Control Engineering Practice 118 (2022) 104957

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

A flexible manufacturing assembly system with deep reinforcement learning☆

Junzheng Li [a], Dong Pang [a], Yu Zheng [a], Xinping Guan [b,c,d], Xinyi Le [b,c,d,*]

[a] School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 20040, China
[b] Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China
[c] Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai 200240, China
[d] Shanghai Engineering Research Center of Intelligent Control and Management, Shanghai 200240, China

ARTICLE INFO

Keywords:
Reinforcement learning
Digital twin
Flexible manufacture
Assembly line

ABSTRACT

Traditional assembly line requires a significant amount of designs from engineers, especially in the case of multi-species and small-lot production. Recently, intelligent algorithms based on reinforcement learning are proposed to address this issue. However, the lower success rate and safety reasons limit their industrial applications. In this article, we proposed a systematic solution, including the automatic planning of assembly motions and the monitoring system of the production lines. In the planning stage, we built the digital twin model of the assembly line, then trained a deep reinforcement learning agent to assembly the workpieces. In the production stage, the digital twin model is used to monitor the assembly lines and predict failures. To validate the system we proposed, we conducted a peg-in-hole assembly experiment, and reached a 90% success rate for a single assembly attempt. During the whole experiment, no collision happens in the real world.

1. Introduction

With the development of the economy, the demand for manufacturing is becoming more and more customized and individualized. Thus, improving the flexibility of manufacturing becomes an important issue in Germany Industry 4.0 (Lu, 2017) and Made in China 2025 (Liu, 2016). Among all stages of manufacturing, improving the flexibility of the assembly process can significantly reduce equipment and labor costs, thus, flexible assembly has become a new research trend.

Many works aims to improve the flexibility of assembly and propose their controlling models (Sun et al., 2020; Zhang, Shi et al., 2017), and recently, reinforcement learning is discovered as a controlling model with higher flexibility for assembly tasks (Li et al., 2019a; Luo et al., 2019;Luo et al., 2018;Inoue et al., 2017). Compared with traditional human-designed process-oriented assembly controlling models, reinforcement learning aims to provide a uniform and self-learning programming paradigm for varied assembly tasks. Reinforcement learning is result-oriented. As shown with Fig. 1, a policy network fits a mapping from the observed state to the next action, as the controlling model. The training process optimizes the policy network with the supervision of the correct assembly relationship. During the whole process, engineers only provide digital models of workpieces and assembly relationships, increasing the flexibility greatly with the help of reinforcement learning.

However, despite much research in the field of flexible assembly, the assembly process still requires human involvement due to some difficulties. We summarize these difficulties as follows:

- On flexible production lines, the robot's movements are complex and varied. The flexible assembly needs to cope with a wide variety of assembly relationships. Also, assembly robots need to sense obstacles in the environment and avoid them. As a result, flexible assembly often requires extensive manual programming or human–machine collaboration.
- Assembly requires a high degree of precision. Workers often need to adjust their movements based on force-feedback. Although some force-feedback robots have been introduced in recent years, the corresponding assembly control models are still complex, inaccurate, and immature.
- Industrial applications require high reliability and success rate. These are the main drawbacks of reinforcement learning approaches due to their poor interpretability. The unknown mechanism of reinforcement learning may lead to risky actions that

☆ The work described in the paper was sponsored by National Nature Science Foundation of China (No. 62176152) and Shanghai Rising-Star Program (No. 20QC1401100).

* Correspondence to: Department of Automation, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China.
E-mail addresses: lijunzheng@sjtu.edu.cn (J. Li), pangdong@sjtu.edu.cn (D. Pang), yuzheng@sjtu.edu.cn (Y. Zheng), xpguan@sjtu.edu.cn (X. Guan), lexinyi@sjtu.edu.cn (X. Le).

https://doi.org/10.1016/j.conengprac.2021.104957
Received 25 June 2021; Received in revised form 24 September 2021; Accepted 25 September 2021
Available online 25 October 2021

are not affordable for factories. Moreover, it is difficult to further optimize the success rate with the unknown mechanism of neural networks.

Thus, an integrated system should be proposed in response to the above difficulties. Given the digital model of workpieces and the final assembly result, a flexible assembly algorithm is expected to learn how to assemble correctly while avoiding obstacles. The algorithm should also be able to fuse data from vision and force sensors to achieve higher precision. Based on some researches on reinforcement learning and digital twin, we trained an assembly agent on a digital twin model of the assembly line to achieve flexible assembly. Reinforcement learning provides a concise model for flexible assembly. However, due to modeling accuracy issues, most reinforcement learning agents that can assemble well in a simulated environment often fail in a real one. Thus, we applied a digital twin in this paper to achieve a higher success rate, rather than the traditional simulation engines. Another issue is the reliability of reinforcement learning. A monitoring unit is applied to predict risky actions made by the agent.

The main contributions of this article are as follows.

- We establish an accurate digital twin model of a flexible assembly system.
- We use deep reinforcement learning to implement the peg-in-hole assembly with visual and force sensors in a virtual environment.
- We introduce the reward from real scenarios gradually during training. As a result, our model can be rapidly deployed in real scenarios with a high assembly success rate.
- We establish an online monitoring system with a digital twin model to predict risky factors in advance.

The rest of this article is organized as follows. Section 2 provides a brief literature review of the digital twin, robot-based assembly, and reinforcement learning. Section 3 proposes the overall architecture of the intelligent flexible assembly system. Section 4 introduces the assembly system in detail, including the digital twin model, reinforcement learning-based policy model, and online monitoring system. Section 5 validates the intelligent assembly algorithm with a peg-in-hole assembly experiment. Finally, Section 6 concludes this article.

2. Related works

2.1. Digital twin in industry

The history of the digital twin can be traced to Grieves in 2003 (Grieves and Vickers, 2017). In the original concept, a digital twin model is built from the real object. We can analyze and test with the digital twin model to obtain more information and prediction of the real object. Due to the rapid growth of communication, modeling, and sensor technology, nowadays the concept of the digital twin is broadened and applied in many industrial areas (Tao et al., 2018).

The digital twin can be applied in the whole life cycle of products including the design, production, and using stages.

