DigiMobot: Digital Twin for Human-Robot Collaboration in Indoor Environments

Yuto Fukushima[1], Yusuke Asai[1], Shunsuke Aoki[2], Takuro Yonezawa[1], and Nobuo Kawaguchi[1]

***Abstract*— Human-robot collaboration and cooperation are critical for Autonomous Mobile Robots (AMRs) in order to use them in indoor environments, such as offices, hospitals, libraries, schools, factories, and warehouses. Since a long transition period might be required to fully automate such facilities, we have to deploy AMRs while improving safety in the mixed environments of human and mobile robots. In addition, human behaviors in such environments might be difficult to predict. In this paper, we present a Digital Twin for Autonomous Mobile Robots system named *DigiMobot* to support, manage, monitor, and validate AMRs in indoor environments. First, DigiMobot can simulate human behaviors and robot movements to verify and validate AMRs to improve safety in a virtual world. Secondly, DigiMobot can monitor and manage AMRs in the physical world by collecting sensor data from each robot in real-time. Since DigiMobot enables us to test the robot systems in the virtual world, we can deploy and implement AMRs in each facility without any modifications. To show the feasibility of DigiMobot, we develop a software framework and two different types of autonomous mobile robots. Finally, we conduct real-world experiments in a warehouse located in Saitama, Japan, in which more than $400,000$ items are stored for commercial purposes.**

I. INTRODUCTION

Autonomous Mobile Robots (AMRs) have attracted many researchers and practitioners [1] to improve our productivity and quality of life. In particular, AMRs are highly expected to be deployed and used in indoor environments, such as offices, hospitals, libraries, schools, factories, and warehouses. Such autonomous robots support human workers and/or users to collect, carry, and/or deliver items. However, such facilities are originally designed for human workers and/or users, and they are not for mobile robots. Unlike autonomous vehicles on public roads [2], mobile robots can lead to deadlocks almost anywhere in indoor environments, because the road structures might be more complex and narrow. In addition, to make these facilities fully automated, a long transition period and huge economic investment might be required [3]. Hence, we have to account for safe human-robot collaboration and cooperation [4], in order to use autonomous robot systems for mixed environments where humans and robots cooperate and collaborate safely and effectively.

In addition, human behaviors and movements [5] might be difficult to predict in these facilities. To safely use AMRs with human workers, Digital twins [6] might be beneficial to validate and verify such robot systems. Digital twin is used for reproducing virtual counterpart in a cyber world for validation and/or verification, and it also supports the applications in a physical world for deployment [7]. By keeping the consistency between the cyber world and the physical world, we can easily transplant the developed systems and/or protocols from the cyber world to the physical world.

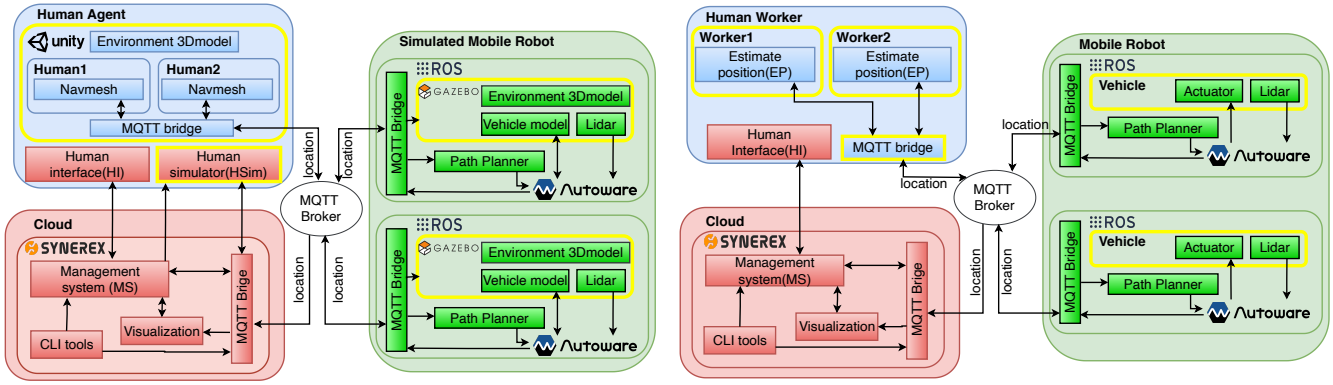
In this paper, we present a Digital Twin for Autonomous Mobile Robots system named *DigiMobot* to support, manage, monitor, and validate AMRs in indoor environments. DigiMobot is capable of simulating both human behaviors and robot movements to verify and validate AMRs to improve safety. In addition, DigiMobot can monitor and manage AMRs in a physical world by collecting sensor data from each robot in real time. Hence, DigiMobot helps us to understand the overall status of the systems. In addition, since DigiMobot enables us to test the robot systems in the virtual world, we can deploy and implement AMRs in each facility without any modifications. By using DigiMobot, multiple AMRs can collaborate and cooperate with multiple human workers in indoor environments.

