2024 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)
July 15-19, 2024. Boston, MA, USA

Collision Detection for Robot Arm Assisted with Digital Twin System

Kengo Tajiri[1] and Takuya Iwamoto[2]

***Abstract*— In industrial and disaster recovery operations, robot arms take over tasks in hazardous areas. Collision detection methods are essential to prevent damage to the surrounding environment and the robot arm itself. Traditional approaches, such as environmental cameras and tactile sensors, face limitations like installation challenges and high costs. Model-based and data-driven strategies also struggle with accuracy, hindered by external disturbances like friction and the need for extensive data collection. Addressing these challenges, this paper introduces a digital twin-based approach for more accurate collision detection. Digital twins, virtual replicas of physical environments, enable cost-effective and safe data collection. This research utilizes a physical simulator within the digital twin to simulate the efforts of a real robot arm based on the robot's physical characteristics and motion trajectory, enhancing detection accuracy by adjusting for modeling errors and disturbances through linear regression. Experiments with the xArm 7 robot arm, simulating collisions by applying manual force, demonstrate the proposed method outperformed conventional model-based and machine-learning methods.**

I. Introduction

In industrial environments and disaster recovery sites, robot arms are anticipated to perform tasks instead of humans in areas where human entry is hazardous. However, when the working environment is not fully understood, robots risk unexpected collision with obstacles, leading to potential damage or operational failure. To mitigate damage to the robot itself and surrounding objects during a collision, collision detection in robotic control is a crucial task, and various studies have been conducted in this field [1].

The most basic collision detection methods are monitoring the operational environment with cameras [2], [3] and equipping robots with tactile sensors [4], [5]. However, installing cameras in work environments inaccessible to humans is difficult, and outfitting the entire robot with sensors to detect collisions is costly. Other approaches involve model-based methods that estimate external joint torques by inputting the robot's characteristics, such as mass, along with measured voracity [6], energy [7], and momentum [8], [9] into motion equations. In model-based methods, isolating external torques is difficult due to including forces generated by friction, modeling errors, etc. Additionally, data-driven methods exist, where machine learning models are trained to detect collisions based on the robot's operational data. In machine learning-based methods, supervised learning techniques are proposed for classifying collision and non-collision data [10], [11], and unsupervised learning techniques for training collision detection models using only non-collision data [12], [13]. However, both approaches require substantial data to capture the motion characteristics of a robot arm accurately.

This paper proposes a method for robot collision detection using digital twins, a concept increasingly being adopted for the realization of Industry 4.0 [14], [15]. In digital twins, a virtual environment that simulates real-world conditions is created, and the data obtained therein can be used for various tasks in actual environments. The advantage of collecting data in a virtual environment is that it can be done at a lower cost than in real settings, and it can gather data on operations that are prohibited in systems already in operation. However, to utilize data generated in a virtual environment for tasks in real environments, the virtual environment must be made as close to the real environment as possible.

In the task of robot collision detection discussed in this paper, a physical simulator is defined within the digital twin's virtual environment to calculate the efforts of a robot arm from the robot's physical characteristics and intended motion trajectory based on motion equations. While prior research exists on collision detection by comparing estimated efforts with actual efforts measured from a robot [16], [17], these do not account for modeling errors or disturbances such as joint friction, resulting in inaccurate simulations. Therefore, to enhance the accuracy of the virtual robot's simulation, the efforts of the virtual and real robot during non-collision are compared, and linear regression is performed. This approach brings the virtual robot simulation closer to the actual behavior of the real robot, thereby improving collision detection accuracy.

In the experiments, a lightweight robot arm, xArm 7 [18] was operated, and data was generated by manually applying force at various points to simulate collision. Comparative experiments were conducted with several model-based and machine learning methods, demonstrating that the proposed method surpasses these existing methods.

II. Related Works

Several related studies, including those cited in Section I, are discussed in detail. Among model-based methods, the energy observer-based collision detection [7] estimates power related to external joint torque from measured kinetic energy. However, the method does not directly estimate external torque. The velocity observer-based collision detection [6] estimates external torque from measured joint velocities, requiring the computation of the inverse matrix of the moment of inertia. To circumvent calculating the second derivative of joint positions and the inverse matrix of inertia,

[1]Kengo Tajiri is with NTT Network Service Systems Laboratories, NTT Corporation, Tokyo 180-8585, Japan kengo.tajiri@ntt.com

[2]Takuya Iwamoto is with National Institutes for Quantum Science and Technology, 801-1 Mukoyama, Naka, Ibaraki, 311-0193, Japan iwamoto.takuya@qst.go.jp

methods were proposed that estimate external torque from momentum measured by a Momentum Observer (MOB) [8], [9]. These methods, due to model simplifications, often overlook the effects of friction and noise, which can lead to decreased accuracy when using estimated values for collision detection.

