Mechatronics and Intelligent Transportation Systems
https://www.acadlore.com/journals/MITS

Design and Implementation of a Digital Twin System for Monitoring Automated Container Terminal Equipment

Houjun Lu*, Bo Zhang, Leike Hou

School of Logistics Engineering College, Shanghai Maritime University, 201306 Shanghai, China

* Correspondence: Houjun Lu (hjlu@shmtu.edu.cn)

Received: 01-06-2024 **Revised:** 02-28-2024 **Accepted:** 03-10-2024

Citation: H. J. Lu, B. Zhang, and L. K. Hou, "Design and implementation of a digital twin system for monitoring automated container terminal equipment," *Mechatron. Intell Transp. Syst.*, vol. 3, no. 1, pp. 55–72, 2024. https://doi.org/10.56578/mits030105.

Abstract: To address the lack of multi-perspective, real-time monitoring and management of operations and equipment in automated container terminals, a digital twin system targeted at monitoring automated container terminal equipment has been designed and developed. Based on the concept of a five-dimensional model of digital twins, a digital twin framework for monitoring automated container terminal equipment was constructed. The system's maintainability is enhanced through a layered design, which also reduces coupling between different functional modules. A multi-dimensional, multi-scale virtual scene was built and model consistency evaluations were conducted to verify the system. The system's operational efficiency was improved by optimizing model rendering with discrete level of detail (LOD) techniques. A multi-layered distributed solution for the digital twin system was proposed to achieve multi-perspective monitoring. Ultimately, using a specific automated container terminal as a case study, a system prototype was developed, realizing multi-perspective digital monitoring of terminal operations and equipment. This project offers a solution for the application of digital twin technology in the field of automated container terminals and promotes the development of intelligent digital terminals.

Keywords: Automated container terminal; Equipment monitoring; Digital twin; Data-driven; Distributed

1 Introduction

In recent years, with the accelerated development of global trade, domestic and foreign ports have successively started to build automated container terminals to improve port operational efficiency and increase throughput [1–3]. Unlike traditional terminals, the operation equipment of automated container terminals, such as quay cranes and yard cranes, generally adopts an "automatic operation + remote control" automation mode, where equipment control and monitoring are centralized in the terminal control center [4]. Since the loading and unloading operations at automated container terminals are controlled remotely by management personnel, it is impossible to enter the operation site in a timely manner to inspect unmanned operation areas and handle exceptions [1], making real-time monitoring of frontline operation equipment especially important. In this context, Bozzo et al. [5] designed a new system called MOCONT (Monitoring the Yard in Container Terminals), which integrates a positioning module, a visual recognition module, and a synchronization and communication module to automatically track the entire container loading and unloading process. Lee et al. [6] used machine vision technology and radio frequency identification technology to provide crane operators with an enhanced workspace view, significantly improving crane operation efficiency. Yang et al. [7] proposed a multi-sensor intelligent monitoring system, composed of multiple sensor clusters, signal transmission, and data acquisition systems, using Wi-Fi or Zigbee wireless transmission technology to obtain structural parameters of key parts of equipment, and using a central computer to manage and analyze these parameters for real-time monitoring of crane working conditions. Kim [8] proposed a TCP/IP-based remote monitoring system for remote monitoring of port container terminal yard cranes. Awad et al. [9] explored different applications of intelligent sensor networks in mobile port cranes, using an IoT-PLC control system to monitor and control mobile port cranes in the cloud. Li and Liu [10] developed a data-driven remote monitoring and alarm system for tower cranes to replace camera monitoring systems, using the Module-Position-Attitude-Scale (MPAS) method to describe the parts of tower cranes, and through parameterization and modularization, made the construction of virtual cranes simpler.

Digital twin is an emerging technology for digital transformation and intelligent upgrading [11], making the interaction between virtual and physical spaces possible. In 2003, Professor Michael Grieves first introduced the concept of digital twins in his Product Lifecycle Management course at the University of Michigan [12], and later in the first digital twin white paper, Professor Grieves proposed that the three-dimensional model of digital twins consists of a physical product in real space, a virtual product in virtual space, and data and information links that connect virtual and physical products [13]. Driven by data and models, digital twins can perform monitoring, simulation, prediction, optimization, and other functions [11]. As a highly intelligent product of the manufacturing and shipping industries, automated container terminals are an excellent stage for exploring digital twin technology [4]. How to apply digital twin technology to automated container terminals to effectively monitor and manage terminal operation equipment has attracted many scholars for in-depth research. Martínez-Gutiérrez et al. [14] proposed a new digital twin design concept based on external services to monitor the accuracy of the total time for automated guided vehicle (AGV) tasks in automated container terminals. Hofmann and Branding [15] proposed an IoT and cloud-based digital twin for intelligent monitoring and real-time decision support in port operations. Yang et al. [16], based on the analysis of the characteristics of large-scale comprehensive port operations, proposed a framework for a digital twin application system, elaborating on the overall design process and functions of the digital twin system, with three-dimensional visual monitoring and real-time perception data-based optimization scheduling as core functions, achieving intelligent port operations. Szpytko and Duarte [17] proposed a method for building and online monitoring of a digital twin model for container terminal cranes from the perspective of enhancing the effectiveness of maintenance decision conceptual models. Agostinelli et al. [18], focusing on the Port of Anzio in Italy, analyzed the control of a port renewable energy system using open-source tools and digital twin models, developed energy-saving programs and port equipment monitoring strategies, and integrated renewable energy production systems to achieve sustainability. Zhou et al. [19] proposed a digital twin-based framework for monitoring the operation of port cranes, combining multi-sensor data collection methods, Open Platform Communications Unified Architecture (OPC UA) information models, and other methods to achieve fusion of multi-source heterogeneous virtual and physical data, and using this method, constructed a physical platform for monitoring track-based gantry cranes based on digital twins.

However, in current implementation cases, systems generally do not have distributed functionality, and can only be displayed and operated on one computer at a time, limiting the system's convenience and scalability. When the dock production site needs to establish monitoring screens from multiple angles simultaneously, the system cannot perform its intended function. Therefore, this paper, based on the theory of the five-dimensional architecture of digital twins, designed a five-dimensional architecture of a digital twin system for monitoring automated container terminal operation equipment, and analyzed the composition and specific functions and technologies of the five-dimensional architecture. The construction and optimization methods of the digital twin virtual scene for automated container terminals were discussed, and the distributed implementation process of the system was detailed. Finally, using a certain automated container terminal as a prototype, a digital twin system for monitoring automated container terminal operation equipment was designed and developed, and its feasibility was verified through testing.

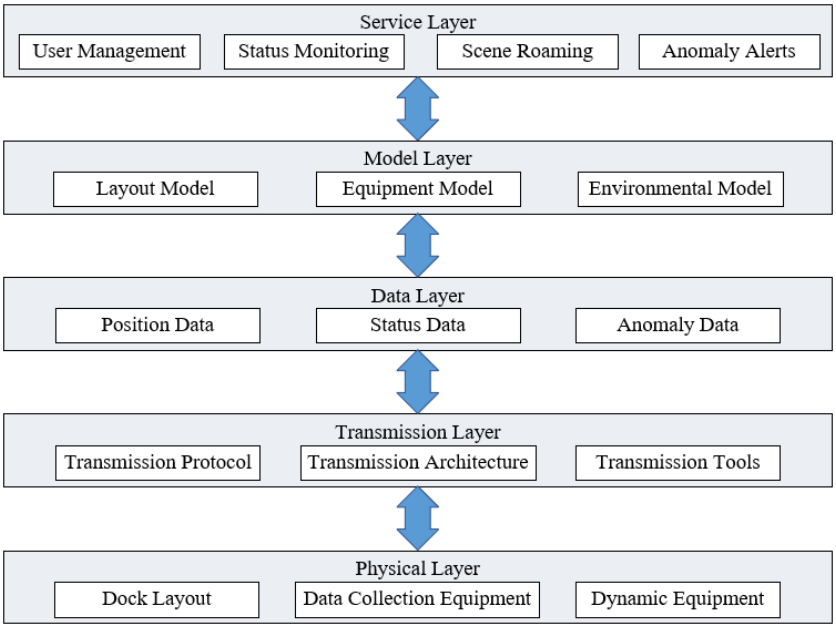


Figure 1. Overall architecture of the digital twin system for monitoring equipment at automated container terminals

2 Architecture of the Digital Twin System for Monitoring Equipment at Automated Container Terminals

To achieve efficient monitoring of equipment at automated container terminals, a five-dimensional architecture of the digital twin system for monitoring automated container terminal operation equipment was constructed based on the concept of the five-dimensional model proposed by Tao et al. [20]. This architecture, shown in Figure 1, is composed of physical space, the transmission layer, the data layer, the model layer, and the service layer. Through layered design, not only is the system's maintainability enhanced, but the encapsulation of specific functions within each level also significantly improves the decoupling between functional modules [21]. Therefore, when modifying one layer independently, the impact on other layers is effectively reduced, thereby enhancing the system's flexibility and scalability [22].