To show the feasibility of DigiMobot, we develop a software framework and two different types of autonomous mobile robots. As shown in Figure 1, the software framework consists of three components: human, robot, and cloud server. For simulations in a virtual world, DigiMobot generates and uses human agents and simulated robots. On the other hand, for real experiments, DigiMobot just supports human workers and robots by collecting the sensor data and providing commands. In both cases, DigiMobot relies on the common components for cloud servers and robot interfaces, and therefore, we can easily deploy and implement robot systems in the real world by using DigiMobot. Since the communication resources and computational resources for mobile robots might be limited, we use MQ Telemetry Transport (MQTT) [8] for the communications. In addition, we modify two types of the existing mobile robots: *Thouzer* and *Logiee*. They have different physical features, such as sizes, weights, and kinematic models, but DigiMobot accommodates such different types of robots.

Overall, the contributions of this paper are described as follows:

- We present a digital twin system and architecture to support, manage, monitor, and validate autonomous mobile robots.
- We develop software and hardware frameworks for safe human-robot collaboration in indoor environments.
- We show the feasibility of our system by conducting

[1] Department of Information and Communication Engineering, Graduate School of Engineering, Nagoya University, Japan. fukurin, asayu, kawaguchi@ucl.nuee.nagoya-u.ac.jp, takuro@nagoya-u.jp [2] Electrical and Computer Engineering, Carnegie Mellon University shunsuka@andrew.cmu.edu



(a) Architecture for simulation.

(b) Architecture for real deployment.

Fig. 1: Software architecture for DigiMobot.

real-world experiments in a warehouse that is commercially used in the society.

The remainder of this paper is organized as follows. Section II describes our software framework and the concept of Digital twin. Section III presents the designs and concepts for our DigiMobot. Section IV presents the software and hardware implementation. Section V shows the results and evaluations for real experiments. Also, section VI reviews the related studies. Finally, section VII presents the conclusion and future work.

II. Assumption

We first we present the basic concept for Digital Twin. In addition, we describe Synerex [9] framework that is used for DigiMobot.

A. Digital Twin

Digital twin concept is to reproduce a software counterpart of a physical object [7], [10]. The digital twins are also expected to improve the feasibility by extending the simulation capacity [11], [12]. We simulate human behaviors and robot movements in the virtual world to test the robot systems. This simulation reproduces the physical objects as digital twins. It enables us to develop the safe robot systems and deploy in the real world without any modifications. This approach is a kind of Experimentable Digital Twins (EDTs), which is a combination of Virtual Testbeds and Digital Twins or eRobotics and Industry 4.0 and first introduced by Schluse et al. [6]. Therefore, our proposed architecture has two cases: simulation and real experiments as shown in Figure 1. In both two architectures, the system is designed to be controlled by the same processes.

B. Synerex

We use an open-source framework called *Synerex* to enable the internal communications of DigiMobot. Synerex has been originally designed for balancing the supply and demand for multiple players, and it provides various functions, including communications, visualizations, and data managements. Synerex mainly consists of three types of components: (i) a synerex server, (ii) a node server, and (iii) synerex providers. The synerex server enables the data exchanges, and the node server manages all nodes in the system. Also, each synerex provider enable each service. For communication protocol, Synerex allows channel-based communications where each node can exchange the data with nodes in the same channel. The protocol is based on a modern Remote Procedure Call framework called gRPC. As we discuss in Section III, our system extends this Synerex framework for autonomous mobile robots.

III. DigiMobot: Architecture Design

In this section, we present the basic concepts and architecture design for DigiMobot. DigiMobot is a digital twin system for autonomous mobile robots, and this includes two parts: (i) components for virtual environment and (ii) components for real deployment.

A. Concepts

The objective of DigiMobot is to support, manage, monitor, and validate Autonomous Mobile Robots (AMRs) to collaborate and cooperate with human workers. To achieve this objective, DigiMobot includes two main features: (i) realistic simulation for test and validation and (ii) monitoring and management for real-world deployment. First, DigiMobot is capable of reproducing the real-world situations in the

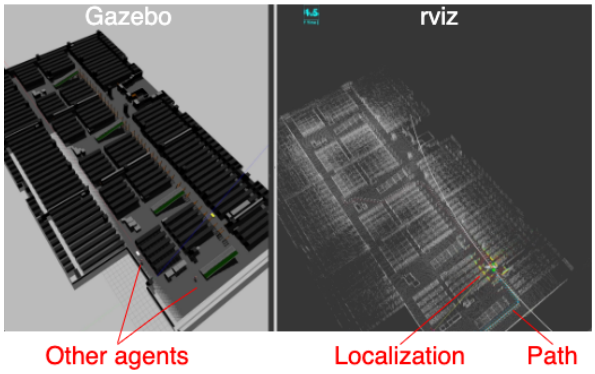


Fig. 2: Simulation on DigiMobot for a single mobile robot. Left one is on Gazebo; Right one is on ROS rviz.

TABLE I: Messages for DigiMobot.