In the design stage, Tao et al. (2019) proposed a framework of digital twin-driven product design (DTPD), suggesting to involve digital twin in product planning, conceptual design, and detailed design. Canedo (2016) suggests adding feedback from digital twin to improve the design. Schleich et al. (2017) argued that digital twin makes it possible for designers to evaluate the product at early stages when managing geometrical variations.

In the production stage, The digital twin model can be used to monitor the physics equipment, make decisions, and optimize the production process. Weyer et al. (2016) predicted that digital twin represents the simulation of the next generation with vast IoT sensors. Fang, Peng et al. (2019) proposed a online scheduling digital twin system. Vachálek et al. (2017)argued simulation with digital twin helps to reduce material waste and extend the life of machines. Leng et al. (2020) proposed an upper-level digital twin model for coarse-grained optimization.

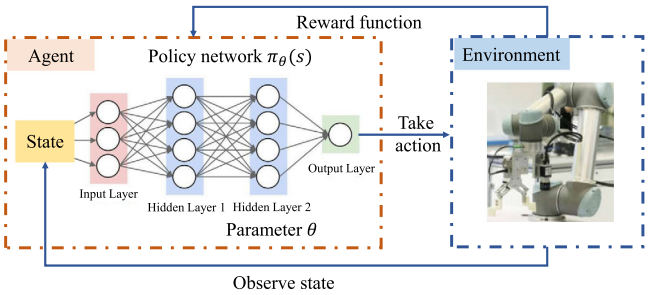

Fig. 1. General framework for reinforcement learning. Then agent learns skills with the feedback of environment.

In the using stage, the digital twin is commonly used in prognostics health management (PHM). Tuegel et al. (2011) built a digital twin system with a structural finite-element model (FEM) and damage model to predict the damage and manage the service life cycle of aircraft. Li et al. (2017) built a digital twin model with a dynamic Bayesian network to predict the leading edge of aircraft wings.

Most researches focus on the use of digital twin in PHM (Tao et al., 2018). Digital twin together with reinforcement learning is just at the beginning, however, we suspect that they can complement each other to further enhance the flexibility of manufacture.

2.2. Robot-based assembly and reinforcement learning

Automated assembly system with robots has been widely used in the industry (Fleming and Purshouse, 2002). Through the use of robots, equipment flexibility is increased. The traditional programming method for assembly tasks is setting key points and motions through teach pendant or computer. For high-precision assembly tasks, it is often time-consuming but necessary to make several fine adjustments to the assembly motion parameters (Li et al., 2019a; Song and Chu, 1998). Sun et al. (2020) proposed a human–robot collaborative system to address the issue as well.

Due to the rise of the collaborative robot with force sensors, some control models were proposed to reduce the workload of parameter adjustment(Liu et al., 2017; Zhang, Shi et al., 2017; Fang, Pang et al., 2019;Le et al., 2017;Zhang et al., 2019; Jasim and Plapper, 2014; Roveda et al., 2018). During the contact phase, these models can perceive and control the assembly force, and accomplished assembly tasks such as peg-in-hole assembly. Still, these models only focus on the contact phase, ignoring the searching phase with visual sensors. To reduce the complexity of the system, we expect to use a single model for all stages of the assembly process.

Reinforcement learning is a potential solution to accomplish assembly tasks with a single model. It refers to a “trial and error" learning fashion as shown in Fig. 1. The learning agent automatically learns assembly skills only with an experimental environment and a correct assembly result. In terms of implementation, a reward function is given by humans to measure how the agent’s action is close to the correct assembly result. Then, the agent tries to learn policy to maximize the reward function. Reinforcement learning approaches have been widely used in reaction process controlling (Zhu et al., 2020), autonomous vehicles (De Morais et al., 2020), motion controlling (Pi et al., 2020), decision making (Kolodziejczyk et al., 2021) and low-energy building (Yu and Dexter, 2010).

With the development of reinforcement learning and the widespread need for flexible manufacturing, several prototyping technologies based on reinforcement learning have emerged in the lab, such as grabbing and assembling. Zhang, Leitner et al. (2017) used the Deep Q Network (DQN) algorithm to train a tri-joint robot in a simulation

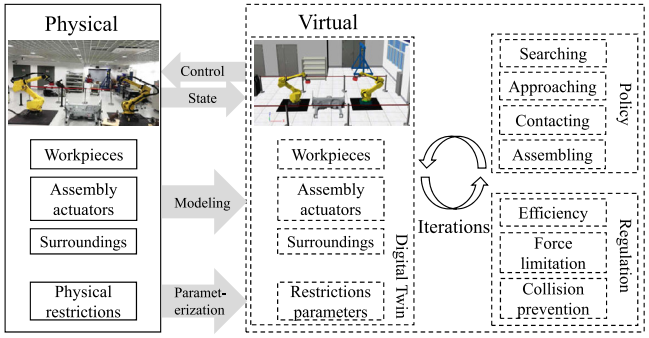

Fig. 2. Intelligent digital twin assembly system in the planning stage.

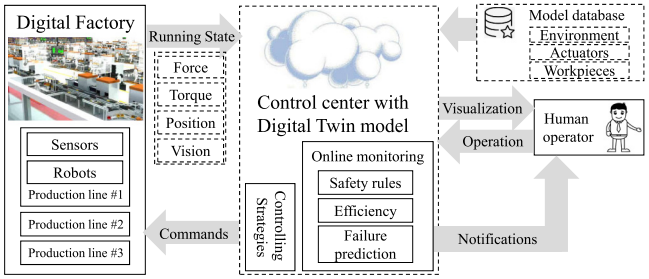

Fig. 3. Intelligent digital twin assembly system in the production stage.

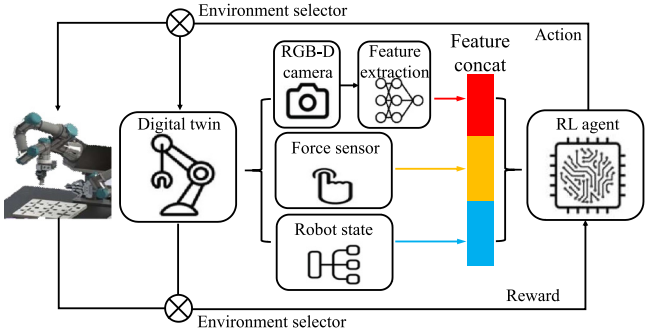

Fig. 4. Reinforcement learning for assembly task.

environment to achieve the grasping task. Gu et al. (2016) improved the DQN algorithm by introducing continuous motion space and realized similar manipulations with higher precision. With continuous improvements, the agents are now capable of multiple assembly tasks in a real laboratory environment, such as multiple peg-in-hole assembly (Li et al., 2019a), gear assembly (Luo et al., 2019), deformable–rigid objects assembly (Luo et al., 2018), and high precision assembly tasks (Inoue et al., 2017).