Several machine learning-based methods are introduced next. Heo et al. [10] proposed Collision Net, constructed with a 1D CNN, which takes joint positions, joint velocities, and reference torque as inputs to output whether a collision has occurred. Zhang et al. [11] created features from statistical processing of measured motor torque, such as mean and variance, and selected features to input into machine learning models, evaluating collision detection with four supervised machine learning models, including kNN and FNN.

Nakamura et al. [12] proposed a method using the measurements from a force-torque sensor as inputs to an unsupervised machine learning model, One Class SVM (OCSVM), for collision detection. Park et al. [13] evaluated collision detection using OCSVM and AutoEncoder (AE) as unsupervised machine learning models, without using force-torque sensors but rather motor current as inputs. Lim et al. [19] proposed a method combining model-based and machine learning-based approaches for collision detection, training a Long Short Term Memory (LSTM) model for estimating external torque from joint positions and joint velocities data collected when no collision occurs, and comparing the estimated external torques with those calculated from MOB during actual robot operation. Supervised methods require training with various collision points and intensities. Hence, this paper proposes an unsupervised method and compares it with existing unsupervised ones.

Finally, prior research on anomaly detection using digital twins is discussed. In smart manufacturing, Xu et al. [20] proposed a method to train anomaly detection models using data from a virtual space modeling the physical space and then performing transfer learning with a small amount of real data for anomaly detection in the physical space. Xu et al. [21] proposed a method in Cyber-Physical Systems using an LSTM-based Generative Adversarial Networks (GAN) to create virtual data closely resembling those from real environments, employing the GAN's discriminator for anomaly detection. Our study assumes a reasonably accurate model can be derived from the motion equations for robot motion, and simple linear regression is used to correct and bring the virtual environment's robot closer to the real environment's robot.

III. Experimental Setup and Proposed Method

We used a lightweight robot arm and VR robot simulator for dataset collection. The xArm 7 is a manipulator having 7 DoF, as shown in Fig. 1. As a VR robot simulator, we used VR4robots [22]. ROS2 [23] environment on the Linux PC controlled the real robot and VR simulator. We collected collision and no-collision data with a real robot (Section III-B) and no-collision data with a VR robot simulator (Section III-C).

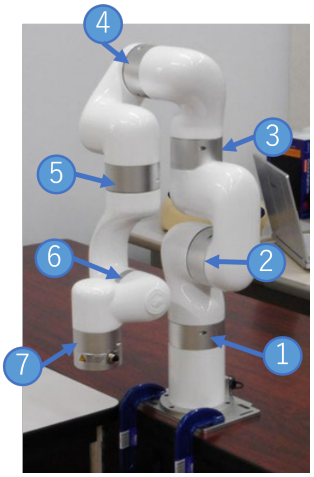

Fig. 1. xArm 7 manipulator having 7 DoF. Silver parts of the robot are rotational joints.

A. Data collection target

xArm 7 does not have joint torque sensors on its joints. We collected "efforts" $\boldsymbol{\tau}_{\text{eff}}$ instead of real motor torque $\boldsymbol{\tau}_F$. xArm API provides torque estimation function from motor current, and we used these values as $\boldsymbol{\tau}_{\text{eff}}$.

The dynamics of a robot can be described as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+\boldsymbol{g}(\boldsymbol{q})=-\boldsymbol{\tau}_{\text{F}}+\boldsymbol{\tau}_{\text{m}}+\boldsymbol{\tau}_{\text{ext}}, \quad (1)$$

where:

- $\boldsymbol{q}$ joint positions in the generalized coordinates
- $\boldsymbol{M}$ inertia matrix
- $\boldsymbol{C}$ Coriolis matrix
- $\boldsymbol{g}$ gravity torque vector
- $\boldsymbol{\tau}_{\text{F}}$ frictional torque in joints
- $\boldsymbol{\tau}_{\text{m}}$ motor torque
- $\boldsymbol{\tau}_{\text{ext}}$ external joint torque

