Advanced Engineering Informatics 47 (2021) 101209

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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

Evolutionary digital twin: A new approach for intelligent industrial product development

Ting Yu Lin [a,b,c], Zhengxuan Jia [a,b,c], Chen Yang [d,*], Yingying Xiao [a,b,c], Shulin Lan [e], Guoqiang Shi [a,b,c], Bi Zeng [a,b,c], Heyu Li [a,b,c]

[a] State Key Laboratory of Complex Product Intelligent Manufacturing System Technology, Beijing Institute of Electronic System Engineering, Beijing, PR China
[b] Beijing Complex Product Advanced Manufacturing Engineering Research Center, Beijing Simulation Center, Beijing, PR China
[c] Science and Technology on Special System Simulation Laboratory, Beijing Simulation Center, Beijing, PR China
[d] School of Computer Science and Technology, Beijing Institute of Technology, Beijing, PR China
[e] School of Economics and Management, University of the Chinese Academy of Sciences, Beijing, PR China

ARTICLE INFO

Keywords:
Evolutionary digital twin
Intelligent industrial product
Collaborative evolution
Approximate world
Multiple cyber spaces
Simple evolution paradigm
Model evolution paradigm

ABSTRACT

To fulfill increasingly difficult and demanding tasks in the ever-changing complex world, intelligent industrial products are to be developed with higher flexibility and adaptability. Digital twin (DT) brings about a possible means, due to its ability to provide candidate behavior adjustments based on received "feedbacks" from its physical part. However, such candidate adjustments are deterministic, and thus lack of flexibility and adaptability. To address such problem, in this paper an extended concept – evolutionary digital twin (EDT) and an EDT-based new mode for intelligent industrial product development has been proposed. With our proposed EDT, a more precise approximated model of the physical world could be established through supervised learning, based on which the collaborative exploration for optimal policies via parallel simulation in multiple cyberspaces could be performed through reinforcement learning. Hence, more flexibility and adaptability could be brought to industrial products through machine learning (such as supervised learning and reinforcement learning) based self-evolution. As a primary verification of the effectiveness of our proposed approach, a case study has been carried out. The experimental results have well confirmed the effectiveness of our EDT based development mode.

1. Introduction

A new round of global scientific and technological revolution and industrial revolution is coming and will bring about a subversive impact on industries around the world. In particular, the rapid development of new generation information and communication technologies (ICTs) such as 5G, Internet of Things, cloud computing, big data, as well as artificial intelligence (AI) is reinforcing the connection between human, computing machine, and the physical world through data and information to promote the intelligence of industrial products to higher levels defined in literature [1]. The industrial product with the capability of cognition, cooperation and quick adaption to the complex changing world is called an intelligent industrial product (IntelliIndusProd) in this paper. Such IntelliIndusProds are of increasing importance nowadays and in the future, as our systems are required to accomplish diversified and complex tasks more autonomously in complex world that has the characteristics of partially observable, non-cooperative, and dynamically changing. The core of nuclear power plant, multi-AGVs of logistics system, and multi-vehicles of vehicle networking [2,3,4] are all good examples of such kind of industrial products which should run autonomously in a rapidly changing environment through dynamic gaming based on part of the information.

However, the R&D of IntelliIndusProds is much more difficult than that of traditional industrial products, as they are conceived to work in harsh complex working conditions and accomplish demanding (multiple) missions. On one hand, nonlinearity, uncertainty, self-organization and emergence pose great difficulty and complexity for theoretical modeling; on the other hand, tactic and model design for IntelliIndusProds under complex environments becomes even harder for engineers, due to limited human capability on precise abstraction and rapid cognition of high-dimensional information. It is arduous to develop an IntelliIndusProd in a short time with enough flexibility and adaptability through theoretical modeling, which relies only on human knowledge and experience.

* Corresponding author.
E-mail address: yangchen666@bit.edu.cn (C. Yang).

https://doi.org/10.1016/j.aei.2020.101209
Received 10 January 2020; Received in revised form 7 October 2020; Accepted 16 November 2020
Available online 1 January 2021

Digital twin technology which integrates the cyber space and the physical space [5] has the potential to increase the flexibility, adaptability and intelligence of the industrial product. First, virtual product (digital model) could better perceive the real world through the real-time "feedback" from its physical product (counterparts); second, virtual product could synthesize the sensing data and provide the physical product with better action policies. However, such digital twin could hardly solve the above problem effectively. First, faced with demanding tasks, it may be difficult to construct such a virtual product (only by human knowledge) that could fully reflect its physical counterpart's characteristics, due to the nonlinearity, uncertainty, self-organization and emergence of the complex world. For example, during establishment of the virtual product of a physical robot arm, the working environment of such robot arm, including the production line layout, the process to be executed, etc., is usually dynamic, with perturbation and drastically changing or even unknown for different tasks, and thus extremely difficult to model in advance. Second, the existence of various working conditions makes it difficult to pre-design a policy-making module that can generate the effective action policies for the physical product. Taking again the robot arm for example, as the motion control law is established through human experience based on the virtual product, when faced with totally different tasks, the control law may need to be re-designed, and such process normally takes several months.

Therefore the concept of digital twin needs to be further developed and extended to enhance its self-learning, self-adaptation and self-growing capability based on machine intelligence.

First, it is necessary for us to establish multiple cognitive models of the real world, called the approximate worlds, which keep approximating the different scenes of the real world through continuous learning. Through the approximate worlds, IntelliIndusProd could fully understand the real world, and could explore different scenes in parallel and in a low-cost, risk-free and super real-time way when it has to deal with the uncertainty in the partially observable, non-cooperative, and dynamically changing world.

Second, it is necessary to train the virtual product to adapt to approximate worlds. With the approximate worlds gradually approaching the real world, the self-growing virtual product could finally adapt to the real world and give better action policies for the physical product. In addition, we could train multiple virtual products in parallel in a single approximate world which could generate more data through simulation to speed up the exploration process. In other words, the new kind of digital twin is an evolving digital twin equipped with multiple cyber spaces, namely multiple approximate worlds, where the virtual product could evolve its behavior and policy to better approximate the physical product and make better decisions.

This paper proposes an Evolutionary Digital Twin (EDT) approach for the IntelliIndusProd development, and is organized as follows: Section 2 presents related work; Section 3 and 4 introduces the EDT concept and the architecture of an EDT; Section 5 discusses the concrete establishment of a product design oriented EDT in details. Section 6 describes an application case study and discusses related results. Section 7 concludes the paper with future work.

2. Related work

2.1. Simulation driven product design

The innovation of digital design and manufacturing is the key to this new round of industrial revolution [23]. Currently, digital methods for product development can be used to quickly define and design a prototype, not only in terms of product structure, but also product function and behavior [24]. The multi-disciplinary virtual prototype could support unified modeling and collaborative simulation across different disciplines to evaluate and optimize virtual products in the cyber space [25]. Model-based system engineering could establish the entire lifecycle model of a product through multiple views, supporting the continuous evolution and verification of the product [26]. An advanced parallel simulator under multi-core environments is proposed to address the challenges of collaborative simulation of complex variable-structure systems that exhibit changes both at structural and behavior levels, with increasingly big and complex models [31]. Recently, digital twin-driven product design approaches has been proposed to optimize the design of the virtual product based on the data feedback from the physical product [6]. However, the above approaches mainly depend on human knowledge and abstraction capability without better utilizing the power of new AI algorithms and models.

2.2. Digital twin and parallel control

The first formal definition of digital twin dates back to 2012, given by Glaessgen and Stargel [5] in NASA. A digital twin is an integrated multi-physics, multiscale, probabilistic simulation of an as-built system that uses the best available physical models, sensor updates, history data, etc., to mirror the life of its corresponding physical product [5]. Rosen et al. [8] gives another definition from the perspective of autonomous systems which are required to be able to respond rapidly to unexpected events without central re-planning. According to [8], a digital twin is an ultra-realistic model that reflects the state of the process and the behavior of the autonomous system in interaction with its environment in the real world.

From the above definitions, it could be summarized that a digital twin is an ultra-realistic model of the physical system, which is established in different scales, synthesizing knowledge, data, and physical model from different domains and equipment. Such a definition, in fact, emphasizes a lot on the establishment of model validity which is the core value of digital twin, but is difficult to achieve in engineering, though related concepts, architectures [9], and even key technologies have already been proposed. This is mainly due to the fact that human knowledge is limited facing the complex real world. Thus starting from current human cognition of the world with traditional theories, we could scarcely build such a digital twin that could reproduce 100% the behavior and characteristics of its physical product, especially the details. However, when complex systems are taken into consideration, a minor mistake in such details could result in huge differences in behavior. Furthermore, to our knowledge, most of the research on digital twin concentrates on optimizing the performance of the physical product through human knowledge based optimization of the virtual product [8,10]. Similar to the modelling precision limit totally based on human knowledge, such optimization ability is also limited.

Faced with similar difficulty in the control law design domain, Feiyue et al. [11] proposed the parallel control theory. In this theory, they proposed constructing an equivalent artificial system (not exactly the same as the physical one) that works in parallel with the real world system. In this artificial system, planning and optimization algorithms could be designed and experimented for better control laws. Meanwhile, the behavior of this artificial system is further corrected by the data collected from the real system. Although such parallel control theory is promising and potentially effective in control law design where the core purpose is to eliminate deviations, it might be less applicable in the cases where policies should be made based on human knowledge or precise knowledge about the state or evolution dynamics of the system, for example the autonomous collaborative robot arm pairs on the production line. Even so, the concept of parallel control has provided good insights to the design of our EDT.

2.3. Reinforcement learning based design

Recent years have witnessed rapid development of machine learning, especially deep learning and deep reinforcement learning. For example, in 2016, AlphaGo designed by DeepMind of Google mastered the game of Go and easily defeated the world champion Sedol Lee and Jie Ke by big scores [12]. With this victory, DeepMind published their research on

development of AlphaGo on Nature, followed by another one on AlphaZero which mastered the game of Go without human supervision. In 2019, faced with the more challenging task Starcraft II, AlphaStar [13] developed by DeepMind defeated again professional human players with score 10:1. From the details revealed in the articles [13], it could be seen that provided with perfectly described interaction environment and task targets, the agent could converge to behavior strategies largely superior than those manually developed, or even to optimal strategies, through autonomous learning. Hence, reinforcement learning based design pattern starts to attract increasing attention both in industry and academy, and is being applied to designs in different domains.