(1) Physical space: The physical space is the foundation of the entire digital twin system architecture, encompassing various dynamic equipment at automated container terminals that need monitoring, such as quay cranes, yard cranes, and AGVs. It also includes sensory devices and control devices that collect equipment data, providing real-time data for the system, enabling positioning, tracking, and status monitoring of the physical equipment at automated container terminals.

(2) Transmission layer: The transmission layer serves as the bridge between physical and virtual spaces, connecting the information space with the physical space through the OPC UA communication protocol. A data transmission network is established, using Wi-Fi, wireless routers, and other network communication equipment to transmit the large amount of data generated during the operation of automated container terminals, ensuring the data's timeliness and high fidelity.

(3) Data layer: The data layer is the driving force of the entire system, primarily storing and managing the various types of data information collected from the physical space. It also communicates in real-time with the virtual three-dimensional scene of the container terminal, driving the movement of the corresponding twin models, thus achieving virtual three-dimensional visualization monitoring of terminal operation equipment.

(4) Model layer: The model layer represents the virtual entity of automated container terminals, characterized by precision, standardization, lightweight, and visualization. Through virtual models, it provides an accurate mapping of automated container terminals. This layer serves as the three-dimensional visualization expression channel for the data collected by the data layer, acting as the main receptor of data and a significant component of the front end of the automated container terminal digital twin system.

(5) Service layer: The service layer is the system's external service interface, establishing service modules within the automated container terminal's virtual three-dimensional scene to fulfill the "monitoring" and "control" functions of the equipment monitoring digital twin system. Utilizing digital twin system data, it provides user management services, status monitoring services, scene roaming services, and anomaly alert services, among other functional modules.

3 Construction of Virtual Scenes for Automated Container Terminals

3.1 Multidimensional and Multiscale Analysis of Digital Twin Models

Digital twin theory requires that virtual space scene models comprehensively reflect production reality, possessing multidimensional and multiscale characteristics, as shown in Figure 2. Different colors represent different model types, P_{ij} represents the j-th component of the i-th model, and x, y, z represent the spatial position attributes of each model type. From a multidimensional and multiscale perspective, the construction of the digital twin system scene model for monitoring equipment at automated container terminals is carried out.

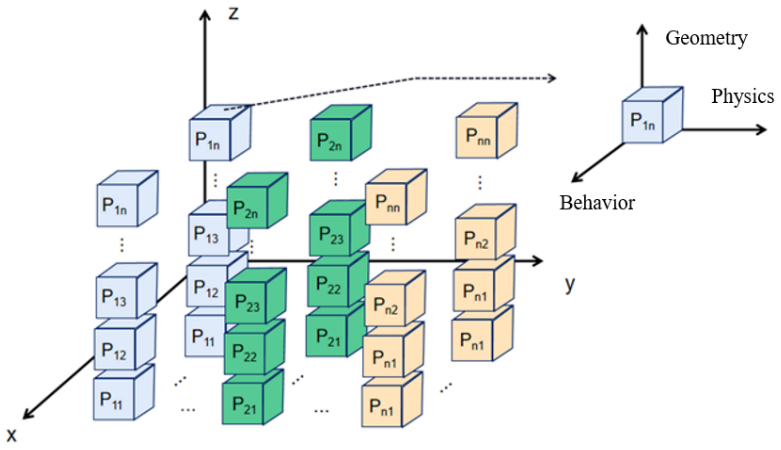


Figure 2. Multidimensionality and multiscale of the digital twin model

Multidimensionality Analysis: For the geometric dimension, virtual models are constructed based on the geometric parameters and features of physical entities [8]. Building on these geometric features, the models are further enhanced by adding physical attributes derived from the physical properties of the entities, allowing for physical dimension modeling of the models. Based on the interrelationships among various components of dock equipment, the behavioral relationships and triggering effects between different parts of the models are designed, leading to the development of a behavioral dimension model.

Multiscale Analysis: The constructed multidimensional models (unit-level models), which include geometry, physics, and behavior, are systematically integrated and assembled [23], and constraints or spatial relationships between unit-level devices are added. Through integration and assembly, the models acquire a systematic nature, forming a comprehensive description of various equipment and environments within the automated container terminal virtual scene, which constitutes the multiscale integration of the model system.

3.2 Construction of Virtual Scenes

The construction process of the virtual scene at automated container terminals is shown in Figure 3. Geometric modeling forms the basis of the virtual scene construction, and after static scene modeling, the features are identified through the digital twin's theory of virtual-real consistency verification to ensure the model possesses digital twin characteristics. Dynamic equipment, as the main monitoring objects of the digital twin system, require the addition of physical properties on top of geometric modeling to simulate their real behavior. The realism of the scene is enhanced by incorporating auxiliary elements such as water surfaces and greenery. Model optimization aims to improve rendering efficiency and conserve system resources, and finally, the multiscale nature of the digital twin model is used for model integration and assembly, forming a three-dimensional visualization interface.

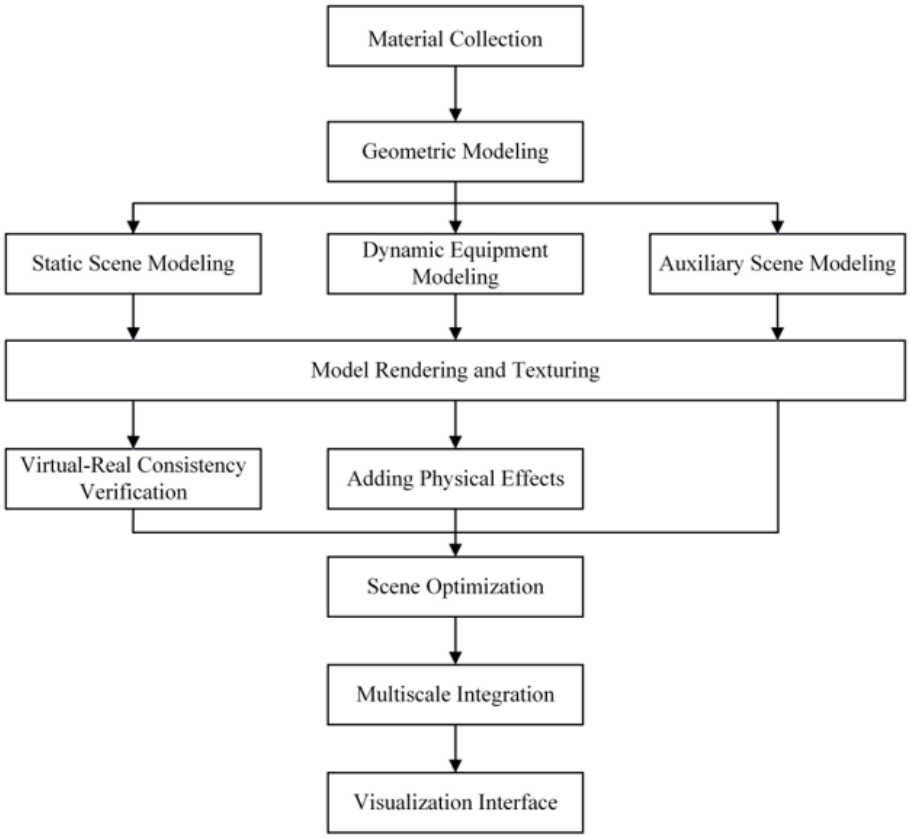


Figure 3. Construction process of virtual scenes

To effectively manage the geometric models of virtual scenes, based on the concept of a scene tree management [24], a root-stem-leaf three-tier structure is used to organize the resource model of automated container terminals, as shown in Figure 4. The main body of the terminal serves as the root level, with static scene models, dynamic equipment models, and auxiliary scene models forming the stem level, and the equipment or sub-scenes that constitute each stem level acting as the leaf level. This structure allows for accurate mapping of the physical space of the terminal in geometric dimensions, covering equipment, environment, and facilities involved in automated container terminal operations.