Effort $\boldsymbol{\tau}_{\text{eff}}$ is equal to the sum of torque terms as follows,

$$\boldsymbol{\tau}_{\text{eff}}=-\boldsymbol{\tau}_{\text{F}}+\boldsymbol{\tau}_{\text{m}}+\boldsymbol{\tau}_{\text{ext}}. \quad (2)$$

Joint position, velocity, and effort were defined as ROS2 messages, and this message was recorded by a recording function implemented on ROS2 known as rosbag2 (Fig. 2). We collected data under the conditions described in Table I. Here, $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$, $\boldsymbol{\tau}_{\text{F}}$, $\boldsymbol{\tau}_{\text{m}}$, $\boldsymbol{\tau}_{\text{ext}}$, and $\boldsymbol{\tau}_{\text{eff}}$ are physical quantities that depend on time t and should be written as $\boldsymbol{q}(t)$, etc. precisely. However, they will usually be written simply as $\boldsymbol{q}$, etc., and only when it is necessary to show time t dependence will they be written as $\boldsymbol{q}(t)$, etc., after that.

TABLE I
Acquisition Condition for Data Collection

Sampling period	10 ms
Target average velocity	30 deg/s
Maximum radius of robot motion	200mm

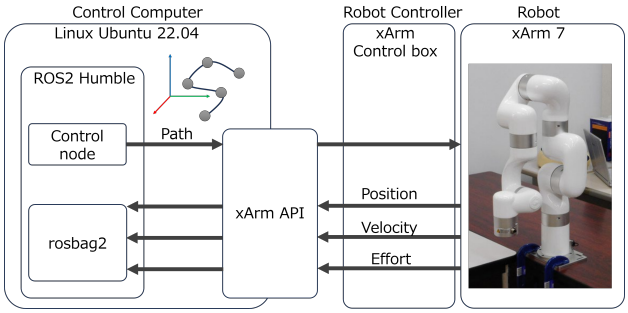

Fig. 2. System configuration diagram for the robot control

B. Data collection with real robot

In the no-collision data collection, random teaching points and paths through these points were generated, and the real robot moved on the path (Fig. 3). We used the xArm API to obtain the path, and no-collision data were acquired by moving the xArm 7 along the paths without collision.

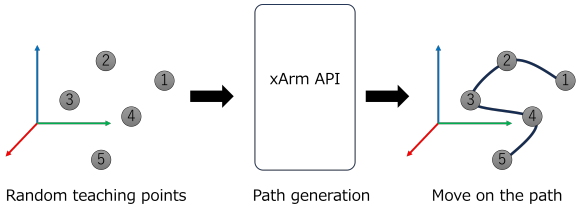

Fig. 3. Path generation from given teaching points

To collect collision data, an experimenter intentionally collided with the moving robot by hand. The collision data were generated by contacting one of the following four points: between the third and fourth joints, between the fourth and fifth joints, between the fifth and sixth joints, and between the sixth and seventh joints. During the intentional collision, the experimenter turned on a mechanical switch to add a flag on the joint state time history (Fig. 4).

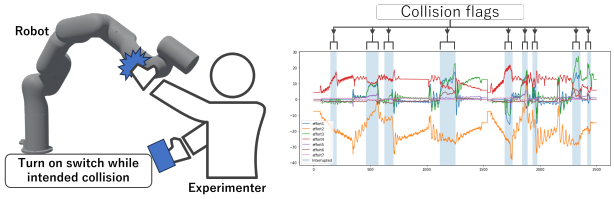

Fig. 4. Anomalies data collection

C. Data collection with VR robot simulator

VR4robots, the robot physics simulator for fusion device remote handling environment, was used for the virtual data collection. The real robot velocity history was used as the input to the VR simulator, as shown in Fig. 5. VR simulator calculated $\boldsymbol{q}$ and $\ddot{\boldsymbol{q}}$ based on the real robot velocity history. Then, the left-hand side of (1) was determined, and its value was recorded as VR effort $\boldsymbol{\tau}_{\text{eff}}$.

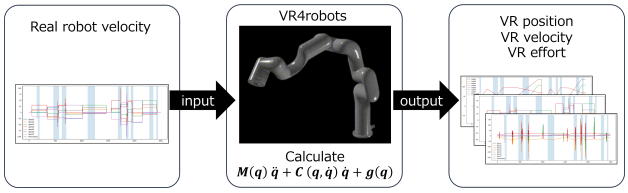

Fig. 5. VR data collection

D. Comparison of collected data in each configuration

Fig. 6 through Fig. 9 are examples of collected data. The physics engine in the VR software re-calculated the joint positions of VR data based on the real robot velocity history; no errors are shown between the real robot data and VR data. Regarding the joint efforts, although there are differences due to modeling errors between the VR model and the actual robot, the trends are consistent.