Among these application domains, the design of robot control algorithms has been widely studied and has been practically applied in the control of real robots. Asada et al. [14] developed a ping-pong player robot based on Q-learning algorithm [16]. Their developed robot control algorithm could drive the robot to beat the ball to required positions based only on visual information. Deisenroth et al. [15] proposed a model based policy search method to train the robot to accomplish building blocks task. Although the application systems in the above work were real robot machines, the robot control commands were constrained and limited by human scripted rules when applied. This may hinder better policy discovery in complicated tasks, as human knowledge or cognition is, to some extent, limited. Deep reinforcement learning based robot control design (without human scripted rules in control commands), on the other hand, could hardly be applied to real robots, and still stays in simulation stage [17,18]. Zhang et al. [19,20] applied DQN algorithm to train a three joint robot for grasping task in simulation environment. However, when applied to physical robot machine, the performance was less satisfactory due to the differences between the simulation environment and the real world. Similar problems also exist in others work [21].

Hence, for the purpose of addressing the above problems encountered in simulation driven product design, we propose the concept of EDT which allows the designed product to persistently and intelligently optimize itself in its whole lifecycle. With our proposed EDT, learning agents could be equipped with a model with high and gradually increasing confidence, which in turn converge to behavior policies with higher performance in real products.

3. EDT and persistent reinforcement based product design paradigm

3.1. Common product development process

The common product development process follows a step-by-step process which is mainly driven and led by human in three worlds including expected world, interpreted world and external world [24]. The product lifecycle could be roughly divided into 3 steps, as shown in Fig. 1: first, in the primary design step, according to the ideal world (corresponding to the expected world in [24]) which is based on the human understanding of the real world (corresponding to the external world in [24]), an ideal product model is established by means of mathematical and physical modeling in the cyber space. Second, in the detailed design, simulation and test step, an approximate world (corresponding to the interpreted world in [24]) of the real world is constructed through modeling and simulation, together with possible semi-physical simulation in verification. In this step, a digital or semi-physical prototype product based on the approximate world in the cyber space or the physical space is established. Finally, in the operation and service step, a real product is fabricated and released to provide services. In the traditional development mode, the function, performance and structure of the final product are fixed, with extremely limited flexibility and adaptability. Facing new scenarios and requirements, corrections or innovations from humans are needed, and then the above three steps needs to be repeated. Usually, each product development cycle will incur high cost and take a long time (several years in extreme cases), which cannot meet user's need for fast product delivery, upgrade and iterations. In other words, this product development mode cannot support the fast product innovation and development (within a relatively short timeframe) in the coming 4th industrial revolution era.

3.2. Connotation of EDT

To address the above issue, based on the industrial Internet, integrating the new generation ICT, the new generation AI technology and the product development field technology, the EDT provides a novel product development mode where different forms of learning and searching means in multiple parallel cyber spaces are introduced, which allows the IntelliIndusProd to evolve and possess better adaptability.

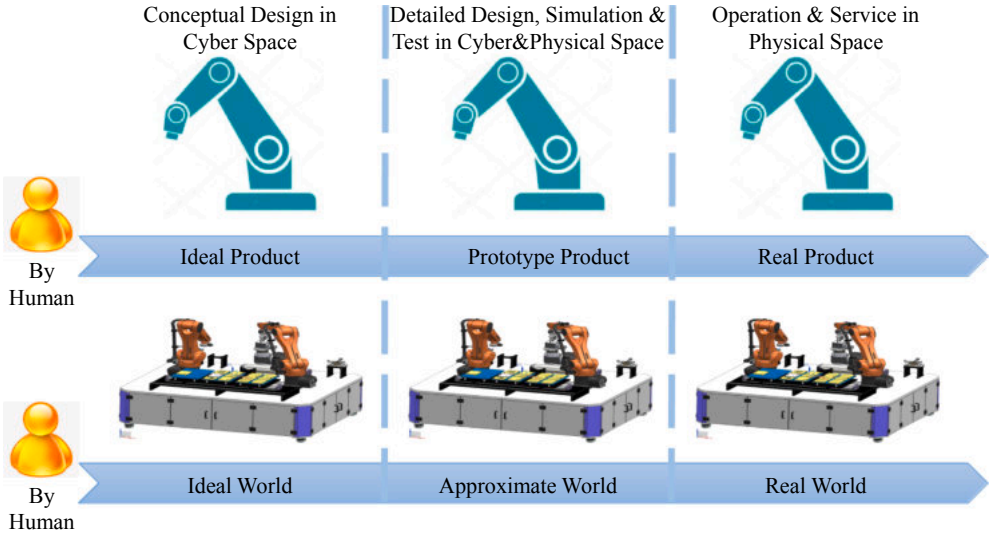

Fig. 1. Common process of product development.

The EDT, as an extension of DT, is also composed of the virtual and physical parts, namely the virtual product and physical product in terms of industrial product development. Mainly based on machine intelligence and supplemented by human intelligence, the EDT could help effectively establish the operation law under various uncertainties and gradually achieve a well approximated model of the real world. Such capability could well guide product design improvement and operation optimization, and hence support to develop IntelliIndusProds with enough flexibility and adaptability for the complex world and tasks.

The EDT, as shown Fig. 2, includes some notable features:

(1) Compared with the current digital twin, the EDT clearly builds an approximate world corresponding to the real world, which could focus and simulate some specific aspects (views) of the real world according to the R&D requirements. Besides, via EDT, such established approximate world can support the repeated tests and experiments of the designed products and evolve with the new product designs at the same time.
(2) Compared with the current digital twin whose cyber space and physical space are mapped through bijection, the EDT builds multiple cyber spaces to construct different models of the real world with uncertainties and of the product at different resolutions from different aspects (views) of the real world according to the development demands.
(3) Compared with the current digital twin which mainly use a cyberspace to predict the operation effects of the product, the EDT builds a development method using multiple cyber spaces for fast parallel learning and searching, which allows the approximate world to keep approaching the real world, and the product scheme to keep adapting to the real world.

3.3. A new product development paradigm

3.3.1. New process of product development

The process of new product development mainly using machine intelligence and assisted by human intelligence, is an evolving process based on the EDT, as shown in Fig. 3. First, if it belongs to a product family, evolution could be performed through learning from the data of similar products fed back by their EDT in the early step of design, which is similar to the in-use product [29]. Initial models of both the product and the approximate world may be different from or cannot fully reflect the real ones or the physical ones. However, through evolution and learning process, such differences can be gradually decreased. Moreover, together with such evolution process, action policies of the physical product should be optimized. Second, based on the EDT, the traditional simulation, test, operation and service would not be a separated process. Instead, this process is combined with the evolution of both the prototype product (virtual product) and the approximate world. Emphasizing on autonomous evolution, the above development process, however, does not negate the value of human and theoretical modeling which may also play an important role in extracting more value from the data on the contrary [7].

In this new product development process, there are two paradigms of digital twin evolution: simple evolution paradigm and model evolution paradigm, which are discussed in details as follows.

3.3.2. Simple evolution paradigm

In the simple evolution paradigm, the models & parameters of the product and the world are deterministic, the operation policies is the only adjustable factor. As shown in Fig. 4, at the beginning, the state of the approximate world is mapped from the real world's state. In multiple cyber spaces, the virtual product could execute actions from different operation policies, and bring the state changes in the corresponding approximate world. The process is a super real-time process. If the policy space is huge, searching and planning algorithms like Monte Carlo Tree Search technology would be applied to explore the feasible solution space in parallel in multiple cyber spaces. Finally, the policy used in the physical space can be optimized using the searching and evaluating results generated in the cyber space.

3.3.3. Model evolution paradigm

In the model evolution paradigm, the behavior policies are not the only variable for the virtual product of an EDT, the cognition models of the real world would also change due to the gradually increased information completeness and certainty.

To this end, as shown in Fig. 5, supervised learning and unsupervised learning could be adopted to construct different cognition models of the real world, supporting the evolution of the approximate world towards the real world, while reinforcement learning approach could be applied for policy model construction, allowing the search of effective and even optimal behavior policies through interaction between the virtual product and the approximate world.

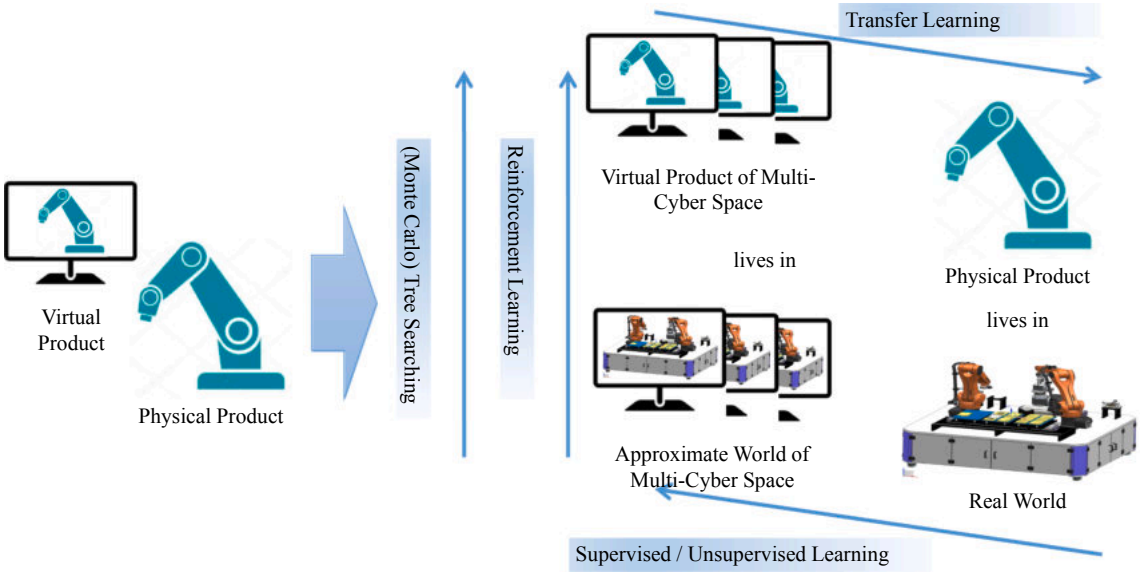

Fig. 2. Core features of the EDT.