Although these studies are excellent and innovative, it is still difficult for them to replace traditional assembly solutions in the factory. The studies concentrate more on the contact phase in the assembly process, while a complete process should also include rough positioning in the assembly space. Another problem is that the success rate of the agents is relatively low and they may even cause safety problems, which do not meet the demand for actual industrial sites.

3. Overall architecture

Our flexible assembly system runs with two stages, the planning stage, and the production stage, as shown in Figs. 2 and 3. Firstly, a digital twin model is built from physical objects and used in both stages. In the planning stage, policy and regulation models are given by engineers. Policy model refers to a strategy, with which the algorithm decides what action to take when facing different observed states. In this paper, a reinforcement learning agent is a single policy model for all 4 stages, including searching, approaching, contacting, and assembling. The regulation model refers to physical or industrial restrictions to restrict the policy. When the policy gives an action that is out of regulation, this action should not be taken. Most of the time, the planning activities take place in the digital twin space with simulation engines, which is more efficient. Physical objects are involved only when milestones are achieved to ensure convergence. If the policy model is a reinforcement learning agent, the planning iterations run automatically with the learning happens.

After being tested in the physical assembly line, the assembly policy is then applied in the production stage. Running states of each assembly line are collected from sensors. To ensure the reliability, digital twin model in this stage is used for online monitoring. On the one hand, the digital twin model provides visualization for human operators. On the other hand, some self-monitoring rules run within the digital twin model.

Compared with traditional assembly systems, this architecture is more flexible and reliable because of the reinforcement learning algorithm and monitoring system with digital twin, respectively.

4. Intelligent assembly system

4.1. The digital twin model

The digital twin model is a digital representation of the elements involved in the assembly process. The digital twin model contains digitized physical entities in the assembly process, including the 3D models of workpieces, assembly actuators, and their surroundings. In some special cases such as assembling elastic workpieces, crucial material features including elastic modulus should also be modeled. Surroundings refer to the workplace where assembly happens. Objects around the actuators should be included as "surroundings" to ensure collision safety in the digital twin model because they are essential components for robot trajectory planning. The digital twin model should also include the physical restrictions, such as the speed and force limitation of actuators. These physical restrictions are represented in the parametric form in the digital twin model. The digital twin requires kinematics and dynamics engines to operate. The engines update the state of the model according to its previous states and controlling sequences, and output the state values that we are concerned with.

The simulation of the digital twin model is data streams. In the planning stage, the data source is virtual operations from the policy model. The digital twin engines then output state values to the policy model so that the policy model decides which action to take accordingly. In the production stage, the data source is production lines in the factory. Through the digital twin, human operators know what is happening, in the form of visualization, notification, and alerts.

In the planning stage of the assembly policy, the use of digital twin models plays an important role in balancing efficiency and quality. On the one hand, time in virtual space can flow faster without being constrained by physical time. Engineers or reinforcement learning agents can iterate on prototypes faster. On the other hand, to compensate for the accuracy of the simulation engines and factors that are not taken into account when modeling, the physical assembly system replaces the simulation engines when there is a necessity to ensure the convergence of the model on physical system.

Most reinforcement learning-based assembly approaches trained their models with simulated environments to save training time. Due to the inaccuracy of simulation, however, the success rate of assembly with real scenarios is relatively lower than that with the simulated environment. That is why we use the digital twin in the planning stage. Our reinforcement learning agent is trained with simulation engines firstly. When the agent has learned basic assembly skills, the simulation engines are replaced by a physical assembly system to optimize the success rate of the agent.

In the production stage, the digital twin model ensures the reliability of the system as Cecil et al. (2019) suggests. On the one hand, operations are validated in digital twin space before the commands are sent to the real production lines. If an operation is found to be risky in the digital twin space, warnings can be given in advance and emergency operations can be taken. This validation and correction are crucial for learning-based or statistic-based systems because they have poorer interpretability and success rate in nature. On the other hand, with the digital twin model, operators and online monitoring systems can perceive a more straightforward and detailed picture of production lines than raw data from sensors. Visualization provides a better interface between operators and data from sensors. The digital twin model also integrates information from the 3D model database of production lines and therefore provides more comprehensive safety protection than simple sensor data.

4.2. Policy model based on reinforcement learning

A typical assembly task can be divided into four stages as shown in Fig. 2. At the very first stage, the workpieces to assembly are separate from each other, and the relative position is not precise. Searching is the first stage as the result, to locate the workpieces including the relative translation and rotation distance. After searching, the actuator carrying a workpiece then approaches another workpiece until they contact. Finally, the actuator tries to assemble the workpieces with its sensors. Different policies can be adopted at different stages of the assembly, or one overall policy can be adopted at all stages, such as reinforcement learning agents. However, most studies focus on the last two stages. In our approach, we implemented the whole four stages within a single reinforcement learning model.

6R robot is the assembly actuator in our implementation. We aimed to train an optimal policy with fault-tolerant and corrective capabilities for unknown situations. As shown in Fig. 4, the state information collected includes the robot state, force signal, and the image from the RGB-D depth camera (including RGB image and depth image). The signals from the three modes are fused after feature extraction. The policy model is then trained accordingly. The policy learned here can be understood as a mapping from state space to action space. Each state-input corresponds to an action signal output. The reward is generated by the environment, but the reward function is designed artificially and carefully. When the cumulative reward is maximized, it indicates that the agent has learned the optimal assembly policy.

In the mainstream of deep reinforcement learning algorithms, the most widely used approach is the Deep Q Learning Network (DQN) (Zhang, Leitner et al., 2017). However, DQN can only deal with discrete, low-dimensional motion space, whereas tasks such as assembly with 6R robots have high-dimensional continuous motion space. Thus, we use the Deep Deterministic Policy Gradient (DDPG) (Li et al., 2019b). DDPG combines the Actor–Critic framework with the ideas of DQN and applies a deep neural network to learn policy in high-dimensional continuous action space.