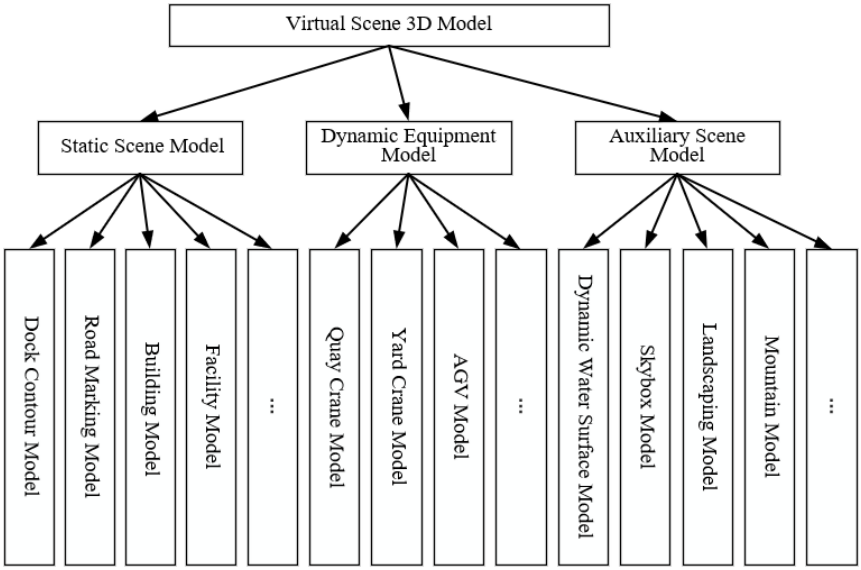


Figure 4. Scene tree management of virtual space models

3.3 Model Virtual-Real Consistency Evaluation

Due to the large scale and complex layout of automated container terminal scenes, the degree of similarity between the virtual space and physical space of the terminal determines whether the digital twin system can accurately reflect the actual production state of the terminal and achieve precise monitoring. Therefore, a virtual-real consistency verification is conducted for the multi-dimensional, multi-scale digital twin model of the terminal [25]. Based on the explicit feature virtual-real consistency verification method proposed by Qian et al. [26], this study focuses on the layout characteristics of the automated container terminal model, conducting a consistency assessment by comparing the positional errors of different functional areas in virtual and actual spaces.

Position data of the monitored areas are taken, for instance, the AGV lanes, berths, and container areas of the double trolley quay crane unloading zones in the physical space of the terminal are taken as verification objects. First, the absolute zero points in virtual and physical spaces are selected respectively, ensuring their alignment. $B_i\left(x_i, y_i\right)$, $L_j\left(x_j, y_j\right)$, and $C_k\left(x_k, y_k\right)$ represent the coordinates of the physical space for the i-th double trolley quay crane unloading zone AGV lane, the j-th berth, and the k-th container area, respectively. BL_{ij} represents the distance between the i-th double trolley quay crane unloading zone AGV lane and the j-th berth in physical space, while BC_{ik} represents the distance between the i-th double trolley quay crane unloading zone AGV lane and the k th container area in physical space. The calculations are as follows:

$$BL_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \quad i \in \left[1, i^*\right], j \in \left[1, j^*\right] \quad i, j \in Z \tag{1}$$

$$BC_{ik} = \sqrt{\left(x_i - x_k\right)^2 + \left(y_i - y_k\right)^2} \quad i \in \left[1, i^*\right], k \in \left[1, k^*\right] \quad i, k \in Z \tag{2}$$

VBL_{ij} represents the distance between the i-th double trolley quay crane unloading zone AGV lane and the j-th berth in the virtual scene, while VBC_{ik} represents the distance between the i-th double trolley quay crane unloading zone AGV lane and the k-th container area in the virtual scene. ER_{ij} is used to denote the positional error of the distance between the double trolley quay crane unloading zone AGV lane and the berth in both the digital twin virtual scene and the physical entity scene, and EOR_{ik} denotes the positional error of the distance between the double trolley quay crane unloading zone AGV lane and the container area in both the digital twin virtual scene and the physical entity scene. i^*, j^*, and k^* respectively represent the maximum numbers of double trolley quay crane unloading zone AGV lanes, berths, and container areas. The verification formulas are as follows:

$$ER_{ij} = \left|BL_{ij} - VBL_{ij}\right| \quad i \in \left[1, i^*\right], j \in \left[1, j^*\right] \quad i, j \in Z \tag{3}$$

$$EOR_{ik} = \left|BC_{ik} - VBC_{ik}\right| \quad i \in \left[1, i^*\right], \quad k \in \left[1, k^*\right] \quad i, k \in Z \tag{4}$$

$$ER_{ij} < \theta_1 \quad EOR_{ik} < \theta_2 \tag{5}$$

θ_1 and θ_2 are controllable thresholds, which can be determined according to actual needs. During layout similarity verification, it is advisable to select the central points of each candidate area for verification to avoid edge points, thus enhancing the validity of the results.

3.4 Optimization of Virtual Scenes

Given the large number and extensive coverage of equipment at automated container terminals, optimizing the virtual terminal scene is essential to ensure smooth display and alleviate the pressure on the digital twin system. This paper employs a discrete LOD rendering method [27, 28] to reduce the number of model faces, lower the computational burden on rendering, and increase the interaction speed of the 3D scene. As shown in Figure 5, during the modeling process, three different levels of detail are created for all dynamic equipment. Models are classified into three levels from detailed to rough: LV0, LV1, and LV2, with LV2 being the simplified model.

During system operation, the precision of models is dynamically switched based on their distance from the camera. Models closer to the camera are displayed using a higher resolution, while those farther away are shown using a lower resolution. By altering the original model's texture maps, models of different levels of detail can be displayed depending on the camera's viewing distance.

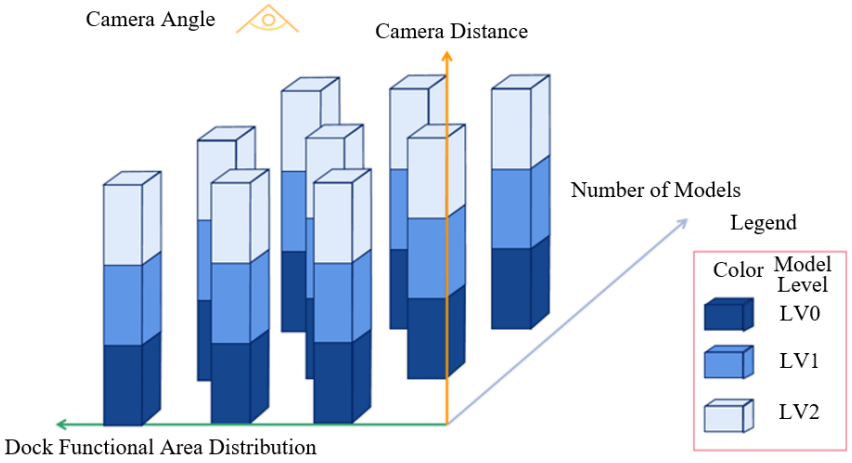


Figure 5. Discrete LOD-based graded model rendering optimization strategy

4 Distributed Study of the System

Traditional digital twin systems have limited research on distributed architecture at the top layer, such as the service layer or display layer, and the user's viewing perspective is relatively limited. Typically, the system can only accommodate one user, which is insufficient for monitoring complex environments like automated container terminals. The greatest advantages of distributed systems are resource sharing and scalability, allowing multiple users to participate in monitoring terminal production equipment simultaneously.

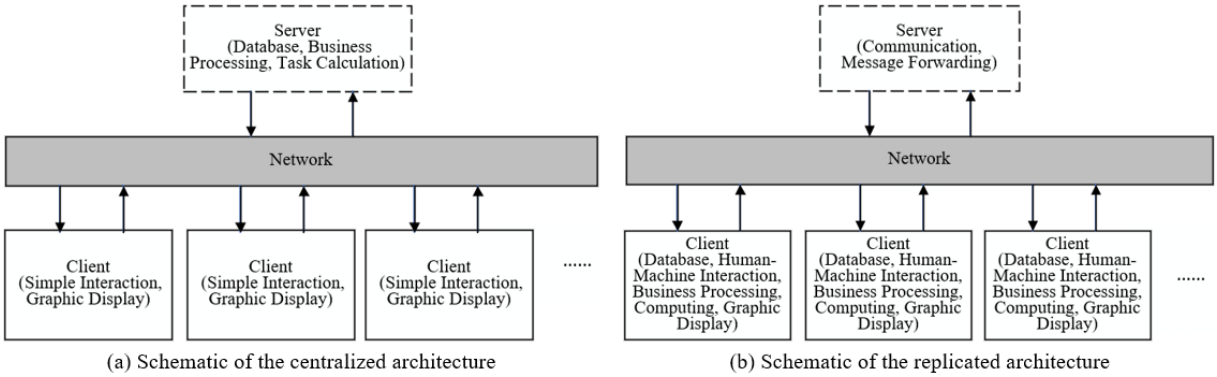


Figure 6. Schematics of centralized and replicated architectures

4.1 Distributed Implementation Plan for the System

Based on state synchronization and frame synchronization, two mainstream network synchronization methods [29], common architectures for distributed systems include centralized architecture and replicated architecture. As shown in subgraph (a) of Figure 6, the centralized architecture places core functions on the server side, requiring real-time dynamic data to be forwarded from the server to the client, which places high demands on the server's computing power and network performance [30]. The structure of the replicated architecture, as shown in subgraph (b) of Figure 6, mainly implements the core logic on the client side. This not only demands higher processing capability from the client but also increases the difficulty of maintaining data synchronization across clients.

Starting from the functional requirements of monitoring equipment at automated container terminals, this paper separates the database from the server and clients, proposing a multi-layer distributed system plan, as shown in Figure 7. Based on the Client/Server (C/S) model, it is divided into four layers: main client, server, sub-client, and database, which communicate and transfer data through network protocols.