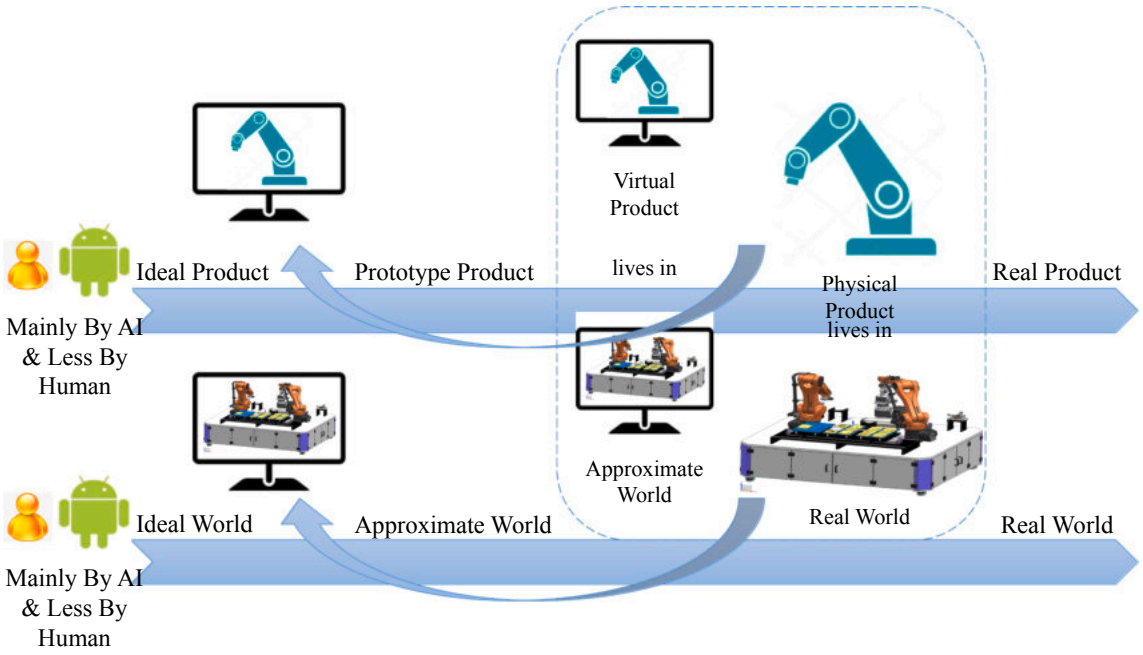

Fig. 3. New product development process.

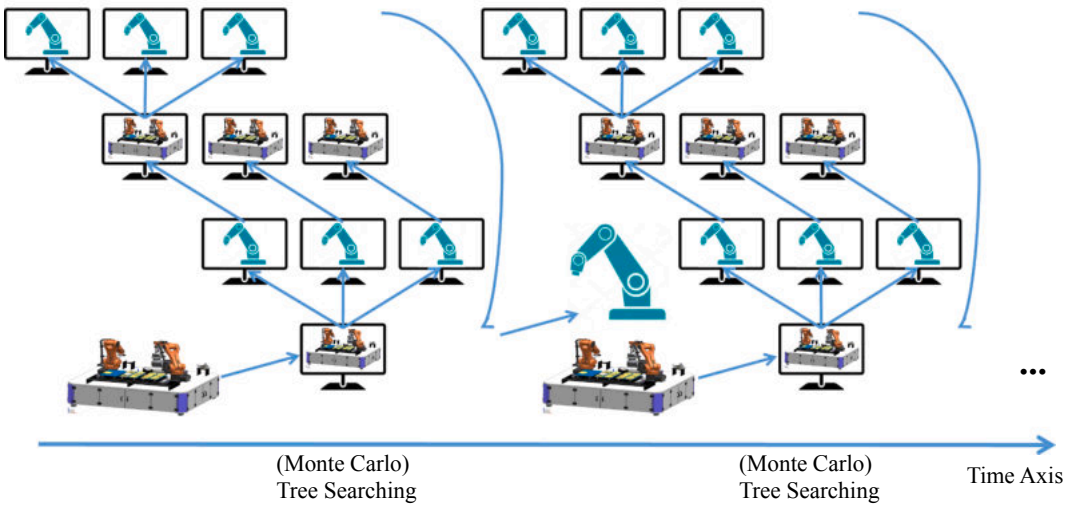

Fig. 4. Simple evolution paradigm.

However, differences still exist between the behavior of the virtual product and that of the physical product, due to the existence of errors in training, forming a gap between the cyber space (including the virtual product and the approximate world) and the physical space (including the physical product and the real world). Thus, to mitigate this gap, the transfer learning approach could be applied to develop a product with enough adaptability to the errors above, allowing the converged policy models to be applied in physical world.

4. EDT based intelligent industrial product development system architecture

4.1. Overview

The intelligence of the IntelliIndusProd is not possible without the effective fusion of big data, algorithm and computing power, for both supervised learning of the approximate world and the reinforcement learning of the virtual product.

(1) Big data: big data is not only collected from the physical space, but also generated from cyber space. (2) Algorithm: algorithm mainly operates in data analysis engine, intelligent optimization engine and machine learning engine. (3) Computing power: enough computing power is needed to support the data generation in the approximate world and the virtual product, and to support the costly operation of corresponding engines.

Furthermore, as the EDT contains multiple cyber spaces, much more computing power is needed to process big data from different cyber spaces. An IntelliIndusProd often runs on the industrial edge where computing resources and data processing capability are limited. Therefore, the development system should adopt the architecture that integrates the cloud and the edge computing power.

As shown in Fig. 6, for both simple evolution paradigm and model evolution paradigm, the edge would collect and pre-process the data of the physical product and the real world, and then feed the data back to the cloud; the cloud could support the supervised learning of multiple approximate worlds and the reinforcement learning of multiple virtual

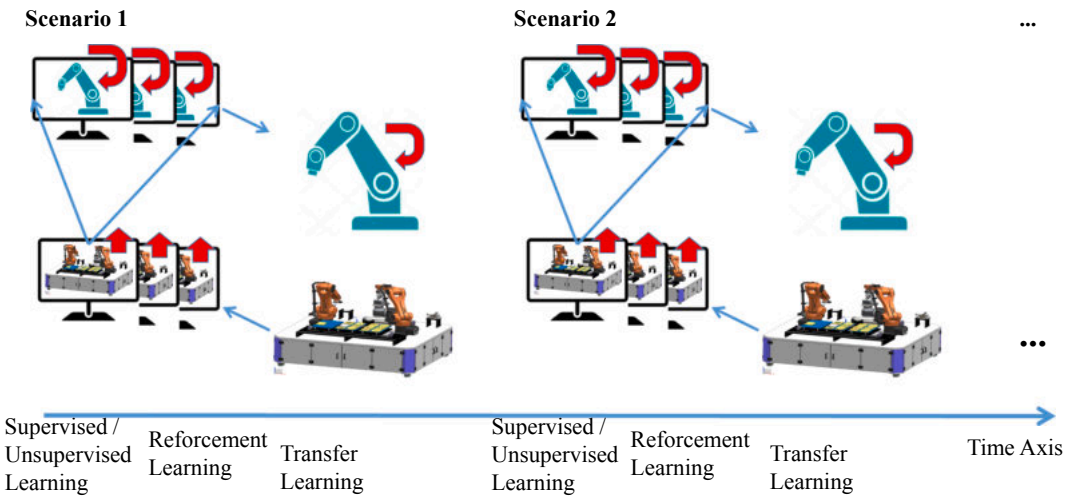

Fig. 5. Model evolution paradigm.

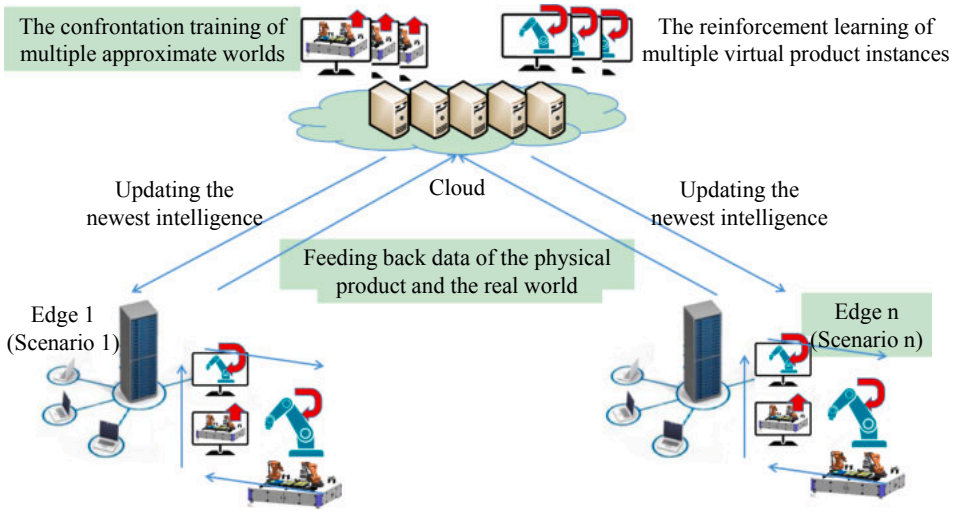

Fig. 6. Cloud-edge integrated architecture.

product instances efficiently. The edge would always update the latest intelligence to improve the flexibility and adaptability.

4.2. Hierarchical description

As shown in Fig. 7, the system architecture of the EDT based IntelliIndusProd development consists of several layers including Physical layer, Access/communication layer, Edge processing platform layer, Cloud development service platform layer and Application layer, along with two parts about Information security management, Standard and specification. The Cloud development service platform layer further includes Virtualizing layer, Cloud development service support layer and Portal layer. The details of this architecture are explained as follows.

(1) Physical layer

It not only includes the physical product and the real world, but also includes the related development resources and capabilities (such as computing power) required by the operation of multiple cyber spaces.

(2) Access/communication layer

It not only includes the access and communication of the edge to the physical product and the real world, but also includes the access and communication of the cloud to each edge.

(3) Edge processing platform layer

It not only supports the time-efficient operation of the intelligent industrial product, but also supports the local evolution of the intelligent industrial product by providing data analysis engine, intelligent optimization engine and machine learning engine along with local data and computing power.

(4) Cloud development service platform layer

1) Virtualizing layer

It not only includes the virtualization of traditional development resources and capabilities such as computing power, but also includes the virtualization of the physical product and the real world, namely the virtual product and the approximate world. The virtual product and the approximate world could be encapsulated by the container technology such as Docker, and could be created as multiple instances in the cloud or be deployed to the edge computing devices.