In the basic paradigm of reinforcement learning, the agent completes one interaction with the assembly system per time unit. At step t, the agent receives an observation vector x_t from the assembly system, then performs an action a_t and receives a reward scalar r_t, representing the feedback. In the assembly task, the environment is fully observable. The behavior of the agent is defined as a policy π, where π is a mapping from a state to an action probability distribution, which is, $\pi : S \rightarrow P(a)$. The state observation, action, reward function, and policy are discussed as follows.

4.2.1. State observation

The state variable s_t comes from the digital twin model. It is provided to the agent to make decisions as follows:

$$\begin{aligned} s_t = \{ & P_E, O_E, J, F, M, P_H, O_H, \\ & P_E - P_H, \text{Safety}, \text{G}(\text{Color}, \text{Depth}) \}, \end{aligned} \tag{1}$$

where

$P_E = (P_E^x, P_E^y, P_E^z)$	Position of robot, 3-dim vector
$O_E = (O_E^x, O_E^y, O_E^z)$	Rotation vector of robot, 3-dim vector
$J = (\theta^1, \theta^2, \theta^3, \theta^4, \theta^5, \theta^6)$	Joint angle of robot, 6-dim vector
$F = (F^x, F^y, F^z)$	Force, 3-dim vector
$M = (M^x, M^y, M^z)$	Torque, 3-dim vector
$P_H = (P_H^x, P_H^y, P_H^z)$	Position of hole, 3-dim vector
$O_H = (O_H^x, O_H^y, O_H^z)$	Orientation of hole, 3-dim vector
Safety	Safety status, bool
Color	RGB image, $128 \times 128 \times 3$ tensor
Depth	Depth image, $128 \times 128 \times 1$ tensor
G(∗)	Feature extraction network

The RGB image and depth image are 3-dim tensors. A pre-trained convolutional neural network G(∗) is used to extract features of the input image. After feature extraction, we concat the state variables together. Hence, we get the state s_t.

4.2.2. Action

Each action that the agent responses is formulated as:

$$a_t = \{\delta_x, \delta_y, \delta_z, \delta_\alpha, \delta_\beta, \delta_\gamma\} \tag{2}$$

Action is the controlling sequence from the agent. Controlling in either joint space or Cartesian space is feasible. However, to reduce the complexity of the actor network fitting, the action in our approach is based on the Cartesian space. To ensure precision and contiguity, the maximum value of action is restricted below 1 mm or 0.001°.

4.2.3. Reward function

The reward function is the core of reinforcement learning, as it monitors the learning of the agent. For the assembly task, the simplest reward function is to give a reward for successful assembly and no reward for unsuccessful assembly. That is, the reward function $r(s)$ equals 1 if it successes, 0 if otherwise.

However, the reward function of this kind is a sparse reward, and it is difficult for the robot to learn the assembly skill during training because the gradient of reward is zero in most areas. Several approaches are proposed to solve the problem, such as reward shaping, curiosity mechanism, curriculum learning, and hierarchical reinforcement learning. In our approach, the reward shaping (Ng et al., 1999) method is involved to refine the reward function so that the agent gets reward gradually in each stage.

To alleviate the problem of sparse reward, reward shaping suggests giving progressive rewards rather than a solely final reward. The principle of designing a reward, in this case, is that the reward value is greatest when the peg and hole fit perfectly (the peg is completely inserted into the hole). The reward gradually decreases when the robot is away from this state. Two criteria are used to measure the process and quality of assembly, the relative posture and the force. In the situation of perfect assembly, the posture of the peg and hole coincide, and the force is zero. Thus, the reward goes lower when the posture goes more distant or the force increases. In our experiment, the reward function is designed artificially as follows:

$$\begin{aligned} r(s) = & -\lambda_1 \|P_r\|_2 - \lambda_2 \|P_{rxy}\|_2 - \lambda_3 \|O_r\|_2 - \\ & \lambda_4 \|F_r\|_2 - \lambda_5 \|M_r\|_2 - \lambda_6 \cdot \text{Safety}, \end{aligned} \tag{3}$$

where

$\lambda_i,\ i = 1, 2, \ldots, 6$	Scale factors, hyperparameters
P_r	Relative position
P_{rxy}	Relative position in vertical plane
O_r	Relative orientation
F_r	Force
M_r	Torque
Safety	Punishment for unsafe actions

The P_r, P_{rxy}, and O_r terms measure the relative position of peg and hole. This term guides the assembly process when the workpieces are far away and the agent learns to move them together. During this coarse-grained approaching phase of assembly, the alignment of axes on peg and hole is critical for further assembly of contacting phase, while the distance along the axis is comparably tolerable. As the result, we repeat P_{rxy} to strengthen the alignment of axes although P_r has included all information. In contacting phase, P_r term pushes the progress of assembly and the signals from force sensors F_r and M_r guide the agent to avoid pressing between peg and hole.

4.2.4. DDPG-based policy model

Each decision of agent is a Markov decision process. The total return in the Markov decision process is defined as:

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i) \tag{4}$$

where $\gamma \in [0, 1]$ is the discount factor. The discount factor is introduced here because of the randomness of the environment. Taking the same action may not necessarily jump to the same state, thus, a discount should be applied to future rewards.

Take expectation of the reward (4), then we get the action value function (Q-function) to evaluate each possible action as follows:

$$Q(s_t, a_t) = E[R_t | s_t, a_t] \tag{5}$$

From the recurrence of the Bellman equation:

$$Q(s_t, a_t) = E\left[r(s_t, a_t) + \gamma E[Q(s_{t+1}, a_{t+1})]\right] \tag{6}$$

If the target policy is deterministic, in other words, the policy is mapped directly from state to action, rather than to a probability distribution, then the target policy can be described as a function: $\mu : S \rightarrow A$. Thus, the expectation symbol can be taken away as follows:

$$Q(s_t, a_t) = E[r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}))] \tag{7}$$

Essentially, the DDPG algorithm is using neural networks to fit a policy network (Actor) and a value network (Critic). The policy network corresponds to the policy function $\mu(s_t)$ above, and the value network corresponds to the value function $Q(s_t, a_t)$ above, hence it is also known as a Q network. Off-policy data and Bellman's equation are used to train the value network. Then the gradient of the value network is used to train the policy network.