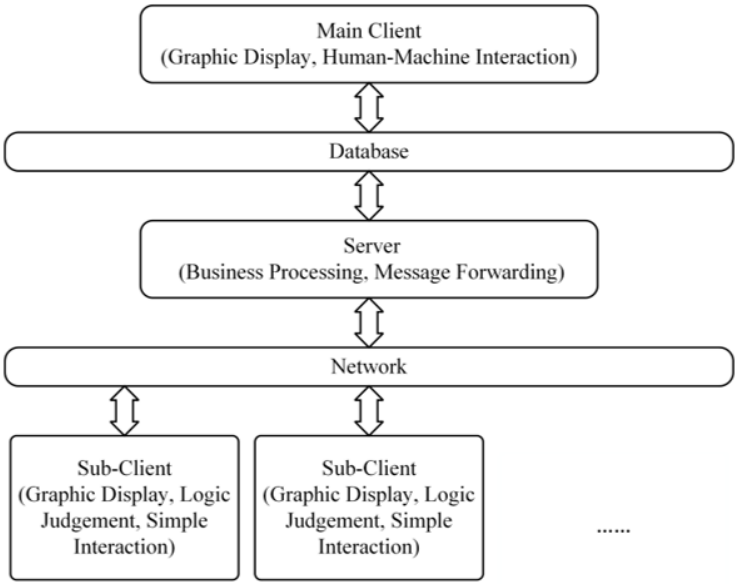


Figure 7. Schematic of the multi-layer distributed system

The main client is assigned two major function categories, including graphic display and human-machine interaction functionalities. These encompass scene interaction, status monitoring, real-time alert, and user roaming functionalities, responsible for user input processing, model behavior display, and data presentation services. The main client communicates directly with the database through business logic and does not go through the server for data reception, hence it does not possess network attributes.

The sub-client is assigned three major function categories, including graphic display, logic judgment, and simple interaction functionalities. This includes real-time alert and user roaming functionalities within the scene interaction feature. When a user performs scene roaming, the sub-client first makes a logic judgment before conveying the user's commands. The sub-client has network attributes. In user roaming functionality, the sub-client can send the user's position and posture information within the scene to the server, while other sub-clients constantly listen for data sent back from the server, receiving and updating the user's position and posture changes in the scene in real-time.

The primary functions of the server include real-time communication with the database and continuous monitoring of device positions and statuses in the database. Additionally, for logic issues executed at the sub-client, such as user joining, user roaming, and scene refreshing, the server only forwards relevant change information and does not perform logic judgment or calculation on changes in user position and posture.

4.2 Synchronization Changes Between the Model Layer and Service Layer

Once the multi-layer distributed system is implemented, some behaviors in the model layer and service layer transition from being local to network behaviors. For changes in the model layer, the behavioral attributes of each dynamic model change, and the movement of dynamic models becomes a network behavior. Therefore, a network identity is assigned to dynamic models. This identity can control the uniqueness of dynamic models on the network. When using this identity, the network system can determine the position of dynamic models at any time, thus enabling the forwarding of dynamic model positions to all sub-clients to ensure that all sub-clients see a consistent data-driven position of dynamic models.

Regarding changes in the service layer, the identity of users in the service layer changes, and the user roaming operation performed by sub-clients becomes a network behavior. Since the user roaming operation of sub-clients is a form of human-machine interaction, this behavior first needs to be generated and logically judged at each sub-client, then communicated to the network, which then transmits the user behavior to the server. The server synchronizes and forwards the user's behavior to all clients, allowing other clients to see that user's behavior.

Considering the large overall scene of the automated container terminal and starting from the perspective of reasonably saving computer system resources, after implementing the system in a distributed manner, only one character entity is established in the system for all sub-clients to invoke. The main way sub-clients provide services is through the real-time display of the equipment production process, which primarily relies on the camera mounted on the same character entity used by all sub-clients for viewing. Therefore, it is necessary to set up a user management script to prevent chaos in the user roaming functionality. Before each user performs a movement operation with the character entity, the management script checks whether the sub-client is a local user and then decides whether to allow the operation.

5 System Prototype Design and Implementation

5.1 System Development Tools

Unity3D facilitates the addition of physical properties to 3D models, such as gravity, rigid body, collision properties, and material properties [31]. Based on a script component mounting mechanism, Unity3D supports dynamic equipment data-driven modes, greatly enhancing the intuitiveness and flexibility of model dynamics. Additionally, Unity3D provides a rich resource library, offering substantial support to developers in implementing various functional modules of the system [32]. The Unity3D user interface is shown in Figure 8. This integrated development environment not only simplifies the 3D model processing workflow but also provides effective tools for highly customizable and dynamic data-driven model presentation, promoting the efficiency of the development process and rapid implementation of system functions.

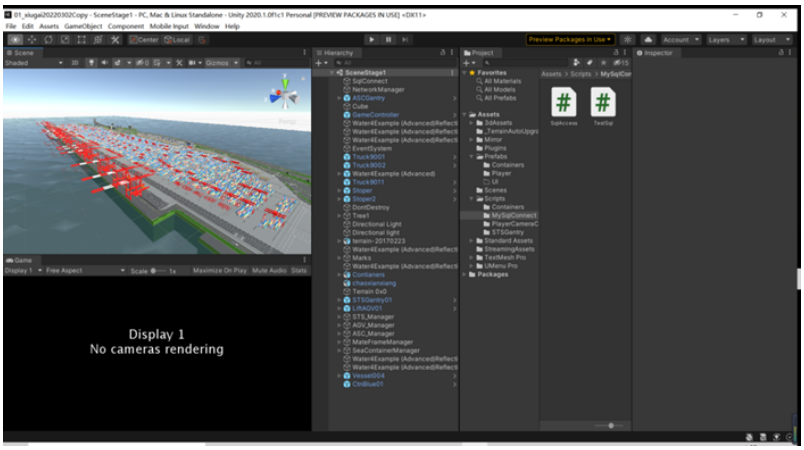


Figure 8. Unity3D user interface

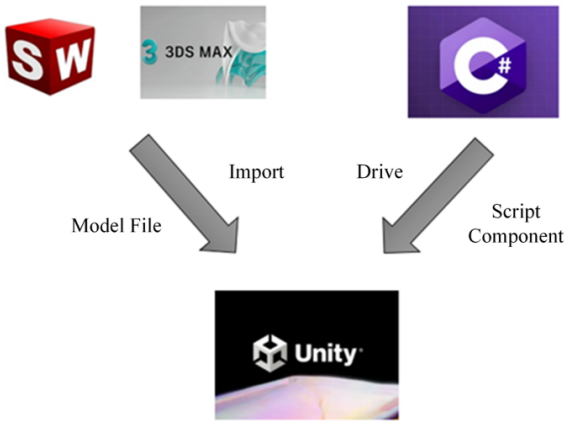


Figure 9. Relationship between Solidworks, 3DS Max, C#, and Unity3D software platform

Therefore, this paper used Unity3D as the development engine, combined with the Mirror framework to achieve system distribution. The modeling platform integrated Solidworks and 3DS Max. Scripting was developed using C#, and MySQL was chosen for managing data at automated container terminals. The relationships among Unity3D, Solidworks, 3DS Max, and C# are illustrated in Figure 9. Models exported in FBX format from Solidworks and 3DS Max were imported into Unity3D, where C# scripts were edited and mounted on the corresponding models to enable model motion.

5.2 Construction of Automated Container Terminal Models

This paper designed a digital twin prototype system for monitoring equipment at an automated container terminal, using the previously designed modeling process to construct the 3D scene. The static scene model and auxiliary scene model are large in scale, hence they are broken down into five parts: contour model, road marking model, building model, facility model, and auxiliary model. Each part was modeled and material effects were added separately, followed by organizational integration. Dynamic models include various types of operational equipment at automated container terminals, such as double trolley quay cranes, Automated Rail-Mounted Gantry Cranes (ARMG), and AGV. Model parent-child relationships were set according to the operational logic of each dynamic device. Using Unity3D's skybox and lighting modules, the main scene was rendered to make its environment closer to physical reality. The overall effect of the automated container terminal model is shown in Figure 10.

Figure 10. Overall modeling effect of automated container terminal

Model layout virtual-real consistency evaluation was conducted on the modeling effect. Initially, the southeastern corner of the terminal's production frontline land area was selected as the absolute zero point for calculating the positions of various functional areas. The automated container terminal's automated stacking yard comprises 61 container zones, primarily composed of ARMGs with and without cantilevers. For simplicity in calculation, the 61 container zones are divided into 11 groups using the single cantilever ARMG as a dividing line.