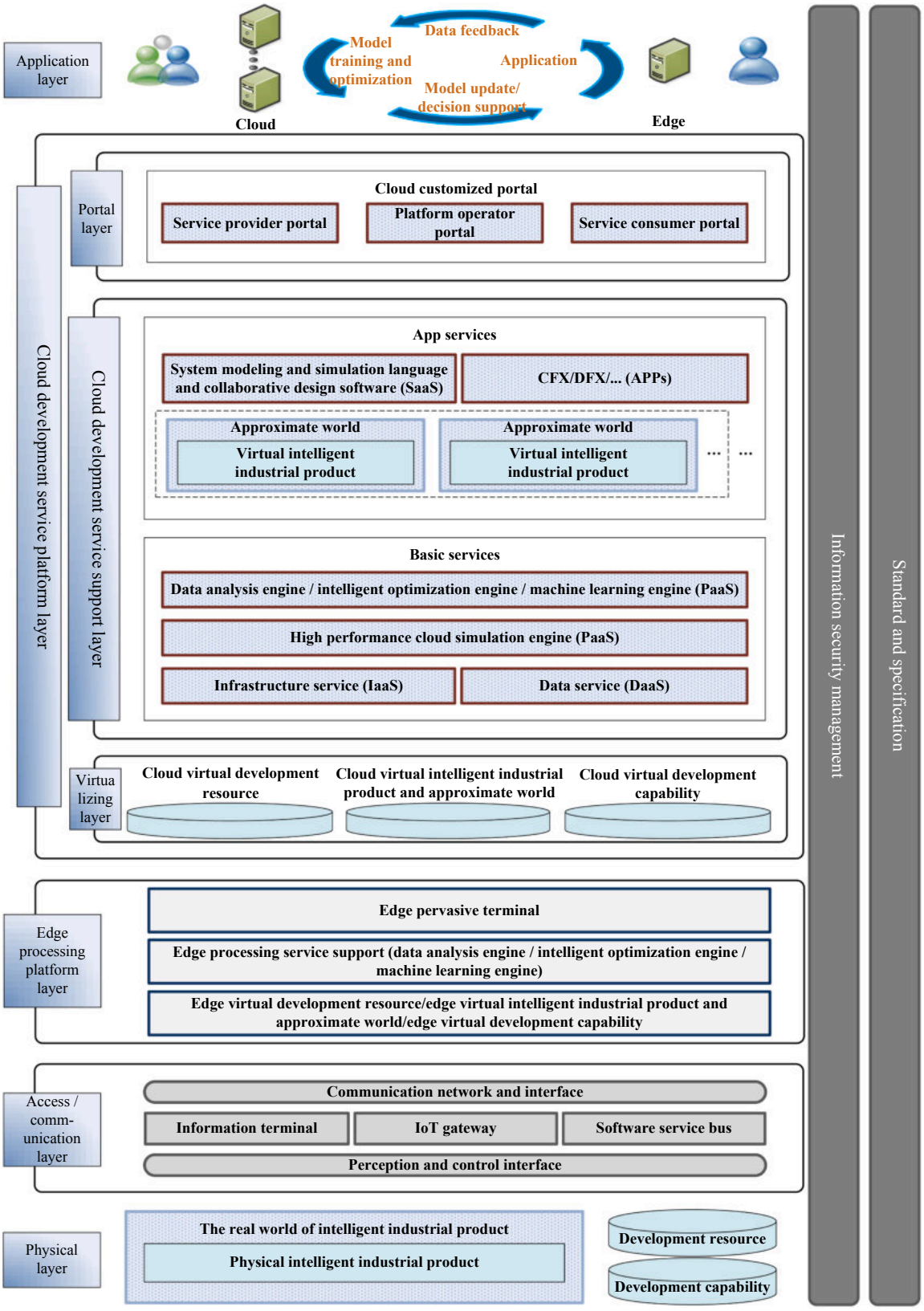

Fig. 7. Global architecture of the EDT based product design system.

2) Cloud development service support layer

(a) Basic services

The basic services module provides the core support of the whole system, including the data support, the computing power support, and the algorithm support in form of services. Data as a service (DaaS) manages and provides the data collected from both the real world systems and the virtualized approximated world systems to support the data driven evolution process of the virtual intelligent industrial product. Infrastructure as a service (IaaS) provides elastic computing power to large scale parallel simulation of multiple virtual intelligent industrial products in multiple approximated world instances, which is further supported by the high performance cloud simulation engine based on the Docker technique. Platform as a service (PaaS) realized via the data analysis engine, the intelligent optimization engine and the machine learning engine supports the evolution of the EDT with abundant powerful algorithms.

(b) Application services

The application services module provides shared domain related applications. With this module, users could define data driven intelligent industrial product development tasks with the assist of system modelling and simulation language and collaborative design service (Software as a Service, SaaS). Large scale simulation instances are defined and executed in this part, continuously generating data from the interaction between the virtual intelligent industrial product and the approximated world, supporting the collaborative evolution of the virtual model.

3) Portal layer

Stakeholders, including end users, could jointly carry out various activities in the whole life cycle of the intelligent industrial product, such as describing the ideal product and supervising the evolution of the prototype product.

(5) Application layer

It reflects that, for both simple evolution paradigm and model evolution paradigm, the EDT could evolve the intelligent industrial product to develop the real intelligence based on the integration of cloud and edge.

5. Product development-oriented EDT construction and evolution

5.1. EDT construction design

As described above, the EDT has the characteristics of persistent enhancement in terms of both its virtual product precision and its physical product policy optimization. Thus, it should be constructed naturally equipped with the ability to evolve, in terms of both the virtual product and the physical product.

To achieve this, the virtual product is designed to be composed of a totally or partially parameterized policy module and a behavior module also parameterized. The function of the latter is to approximate the behavior of the physical product, while that of the former is to generate action commands for the latter to accomplish tasks. The physical product, on the other hand, is designed to be composed also of a totally or partially parameterized policy module whose parameters are a copy of those of the virtual product's policy module through update, as shown in the example of a robot arm EDT in Fig. 8. As for the approximate world, it is also equipped with a behavior module with similar kind and function as that of the virtual product.

With such construction design, the upgrade of the virtual product relies on the model adjustment based on the measurement data collected from the physical world and the simulation data from the approximate world, while that of the physical product relies on the update of its behavior policy and the application of the generated policy.

It is thus required that the physical product is able to apply the generated policy and that the physical product, the virtual product, and the approximate world are adjustable through programs. Based on this, the construction of the dual parts of the EDT is discussed in details in the following parts.

5.1.1. Physical product construction

In an EDT, the physical product serves as a collector of the physical

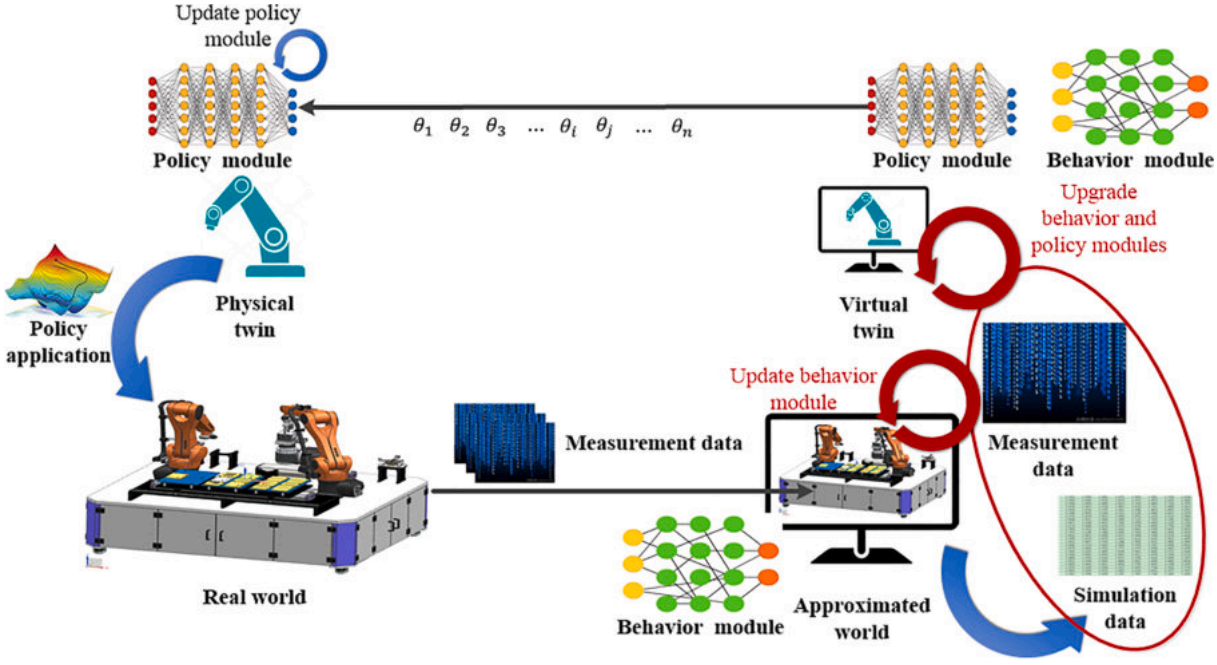

Fig. 8. Global design of the EDT.

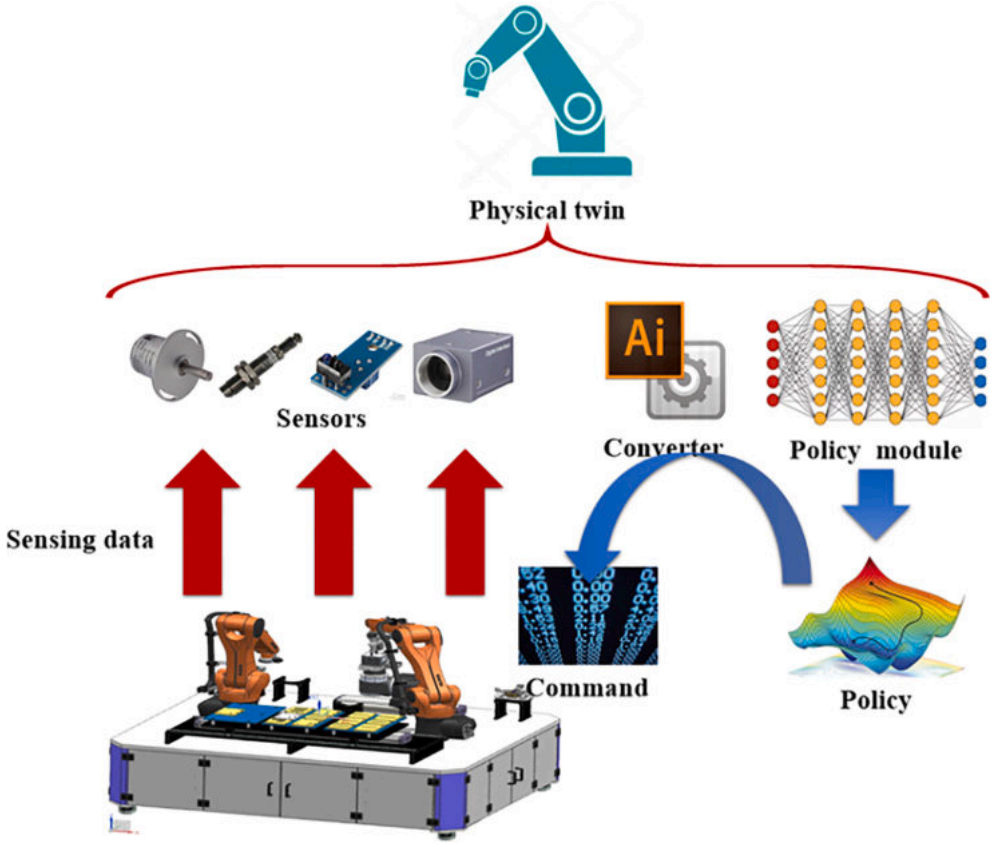

Fig. 9. Physical product construction for robot arm EDT.

measurement data and an actuator and verifier of the policy learned by the virtual product. Thus, the physical product should have the ability to convert the policy into commands directly applicable to its physical sub-components and continuously collect measured data about itself and the surrounding environment. Based on these functional requirements, the following key points should be well determined in the construction of the physical product in an EDT.