4.2.5. Learning of policy model

The Bellman equation describing the action value function is stated as follows:

$$\begin{aligned} Q^*(s_t, a_t) = & E\Big[r(s_t, a_t) \\ & + \gamma \max_{a_{t+1}} Q^*\left(s_{t+1}, a_{t+1} = \mu(s_{t+1})\right)\Big] \end{aligned} \tag{8}$$

The Bellman Eq. (8) is the basis for learning the value function $Q^*(s_t, a_t)$. Since the optimal action value function cannot be obtained through iterative optimization, a neural network $Q_\phi(s_t, a_t)$ with parameters ϕ is constructed here to fit the value function. To determine the proximity to the Bellman equation, the Mean Squared Bellman Error (MSBE) function can be used as the loss function as follows:

$$\begin{aligned} L(\phi) = & E\Big[\Big(Q_\phi(s_t, a_t) - \big(r(s_t, a_t) \\ & + \gamma \max_{a_{t+1}} Q_\phi\left(s_{t+1}, a_{t+1} = \mu(s_{t+1})\right)\big)\Big)^2\Big] \end{aligned} \tag{9}$$

In algorithms containing deep Q networks, the process of learning the value function is primarily to minimize this MSBE loss function.

Before DQN, it was widely believed that learning value functions using large neural networks was difficult and unstable. Innovation in two areas is taken to solve the problem, which is also borrowed by DDPG.

Algorithm 1 DDPG-based assembly algorithm

```
Initialize parameters of policy network θ , Q network φ
Copy parameters to target networks φ → φ_targ, θ → θ_targ
Initialize empty replay buffer
for episode=1:MAX_EPISODES do
    Reset the robot and observe the state
    for t=1:MAX_EP_STEPS do
        Actor selects the action a_t = μ_θ(s_t) + noise
        The robot take action a_t, return the reward r_t and new state s_{t+1}
        Actor stores (s_t, a_t, r_t, s_{t+1}) into replay buffer
        Random sample from replay buffer with size N, denoted as (s_i, a_i, r_i, s_{i+1})
        Calculate Q value with target network
        y_i = r_i + γ Q_{φ_targ}(s_{i+1}, μ_{θ_targ}(s_{i+1}))
        Update the parameters of Q network using gradient descent
        ∇_φ (1/N) Σ_i (Q_φ(s_i, a_i) − y_i)^2
        Update the parameters of policy network using gradient ascend
        ∇_θ (1/N) Σ_i Q_φ(s_i, μ_θ(s_i))
        Update the parameters of target Q network and target policy network with soft update
        φ_targ ← ρ φ_targ + (1 − ρ) φ
        θ_targ ← ρ θ_targ + (1 − ρ) θ
    end for
end for
```

- Adopting the idea of empirical replay. The samples collected during the training are stored in a replay buffer, which is used to train the network offline.
- Training with an additional target network as follows:

$$y_t = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q_\phi\left(s_{t+1}, a_{t+1} = \mu(s_{t+1})\right) \tag{10}$$

If the network generating the target depends on the parameters of the current Q network: ϕ, the training process will be unstable. So the target Q network uses a set of parameters close to ϕ, denoted as ϕ_{targ}, and the updates of ϕ_{targ} have a time delay. The target Q network copies the parameters from the main network every certain fixed step and updates them in the following manner as follows:

$$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1 - \rho)\phi \tag{11}$$

where $\rho \in (0, 1)$ representing soft update, is a hyper-parameter that usually tends to 1.

It is difficult to find the target action by calculating the maximum Q value in continuous action space, and DDPG approximates this problem by using a deterministic target policy network ($\mu_{\theta_{\text{targ}}}(s_{t+1})$) to decide actions. The parameters of the target policy network do not participate in gradient optimization. With the target network, the loss function in the Q learning part of the DDPG can be reformulated as follows:

$$\begin{aligned} L(\phi) = & E\Big[\Big(Q_\phi(s_t, a_t) - \big(r(s_t, a_t) + \\ & \gamma Q_{\phi_{\text{targ}}}\left(s_{t+1}, a_{t+1} = \mu_{\theta_{\text{targ}}}(s_{t+1})\right)\big)\Big)^2\Big] \end{aligned} \tag{12}$$

The learning of policy function is relatively simple. Since the action space is continuous, and the Q-function is assumed to be differentiable with respect to the action variable, learning the policy function is training with gradient ascend.

$$\max_\theta E[Q_\phi(s_t, \mu_\theta(s_t))] \tag{13}$$

With our experiments, we found that there is always some error between the digital twin model and the real scene. Thus, we did some optimizations accordingly and the overall training method is shown in Fig. 5.

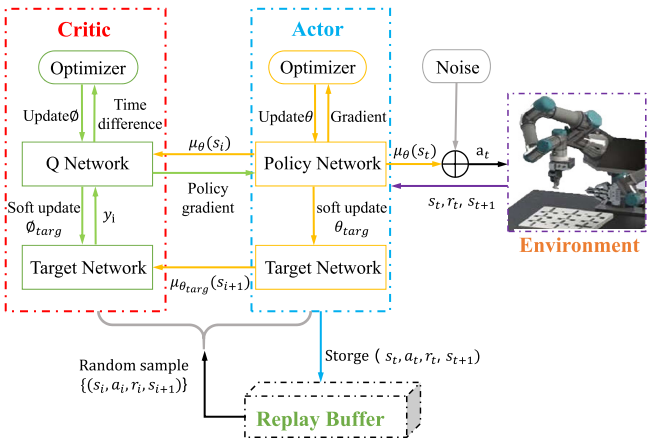

Fig. 5. Overall training procedure of reinforcement learning agent.

- Add some noise when the agent interacts with the simulation engines.
- Replace the simulation engines with the physical assembly system when the agent has learned basic skills.

4.3. Online monitoring with digital twin model

As the integration level of the factory increases, the amount of data collected by sensors is also increasing, and how to utilize the data is a pressing issue. We suggest in this article to handle the problem with the digital twin model. The digital twin model provides a visualization tool to present the raw data for human operators friendly. More importantly, with the analysis in virtual space, risky operations can be distinguished before they are sent to physical assembly lines. This character of digital twin simulation is crucial in the assembly process because precision problems could cause some damages to workpieces and actuators.