Message	Topic (in MQTT)	Contents	From	To	Protocol
Robot position	pos/robot/[id]	location, orientation	Robot	All	MQTT
Robot destination	cmd/robot/[id]/go	destination	MS	Robot	MQTT
Worker position	pos/human/[id]	location, orientation	Unity, EP	All	MQTT
Worker destination	cmd/human/[id]/go	destination	HSim	Unity	MQTT
Initialization	cmd/sim/set_state	initial workers/robots position	CLI tools	MS, Unity	MQTT
Picking order	-	picking order list	CLI tools, MS	MS, HSim	DT (on Synerex)
Agents Status	-	workers/robots status	MS	HI, Hsim	DT (on Synerex)
Intervention	-	robotID	HI, CLI tools	MS	DT (on Synerex)
Picking	-	workerID, shelfID, batchID	Web client	HI	WebSocket
Picking	-	workerID, shelfID, batchID	HI	MS	DT (on Synerex)
Handing	-	workerID, robotID, batchID	Web client	HI	WebSocket
Handing	-	workerID, robotID, batchID	HI	MS	DT (on Synerex)

virtual world to test and validate the system configurations and cooperative protocols. Secondly, DigiMobot can keep collecting the sensor data from each AMR, and it is also able to send the commands for the operation of each AMR. These features allow us to deploy the robot systems while saving costs.

In addition, we design and develop DigiMobot to support human workers in indoor environments, such as offices, hospitals, libraries, schools, factories, and warehouses. Hence, in DigiMobot, each AMR collects, carries and delivers items from a pickup position to their destination. The human workers select the items to hand over to the mobile robots at the pickup position. Also, at the destination, the human workers collect the items from the mobile robots. Overall, DigiMobot frees human workers from heavy workloads, including carrying and delivering tasks.

B. Overall Architecture

We present the software architecture for DigiMobot in Figure 1. Since DigiMobot is a digital twin system, we show the components for the virtual world in Figure 1a, and we also show the components for the physical world in Figure 1b. As shown in Figure 1a, DigiMobot for simulation includes three components: (i) Human agent, (ii) Simulated robot, and (iii) Cloud. To test and validate the mobile robot systems before the real deployment, these components simulate human behaviors and robot movements. We discuss the procedures more in Section III-C. Also, as shown in Figure 1b, DigiMobot for real deployment has three components: (i) Human worker, (ii) robot, and (iii) Cloud. To deploy the tested and simulated robot systems in the real world, the architecture for real deployment is very similar to that for simulation. We present the architecture more in Section III-D.

One of the common components for these two software is cloud, as shown in Figures 1a and 1b. In DigiMobot, we use Synerex to manage and monitor multiple mobile robots and human workers, as we covered in Section II-B. Since Synerex uses channel-based communications, we build two channels for DigiMobot: MQTT channel and Digital Twin (DT) channel. The MQTT channel is used for reliable interface of MQTT to communicate with other components. On the other hand, the Digital Twin channel is used for exchanging the information, such like the picking orders, picking messages, and agent status, as shown in Table I. The cloud has six components: (i) CLI tools, (ii) Management System (MS), (iii) Visualization, and (iv) MQTT bridge, (v) Human Interface (HI), and (vi) Human Simulator (HSim).

First, CLI tools is for sending the initialization settings and predefined picking orders by commands. Secondly, MS is for managing the agents status and allocating tasks. Visualization provider is to visualize the agents movements as shown in Figure 3. The graphical user interface in Figure 3 can control the time such as pause and fast-forward. Human Interface (HI) provider is a Web server and works as user interfaces for humans. It also enables to support and monitor the robot systems. Figure 4 shows an example of the Human Interface. Instructions such as shelf locations and assigned AMR and buttons to send picking or handing message for a specific worker are shown in Figure 4a. Besides, Figure 4b shows the overall status such as picking states and current locations. This monitoring window also enables an administrator to send intervention command which shift robots status managed in the MS. Human Simulator (HSim) provider is for simulating human agents by communicating with the Unity application. This distributed cloud system is flexible to unpredictable disturbs in the real world, which makes it difficult to deploy human-robot cooperative system. For example, we can correct a status gap between the physical and virtual world by sending an intervention message with human interface.

Table I shows the messages and the picking orders examples in DigiMobot. The communication protocol uses MQTT which is a publish-subscribe network protocol and high affinity with Robot Operating System (ROS). The locations of robots and humans are shared at 10 Hz and the destination is specified in MQTT.

C. Components for Virtual Environment

We use an open-source simulation software Gazebo [13] for simulating robot systems. One of the goals of Gazebo is to simulate robotics systems on ROS in the same process as in the real robots [13]. Therefore, we just describe the mechanical model of the robot, then we can simulate the robots processes. As mentioned above, the robot simulation is performed on each AMR. It means multiple robots simulation with high loads can be used by distributing computer resources respectively.