(1) Measurement data collection and preprocessing

Measurement data of a physical system originate from measurements from different sensors which, taking the robot arm EDT shown in Fig. 9 as an example, include angular sensors, angular velocity sensors, camera, infrared sensor etc. Therefore, in order to gather sufficient data with a wide enough range for training the virtual product, the physical product needs to be equipped with a wide range of accurate sensors. Meanwhile, as different sensors may have different sample frequency, a pre-processing module in the edge processing platform layer is needed to unify different measurement data time line through either unifying the sample frequency or introducing sample interpolation for measurement data from sensors with a lower sample frequency. Through this coordination process, the data are collected at the same frequency and could be merged together to provide the virtual product with a comprehensive measurement data set of both the physical product and its living environment.

Moreover, besides the data time line unifying processing, further data processing is required before the collected data could be transmitted to the virtual product and is utilized for training, due to the existence of noise, abnormality and misalignment in the preprocessed data. Thus, in the operation mode, the data flow from the physical world into the sensors, pass through the pre-processing module, the filter and alignment module, and finally transmitted to the cloud where they could be utilized for virtual product training.

(2) Policy update and application

To automatically update the policy of the physical product, a parameterized policy module which is the same with that of the virtual product is constructed. At each policy update, the parameters of the physical product's policy module are updated with those of the virtual product's policy module, and hence update the physical product behavior. Moreover, as the policy generated by the policy module is digital, and that the motion or dynamics of a physical system is usually continuous, converting modules, such as stepping motors, that convert the digital policy into continuous behavior policy are equipped, as shown in Fig. 9. These converting modules could convert online the behavior policy, and the behavior of the robot arm in the example in Fig. 9 could be controlled directly or indirectly by frequency of the pulse signal generated by the policy module.

5.1.2. Virtual product construction

In an EDT, the virtual product, composed of a policy module and a behavior module, serves as both an imitator and an optimal policy searcher of the physical product. In the operation mode, the behavior module corrects its behavior based on the measurement data received from its physical counterpart and the policy module searches for an optimal behavior policy in the approximate world with the support of the high performance computer cluster in the cloud. Accordingly, both the behavior module and the policy module of the virtual product should be designed as auto-adjustable through deterministic or stochastic learning programs.

To make the behavior of the virtual product adjustable through measurement data based learning and simulation based reinforcement learning in the approximate world, both the policy module and the behavior module of the virtual product can be modelled in different parameterized forms: partially parameterized and totally parameterized, as shown in Fig. 10.

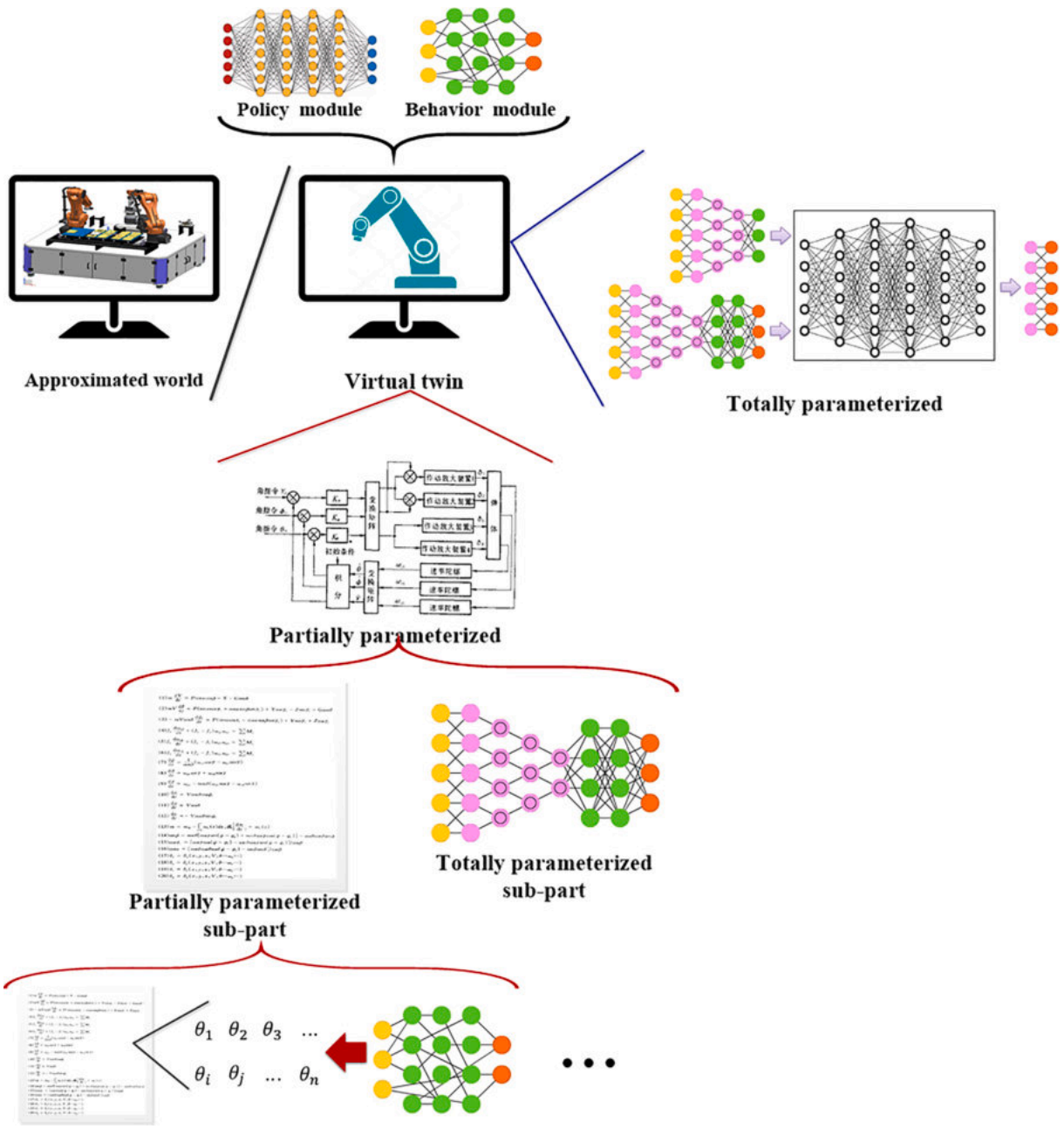

Fig. 10. Behavior module construction of the EDT.

In the totally parameterized modelling mode, the virtual product is either modelled by different neural networks according to its desired function or established by different components, each of which is further constructed by a neural network imitating its behavior. Such a modelling mode normally results in a large number of parameters to be optimized when the behavior of the physical product is complex. Such non-structural characteristics combined with a large number of adjustable parameters make the model hard to optimize.

Different from the totally parameterized modelling mode, the partially parameterized mode introduces the prior information of the physical product into the modelling process, including the system structure, some deterministic theory based part models, some manually behavior fitting functions, etc., as shown in Fig. 10. In this mode, the non-parameterized approach and the parameterized approach are combined in two ways. In the first way, part of the model is established based on theories with some influencing parameters determined from outside, for example by a neural network. In the second, part of the model further consists of a non-parameterized or partial parameterized sub-part and a totally parameterized sub-part. In such a construction form, parameters to be optimized could be drastically decreased, as the model is restricted to only some parts of the entire system, and that the parameterized parts are based on some empirical models or theory based models. But such modelling mode has its drawbacks compared to the first mode, such as sophisticated training process, relatively limited behavior fitting capability.

Hence, in the construction of the virtual product's behavior module of an EDT, the extent and the approach of parameterization needs to be considered delicately based on the features of the system and the extensibility of it. The same also applies to the construction of the approximated world models.

5.2. EDT evolution design

Currently, model evolutions are achieved mainly using machine

learning, deep learning, and reinforcement learning approaches. As for the first two approaches, models are trained mainly through supervised or semi-supervised mode based on collected data, while model evolution in the last approach is accomplished based on data collected via interactions between the current agent and the environment. All three approaches have achieved good results in various tasks. However, in the evolution process of an EDT, none of these evolution approaches could succeed alone. Hence, in this paper, a coordinated evolution approach designed for the EDT is proposed.

An EDT is composed of a virtual product and a physical product. However, the evolution of an EDT mainly concentrates on the virtual product side, while the evolution of the physical product is achieved through a simple transmission of parameters, given that both the virtual product and the physical product have a policy module with the same structure. Meanwhile, the parameter transmission could be performed once the difference between the policy generated by the up-to-date model and that given by the previous version of model exceeds a certain threshold. In such a way, the physical product is gradually evolved, guaranteeing its behavior stability at the same time.

The evolution of the virtual product concerns the evolution of its behavior module through supervised learning and that of its policy module through reinforcement learning, which optimizes the behavior of an EDT according to different tasks. Moreover, as both evolutions take place simultaneously, a coevolution strategy that coordinates these two evolution processes also plays a key role in the evolution of the virtual product.

5.2.1. *Supervised learning based behavior module evolution*

The purpose of the behavior module evolution is to achieve the consistency between the behavior of the physical product and that of the virtual product. Taking the robot arm EDT as an example, in a well evolved virtual robot arm, each ankle of it should turn exactly the same angle as the physical robot arm, given the same input command. This consistency could result in an exact virtualization of the physical robot, which could serve to optimize the control policy of the real robot through reinforcement learning.

Therefore the training of the behavior module follows the typical supervised learning process. The behavior module is established according to the Section 5.1. As shown in Fig. 11, with the parameterized parts built by using neural networks, such process takes the input commands labelled by the output behavior of the physical product as training data, and takes the mean square error between the output behavior of the virtual product and that of the physical product as the loss function. The parameter optimization is accomplished through minimizing the loss function by tuning the neural network parameters.

5.2.2. *Reinforcement learning based policy module evolution*

The reinforcement learning process aims to optimize the policy adopted by the EDT. Different from supervised learning processes based on well labelled data, the reinforcement learning process learns through interaction between the agent and its living environment with no pre-collected labelled data, and relies on the virtual product behavior module. The complete training process under the collaborative training framework is shown in Fig. 12.

As shown in Fig. 12, the exact virtualization of the physical product allows the creation of multiple cyber spaces, where the policy module accumulates experience (in the form of collected data, accumulated parameter gradients, etc.) and updates its policy model through parallel interaction with different virtual product models with multiple instances.

In each of the multiple parallel interactions and trainings, the policy module collects observations of the environment and the reward it gains after applying its output policy based on previous observations. With batches of such interaction data collected, the policy module accumulates experience through policy gradient [22] or policy optimization [17] methods. Finally, with the accumulated experience collected from different interaction and training processes, the parameters of the policy module is updated through application of the synthesized gradient or direct parameter assignment.