Taking the center points of 7 quay crane unloading zone AGV lanes as measurement points, and the absolute zero point as a reference, the position coordinates $B_i\,(x_i, y_i)$ of each quay crane unloading zone AGV lane were obtained, $i \in [1, i*], i \in Z$. Taking the center points of 7 container ship berths as measurement points, the center position coordinates $L_j\,(x_j, y_j)$ of each berth were obtained, $j \in [1, j*], j \in Z$. Taking the center of the container zone groups as measurement points, the center position coordinates $C_k\,(x_k, y_k)$ of each container zone group were obtained, $k \in [1, k*], k \in Z$. Where i^*, j^*, and k^* respectively represent the maximum number of double trolley quay crane unloading zone AGV lanes, berths, and container zone groups.

Physical space distances between each quay crane unloading zone AGV lane and berth, and between each quay crane unloading zone AGV lane and the automated stacking yard container zone group, were calculated using Eq. (1) and Eq. (2), and the data are shown in Table 1 and Table 2 (unit: meters).

The calculation of distances in the virtual scene layout was implemented using the Unity3D scripting system. Initially, a vector between two measurement points was created within the constructed 3D scene of the automated container terminal, and the magnitude of the vector was used to calculate the distance between these two points. Using Eq. (3) and Eq. (4), the errors ER_{ij} and EOR_{ik} were calculated, with the results shown in Figure 11. Given that automated container terminals are typical representatives of large-scale production environments [33], this study set the error thresholds θ_1 and θ_2 at 0.1 m to meet the specific needs and accuracy requirements of this scenario. The maximum errors between the physical AGV lane distances at the dock and berth and the virtual AGV lane distances at the dock and berth (ER_{ij}), as well as between the physical AGV lane distances at the dock and the container

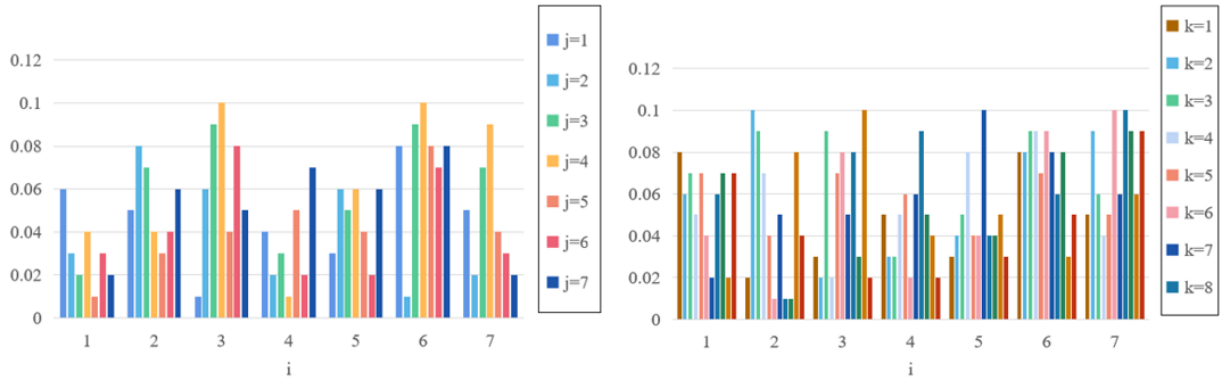
group distances and the virtual AGV lane distances at the dock and container group distances (EOR_{ik}), are all less than 0.1 m. This is below the error threshold, allowing a preliminary assumption that the 3D virtual scene has the characteristics of a digital twin in terms of model layout.

Table 1. Distances between AGV lanes and berths in dock physical space

Quay Crane Unloading Area AGV Lane Number							
Berth Number	1	2	3	4	5	6	7
1	1181.71	1181.34	1180.86	1180.51	1180.10	1179.74	1179.40
2	788.12	787.53	786.91	786.31	785.74	785.20	784.77
3	361.28	359.91	358.70	357.51	356.44	355.37	354.36
4	108.35	103.91	99.59	95.15	90.84	86.52	95.13
5	439.80	438.70	437.71	436.74	435.82	435.03	434.18
6	821.97	821.49	820.80	820.31	819.87	819.41	818.94
7	1207.61	1207.27	1206.84	1206.56	1206.15	1205.82	1205.53

Table 2. Distances between AGV lanes in quay crane unloading areas and automated stacking yard container groups

Quay Crane Unloading Area AGV Lane Number							
Container Group Number	1	2	3	4	5	6	7
1	1592.90	1593.54	1594.15	1594.71	1595.32	1596.00	1596.61
2	1422.04	1423.17	1424.12	1425.29	1426.31	1427.52	1428.69
3	1133.97	1135.33	1136.72	1138.23	1139.64	1141.16	1142.61
4	828.60	830.54	832.57	834.50	836.50	838.51	840.54
5	503.87	506.92	509.93	513.01	516.10	519.37	522.41
6	300.91	305.75	310.57	315.30	320.09	324.85	329.64
7	457.71	460.90	464.17	467.35	470.54	473.82	477.13
8	732.39	734.29	736.27	738.24	740.21	742.26	744.35
9	943.11	944.47	945.72	947.19	948.51	949.80	951.37
10	1131.61	1132.77	1133.84	1134.93	1136.04	1137.28	1138.45
11	1256.90	1257.94	1258.95	1259.94	1260.96	1261.94	1263.01



(a) ER_{ij} calculation results (b) EOR_{ik} calculation results

Figure 11. Results of the virtual-real consistency evaluation for model layout

Finally, the LOD Group component was used to implement the discrete LOD algorithm, optimizing the virtual scene. Various dynamic equipment types such as double trolley quay cranes, ARMG, and AGV were constructed in three different rendering levels. Under a top-down view, rendering tests for the number of faces of Level LV0 and LV2 models of double trolley quay cranes were conducted, as shown in Figure 12. Test results show that the number of faces (Tris attribute) of the LV2 model is more than ten times lower than that of the LV0 model, and the number of vertices (Verts attribute) is reduced nearly tenfold. With the equipment's main form intact, rendering efficiency has been significantly improved.

(a) Number of faces and vertices for LV0 model

(b) Number of faces and vertices for LV2 model

Figure 12. Comparison of rendering faces and vertices between LV0 and LV2 models

5.3 Data-Driven Implementation

The overall architecture of the data-driven mode can be described in three layers: service layer, logic layer, and data layer, as shown in Figure 13. The service layer, as the interface between the system and users, provides the function of displaying the operational status of the port production in a 3D visual format. User commands generated in the main client are passed from the service layer to the logic layer, which returns relevant parameters to the service layer and makes changes to the database while also updating the model's position and state attributes in real time.

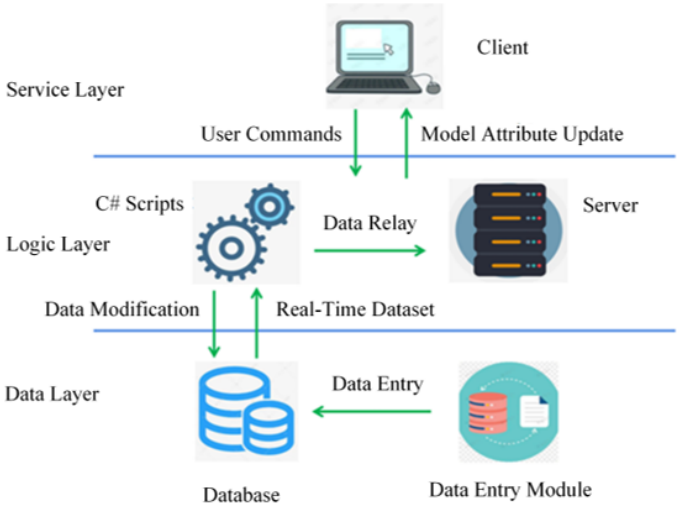


Figure 13. Overall architecture of the data-driven model

The logic layer, implemented through C# backend scripts on the Unity3D platform, serves as a bridge between user actions and model responses. This layer, by using object-oriented dynamic models and script components mounted on their sub-assemblies, transforms operations at the service layer into updates of model states, achieving real-time data processing and dynamic feedback. Additionally, the logic layer is responsible for dynamically reading information from the data layer, driving models based on this data, and monitoring the status of newly added sub-clients to ensure the system's timeliness and accuracy.

The data layer forms the foundation of the system's information storage and management, including the database and data entry module. The database primarily stores the positions and status data of operational equipment, providing essential input for the logic layer. The data entry module is responsible for real-time collection and updating of operational data from port equipment, ensuring data accuracy and timeliness. Through efficient data management, the data layer supports the entire digital twin system's data-driven mechanism, ensuring that the logic and service layers can perform accurate model driving and user feedback based on the latest data.

Based on the concept of a model tree for geometric modeling, the management of equipment position data and status data is controlled using two separate tables for each piece of equipment, with a uniform database table structure design for all equipment. For instance, Table 3 is the structure of the double trolley quay crane equipment position data table, and Table 4 is the equipment status data table structure. Using the same two-table structure, position and status data tables have been established for each AGV and ARMG.