Concerning simulating human agents, we use Unity as the platform to simulate human agents. Unity is a game engine developed by Unity Technologies. We adapt Unity because it is a general-purpose 3D game engine, and it might be more suitable for the physical simulation of human agents. In addition, NavMesh provided by Unity can make an object move to the set destination through the shortest path on a predefined walking area, so suitable to represent the picking work to move to specific locations. The workers behaviors include picking, handing the items to the AMR, and understanding the instructions by using human interfaces. Since these behaviors are managed in the cloud, the AMR does not need to grasp such status. Hence, we design the status of human agents to be simulated in the human simulator on the cloud server, and only the movements to be simulated in Unity.

D. Components for Real Deployment

As shown in Figure 1b, DigiMobot accounts for two components: human workers and mobile robots. First, to share the information between the cloud and the human workers, DigiMobot has a Human Interface (HI). The human interface provides safety and non-safety information through Graphical User Interfaces. In addition, DigiMobot includes Estimate Position (EP) modules. By using the estimate position modules, the cloud can estimate the 3-D positions of human workers in the facilities. As shown in Table I, the human interface uses Web Socket for the communications, and the estimate position modules use MQTT. By enabling the communications with the cloud, DigiMobot can support the human workers to guarantee their safety.

Secondly, we assume that each mobile robot has a computer, sensors, and network interfaces for wireless communications. Each robot has the MQTT bridge to communicate with the human workers and the cloud server to guarantee safety. To safely operate these mobile robots, the management system in the cloud server can send the commands for each robot. Also, we can halt the operation of mobile robots locally, for safety purposes.

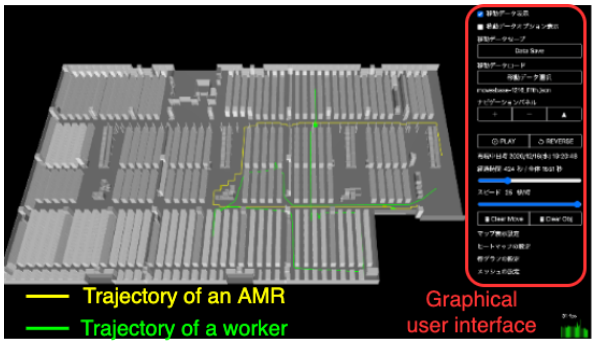


Fig. 3: Visualization for movements of an AMR and a human worker.

IV. Implementation

In this section, we present the software and hardware implementation of DigiMobot. We first present the implementation for Autonomous Mobile Robots (AMRs) and its management, and we then describe the implementation for human agents in simulations and for supporting human workers. In addition, we present the detailed information for the management system in DigiMobot.

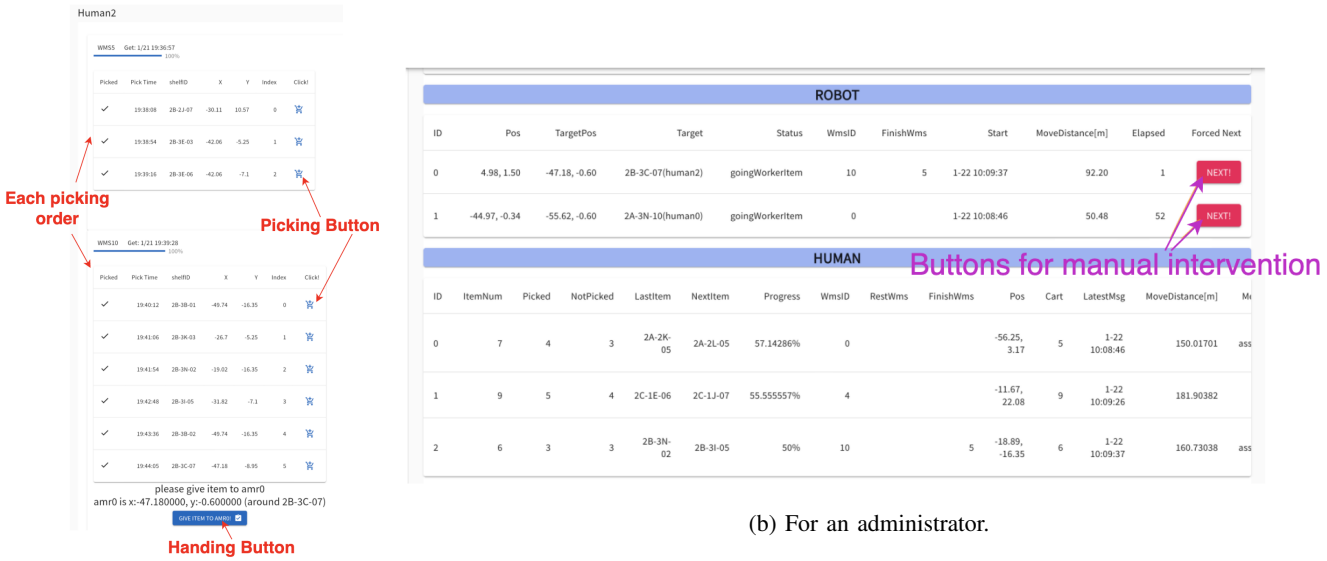
A. Autonomous Mobile Robots (AMRs)

In DigiMobot, we assume that each AMR is installed with some LiDARs, a computer, and a wireless network adaptor to communicate wirelessly. We use ROS [14] as middleware and the processes on AMRs are independent except for receiving the destination, so additional sensors such as cameras are possible. Since each AMR knows the position/orientation of the other agents(robots and humans), it allows for separate simulation on each AMR. Path planning can also perform by taking into account the locations of the other robots. Figure 2 shows the other agents seen by an AMR in the simulation.