Collision is commonly a potential risk in flexible assembly systems. A large number of sensors are integrated with modern robots, however, it is still quite difficult for robots to perceive their surroundings. Robot manufacturers have done a lot of work for collision protection and they can be divided into two categories, safe space-based solutions, and force sensor-based solutions. Safe space refers to the concept that the robot end is prohibited to reach some artificially designed cubes where there may contain some other objects. The most commonly used approach is Axis-Aligned Bounding Box(AABB) algorithm. This is a fast but rough algorithm that prevents the robot from moving flexibly and efficiently. However, more accurate collision detection is difficult to integrate into robot controllers due to some reason. On the one hand, it is difficult to briefly and effectively describe the shape and motion of surrounding objects. On the other hand, precise collision detection algorithms consume much CPU resources. Force sensor-based solution check the torque on each joint periodically. If the joint torque increases abnormally, then a collision could have occurred and the movement should be stopped immediately. However, the robot may have caused some damages to the workpieces or itself. In factories, the surrounding of robots is very complex and changes frequently. The issue has become a major impediment to improve the flexibility of the assembly procedure to a higher stage.

With the digital twin model of a factory, information of assembly robots and their surrounding objects, including their position and motion, can be collected to a centralized computation center, as shown in Fig. 3. The digital twin model is a precise, comprehensive, informative, and dynamic reflection of the factory. With powerful computers, we applied modern anti-collision algorithms with the digital twin model to predict collision. Digital twin-based collision detection can provide a larger zoom for further procedure flexibility from the safety perspective.

The inaccuracy of positioning may cause failures in assembling as well, for example, workpieces may be pressed together wrongly. Even though the pressure does not reach the emergency thresholds, some fragile and plastic components may have been damaged. Building a digital twin model can also predict the failure of this type. When the force and position sensing information collected from the assembly lines is different from the digital twin's simulation data, it can be assumed that there may be a positioning error at the beginning of the assembly.

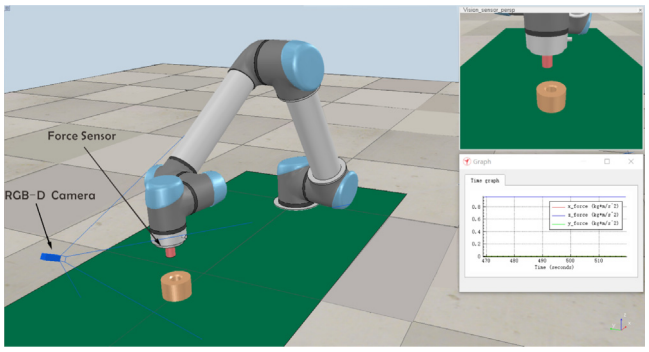

Fig. 6. Digital twin model of the peg-in-hole assembly workspace.

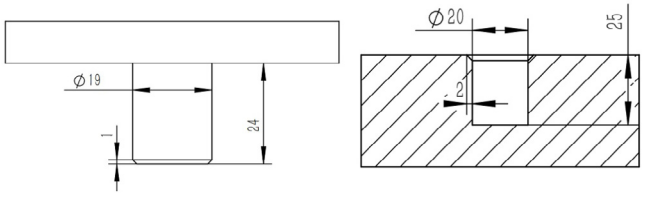

Fig. 7. Engineering drawings of peg and hole.

5. Experiments

To verify the effectiveness of the system we proposed, especially the reinforcement learning-based assembly algorithm, a peg-in-hole assembly experiment is carried out.

The task of this experiment is controlling the UR5 robot and completing assembly from any random initial position. The peg is connected to the end of the robot and the hole is connected to the workbench rigidly. During the whole experiment, monitoring system operates to avoid any collision.

5.1. Experimental model

The digital twin model is built with CoppeliaSim as shown in Fig. 6. 1.2 m × 0.6 m of the green square area represents the workbench. The base of the UR5 robot is fixed in the world coordinate system at the origin $(0, 0, 0)$ (0.5 m from the geometric center of the workbench). The circular hole is fixed at $(-0.6, 0, 0)$, and the axis of the hole is oriented vertically along the z axis. The RGB-D camera is fixed and placed at $(-0.88, 0, 0.22)$. The angle of the camera is manually adjusted so that the hole and the peg are always in the field of view during the assembly process. The sampling accuracy is set to 128 × 128 pixels. The physical objects are laid out in the same position as the digital twin model. The workpieces are made of aluminum with Yong's modulus of 70 GPa, and the detailed size is described with Fig. 7.

Physical restrictions are parameterized as well. The end of the robot is restricted to a hemisphere whose center is the hole and whose radius

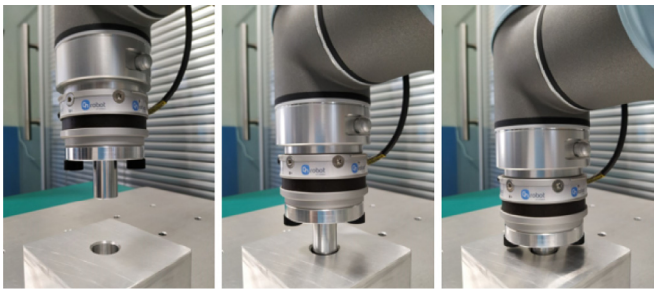

Fig. 8. The peg-in-hole assembly experiment.

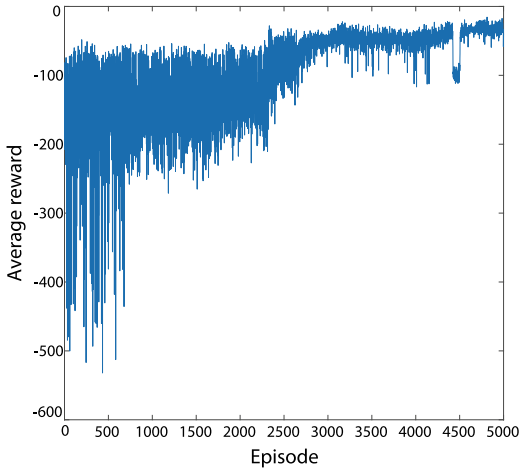

Fig. 9. Digital twin model of the peg-in-hole assembly workspace.

is 300 mm. The force restriction on the end is 30N. Once the agent has acted over the safety limits, or further collision is indicated by monitoring system, the current training iteration will be marked unsafe and terminated.