5.2.3. *Coevolution of the supervised learning and reinforcement learning*

The above two processes of the virtual product evolutions are carried out independently. However, coupling does exist between them. As stated above, the entire policy optimization process of the policy module bases on the correctness of the virtualization, namely the virtual product behavior module. Thus, if the behavior of the behavior module is largely different from that of the physical product, the behavior policy provided by the policy module which is trained based on data collected through simulation interaction between the virtual product in the approximate world would be useless or even dangerous if applied in real world. Accordingly, in the EDT evolution process, the evolution of the policy module will not start until good precision has been achieved by the behavior module.

Furthermore, as described in the Section 5.1, the EDT is designed to support, to some extent, system scalability, which indicates that some functions or parts in the physical system could be modified according to the needs of product upgrade. Under such condition, the behavior module first will be updated, followed by the update of policy module, to adapt to this change, realizing specifically the support of system extensibility.

6. Case study: EDT based development application in robot arm control

In this section, an application of an EDT based development in the robot arm control command calculation is presented. We first introduce

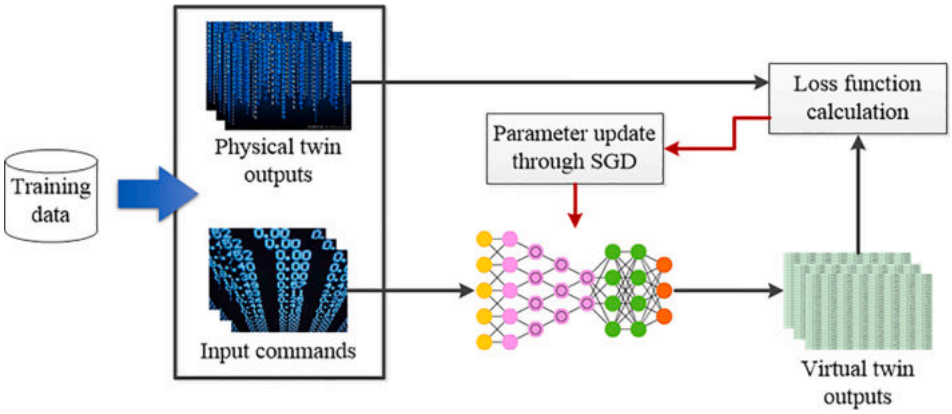

Fig. 11. Supervised learning of behavior module.

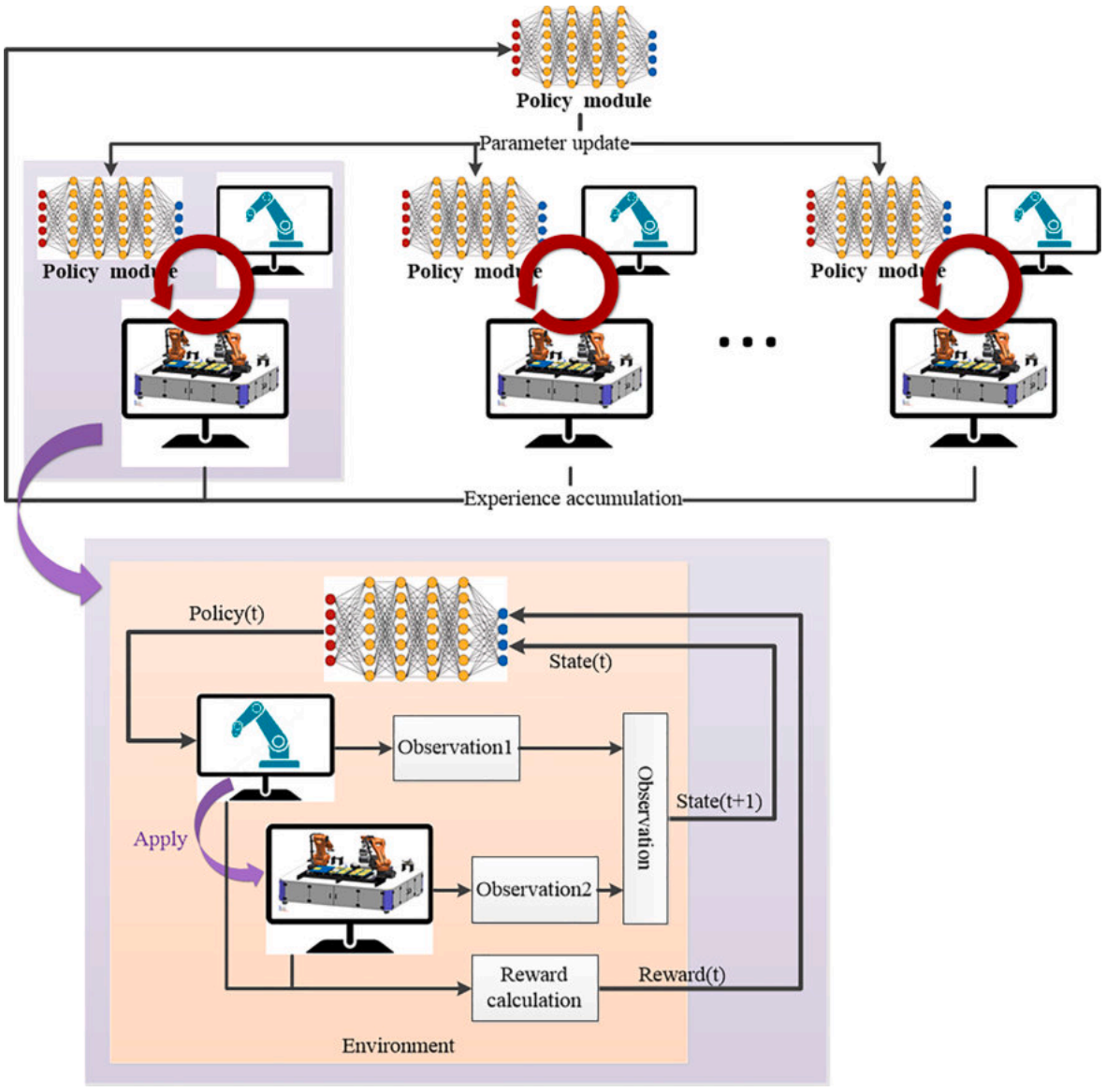

Fig. 12. Reinforcement learning of policy module.

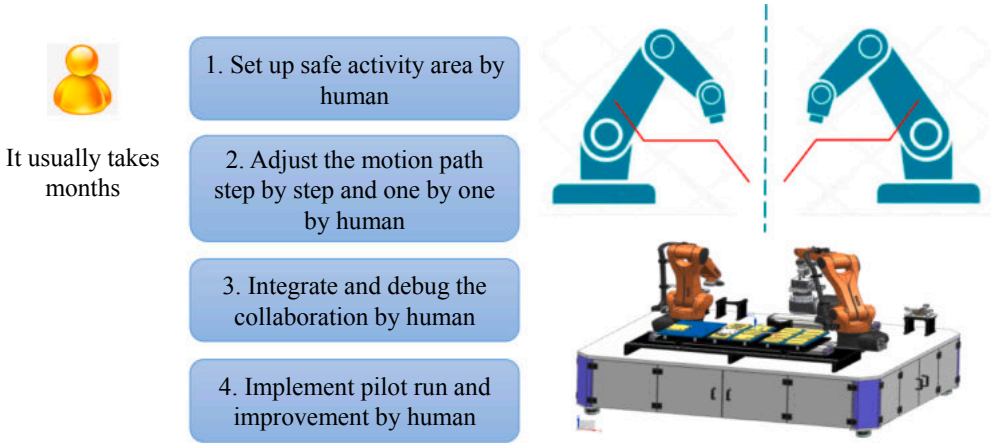

Fig. 13. Manual adjustment of robot arms before equipped in a production line.

a robot arm dynamic model in the production line, followed by the design to construct an EDT for the robot arm control. Finally, some related results on the construction process and system performance are discussed.

6.1. Robot arm control in the production line

Manufacturers around the world are turning to automation to help solve the labor shortage, increase productivity and improve product quality. Robot arms provide a cost-effective, flexible, and safe automation solution for a wide range of production tasks, including machining, product packaging, product sorting, etc.

At present, most of the busy robots on the production line are based on manual pre-adjustment, undertaking fixed tasks, and running in the scope of non-interference. However, with the increasing demand for personalized and intelligent production, they are required to perform diversified and complex tasks, constantly undertake and adapt to new tasks, and can work closely with each other or with people autonomously. Hence, they are becoming the IntelliIndusProd that this paper focuses on.

Normally, a robot arm is composed of five subsystems, namely the driving system, the transmission system, the actuators, the control system, and the detecting system. The control system stands in core position of a robot arm, as it coordinates the dynamics of motors on different axes to accomplish a production related task. Hence, the design of this control system is the core of the design of a robot arm.

However, the control system of most robot arms could only control the robot arm to perform predefined action sequences which are realized through manual guidance and adjustment, as shown in Fig. 13. During this adjustment process, normally, a step-by-step guiding process is needed. In this process, the robot is controlled by the technician through a wired controller to accomplish a given task. The commands sent by the technician through the controller are transformed into executable code lines and stored in the computer. As the task that a robot arm needs to achieve is usually delicate with high precision, the human controlled process needs to be slow enough to protect both the robot and the product from being damaged. After this controlled process, the command execution needs to be further accelerated to meet the production needs. As a result, for each robot arm and each task, it would require about two to three months for a task oriented adjustment before its utilization on product line. Furthermore, as the whole adjustment process is task oriented, any changes on the pre-defined task would require a re-adjustment of the robot arm.

With the increasing needs for individualized and small-lot production [28], flexibility of the production line becomes more and more important. Under such circumstances, the above manual adjustment and re-adjustment of robot arms turn out to be a bottleneck in the efficient production, where our proposed EDT approach can be utilized. Our approach applied in the design of the robot arm, allows the robot arm to adapt itself to different tasks, even changes in the robot arm constitution. We will describe the construction of a robot arm control EDT based on our proposed method in Section 6.2.

6.2. Construction of a robot arm control EDT

6.2.1. Constructing a robot arm EDT model

The construction of the virtual product and physical product is described in Figs. 14 and 15. As the environment is simply an object with a table and a floor, the approximate world could be established deterministically.

For the construction of the virtual product, geometric data and dynamic characteristic data are first collected through product description and geometric measurements. These collected data are further utilized for geometric modelling and dynamic modelling of the virtual product model. Concerning these two modelling processes, Unity [30] is adopted as the modelling tool, as it supports both 3D modelling and dynamic modelling, and hence could combine these two parts together, forming a unique virtual product model.