At first, we present the AMR system and processing of automated driving. Figure 5 shows the overview of controlling processes. The main algorithms are implemented by using Autoware [15]. Autoware is an open-source software for autonomous driving that works on ROS. The processes of autonomous driving consist of localization, path planning, and path following. AMRs will stop when they detect an obstacle in front.

First, as preparations for autonomous driving, we manually drive through the entire environment, obtain rosbag data including point cloud, and then perform Normal Distribution Transform-matching-mapping (NDT-mapping) algorithm to obtain a point cloud map of the environment. Then, an initial position is input manually for localization by NDT-matching. After the preparations, the AMR starts autonomous driving. Voxel_grid_filter downsamples the point cloud data obtained from LiDAR. Then, NDT-matching estimates the position by referring to the downsampled point cloud data and the point cloud map, and vel_pose_connect estimates the pose. When each AMR receives a destination from the cloud server, it plans a path to the destination and performs path following.

In general, when multiple autonomous vehicles run automatically in the same environment, it is necessary to define a protocol to avoid each other without stopping when they face each other. We try to solve this problem by avoiding facing each other and developed a path planning method adapted to this experimental environment. The path planning is based on A* [16] that is used to find the optimal path in the graph. The path planning algorithm is performed when one of the AMRs receives the destination. First, both AMRs judge whether they are likely to face each other based on the location and destination of the other robot. The criterion is whether they are both on the same side of the main two roads and facing each other. If the two robots are judged to face each other, the robot going to the shipping location generates a detour route by drawing some virtual walls that block the



(a) For a human worker.

(b) For an administrator.

Fig. 4: Web application for graphical user interface on DigiMobot.

way of the other robot before performing A*. The generated path contains velocity, and the velocity is set to 0.5 $\mathrm{m/s}$. This velocity is slow down to ensure safety in warehouses with many shelves and limited visibility.

Using the obtained path, the vehicle is controlled by ROS nodes provided by Autoware: waypoint_loader, pure_pursuit, astar_avoid, velocity_set, and twist_filter. The waypoint_loader node loads the path obtained by the path planning node. The pure_pursuit node performs pure pursuit algorithm that determines target velocity and angular velocity. The velocity_set node determines velocity while detecting obstacles. The astar_avoid node is for stopping when obstacles are detected. The twist_filter node adjusts velocity based on lateral acceleration.

B. Human Agents and Human Workers

In this section, we present the implementation of human agents simulation and the Human interface for the real experiments. First, in regard to simulating human agents, the Human Simulator (HSim) on the cloud controls the Navmesh agents on Unity by sending their destination and receiving their locations at 10 Hz. Walking speed of the Navmesh agents is set at 0.91 $\mathrm{m/s}$. This speed is determined by using the speed of the subjects to walk with a cart.

Figure 6 shows the workflow for each human agent in the HSim. First, the human agent receives a single picking order from the management system, then the HSim sends the destination in front of the first shelf of the order. When the worker location is within a three-meter radius of the destination, the HSim sends a picking message to the server after 10 seconds. This 10 seconds of sleep represents the amount of time workers are picking. This flow repeats until the last item of the order is picked, and then the worker is

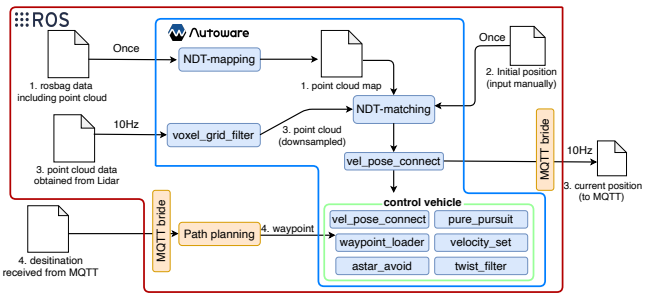


Fig. 5: The overview of controlling AMR processes. Each box represents a ROS node.

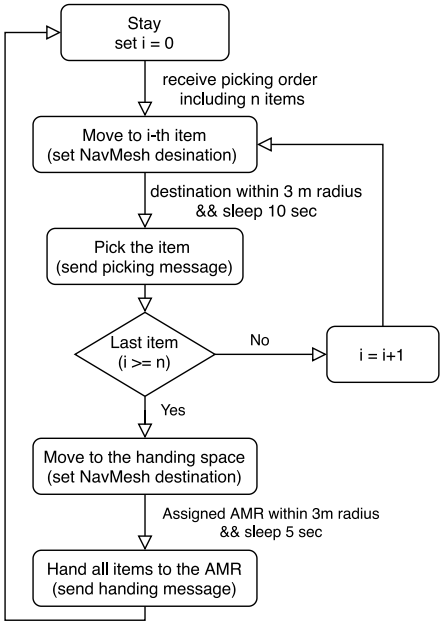


Fig. 6: The workflow for a human agent in simulation.

set the destination to the nearest road that AMR is passable. Finally, when the assigned AMR comes within a three-meter radius of the worker, the HSim sends the handing message that means the items already have been handed to the AMR. Note that the HSim knows the assigned AMR by referencing the AMR status shared by the management system.