Table 3. Double trolley quay crane equipment position data table structure

Attribute	Definition	Data Type	Remarks
id	No actual meaning	int	Primary key, auto-increment
equip_id	Equipment ID	varchar	Unique, uniformly assigned and managed
equip_name	Equipment name	varchar	Unique, uniformly assigned and managed
time	Update time	varchar	Uniform format: xxxx-xx-xx xx:xx:xx
stsgantry_x	Gantry x-coordinate	float	World coordinate of the gantry relative to the dock layout
trolleysea_y	Seaside trolley y-coordinate	float	Relative coordinate of the seaside trolley relative to the gantry parent
seahoist_z	Seaside trolley hoist z-coordinate	float	Relative coordinate of the seaside trolley hoist relative to the seaside trolley parent
trolleyland_y	Landside trolley y-coordinate	float	Relative coordinate of the landside trolley relative to the gantry parent
landhoist_z	Landside trolley hoist z-coordinate	float	Relative coordinate of the landside trolley hoist relative to the landside trolley parent

Table 4. Equipment status data table structure

Attribute	Definition	Data Type	Remarks
id	No actual meaning	int	Primary key, auto-increment
equip_id	Equipment ID	varchar	Unique, uniformly assigned and managed
equip_name	Equipment name	varchar	Unique, uniformly assigned and managed
time	Update time	varchar	Uniform format: xxxx-xx-xx xx:xx:xx
state	Operating state	int	0, Power on; 1, Standby; 2, Running; 3, Fault; 4, Emergency stop; 5, Power off.
stateinfo	Operating state description	varchar	Specific fault codes must be provided if there is a fault or emergency stop
structureinfo	Structural monitoring state	int	0, Normal; 1, Alarm.
mechanisminfo	Mechanical operation state	int	0, Normal; 1, Alarm.
electricalinfo	Electrical control state	int	0, Normal; 1, Alarm.

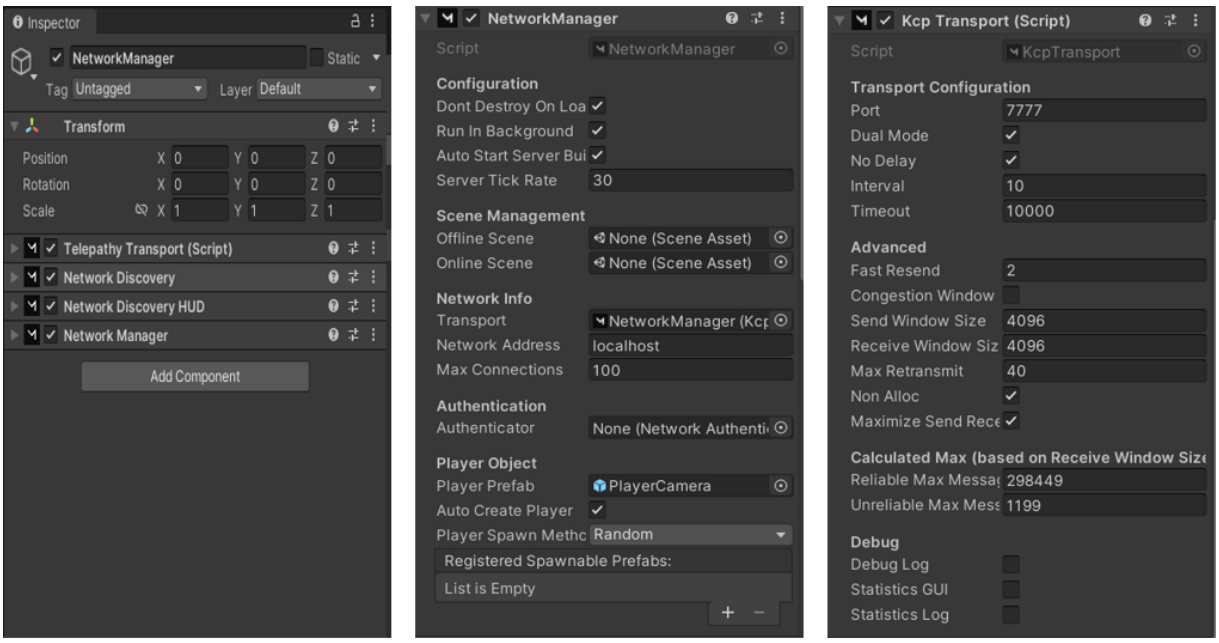
To closely replicate the real working conditions of the transmission layer and data layer in the five-dimensional architecture, the data entry system was designed to regularly enter equipment position data and equipment status data into the database using a two-table structure at set time intervals. Therefore, to facilitate dynamic reading, an adjustable timer was set in the data retrieval stage, allowing dynamic control of the connection between the logic layer and the data layer. This setup enables the logic layer to access data at fixed intervals, and subsequently, data is forwarded to each client through the server.

5.4 Distributed Implementation of the System

In the process of building a distributed system using the Mirror framework and its network components, the first step is to create an empty object named NetworkManager and configure the necessary network framework components as shown in subgraph (a) of Figure 14. The Network Manager component is responsible for coordinating the connection process between clients and the server, synchronizing network scenes and states, and managing the lifecycle of network entities, as depicted in subgraph (b) of Figure 14. To instantiate user character entities in real-time within the network scene, the relevant character entity information is predefined in the Prefab resources, which is further assigned to the Player Prefab property of the NetworkManager. The KCP Transport component implements a network transmission mechanism based on the KCP protocol, ensuring the orderliness and reliability of data, with specific communication parameters set as shown in subgraph (c) of Figure 14. The Network Discovery component uses network broadcasting technology to automatically detect server instances on the same network, thereby simplifying the network session discovery process. The Network Discovery HUD provides a UI interaction interface, allowing users to intuitively select and join available sessions within the network.

To ensure that sub-clients immediately obtain an effective monitoring perspective upon connection, this study

developed a method called CustomNetworkManager by extending the Network Behaviour class. This method, by overriding the OnServerAddPlayer function, specifies the initial position of sub-clients to optimize their access viewpoint, with related pseudocode presented in Table 5.



(a) Network manager object (b) Network manager component (c) KCP transport component

Figure 14. Mirror network framework components

Table 5. Pseudocode for CustomNetworkManager class

```
// Define the CustomNetworkManager method, inheriting from Mirror's NetworkManager class public class
public class CustomNetworkManager : NetworkManager
{
// Declare an array of Vector3 type named startPositions, to store initial positions for clients public Vector3
public Vector3[] startPositions = new Vector3[]
// Override the OnServerAddPlayer method from NetworkManager, executed when a new client is added on
the server
public override void OnServerAddPlayer(NetworkConnectionToClient conn)
{
// Calculate the initial position index for the new client, index is based on the current number of connected
clients
int startPosIndex = numPlayers % startPositions.Length;
// Select the initial position for the new client from the startPositions array using the calculated index
Vector3 startPos = startPositions[startPosIndex];
// Instantiate playerPrefab at the selected initial position
GameObject player = Instantiate(playerPrefab, startPos, Quaternion.identity);
// Add the instantiated player object to the corresponding client connection, completing the network setup for
the player
NetworkServer.AddPlayerForConnection(conn, player);
}
}
```

All dynamic equipment and user character entity scripts have been changed from the MonoBehaviour class to the NetworkBehaviour class, transforming their actions into network behaviors. By using the Network Identity component, each participant and dynamic model is assigned a unique identifier, ensuring their uniqueness in the scene. Simultaneously, the Network Transform component is used to synchronize the positions and statuses of all dynamic equipment and user character entities, ensuring accurate real-time network position synchronization. To allow all sub-clients to see each other and the movement states of dynamic models, network attributes, including

network identity and real-time network position, are assigned to user character entities and all dynamic models. These implementations in the Mirror framework, through network identity and network transform components, ensure interactive and consistent behavior of users and devices on the network.

Taking the double trolley quay crane as an example, its script responsible for grabbing containers has been modified to utilize the Client, Command, ClientRpc, and SyncVar attributes to dictate methods' behaviors in the network. The Client attribute is used to mark methods that should only be executed on the client side. The Command attribute is used to mark a method so it can be called from the client and executed on the server. Methods marked with the ClientRpc attribute are called from the server and executed on all clients. SyncVar is used to mark variables that need to be synchronized across the network. The modified script pseudocode is shown in Table 6.