5.2. Training and results

Adam optimizer is used to train the network with the learning rates for Actor and Critic set to 1×10^{-4} and 1×10^{-3} respectively. For the Q-function, the discount factor is $\gamma = 0.99$. For the soft update, we use $\rho = 0.99$ and update the target network every 32 iterations. The hyperparameters are from grid search, and the model parameters are trained with gradient ascent.

The Actor network contains 2 hidden layers. Each hidden layer has 40 neurons. The activation function of the hidden layers is "relu" while the activation function of the output layer is "tanh".

The Critic network contains 4 hidden layers. Each hidden layer has 40 neurons. The activation function of the first three hidden layers is "relu", while the activation function of the fourth hidden layer is "tanh".

The size of DDPG's minibatch is 64, and the maximum number of steps per round (iteration) is 100. The size of the replay buffer is 50,000, which means that at least 500 rounds are required to fill the buffer at the beginning of training.

The training process is shown with Fig. 9. In the first 2500 training episodes, the reinforcement learning agent hardly ever succeeds. Since about 2500 episodes, the agent has learned basic assembly skills. When the agent is almost capable of assembly with simulation from digital twin, assembly system from the real world is introduced at around 3000 episodes. An oscillation is observed due to the changing of the simulation engine. And finally, the model converges within 5000 episodes and is ready for further validation.

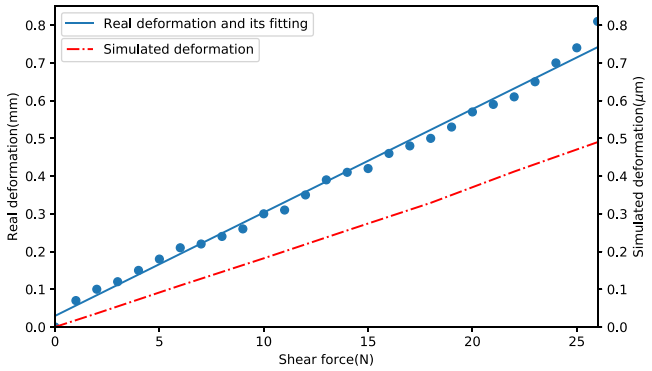

Fig. 10. Stiffness curves of simulated and real peg-in-hole assembly system.

Experiments on simulated assembly system reports the success rate of over 90%. Then we discuss the metric of success rate for real assembly. Considering the existence of monitoring system, when an assembly attempt is detected as failed or unsafe (could be collision, force/torque out of range, or robot's end out of working space), the robot moves to its initial position and try again until it succeeds. Therefore, when taking the agent and monitoring as a whole system, the efficiency drops due to the detected failures but the success rate is always 100%. To validate the efficiency of the system, we would report the success rate of the agent rather than the system in the real scenario. Under this metric, validation in real scenarios also reports a success rate of over 90%. And the assembly process with physical assembly equipment is shown in Fig. 8.

Ablation studies are conducted to validate the key components of this system. In the reinforcement learning framework, we replace the reward function with naive sparse reward and the training process does not converge. For the sensors, when camera and force sensors are removed, the training process converges but the algorithm fails to assembly in the real world. When removing the digital twin and training with a real robot from the beginning, the converged algorithm reaches an equivalent successful rate but takes significantly more time to train.

Different sensors provide essential data in different assembly stages. In the searching stage, the force sensor does not output any meaningful information because there is no contact. However, when the peg contact with the hole, the force sensor guides the agent to assemble. The shifting of the agent's main data source, from vision to force, is smooth, indicating that deep reinforcement algorithm is capable to integrate different policies on different assembly stages.

5.3. Precision analysis of naive simulation

In order to understand the necessity of digital twin as a better simulation, we studied the stiffness curves of naive statistic engine and real assembly system, then show the difference in Fig. 10.

The stiffness is a proportional factor of the data obtained from the robot's force sensor and position sensor when the peg and hole are in contact. It reflects the quality of the simulation system. When the peg is tangent to the hole and the contact surface is a straight line, we apply a force at the contact and measure the corresponding stiffness of this situation. The result is shown in Fig. 10.

The stiffness of the simulated system is a lot bigger than that of the real, indicating that naive simulation has poor precision. This experiment also explains why agents trained in a naive simulation environment sometimes fail in a real environment. Thus, introducing digital twin as a better simulation is crucial.

At last, we would like to explain why there is a difference between the stiffness curves. We assumed the robot, grubbers, and connections to be rigid when modeling to reduce the computational consumption

of the simulation engine. However, in real scenarios, these objects are deformable and thus enlarge the stiffness of the system. Besides, the fitting curve of real stiffness does not start from zero. A possible reason is that only a contact point, rather than a line, is formed at the beginning.

6. Conclusion

To improve the flexibility of assembly lines, we proposed a deep reinforcement learning and digital twin based approach in this article. The main difficulties of flexible assembly are motion planning, precision, and safety. Firstly, we designed a deep reinforcement learning model that can perform the process of end-to-end assembly. The model is trained with the digital twin of the assembly system to balance training efficiency and precision. Considering the safety issue in factory, especially when coping with deep learning, we built a monitoring system based on the digital twin. The experiment shows the efficiency of our approach.

Digital twin provides a framework for applying deep learning in industry by enhancing efficiency and reliability. It is suggested to promote this framework into more sectors of smart manufacturing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Canedo, A. (2016). Industrial IoT lifecycle via digital twins. In *Proceedings of the eleventh IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis* (p. 1).

Cecil, J., Albuhamood, S., Cecil-Xavier, A., & Ramanathan, P. (2019). An advanced cyber physical framework for micro devices assembly. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(1), 92–106.

De Morais, G. A., Marcos, L. B., Bueno, J. N. A., de Resende, N. F., Terra, M. H., & Grassi Jr, V. (2020). Vision-based robust control framework based on deep reinforcement learning applied to autonomous ground vehicles. *Control Engineering Practice*, *104*, Article 104630.

Fang, X., Pang, D., Xi, J., & Le, X. (2019). Distributed optimization for the multi-robot system using a neurodynamic approach. *Neurocomputing*, *367*, 103–113.

Fang, Y., Peng, C., Lou, P., Zhou, Z., Hu, J., & Yan, J. (2019). Digital twin based job shop scheduling toward smart manufacturing. *IEEE Transactions on Industrial Electronics*, *15*(12), 6425–6435.