On the other hand, in the real world, the worker works by checking the instructions from the human interface on the web using wearable devices such as a smartphone. The human interface refers to the agents status shared by the management system and displays instructions. The workers use the picking and handing buttons on the human interface to communicate their status to the management system. When the workers finish picking the last item in the order, the assigned AMR is displayed, and they can grasp AMR to be passed is known.

C. Management System

Finally, we present the implementation of Management System (MS). Figure 7 shows the state transition used by each AMR. We defined the status into shown four categories based on where and when the AMR is moving or stopping. This one-way state transition allows the state inconsistency to be corrected by the command from the Human Interface to shift the state. Whereas, workers status is just updating the status by the latest message that is picking message or handing message because the workers can determine their own actions unlike robots. In particular, other providers in the cloud refers the index number of items that the worker is picking and whether they have already handed over the items to the AMR as workers status.

The picking orders are allocated to workers in predefined order when the worker hands products to AMR. Whereas, tasks for AMRs are allocated when the number of remaining items that a worker picks up is less than three.

V. Experiments in a Warehouse

In this section, we show the results and evaluations for the real experiments in a warehouse.

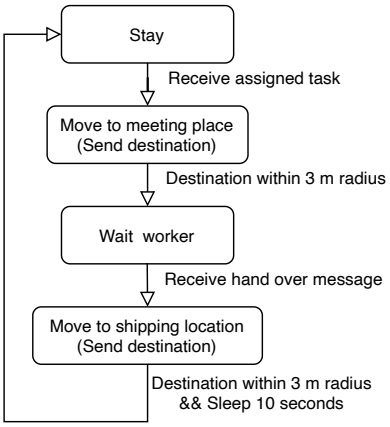
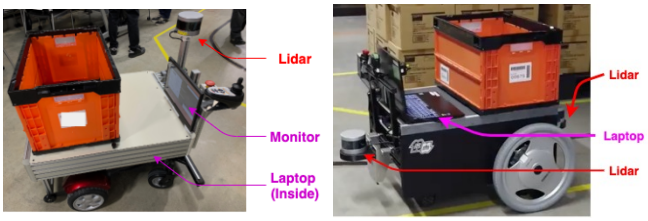


Fig. 7: State transition used by each AMR.

A. Real Experiment Settings

DigiMobit can be applied in various indoor environments such as offices, hospitals, libraries, schools, factories. In this paper, we conduct experiments in a warehouse as an example. This warehouse locates in Saitama, Japan, and stores more than 400,000 industrial items such as work clothes, wrenches, drills, and toolboxes. We experimented in one area of this warehouse, which is about 70 m long and has a total area of about 3200 m^2. The workers in this warehouse pick these items with a cart and box and carry them to the shipping location. We apply DigiMobit to replace this work except the picking for commercial purposes. It took place a day to prepare and a day to conduct.

In the actual experiment, we implement two different modified AMRs shown in Figure 8. These two AMRs are installed LiDAR (Velodyne VLP-16), laptop computer (Intel Core i7-10750H 2.60GHz processer, Ubuntu 18.04), and wireless network adaptor. Besides, we configured a network infrastructure with Wi-Fi wireless routers in three locations.



(a) Thouzer. (b) Logiee.

Fig. 8: Two different types of autonomous robots for experiments.

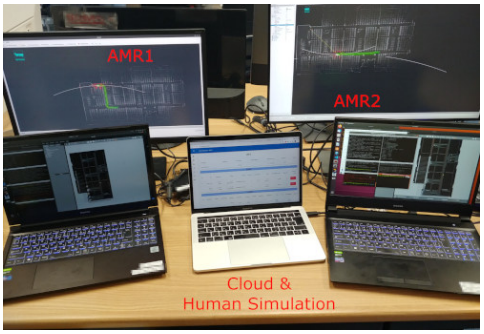
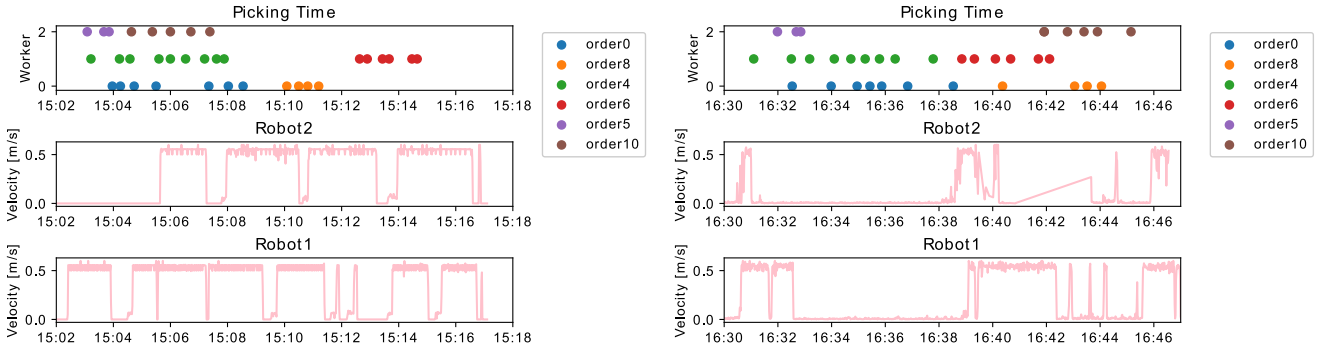