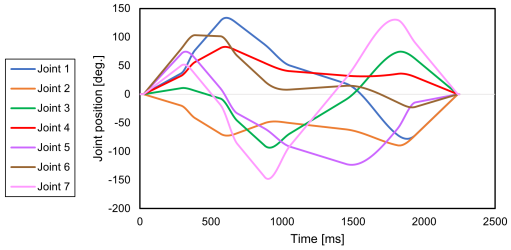

Fig. 6. Joint positions (the actual robot)

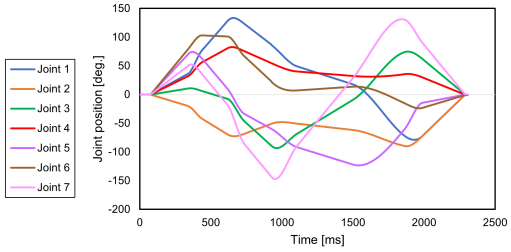

Fig. 7. Joint positions (VR)

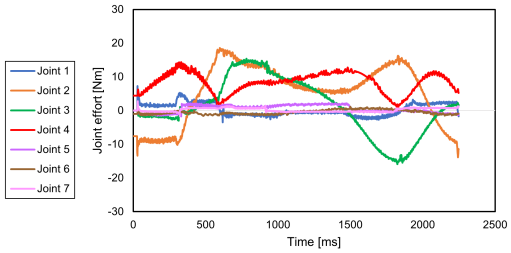

Fig. 8. Joint efforts (the actual robot)

E. Collision Detection

As mentioned in Section III-B and Section III-C, we first generate several target paths and obtain the real efforts $\boldsymbol{\tau}_{\text{eff}}$

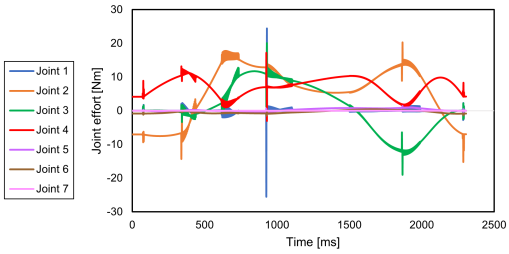

Fig. 9. Joint efforts (VR)

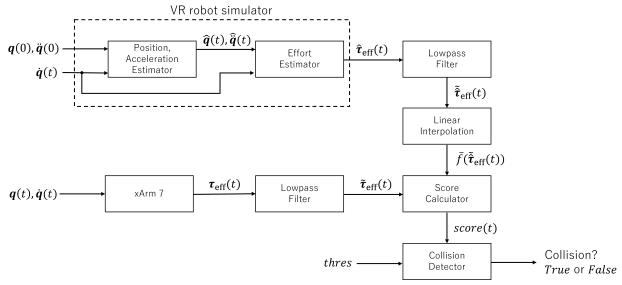

Fig. 10. Block diagram of the proposed framework.

and estimated efforts $\hat{\boldsymbol{\tau}}_{\text{eff}}$ using xArm 7 and VR4robots based on the generated paths. Then, $\boldsymbol{\tau}_{\text{eff}}$ and $\hat{\boldsymbol{\tau}}_{\text{eff}}$ are low-pass filtered, and they are represented as $\tilde{\boldsymbol{\tau}}_{\text{eff}}$ and $\tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}}$, respectively because the noise oscillation in $\boldsymbol{\tau}_{\text{eff}}$ and some spikes in $\hat{\boldsymbol{\tau}}_{\text{eff}}$ (Fig. 9) can decrease the accuracy of collision detection.

Next, to allow $\tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}}$ to represent $\tilde{\boldsymbol{\tau}}_{\text{eff}}$ more accurately, $\tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}}$ is adjusted by linear regression. In detail, the parameters of the linear regression $\bar{\boldsymbol{a}} \in \mathbb{R}^D$ and $\bar{\boldsymbol{b}} \in \mathbb{R}^D$ are determined by the following equation,

$$\bar{\boldsymbol{a}}, \bar{\boldsymbol{b}} = \text{argmin}_{\boldsymbol{a},\boldsymbol{b}} \sum_{n=1}^{N} \sum_{t=1}^{T_n} ||\tilde{\boldsymbol{\tau}}_{\text{eff}}(t) - f(\tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}}(t))||^2, \quad (3)$$

$$f(\tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}}) = \boldsymbol{a} \odot \tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}} + \boldsymbol{b}, \quad (4)$$

where N is the number of the generated paths used for the linear regression, T_n is the measurement time of n-th path data, $\odot$ means Hadamard product, and D is the number of the joints in xArm 7, that is 7 in this paper. In the linear regression, we use only the no-collision data because we need to make $\tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}}$ closer to $\tilde{\boldsymbol{\tau}}_{\text{eff}}$ in the collision-free case in order to make the digital twin for using collision detection.