Table 6. Pseudocode for network behavior and synchronization mechanism of the STSCantry class

```
using Mirror;// Reference to the Mirror networking framework
// Marking the STSCantry class as a network behavior class, inheriting from the NetworkBehaviour class
public class STSCantry: NetworkBehaviour
{
[SyncVar] Declares variables that need to be automatically synchronized between server and client
void Update()//Called every frame to update the system state
{
// Ensure only the local client can perform actions
if (!isLocalPlayer) return;
MoveCrane();
CheckGrabInput();
}
[Client]// Indicates the following methods are executed only on the client, not on the server
MoveCrane(),// Controls the movement of the crane
CheckGrabInput()//Checks and handles user input
[Command]// Indicates the following methods are called on the client but executed on the server
CmdGrabCargo(cargo) // Command to grab a container executed on the server
CmdDropCargo() // Command to drop a container executed on the server
[ClientRpc] // Indicates the following methods are called on the server but executed on all clients
RpcSyncCargo(cargo, isGrabbing) // Synchronizes the cargo grabbing status to all clients
ShowGrabHint(show) // Show or hide grab hint text based on the parameter
}
```

5.5 Implementation of System Functional Modules

(1) Scene Roaming

By defining mouse events and keyboard inputs in the Input class and setting up user behavior logic with if statements, scene roaming can be implemented using Unity3D's Translate method. After establishing server connections and joining multiple sub-clients for testing, each sub-client entity can roam within the scene, observing a clear view of the dock operations without synchronization issues, as shown in Figure 15, depicting four clients simultaneously roaming in the scene.

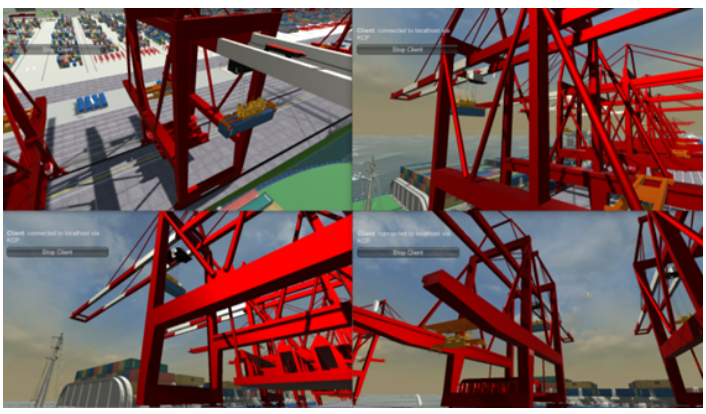


Figure 15. Mirror network framework components

(2) Human-Machine Interaction Functionality

Dynamic equipment is equipped with box-type colliders. When the mouse is placed within the collider range of an equipment model, an interaction event is triggered, displaying the current equipment's information panel. The content of the information panel is composed of the operational status data stored in Table 3 and Table 4. The interaction effects between users and models are shown in Figure 16.

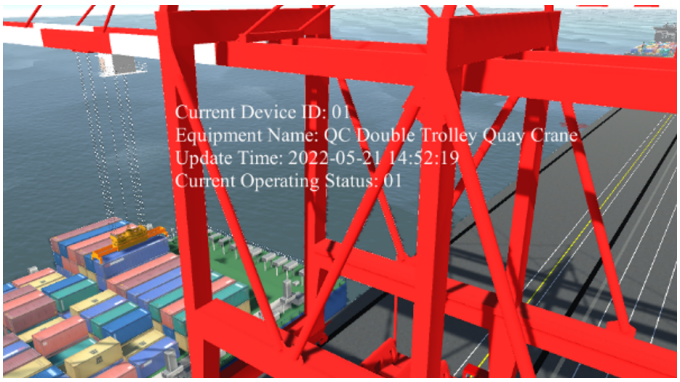


Figure 16. Interaction effects between users and models

(3) Status Monitoring Module

The main client is equipped with a status monitoring module that provides real-time monitoring of the dock environment and equipment, as shown in Figure 17. After entering the system's main interface, users can overview the real-time status of all dock equipment and check for any alarm messages. By clicking the checkbox to close the user interface on the main screen, users can shut down or activate the current user interaction interface.

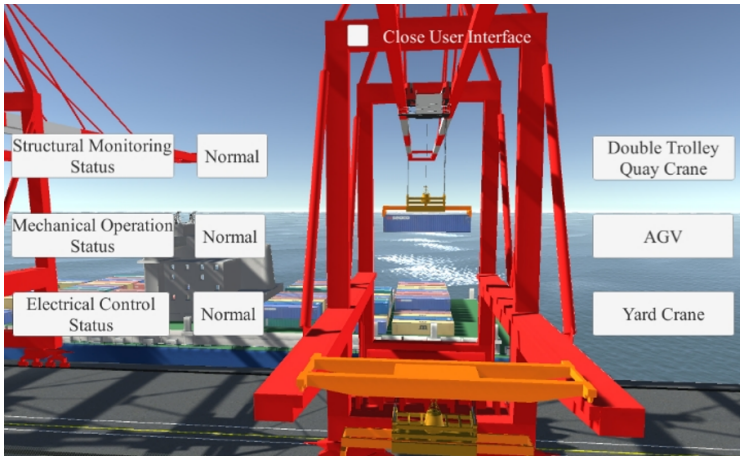


Figure 17. Main interface of the status monitoring module

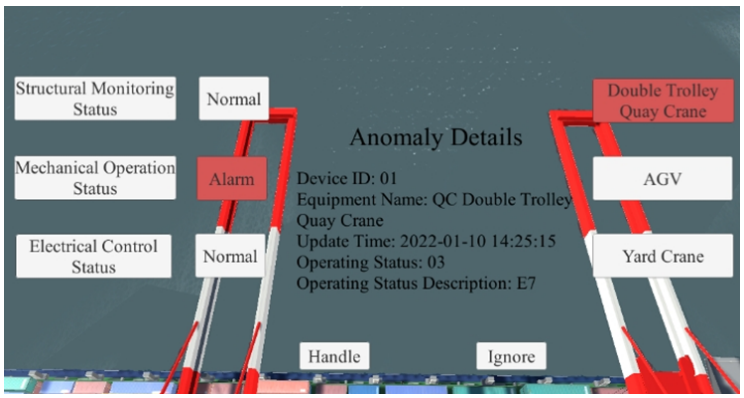
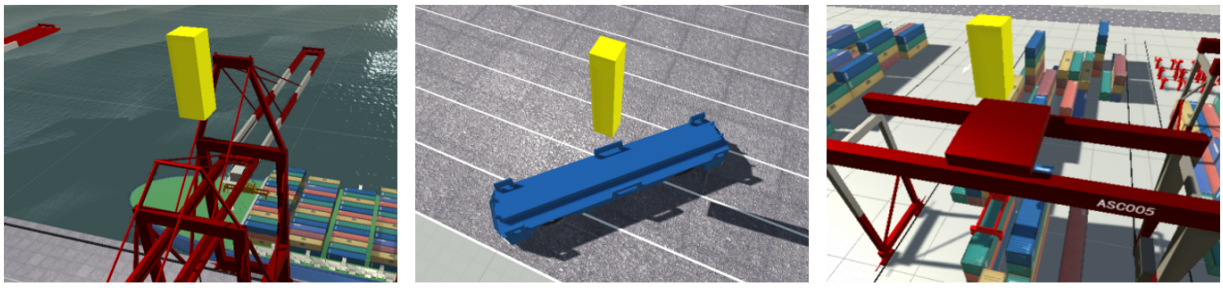


Figure 18. Alarm details interface

(4) Real-Time Alarm Module

When the system database records an alarm for a piece of equipment, the corresponding equipment category button on the right side of the status monitoring interface turns red to signal an alarm, and the specific status type on the left side also turns red, changing the status display from “Normal” to “Alarm”. The alarm details for the equipment experiencing the alarm are displayed in the center of the main interface, as shown in Figure 18, where a double trolley quay crane mechanism has triggered an alarm.

Simultaneously, the main client and all sub-clients will see a distinct yellow alarm marker on the alarmed equipment, as shown in Figure 19. The height of the yellow alarm marker varies depending on the equipment’s height. Once the main client resolves the issue, the yellow marker disappears.