Fig. 9: Hardware setup for simulation on DigiMobot.

Fig. 10: Experiments in a warehouse. A mobile robot meets a human worker.



(a) In the simulation

(b) In the experiment

Fig. 11: Picking time of three workers and velocity of two AMRs. Each scattered color represents a single picking order.

We validate a picking scenario with three workers and two AMRs. The three workers are non-warehouse workers and worked with a smartphone and a cart. The picking orders we used are the actual six data that the warehouse used. We try only six picking orders because it takes a long time to prepare the system and perform the scenario.

B. Results

We show the results obtained from the real experiments comparing with the simulation experiments. Figure 9 and Figure 10 show these progresses. Although our objective in the real experiment was to execute the system tested in the simulation as it was, several problems that did not occur in the simulation arose. First, the communications between the cloud and the robot stopped due to network trouble in the large warehouse with many shelves blocking signals. This problem caused the system workflow to stop until the network reconnected. Picking errors and operation errors also occurred by the untrained non-warehouse workers and caused inconsistencies between the physical and cyber status. Since we had designed this system to be adaptable to such troubles with corrections by an administrator, we addressed such problems through a few attempts. Eventually, we addressed the problems in sending handing commands as correction every time by an administrator.

Figure 11 compares the picking time and velocities of two AMRs recorded in the experiment with the simulation. Picking time indicates when the cloud receives the picking message, and velocities are calculated using location data sent by each AMR. The linear velocity change from 16:40 to 16:44 of Robot2 in Figure 11b shows that communication stopped due to the network error. In addition, both robots stopped for about eight minutes, and worker2 also stopped picking between 16:33 and 16:42 in the experiments. In order to improve efficiency, it was necessary to devise a better way of allocating tasks and increase the number of AMRs.

C. Evaluation of the difference

Finally, we evaluate the difference between virtual environments and actual environments based on the results of the experiments. Table II shows the frequency of localization and vehicle command processed on an AMR, distance traveled by each AMR, and the total time of experiments in the virtual and actual environments. The frequency is obtained by rosbag data and each value represents the processing load of each process. As shown in table II, there were slight differences in the AMRs processes. Although the distance traveled by robot2 is uncertain, the total traveled distance of two AMRs was also almost the same. Total time is in the duration from sending orders until being handed all items. The problem here is that there was a significant difference in the behavior of humans. Unfortunately, actual humans movements could not fully collect due to network trouble. The humans did not actually travel the shortest path (they also sometimes make mistakes or slack off). Therefore, it was insufficient to reproduce human work only by the elapsed time, and despite the short experiment scenario, it took about twice as long.

VI. RELATED WORK

Over the past last few decades, automated vehicle systems have been studied from various perspectives. In particular, there has been much research on simulation methods [17], [18], multiple vehicles [19], [20], [21], and human-robot cooperation [22], [23]. However, challenges remain in achieving a fully autonomous system using mobile robots that move in the same space as humans. For instance, Paudel et al. proposed Kiva System using hundreds of mobile robots that lift and carry specific shelves [3]. Since this system is targeted only at a specific environment and requires capital investment to install, it is difficult to apply to a wide range of environments. Besides, Singhal et al. proposed three different types of fleet management systems for autonomous

TABLE II: Difference between virtual environment and actual environment.

	Virtual	**Actual**
Freq. of localization [Hz]	9.67	9.915
Freq. of vehicle command [Hz]	30.03	30.035
Robot1 traveled distance [m]	299.7	347.6
Robot2 traveled distance [m]	256.9	more than 124.6
Total Time	13 m 1.12 s	25 m 4.11 s

ground vehicles using a cloud platform including global path planner and environmental updating [19]. Hu et al. proposed a heterogeneous multi-robot system and demonstrated the feasibility [20]. Since these systems require human operation for indicating the destination, it shows the difficulty of full automation of multiple mobile robots cooperating with humans. On the contrary, our system objects to improve the feasibility of a safe and fully automation for human-robot collaboration and cooperation.