After the parameters $\bar{\boldsymbol{a}}$ and $\bar{\boldsymbol{b}}$ are determined from (4), collision detection is performed. When the real effort $\boldsymbol{\tau}_{\text{eff}}$ and estimated effort $\hat{\boldsymbol{\tau}}_{\text{eff}}$ are obtained, performing the xArm 7 and VR4robots, the value $score(t)$ to verify whether xArm 7 is in collision at a certain time t is calculated as follows,

$$score(t) = ||\tilde{\boldsymbol{\tau}}_{\text{eff}}(t) - \bar{f}(\tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}}(t))||^2, \quad (5)$$

$$\bar{f}(\tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}}) = \bar{\boldsymbol{a}} \odot \tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}} + \bar{\boldsymbol{b}}. \quad (6)$$

Since in the case of a collision, $\tilde{\boldsymbol{\tau}}_{\text{eff}}(t)$ is different from $\bar{f}(\tilde{\hat{\boldsymbol{\tau}}}_{\text{eff}}(t))$ calculated assuming no collision, $score(t)$ becomes larger when a collision occurs. Therefore, we can perform collision detection based on $score(t)$.

Finally, we calculate a $score(t)$ for no-collision data to determine the threshold $thres$. In this paper, we define the threshold $thres$ as the $\mu + 4\sigma$ of the $score$ for no-collision data, where μ and σ are the mean and standard deviation of the $score$, respectively. In collision detection terms, whether a collision occurs or not is determined based on the following,

$$cd(score(t)) = \begin{cases} True & (score(t) \geq thres) \\ False & (score(t) < thres). \end{cases} \quad (7)$$

The block diagram of the proposed method is shown in Fig. 10.

IV. Experiment

A. Dataset

As described in Section III-B to III-C, we collected the data for the collision detection experiments with xArm7 and VR4Robots. Training data were collected by running the robot on xArm 7 and VR4Robots without collisions along 30 different paths, and validation data were also collected without collisions along the other five paths. Test data were collected with collisions along the other five paths. However, the test data also included data points without collisions because xArm does not always collide with the experimenter's hand (Fig. 4). The minimum measurement time per path was 21.8 seconds, and the maximum was 34.7 seconds. Then, 2183 to 3467 data points were collected per path.

We computed $\boldsymbol{\tau}_{\text{ext}}$ using MOB [8], [9] from the $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ collected on the paths for use in the comparison methods. In the calculation of $\boldsymbol{\tau}_{\text{ext}}$, we fixed the observer gain to 100.0.

B. Comparison Methods

We compared the proposed method with the following seven prior methods. For all methods, low-pass filtered data were input. In the first two methods, we used the validation data to determine the threshold and the test data to evaluate the method, and the training data were not used. For all of the remaining methods, the models' parameters were trained with training data, thresholds were determined with validation data, and evaluations were performed with test data Since the methods except the first two methods used machine learning model for collision detection, the input data was normalized by dimension as follows:

$$x_{[0,1]} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad (8)$$

where x was the value by dimension in the input, and $x_{\min}$ and $x_{\max}$ were the minimum and maximum values by dimension in the training data, respectively. The used quantities in each method are shown in Table II.

1) Naive Diff $\boldsymbol{\tau}_{\text{eff}}$ [16], [17]: The method simply compares between the real effort $\boldsymbol{\tau}_{\text{eff}} \in \mathbb{R}^7$ and the simulated effort $\hat{\boldsymbol{\tau}}_{\text{eff}} \in \mathbb{R}^7$. This method is equal to the substitution $\bar{\boldsymbol{a}} = 1$ and $\bar{\boldsymbol{b}} = 0$ in the proposed method.

2) Naive MOB [1]: When $\boldsymbol{\tau}_{\text{ext}} \in \mathbb{R}^7$ is calculated by MOB, a naive collision detection that judges an abnormality if $||\boldsymbol{\tau}_{\text{ext}}||^2$ is large, can be performed.

