transfer methods," *Advanced Engineering Informatics*, vol. 56, 2023, Art. no. 101959, ISSN 1474-0346.