VII. Conclusion and Future Work

In this paper, we presented a digital twin system and architecture to support, manage, monitor, and validate autonomous mobile robots named DigiMobot. DigiMobot supports realistic simulations for human behaviors and robot movements, and it is also able to monitor and manage multiple autonomous robots in a physical world. By using the system, we can understand the overall systems to safely operate autonomous robots in the mixed environments. In addition, we conducted real-world experiments in a warehouse that is used for commercial purposes and stores more than $400,000$ items. To show the feasibility, we developed two different types of autonomous robots, and we also designed and implemented software frameworks.

We note several limitations of our work. First, our system must be robust against communication errors and/or system failures. Since each robot in DigiMobot is connected wirelessly, message delays and/or drops can happen. We will design a robust framework and/or cooperation protocol for such failures. For instance, we need technology that robots can autonomously understand the next task to be performed when a network problem occurs. And we need to automatically reflect this information to the cloud after the network is restored. Secondly, although our system enabled the simulations for human agents, there are many factors to determine human behaviors. To improve the system reliability, we will study human behaviors to design more realistic and configurable human models.

ACKNOWLEDGMENT

This work was partly supported by The National Institute of Advanced Industrial Science and Technology (AIST) and Japan Science and Technology Agency Open Innovation Platform with Enterprises, Research Institute and Academia (JST OPERA).

References

[1] K. Gao, J. Xin, H. Cheng, D. Liu, and J. Li, "Multi-mobile robot autonomous navigation system for intelligent logistics," in *2018 Chinese Automation Congress (CAC)*, pp. 2603–2609, IEEE, 2018.

[2] A. Bhat, S. Aoki, and R. Rajkumar, "Tools and methodologies for autonomous driving systems," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1700–1716, 2018.

[3] D. B. Poudel, "Coordinating hundreds of cooperative, autonomous robots in a warehouse," *Jan*, vol. 27, no. 1-13, p. 26, 2013.

[4] D. M. Ebert and D. D. Henrich, "Safe human-robot-cooperation: Image-based collision detection for industrial robots," in *IEEE/RSJ international conference on intelligent robots and systems*, vol. 2, pp. 1826–1831, IEEE, 2002.

[5] F. Camara, N. Bellotto, S. Cosar, F. Weber, D. Nathanael, M. Althoff, J. Wu, J. Ruenz, A. Dietrich, G. Markkula, *et al.*, "Pedestrian models for autonomous driving part ii: high-level models of human behavior," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[6] M. Schluse and J. Rossmann, "From simulation to experimentable digital twins: Simulation-based development and operation of complex technical systems," in *2016 IEEE International Symposium on Systems Engineering (ISSE)*, pp. 1–6, IEEE, 2016.

[7] R. Minerva, G. M. Lee, and N. Crespi, "Digital twin in the iot context: a survey on technical features, scenarios, and architectural models," *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1785–1824, 2020.

[8] "Mqtt - the standard for iot messaging." https://mqtt.org/. Accessed: 2020-02-01.

[9] "Synerex project: Service integration platform for smart cities and society 5.0." https://github.com/synerex, 2021. Online; accessed 1 February 2021.

[10] 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.

[11] S.-K. Jo, D.-H. Park, H. Park, and S.-H. Kim, "Smart livestock farms using digital twin: Feasibility study," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1461–1463, IEEE, 2018.

[12] B. Korth, C. Schwede, and M. Zajac, "Simulation-ready digital twin for realtime management of logistics systems," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 4194–4201, IEEE, 2018.

[13] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2149–2154, IEEE, 2004.

[14] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[15] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 287–296, IEEE, 2018.

[16] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[17] Y. Mizuchi and T. Inamura, "Cloud-based multimodal human-robot interaction simulator utilizing ros and unity frameworks," in *2017 IEEE/SICE International Symposium on System Integration (SII)*, pp. 948–955, IEEE, 2017.

[18] S. Aoki, L. E. Jan, J. Zhao, A. Bhat, R. R. Rajkumar, and C.-F. Chang, "Co-simulation platform for developing inforich energy-efficient connected and automated vehicles," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1522–1529, IEEE, 2020.

[19] A. Singhal, P. Pallav, N. Kejriwal, S. Choudhury, S. Kumar, and R. Sinha, "Managing a fleet of autonomous mobile robots (amr) using cloud robotics platform," in *2017 European Conference on Mobile Robots (ECMR)*, pp. 1–6, IEEE, 2017.

[20] C. Hu, C. Hu, D. He, and Q. Gu, "A new ros-based hybrid architecture for heterogeneous multi-robot systems," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pp. 4721–4726, IEEE, 2015.

[21] S. Aoki and R. Rajkumar, "V2V-based synchronous intersection protocols for mixed traffic of human-driven and self-driving vehicles," in *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 1–11, IEEE, 2019.

[22] Z. Li, S. Zhao, J. Duan, C.-Y. Su, C. Yang, and X. Zhao, "Human cooperative wheelchair with brain–machine interaction based on shared control strategy," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 1, pp. 185–195, 2016.

[23] J. Berg, A. Lottermoser, C. Richter, and G. Reinhart, "Human-robot-interaction for mobile industrial robot teams," *Procedia CIRP*, vol. 79, pp. 614–619, 2019.