3) Point OCSVM: We adopted OCSVM, which was used in [13] as the comparison method. However, the proposed method performed collision detection for each data point, whereas Park et al. [13] did for each time window consisting of 16 data points, making comparisons impossible. Then, we performed collision detection for each data point with OCSVM. The input data was $\boldsymbol{\tau}_{\text{ext}}$ calculated by MOB. We set the kernel of OCSVM to rbf and the hyperparameter ν, which is an upper bound on the fraction of training errors, to 0.1.

4) Point AE: Since Park et al. [13] used AE as the collision detection method, we also adopted it as the comparison method. However, because AE was also used as a collision detection method for time windows, we inputted $\boldsymbol{\tau}_{\text{ext}}$ of each data point calculated by MOB to AE for collision detection to use AE for comparative evaluation. AE consisted of three middle layers, and these dimensions were set to 5, 3, and 5, respectively. Other hyperparameters were set as follows: the bath size was 1000, the active functions in all middle layers were ReLU, the dropout rate was 0.1, the optimizer was Adam, and the learning rate was 10^{-4}.

5) LSTM MOB [19]: Lim et al. [19] proposed a collision detection method that compares $\boldsymbol{\tau}_{\text{ext}}(t)$ calculated by MOB with the output of LSTM, which was trained to output $\boldsymbol{\tau}_{\text{ext}}(t)$ from $\boldsymbol{q}(t)$, $\dot{\boldsymbol{q}}(t)$, and $\dot{\boldsymbol{q}}(t-1)$ in the case of no collisions. The several hyperparameters of the LSTM model were determined based on Lim et al. [19]. However, the optimizer and learning rate were changed because the training did not proceed in the original setting with our data. The details of the settings were as follows: the sequence length of the input was 100, the dimensions in the middle layers of LSTM were 100, the batch size was 1000, the dropout rate was 0.0, the optimizer was Adam, and the learning rate was 10^{-4}.

6) Time Window OCSVM [13]: Although the methods proposed in Park et al. [13] could not be compared with the proposed method directly, collision detection using these methods was conducted as a reference. For each time window, we assigned a label of collision if the robot was touched by the experimenter's hand in the time. Otherwise, we assigned a label of no collision. We set the time window size to 16, the kernel of OCSVM to rbf, and the hyperparameter ν to 0.1. From the time window size, the dimension of input data was calculated as $7 \times 16 = 112$.

7) Time Window AE [13]: The label settings and time window size were the same as in the case of Time Window OCSVM. AE consisted of three middle layers, and these dimensions were set to 32, 11, and 32, respectively. Other hyperparameters were set as follows: the bath size was 128, the active functions in all middle layers were ReLU, the dropout rate was 0.1, the optimizer was Adam, and the learning rate was 10^{-4}.

C. Evaluation Metric

We used the F1 score and AUC as the evaluation metric of collision detection. The F1 score is a value calculated by the harmonic mean of Precision and Recall, which are calculated with the score output from the collision detection model and the threshold value for the score. Since the F1 score has a threshold dependence, we calculated the mean μ and standard deviation σ of the score calculated from the validation data for each method and then calculated the F1 score for each threshold value $\mu + i\sigma$ in 0.1 increments from $i = 0$ to 1 and in 1 increments from $i = 2$ to 5. The best F1 score for each method was compared with each other. To evaluate the methods independently of the threshold, we also used the AUC. For both metrics, the higher the values, the better the method.

D. Comparison Results

The experimental results are shown in Table II. Because the methods using the deep learning model (Point AE, LSTM MOB, and Time window AE) included randomness in the results, five experiments were conducted, and the results are shown in the form of mean $\pm$ standard deviation. The training time is the time it took to train the model, and the test time is the time it took to input test data into the model and receive the output. Since the first two methods did not involve training, the training time column is blank in Table II. A GPU server equipped with an NVIDIA V100-PCIe GPU was used to train the deep learning model, while the other methods were experimented on a PC equipped with an Intel(R) Core(TM) i9-10900K CPU.

Despite the simplicity of the proposed method, it achieved the highest score in the comparison methods, including the methods using a deep learning model for both the F1 score and AUC. The high accuracy of the proposed method is thought to be attributed to simple linear regression, which allowed the model to be effectively trained with a minimal dataset comprising only 30 paths. Moreover, the training time was the shortest among the comparison methods, and the test time was as short as that of the naive methods Naive diff $\boldsymbol{\tau}_{\text{eff}}$ and Naive MOB. The short training time of the proposed method is useful for implementing collision detection systems, and the short detection time means that real-time contact detection is possible.