Fleming, P. J., & Purshouse, R. C. (2002). Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, *10*(11), 1223–1241.

Grieves, M., & Vickers, J. (2017). Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary perspectives on complex systems* (pp. 85–113). Springer.

Gu, S., Lillicrap, T., Sutskever, I., & Levine, S. (2016). Continuous deep Q-learning with model-based acceleration. In *International conference on machine learning* (pp. 2829–2838).

Inoue, T., De Magistris, G., Munawar, A., Yokoya, T., & Tachibana, R. (2017). Deep reinforcement learning for high precision assembly tasks. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 819–825).

Jasim, I. F., & Plapper, P. W. (2014). Contact-state monitoring of force-guided robotic assembly tasks using expectation maximization-based Gaussian mixtures models. *International Journal of Advanced Manufacturing Technology*, *73*(5–8), 623–633.

Kolodziejczyk, W., Zoltowska, I., & Cichosz, P. (2021). Real-time energy purchase optimization for a storage-integrated photovoltaic system by deep reinforcement learning. *Control Engineering Practice*, *106*, Article 104598.

Le, X., Yan, Z., & Xi, J. (2017). A collective neurodynamic system for distributed optimization with applications in model predictive control. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *1*(4), 305–314.

Leng, J., Yan, D., Liu, Q., Xu, K., Zhao, J. L., Shi, R., Wei, L., Zhang, D., & Chen, X. (2020). ManuChain: Combining permissioned blockchain with a holistic optimization model as bi-Level intelligence for smart manufacturing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(1), 182–192.

Li, F., Jiang, Q., Zhang, S., Wei, M., & Song, R. (2019). Robot skill acquisition in assembly process using deep reinforcement learning. *Neurocomputing*, *345*, 92–102.

Li, C., Mahadevan, S., Ling, Y., Choze, S., & Wang, L. (2017). Dynamic Bayesian network for aircraft wing health monitoring digital twin. *American Institute of Aeronautics and Astronautics*, *55*(3), 930–941.

Li, S., Wu, Y., Cui, X., Dong, H., Fang, F., & Russell, S. (2019). Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI conference on artificial intelligence (Vol. 33)* (pp. 4213–4220).

Liu, S. X. (2016). Innovation design: Made in China 2025. *Design Management Review*, *27*(1), 52–58.

Liu, S., Li, Y.-F., Xing, D.-P., Xu, D., & Su, H. (2017). An efficient insertion control method for precision assembly of cylindrical components. *IEEE Transactions on Industrial Electronics*, *64*(12), 9355–9365.

Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, *6*, 1–10.

Luo, J., Solowjow, E., Wen, C., Ojea, J. A., & Agogino, A. M. (2018). Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 2062–2069).

Luo, J., Solowjow, E., Wen, C., Ojea, J. A., Agogino, A. M., Tamar, A., & Abbeel, P. (2019). Reinforcement learning on variable impedance controller for high-precision robotic assembly. In *International conference on robotics and automation* (pp. 3080–3087).

Ng, A. Y., Harada, D., & Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. *International conference on machine learning* (pp. 278–287).

Pi, C.-H., Hu, K.-C., Cheng, S., & Wu, I.-C. (2020). Low-level autonomous control and tracking of quadrotor using reinforcement learning. *Control Engineering Practice*, *95*, Article 104222.

Roveda, L., Pedrocchi, N., Beschi, M., & Tosatti, L. M. (2018). High-accuracy robotized industrial assembly task control schema with force overshoots avoidance. *Control Engineering Practice*, *71*, 142–153.

Schleich, B., Anwer, N., Mathieu, L., & Wartzack, S. (2017). Shaping the digital twin for design and production engineering. *CIRP Annals*, *66*(1), 141–144.

Song, K.-T., & Chu, T.-S. (1998). Reinforcement learning and its application to force control of an industrial robot. *Control Engineering Practice*, *6*(1), 37–44.

Sun, Y., Wang, W., Chen, Y., & Jia, Y. (2020). Learn how to assist humans through human teaching and robot learning in human-robot collaborative assembly. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1–11.

Tao, F., Sui, F., Liu, A., Qi, Q., Zhang, M., Song, B., Guo, Z., Lu, S. C.-Y., & Nee, A. (2019). Digital twin-driven product design framework. *International Journal of Productions Research*, *57*(12), 3935–3953.

Tao, F., Zhang, H., Liu, A., & Nee, A. Y. (2018). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Electronics*, *15*(4), 2405–2415.

Tuegel, E. J., Ingraffea, A. R., & Eason, T. G. (2011). Reengineering aircraft structural life prediction using a digital twin. *International Journal of Aerospace Engineering*, 1687–5966.

Vachálek, J., Bartalskỳ, L., Rovnỳ, O., Šišmišová, D., Morháč, M., & Lokšík, M. (2017). The digital twin of an industrial production line within the industry 4.0 concept. In *International conference on process control* (pp. 258–262). IEEE.

Weyer, S., Meyer, T., Ohmer, M., Gorecky, D., & Zühlke, D. (2016). Future modeling and simulation of CPS-based factories: an example from the automotive industry. *IFAC-Papersonline*, *49*(31), 97–102.

Yu, Z., & Dexter, A. (2010). Online tuning of a supervisory fuzzy controller for low-energy building system using reinforcement learning. *Control Engineering Practice*, *18*(5), 532–539.

Zhang, F., Leitner, J., Milford, M., & Corke, P. (2017). Modular deep Q networks for sim-to-real transfer of visuo-motor policies. In *Proceedings of the Australasian conference on robotics and automation 2017* (pp. 1–10).

Zhang, K., Shi, M., Xu, J., Liu, F., & Chen, K. (2017). Force control for a rigid dual peg-in-hole assembly. *Assembly Automation*, *37*(2), 200–207.

Zhang, K., Xu, J., Chen, H., Zhao, J., & Chen, K. (2019). Jamming analysis and force control for flexible dual peg-in-hole assembly. *IEEE Transactions on Industrial Electronics*, *66*(3), 1930–1939.

Zhu, L., Cui, Y., Takami, G., Kanokogi, H., & Matsubara, T. (2020). Scalable reinforcement learning for plant-wide control of vinyl acetate monomer process. *Control Engineering Practice*, *97*, Article 104331.