Moreover, in the process of dynamic modelling, the aforementioned partial parameterized model is adopted. In this model, the dynamics of each arm and joint are determined through dynamic characteristic based modelling, while the interaction between different arms and joints are modelled through neural networks. Specifically, the mass, and the moment of inertia of each arm, the angle range and the motor torque range of each joint are determined manually, while the joint angles between successive arms, together with their changing rates are provided by the neural networks. Based on this, the behavior of a virtual product model could be described by joint angles, joint angle velocities, and joint positions.

As for the policy module that is in charge of generating task oriented

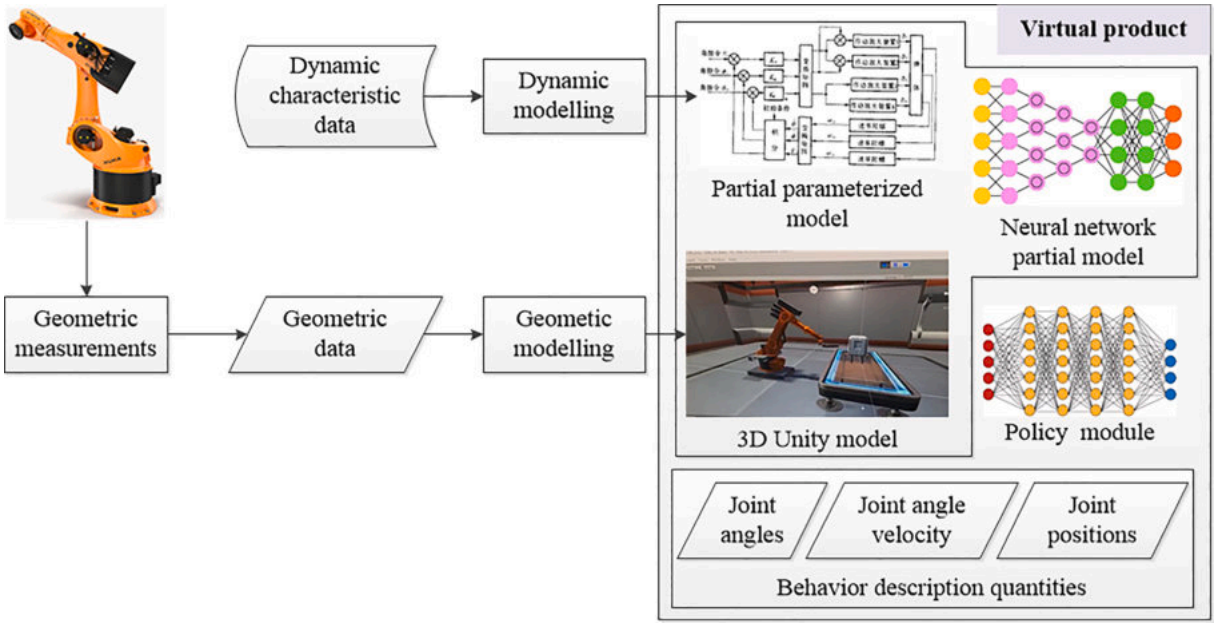

Fig. 14. Virtual product model construction process.

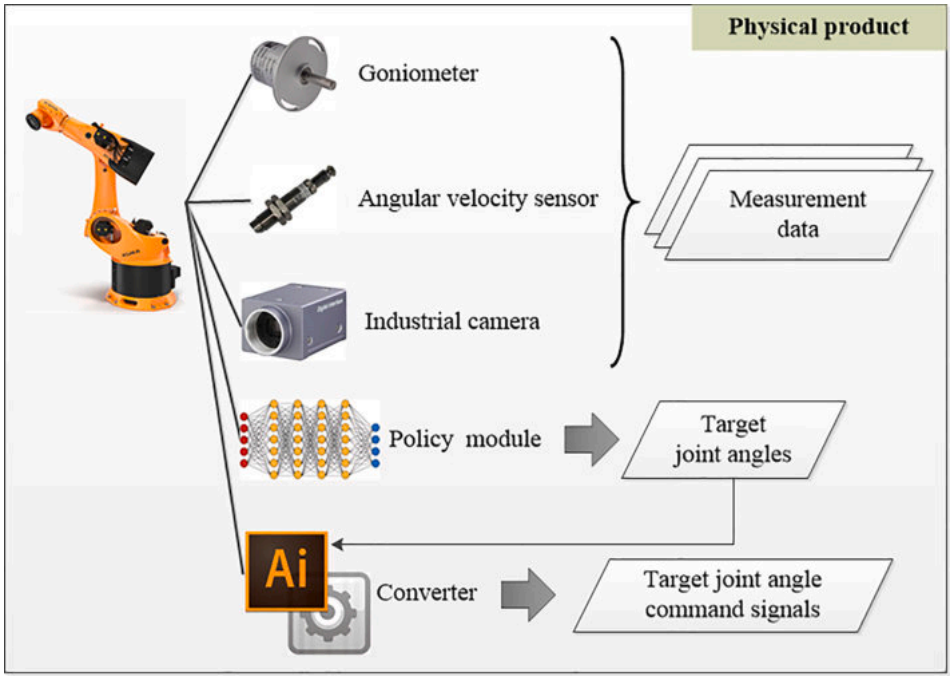

Fig. 15. Physical product construction process.

control commands, on the other hand, it is also modelled as a neural network taking the task related information (for example the position of the cargo in a cargo fetching task) as input and outputs the change rate of joint angles.

Compared to the construction of the virtual product, that of the physical product is less complicated. As depicted in Fig. 15, the raw robot arm is first equipped with different sensors, including goniometers, angular velocity sensors, industrial cameras, etc. These sensors collect measurement data including but not limited to the joint positions, the joint angles and the joint angle change rates, which could be further utilized in the evolution of the virtual product model. Furthermore, the physical product is connected to a computer where the policy module and the converter are equipped. For these two parts, the former is only a copy of the policy module of the virtual product, while the latter converts the target joint angle values generated by the former into target joint angle command signals which could be directly applied to the robot arm machine.

6.2.2. Robot arm EDT evolution

With the EDT model constructed above, evolution of the robot arm EDT is implemented as follows. For the supervised learning based behavior module evolution, the implementation is simply an application of the backpropagation process on a structured neural network, with one neural network part for each correlation between arms. The network training process is as described in Section 4.2.

The evolution of the policy module is a bit more complicated. In this case, the Deep Deterministic Policy Gradient (DDPG) [22] algorithm is adopted as the reinforcement learning algorithm to drive the evolution of the policy module. The DDPG algorithm originating from Deep Policy Gradient (DPG) [27] algorithm is proposed for solving reinforcement learning problems, especially those with a continuous action space, where an agent needs to learn an action policy π through interaction with an environment, aiming at maximizing the expected return from start R_1

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i) \tag{1}$$

the DDPG algorithm utilizes an actor-critic structure, with the actor $\pi(s;\theta^{\pi})$ serving as the policy function and the critic $Q(s,a;\theta^{Q})$ as the action-value function, parameterized respectively with θ^{π} and θ^{Q}, where s denotes the observation state and a represents the action taken. In the training process, the θ^{Q} is updated via minimizing the expected value of temporal difference error

$$\mathrm{E}_{\pi'}\left[Q(s_t,a_t;\theta^{Q}) - \left(r(s_t,a_t) + \gamma Q(s_{t+1},a_{t+1};\theta^{Q})\right)\right] \tag{2}$$

with π' an ε-greedy policy based on policy π, while the policy parameter θ^{π} is updated in the direction of the deterministic policy gradient through the equations

$$\delta_t = r(s_t,a_t) + \gamma Q(s_{t+1},\pi(s_{t+1});\theta^{Q}) - Q(s_t,a_t;\theta^{Q}) \tag{3}$$

$$\theta^{\pi}_{t+1} = \theta^{\pi}_{t} + \nabla_a Q(s,a;\theta^{Q})|_{s=s_t,a=\pi(s_t;\theta^{\pi}_t)} \alpha^{\pi} \nabla_{\theta^{\pi}} \pi(s;\theta^{\pi}_t)|_{s=s_t} \tag{4}$$

With experience replay and target networks mechanisms further introduced, the DDPG algorithm further stabilizes the algorithm evolution process.

Applying the DDPG algorithm to our case, the observation state s is designed as follows. The observation state provided by the behavior module of the virtual product is composed of variables below:

$$\begin{cases} \boldsymbol{diff\_jt}_i = (\boldsymbol{joint}_i - \boldsymbol{tgt})/2 \\ \boldsymbol{diff\_jj}_i = (\boldsymbol{joint}_i - \boldsymbol{joint}_0)/2 \\ \boldsymbol{diff\_th}_j = (\boldsymbol{tpoint}_j - \boldsymbol{hpoint}_j)/4 \; j = 1,2,3,4 \\ \boldsymbol{diff\_hj}_{ij} = (\boldsymbol{hpoint}_j - \boldsymbol{joint}_i)/4 \; j = 1,2,3,4 \\ collision \end{cases} \tag{5}$$

where $\boldsymbol{joint}_i$ denotes the three-dimensional position of joint i, $\boldsymbol{tgt}$ represents the three-dimensional position of the target, $\boldsymbol{tpoint}_j$ is the three-dimensional position of the point j just beneath the object, $\boldsymbol{hpoint}_j$ is the three-dimensional position of the point j on the gripper, $collision$ is the occurrence of collision. By these five parts, the final observation of the agent is

$$\begin{gathered} observation = [s_1 \;\; s_2 \;\; s_3 \;\; s_4 \;\; s_5] \\ s_i = [diff\_jt_i \;\; diff\_jj_i \;\; diff\_hj_{i1} \;\; diff\_hj_{i2} \;\; diff\_hj_{i3} \;\; diff\_hj_{i4}] \; \forall i \in \{1,2,3,4\} \\ s = [diff\_th_1 \;\; diff\_th_2 \;\; diff\_th_3 \;\; diff\_th_4 \;\; collision] \end{gathered}$$

For the design of the reward function $r(s_i,a_i)$ that guides the robot

arm to converge to optimal policy through the DDPG algorithm is designed to guide the gripper to the position below the object with

$$\begin{cases} jre_1 = \left(\sum_{i=0}^{3} \|tpoint_i - hpoint_i\|/4\right) \\ jre_2 = \left(\sum_{i=0}^{3} |hpoint\_x_i|/4\right) \\ jre_3 = \left(\sum_{i=0}^{3} |hpoint\_y_i|/4\right) \\ jre_4 = cos(\boldsymbol{hvect}, \boldsymbol{tvect}) \end{cases} \tag{6}$$

where jre_i $i = 1, 2, 3, 4$ denotes the reward part i, *hpoint_x* represents the x coordinate of *hpoint*, and *hpoint_y* the y coordinate of the *hpoint*. ***hvect*** represents the normal vector of gripper plane, and ***tvect*** is the normal vector of object-bottom plane.