Comparing the proposed method with Naive diff $\boldsymbol{\tau}_{\text{eff}}$, where linear regression was removed from the proposed method, we can see that the proposed method improved both the F1 score and AUC. The result indicates that the numerically calculated effort $\hat{\boldsymbol{\tau}}_{\text{eff}}$ did not represent the real effort $\boldsymbol{\tau}_{\text{eff}}$ well because there were factors that were not taken into account in the calculation.

E. Linear regression Results

In the proposed method, the results of linear regression were as follows:

$$\bar{\boldsymbol{a}} = (1.39, 1.17, 1.20, 1.16, 1.28, 1.30, 21.74), \tag{9}$$

$$\bar{\boldsymbol{b}} = (0.24, -1.27, -0.19, 0.05, 0.15, -0.01, 0.18). \tag{10}$$

The results show that the effort calculated by VR4Robots $\hat{\boldsymbol{\tau}}_{\text{eff}}$ tended to be smaller in absolute value than the actual effort $\boldsymbol{\tau}_{\text{eff}}$, indicating that the linear regression was essential for constructing a digital twin. In Naive diff $\boldsymbol{\tau}_{\text{eff}}$, the detection accuracy was degraded because the absolute value of the

TABLE II
RESULTS OF COLLISION DETECTION

	training time [s]	test time [s]	Input quantities	best threshold	F1 score	AUC
Naive diff $\boldsymbol{\tau}_{\text{eff}}$		3.4×10^{-4}	$\boldsymbol{\tau}_{\text{eff}}(t)$, $\hat{\boldsymbol{\tau}}_{\text{eff}}(t)$	$\mu + 4\sigma$	0.525	0.760
Naive MOB		3.1×10^{-4}	$\boldsymbol{\tau}_{\text{ext}}(t)$	$\mu + \sigma$	0.588	0.815
Point OCSVM	56.4	2.76	$\boldsymbol{\tau}_{\text{ext}}(t)$	μ	0.540	0.803
Point AE	245	5.6×10^{-2}	$\boldsymbol{\tau}_{\text{ext}}(t)$	$\mu + 2\sigma$	0.417 ± 0.097	0.693 ± 0.083
LSTM MOB	747	1.9×10^{-3}	$\boldsymbol{q}(t)$, $\dot{\boldsymbol{q}}(t)$, $\dot{\boldsymbol{q}}(t-1)$, $\boldsymbol{\tau}_{\text{ext}}(t)$	$\mu + \sigma$	0.604 ± 0.018	0.815 ± 0.015
Proposed method	1.2×10^{-2}	5.1×10^{-4}	$\boldsymbol{\tau}_{\text{eff}}(t)$, $\hat{\boldsymbol{\tau}}_{\text{eff}}(t)$	$\mu + 4\sigma$	**0.722**	**0.867**
Time window OCSVM	1.9×10^{-1}	2.6×10^{-2}	$\boldsymbol{\tau}_{\text{ext}}(t-15), \cdots, \boldsymbol{\tau}_{\text{ext}}(t)$	μ	0.585	0.815
Time window AE	110	8.6×10^{-3}	$\boldsymbol{\tau}_{\text{ext}}(t-15), \cdots, \boldsymbol{\tau}_{\text{ext}}(t)$	$\mu + 0.9\sigma$	0.561 ± 0.012	0.768 ± 0.009

difference between $\hat{\boldsymbol{\tau}}_{\text{eff}}$ and $\boldsymbol{\tau}_{\text{eff}}$ became large regardless of whether there was a collision or not, due to the small estimate of $\hat{\boldsymbol{\tau}}_{\text{eff}}$. The large seventh element of $\bar{\boldsymbol{a}}$ is thought to derive from the fact that the seventh joint was located at the tip of the xArm 7, making the numerical calculation of its effect difficult.