(a) Alarm marker on the double trolley quay crane (b) Alarm marker on the AGV (c) Alarm marker on the ARMG

Figure 19. Yellow marker effects on alarmed equipment

6 Conclusion

To achieve multi-perspective real-time monitoring and management of operational equipment at automated container terminals, ensuring the safe operation of automated container terminals, this paper started from the overall architecture of a digital twin system for monitoring automated container terminal operational equipment. It detailed the content of each layer and performs a multidimensional and multiscale analysis of the digital twin model, building high-fidelity virtual models. Addressing the large volume of virtual scene models at automated container terminals, a unified management approach for various models based on the concept of scene tree geometric modeling was used. From the perspective of layout consistency between virtual and physical spaces, an evaluation method for virtual-real consistency was proposed, and combined with discrete LOD technology, virtual models were rendered and optimized. Leveraging the advantages of ”centralized” and ”replicated” architectures, along with the functional requirements for dock operational equipment monitoring, a multi-layer distributed system solution was proposed. Finally, using an automated container terminal as an example, a digital twin system prototype was designed and developed, and testing confirms that the system’s functions operate normally, achieving multi-perspective digital monitoring of dock operational equipment and the site. This provides a case study for the application of digital twin technology at automated container terminals, significantly contributing to the enhancement of digital intelligence levels at automated container terminals.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] C. Mi, Y. Huang, C. Fu, Z. Zhang, and O. Postolache, “Vision-based measurement: Actualities and developing trends in automated container terminals,” *IEEE Instrum. Meas. Mag.*, vol. 24, no. 4, pp. 65–76, 2021. https://doi.org/10.1109/MIM.2021.9448257

[2] X. Xiang and C. Liu, “Modeling and analysis for an automated container terminal considering battery management,” *Comput. Ind. Eng.*, vol. 156, p. 107258, 2021. https://doi.org/10.1016/j.cie.2021.107258

[3] N. Wang, D. Chang, X. Shi, J. Yuan, and Y. Gao, “Analysis and design of typical automated container terminals layout considering carbon emissions,” *Sustainability*, vol. 11, no. 10, p. 2957, 2019. https://doi.org/10.3390/su11102957

[4] Y. Ding, Z. Zhang, K. Chen, H. Ding, S. Voss, L. Heilig, Y. Chen, and X. Chen, “Real-time monitoring and optimal resource allocation for automated container terminals: A digital twin application at the Yangshan Port,” *J. Adv. Transp.*, vol. 2023, p. 6909801, 2023. https://doi.org/10.1155/2023/6909801

[5] R. Bozzo, A. Derito, R. Nurchi, and N. Ackroyd, "MOCONT: A new system for container terminal monitoring and control," in *2001 IEEE Intelligent Transportation Systems, Proceedings*, Oakland, CA, USA, 2001, pp. 1090–1094. https://doi.org/10.1109/ITSC.2001.948814

[6] U. Lee, K. Kang, G. Kim, and H. Cho, "Improving tower crane productivity using wireless technology," *Comput. Aided Civ. Infrastruct. Eng.*, vol. 21, no. 8, pp. 594–604, 2006. https://doi.org/10.1111/j.1467-8667.2006.00459.x

[7] Z. Yang, J. Yang, and E. Yuan, "Safety monitoring of construction equipment based on multi-sensor technology," in *Proceedings of the International Symposium on Automation and Robotics in Construction*, Kitakyushu, Japan, 2020, pp. 677–684. https://doi.org/10.22260/ISARC2020/0095

[8] J. Y. Kim, "A TCP/IP-based remote control system for yard cranes in a port container terminal," *Robotica*, vol. 24, no. 5, pp. 613–620, 2006. https://doi.org/10.1017/S0263574706002694

[9] H. Awad, M. S. Saraya, M. S. M. Elksasy, and A. M. T. Ali-Eldin, "IoT based framework for monitoring and controlling movable harbor cranes," in *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, Cairo, Egypt, 2021, pp. 42–47. https://doi.org/10.1109/MIUCC52538.2021.9447596

[10] Y. Li and C. Liu, "Integrating field data and 3D simulation for tower crane activity monitoring and alarming," *Autom. Constr.*, vol. 27, pp. 111–119, 2012. https://doi.org/10.1016/j.autcon.2012.05.003

[11] F. Tao, B. Xiao, Q. Qi, J. Cheng, and P. Ji, "Digital twin modeling," *J. Manuf. Syst.*, vol. 64, pp. 372–389, 2022. https://doi.org/10.1016/j.jmsy.2022.06.015

[12] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, pp. 85–113, 2017. https://doi.org/10.1007/978-3-319-38756-7_4

[13] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," *White Paper*, vol. 1, pp. 1–7, 2014.

[14] J. Martínez-Gutiérrez, J. Díez-González, R. Ferrero-Guillén, P. Verde, R. Álvarez, and H. Perez, "Digital twin for automatic transportation in industry 4.0," *Sensors*, vol. 21, no. 10, p. 3344, 2021. https://doi.org/10.3390/s21103344

[15] W. Hofmann and F. Branding, "Implementation of an IoT-and cloud-based digital twin for real-time decision support in port operations," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 2104–2109, 2019. https://doi.org/10.1016/j.ifacol.2019.11.516

[16] W. Yang, X. Bao, Y. Zheng, L. Zhang, Z. Zhang, Z. Zhang, and L. Lin, "A digital twin framework for large comprehensive ports and a case study of Qingdao Port," *Int. J. Adv. Manuf. Technol.*, vol. 131, pp. 5571–5588, 2022. https://doi.org/10.1007/s00170-022-10625-1

[17] J. Szpytko and Y. S. Duarte, "Digital twins model for cranes operating in container terminal," *IFAC-PapersOnLine*, vol. 52, no. 10, pp. 25–30, 2019. https://doi.org/10.1016/j.ifacol.2019.10.014

[18] S. Agostinelli, F. Cumo, M. Nezhad, G. Orsini, and G. Piras, "Renewable energy system controlled by open-source tools and digital twin model: Zero energy port area in Italy," *Energies*, vol. 15, no. 5, p. 1817, 2022. https://doi.org/10.3390/en15051817

[19] Y. Zhou, Z. Fu, J. Zhang, W. Li, and C. Gao, "A digital twin-based operation status monitoring system for port cranes," *Sensors*, vol. 22, no. 9, p. 3216, 2022. https://doi.org/10.3390/s22093216

[20] F. Tao, Y. Cheng, J. Cheng, M. Zhang, W. Xu, and Q. Qi, "Theories and technologies for cyber-physical fusion in digital twin shop-floor," *Comput. Integr. Manuf. Syst.*, vol. 23, no. 8, pp. 1603–1611, 2017. http://doi.org/10.13196/j.cims.2017.08.001

[21] E. Tuegel, "The airframe digital twin: Some challenges to realization," in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Honolulu, Hawaii, 2012, p. 1812. https://doi.org/10.2514/6.2012-1812

[22] O. Topçu and H. Oğuztüzün, "Layered simulation architecture: A practical approach," *Simul. Model. Pract. Theory*, vol. 32, pp. 1–14, 2013. https://doi.org/10.1016/j.simpat.2012.11.001

[23] K. Ding, X. Zhang, G. Zhou, C. Wang, H. Yang, F. Zhang, and X. Cao, "Digital twin-based multi-dimensional and multi-scale modeling of smart manufacturing spaces," *Comput. Integr. Manuf. Syst.*, vol. 25, no. 6, pp. 1491–1504, 2019.

[24] Y. Cong, M. Y. Yang, and B. Rosenhahn, "RelTr: Relation transformer for scene graph generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 9, pp. 11 169–11 183, 2023. https://doi.org/10.1109/TPAMI.2023.3268066

[25] F. Tao, H. Zhang, Q. Qi *et al.*, "Theory of digital twin modeling and its application," *Comput. Integr. Manuf. Syst.*, vol. 27, no. 1, pp. 1–15, 2021. https://doi.org/10.13196/j.cims.2021.01.001

[26] W. Qian, Y. Guo, K. Cui, P. Wu, W. Fang, and D. Liu, "Multidimensional data modeling and model validation

for digital twin workshop," *J. Comput. Inf. Sci. Eng.*, vol. 21, no. 3, p. 031005, 2021. https://doi.org/10.1115/1.4049634

[27] C. Li and F. Kang, "Intelligent combination of discrete LOD model for 3D visualization based on visual perception and information entropy fusion," *J. Syst. Simul.*, vol. 27, no. 8, pp. 1815–1823, 2015.

[28] L. Tang, Y. Shen, L. Lin, F. Biljecki, H. Zhu, Y. Zhu, F. Yang, and F. Su, "An application-driven LOD modeling paradigm for 3D building models," *ISPRS J. Photogramm. Remote Sens.*, vol. 161, pp. 194–207, 2020. https://doi.org/10.1016/j.isprsjprs.2020.01.019

[29] K. Hengster-Movric, F. L. Lewis, and M. Sebek, "Distributed static output-feedback control for state synchronization in networks of identical LTI systems," *Automatica*, vol. 53, pp. 282–290, 2015. https://doi.org/10.1016/j.automatica.2014.12.015

[30] S. Chen, A. Ng, and P. Greenfield, "A performance evaluation of distributed database architectures," *Conc. Comput. Pract. Exp.*, vol. 25, no. 11, pp. 1524–1546, 2013. https://doi.org/10.1002/cpe.2891

[31] X. Zhu, "Behavior tree design of intelligent behavior of non-player character (NPC) based on Unity3D," *J. Intell. Fuzzy Syst.*, vol. 37, no. 5, pp. 6071–6079, 2019. https://doi.org/10.3233/JIFS-179190

[32] K. Cheliotis, "ABMU: An agent-based modelling framework for Unity3D," *SoftwareX*, vol. 15, p. 100771, 2021. https://doi.org/10.1016/j.softx.2021.100771

[33] W. Kon, N. S. F. Abdul Rahman, R. Md Hanafiah, and S. Abdul Hamid, "The global trends of automated container terminal: A systematic literature review," *Marit. Bus. Rev.*, vol. 6, no. 3, pp. 206–233, 2021. https://doi.org/10.1108/MABR-03-2020-0016