Finally, these two evolution processes above are combined together according to the coevolution mode designed in Section 4.

6.3. Results and discussions

Based on the settings described above, in our simulation experiment, additional Gaussian noise has also been applied to the observation of the agent, i.e. *observation* introduced in Section 6.2.2, which forces our agent to learn the control policy instead of overfitting to a single application scene. Moreover, with the same models and algorithms, objects placed at different positions around the robot could be fetched successfully by the robot arm controlled by our trained policy model. Results of a single object reaching task is shown in Fig. 17.

Episodic accumulated reward evolution chart suggests the variation of the accumulated reward value that the agent could gain in each episode throughout the model evolution process. As expressed in Fig. 16, at the beginning, the reward seems to be only random noise, which indicates that our designed policy module has not yet capture the rules of the task and it just behaves randomly. Then, with the progress of evolution, the episodic reward starts to rise with gradually narrowing variation range. This shows that our policy module has mastered the optimal control policy, indicating the effectiveness of our approach. This could be further confirmed by the results on distance variation along time with a well-trained policy module, as shown in Fig. 17. From Fig. 17, it could be seen that during the process, with the progress of time, the distance between the robot arm hand and the bottom of the object decreases and reaches under 1 cm rapidly. This result shows that through the control policy learned by the policy module, the robot arm could accomplish cargo fetching tasks as expected, which confirms the results above.

7. Conclusion and future work

Motivated by the core idea of developing intelligent industrial products with autonomous learning and self-adaptation capability, we proposed an EDT approach. The contribution of this paper mainly includes:

First, a new concept of EDT has been proposed in this paper via machine learning approaches, to address the lack of flexibility and adaptability in traditional industrial product development. With the newly proposed EDT, more precise virtual product together with behavior policies with higher performance could be developed.

Second, a coevolution approach of the approximate world and the product has been proposed, where the former converges to the behavior of the real world and the latter explores excellent behavior policies that could be applied in the real world. Moreover, multiple virtual spaces has been designed, allowing for more efficient and effective design of different policy models corresponding to different situations, which in turn could largely improve the adaptability of the established policy model to varying and uncertain environments.

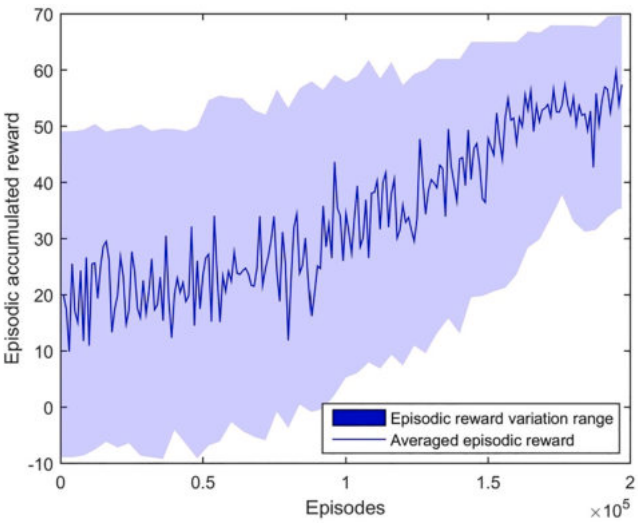

Fig. 16. Episodic accumulated reward evolution.

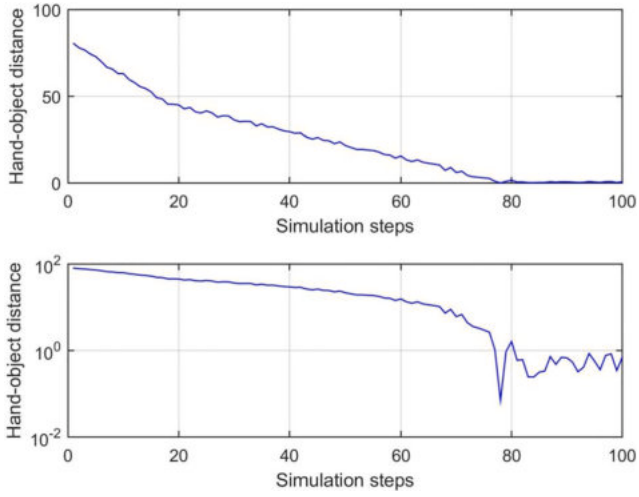

Fig. 17. Control precision of robot arm in cargo fetching task (distance between the robot hand and the bottom of the object).

Third, two evolution paradigms for the virtual product, namely the simple evolution paradigm and the model evolution paradigm, have been proposed. The former allows policy improvement via super real-time deep search based on techniques like Monte Carlo Tree Search, while the latter adopts reinforcement learning approach to ameliorate the policy performance of the product, based on recognition abstracted through supervised learning and unsupervised learning. Moreover, via transfer learning the gap between the virtual space and the physical space was merged for application of the learned policy.

Our future work will concentrate on the following two aspects:

(1) the abstract model and evolution formalism which form the theoretical basis of the intelligent industrial product development.
(2) the system of swarm intelligent industrial product systems developed based on EDT, which supports the collaborative recognition and decision with multiple agents.

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.

Acknowledgement

The research is supported by the National key R&D Program of China under Grant No. 2018YFB1701600 and the Beijing Institute of Technology Research Fund Program for Young Scholars.

References

[1] B.H. Li, X. Chai, B. Hou, et al., An Industrial Internet in the Age of "Intelligence+" - Cloud Manufacturing System 3.0 (Manufacturing Cloud 3.0). International Conference on Industrial Internet, 2019.
[2] G. Xiong, J. Hou, F. Wang, T.R. Nyberg, J. Zhang, M.C. Fu, Parallel system method to improve safety and reliability of nuclear power plants, Intell. Control Automation (2011).
[3] F.-Y. Wang, P.K. Wong, Intelligent systems and technology for integrative and predictive medicine: An ACP approach, ACM Trans. Intell. Syst. Technol. 4 (2) (2013) 1–6, https://doi.org/10.1145/2438653.2438667.
[4] E. Lapira, D. Brisset, H. Davari Ardakani, D. Siegel, J. Lee, Wind turbine performance assessment using multi-regime modeling approach, Renew. Energy 45 (2012) 86–95, https://doi.org/10.1016/j.renene.2012.02.018.
[5] E.H. Glaessgen, D. Stargel, The Digital Twin Paradigm for Future NASA and US Air Force Vehicles, in: 53rd Struct. Dyn. Mater. Conf. Special Session: Digital Twin, Honolulu, HI, US, 2012, 1–14.
[6] J. Leng, D. Yan, Q. Liu, H. Zhang, G. Zhao, L. Wei, ... X. Chen, Digital twin-driven joint optimisation of packing and storage assignment in large-scale automated high-rise warehouse product-service system, Int. J. Comput. Integrated Manuf. 2019, 1–18.
[7] L.R. Goldberg, The book of why: the new science of cause and effect, Notices Am. Math. Soc. 66 (07) (2019) 1, https://doi.org/10.1090/noti1912.
[8] R. Rosen, G. von Wichert, G. Lo, K.D. Bettenhausen, About the importance of autonomy and digital twins for the future of manufacturing, IFAC-PapersOnLine 48 (3) (2015) 567–572, https://doi.org/10.1016/j.ifacol.2015.06.141.
[9] T. Gabor, L. Belzner, M. Kiermeier, M.T. Beck, A. Neitz, A simulation-based architecture for smart cyber-physical systems, in: IEEE International Conference on Autonomic Computing, 2016, 374–379.
[10] R. Söderberg, K. Wärmefjord, J.S. Carlson, L. Lindkvist, Toward a Digital Twin for real-time geometry assurance in individualized production, CIRP Annals 66 (1) (2017) 137–140, https://doi.org/10.1016/j.cirp.2017.04.038.
[11] F.Y. Wang, D.R. Liu, G. Xiong, Parallel control theory of complex systems and applications, Complex Syst. Complexity Sci. 9 (3) (2012) 1–12.
[12] D. Silver, A. Huang, C.J. Maddison, A. Guez, D. Hassabis, Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.
[13] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, et al., Relational deep reinforcement learning, 2018.
[14] M. Asada, S. Noda, S. Tawaratsumida, K. Hosoda, Purposive behavior acquisition for a real robot by vision-based reinforcement learning, Mach Learn 23 (2-3) (1996) 279–303, https://doi.org/10.1007/BF00117447.
[15] M.P. Deisenroth, C.E. Rasmussen, D. Fox, Learning to control a low-cost manipulator using data-efficient reinforcement learning, Robotics: Science and Systems VII, 2011, 57–64.
[16] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533, https://doi.org/10.1038/nature14236.
[17] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, ... D. Silver, Emergence of locomotion behaviours in rich environments. arXiv preprint arXiv: 1707.02286, 2017.
[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
[19] F. Zhang, J. Leitner, M. Milford, B. Upcroft, P. Corke, Towards vision-based deep reinforcement learning for robotic motion control. arXiv preprint arXiv: 1511.03791, 2015.
[20] F. Zhang, J. Leitner, M. Milford, P. Corke, Modular deep q networks for sim-to-real transfer of visuo-motor policies. arXiv preprint arXiv:1610.06781, 2016.
[21] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, W. Zaremba, Hindsight experience replay, Adv. Neural Inform. Process. Syst. (2017) 5048–5058.
[22] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Wierstra, Continuous control with deep reinforcement learning. arXiv preprint arXiv: 1509.02971, 2015.
[23] S.F. Qin, K. Cheng, Future digital design and manufacturing: embracing industry 4.0 and beyond, Chin. J. Mech. Eng. 05 (2017) 12–14.
[24] J.S. Gero, U. Kannengiesser, The situated function–behaviour–structure framework, Des. Stud. 25 (4) (2004) 373–391.
[25] H. Zhang, H. Wang, D. Chen, G. Zacharewicz, A model-driven approach to multidisciplinary collaborative simulation for virtual product development, Adv. Eng. Inform. 24 (2) (2010) 167–179.
[26] J. Estefan, MBSE methodology survey, Insight 12 (4) (2009) 16–18.
[27] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: ICML, Beijing, China, 2014, 21-26 June, 387–395.
[28] C. Yang, S. Lan, W. Shen, G.Q. Huang, X. Wang, T. Lin, Towards product customization and personalization in IoT-enabled cloud manufacturing, Cluster Comput. 20 (2) (2017) 1717–1730.
[29] C. Yang, W. Shen, X. Wang, Application of Internet of Things in manufacturing, in :2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2016, May, 670–675.
[30] Unity User Manual, https://docs.unity3d.com/Manual/ModelingOptimizedCharacters.html.
[31] C. Yang, P. Chi, X. Song, T.Y. Lin, B.H. Li, X. Chai, An efficient approach to collaborative simulation of variable structure systems on multi-core machines, Cluster Comput. 19 (1) (2016) 29–46.