V. CONCLUSIONS

In this paper, we proposed the collision detection method using digital twins, where the real efforts $\boldsymbol{\tau}_{\text{eff}}$ measured by a robot arm and the VR efforts $\hat{\boldsymbol{\tau}}_{\text{eff}}$ numerically calculated from an equation of rigid body motion of a robot are regarded as digital twins. In the experiments, the proposed method achieved the highest score in the comparison methods, including the methods using a deep learning model for both the F1 score and AUC, while the test time was comparable to that of the naive methods. Furthermore, the analysis of the experimental results showed that it was important to approximate the real efforts $\boldsymbol{\tau}_{\text{eff}}$ by applying linear regression to VR efforts $\hat{\boldsymbol{\tau}}_{\text{eff}}$ in order to construct digital twins. In future work, we will develop a method to solve the collision isolation problem on the robot arm, which is the problem of identifying where the collision occurred on the robot arm by using the digital twins system introduced in this paper.

REFERENCES

[1] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[2] D. M. Ebert and D. D. Henrich, "Safe human-robot-cooperation: Image-based collision detection for industrial robots," in *2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2. IEEE, 2002, pp. 1826–1831.

[3] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 338–345.

[4] M. Strohmayr, H. Wörn, and G. Hirzinger, "The dlr artificial skin step i: Uniting sensitivity and collision tolerance," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1012–1018.

[5] A. Cirillo, F. Ficuciello, C. Natale, S. Pirozzi, and L. Villani, "A conformable force/tactile skin for physical human–robot interaction," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 41–48, 2015.

[6] S. Haddadin, *Towards safe robots: approaching Asimov's 1st law*. Springer, 2013, vol. 90.

[7] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the dlr-iii lightweight manipulator arm," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 1623–1630.

[8] A. De Luca and R. Mattone, "Actuator failure detection and isolation using generalized momenta," in *2003 IEEE International Conference on Robotics and Automation (cat. No. 03CH37422)*, vol. 1. IEEE, 2003, pp. 634–639.

[9] A. De Luca and R. Mattone, "Sensorless robot collision detection and hybrid force/motion control," in *2005 IEEE international Conference on Robotics and Automation*. IEEE, 2005, pp. 999–1004.

[10] Y. J. Heo, D. Kim, W. Lee, H. Kim, J. Park, and W. K. Chung, "Collision detection for industrial collaborative robots: A deep learning approach," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 740–746, 2019.

[11] Z. Zhang, K. Qian, B. W. Schuller, and D. Wollherr, "An online robot collision detection and identification scheme by supervised learning and bayesian decision theory," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1144–1156, 2020.

[12] K. Narukawa, T. Yoshiike, K. Tanaka, and M. Kuroda, "Real-time collision detection based on one class svm for safe movement of humanoid robot," in *2017 IEEE/RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 791–796.

[13] K. M. Park, Y. Park, S. Yoon, and F. C. Park, "Collision detection for robot manipulators using unsupervised anomaly detection algorithms," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 5, pp. 2841–2851, 2021.

[14] Q. Qi and F. Tao, "Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison," *Ieee Access*, vol. 6, pp. 3585–3593, 2018.

[15] S. Mihai, M. Yaqoob, D. V. Hung, W. Davis, P. Towakel, M. Raza, M. Karamanoglu, B. Barn, D. Shetve, R. V. Prasad, *et al.*, "Digital twins: A survey on enabling technologies, challenges, trends and future prospects," *IEEE Communications Surveys & Tutorials*, 2022.

[16] S. Takakura, T. Murakami, and K. Ohnishi, "An approach to collision detection and recovery motion in industrial robot," in *15th Annual Conference of IEEE Industrial Electronics Society*. IEEE, 1989, pp. 421–426.

[17] S. Morinaga and K. Kosuge, "Collision detection system for manipulator based on adaptive impedance control law," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 1. IEEE, 2003, pp. 1080–1085.

[18] "xarm 7 manual http://download.ufactory.cc/xarm/en/xArm\%20User\%20Manual.pdf?v=1578910898247 (last visited 2024-02-08)."

[19] D. Lim, D. Kim, and J. Park, "Momentum observer-based collision detection using lstm for model uncertainty learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4516–4522.

[20] Y. Xu, Y. Sun, X. Liu, and Y. Zheng, "A digital-twin-assisted fault diagnosis using deep transfer learning," *IEEE Access*, vol. 7, pp. 19 990–19 999, 2019.

[21] Q. Xu, S. Ali, and T. Yue, "Digital twin-based anomaly detection in cyber-physical systems," in *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2021, pp. 205–216.

[22] "Vr4robots website https://www.tree-c.nl/what-we-do/vr4robots/ (last visited 2024-02-08)."

[23] "Ros2 documentation https://docs.ros.org/en/iron/index.html (last visited 2024-02-08)."