Fruit Picking Robot Arm Training Solution Based on Reinforcement Learning in Digital Twin

Xinyuan Tian[1], Bingqin Pan[2], Liping Bai[1,*], Guangbin Wang[3] and Deyun Mo[1,3]

[1]*Macau Institute of Systems Engineering, Macau University of Science and Technology, Taipa, 999078, Macau, China*
[2]*Sichuan Digital Transportation Tech Co. Ltd., Chengdu, 610095, Sichuan, China*
[3]*School of Mechanical and Electrical Engineering, Lingnan Normal University, Zhanjiang, 524000, Guangdong, China*
E-mail: lipbai@must.edu.mo
[*]*Corresponding Author*

Received 27 May 2023; Accepted 04 July 2023;
Publication 11 September 2023

Abstract

In the era of Industry 4.0, digital agriculture is developing very rapidly and has achieved considerable results. Nowadays, digital agriculture-based research is more focused on the use of robotic fruit picking technology, and the main research direction of such topics is algorithms for computer vision. However, when computer vision algorithms successfully locate the target object, it is still necessary to use robotic arm movement to reach the object at the physical level, but such path planning has received minimal attention. Based on this research deficiency, we propose to use Unity software as a digital twin platform to plan the robotic arm path and use ML-Agent plug-in

Journal of ICT Standardization, Vol. 11_3, 261–282.
doi: 10.13052/jicts2245-800X.1133

as a reinforcement learning means to train the robotic arm path, to improve the accuracy of the robotic arm to reach the fruit, and happily the effect of this method is much improved than the traditional method.

Keywords: Robot arm, digital twin, reinforcement learning, unity, ML-agent.

1 Introduction

Agriculture in the process of Industry 4.0 is also known as Agriculture 4.0 [1]. The first signals of transition are already apparent, and digitization has the potential to fundamentally alter daily life, networks and systems for the provision of food, fibre, and bioenergy, as well as agricultural production techniques [2]. Several ideas have developed in the agricultural sector to express various forms of digitalization in agricultural production systems, value chains, and the larger food system [3]. In the context of such times, fruit picking, an industry that relies heavily on human labour has been iterating. Both companies and scholars want to rely on mechanical methods of fruit picking so as to control labour costs, knowing that labour costs are very expensive today, especially for developed countries [4]. A great deal of research has been put into using Computer Vision (CV) to identify ripe fruit and to pick fruit, and this has largely changed the industry. But to achieve intelligent picking of fruits using fully automated equipment, it is not enough to rely on CV or ripe fruit recognition. After a fruit is identified, how to use the robot arm to pick it accurately is still a problem to be solved. This leads to another area of research which we have focused on, that is, how to dynamically plan the movement path of the robot arm to reach the fruit more precisely.

The automatic planning of paths is never just limited to agricultural scenarios, but can be expanded from there to increasingly complex scenarios. The need for quick and self-correcting machine learning-based navigational capabilities will increase. So, we suggested using reinforcement learning to plan the trajectories of space robots. In contrast to earlier algorithms, reinforcement learning does not give the agent a precise answer to the job. Through interaction with the environment, the reward function, and the task goal, the agent learns the best strategy for maximizing the cumulative reward. Robot users without the necessary specialized knowledge can nonetheless use this strategy. The robot only needs to be given a task to perform; after that,

it will automatically learn how to do it. Robots can learn from their own experiences with little human intervention thanks to reinforcement learning. And it has become a new trend to build digital twins through game engines, such as Unity to model robotic arms and targets in virtual 3D world for path planning and training of generated paths through deep learning methods.

The second chapter of this paper introduces the related work of the robotic arms, digital twins and smart agriculture scenario applications, and summarizes the ideas that inspired our article. The third chapter is the methodology, which is divided into sections to tell the theoretical basis related to the methods and tools we used in turn. Chapter 4 describes the prior information for our specific experiments. It mainly includes specific information about the experimental platform and the building process, as well as the demonstration of the results obtained from the experiments. Chapter 5 is the concluding section, which summarizes our work and gives a theoretical outlook on possible future research directions.

The following are the contributions of this work:

1. Proposed a solution based on digital twin to realize smart agriculture.
2. Use Unity and ML-Agent plug-in to train and optimize the motion path of robotic arm when picking fruits in a smart agriculture scenario.

The feasibility of combining the digital twin with specific industrial scenarios for path planning is verified.

2 Related Work

Robotic and automated systems are being developed to perform jobs currently handled by operators in the fields of industry, medicine, and the military [5]. Crop harvesting is one of the many uses of robots in agriculture that have grown as a result of recent technological advancements in visual identification, 3D reconstruction, placement, and fault tolerance. Zhang et al. proposed that artificial intelligence is used by agricultural robots, like other robotic systems in the field, to carry out a variety of labour-intensive agricultural operations like planting, spraying, trimming, and harvesting [6]. Based on improvements in image perception and orientation detection, the spatial dimensions of fruit targets were determined by segmenting fruits and their provides a basis, then reconstructing 3D fruits using stereo matching. In Wang's research, obstacles for robotic components include fruit localization and hands-free navigation [7].

Finding a route from the current start position to the recognized goal position in the given or partially known environment is the major goal of collision-free pathfinding for the picking manipulator. This path allows the fruit-picking manipulator to proceed securely and without hitting any barriers from the starting point to the target position. Numerous planning strategies, such as the simulated annealing methodology, the A* algorithm [8], the ant colony optimization algorithm [9], and the artificial potential field approach, have been presented by researchers to address this issue. These classic path planning techniques work well for 2–3 dofs robots. The model must, however, adequately characterize barriers in a consistent space for these techniques to work. Like a consequence, the computation cost increases exponentially with the growth of the degrees of freedom of the robot. Multi dofs robot path planning is not acceptable for these conventional techniques. Then, using the hybrid manipulator's high degrees of freedom and intricate structure, Yang et al. [10]. devised a rapidly exploring random tree (RRT) algorithm that has realized the efficient path planning of the hybrid manipulator. Path planning for selecting fruit in the field, which in itself is dynamic and uncontrolled, is a very difficult challenge. In this paper, a better RRT algorithm is suggested as a solution to this issue. Self-made binocular vision system is used to recognize and find the target, while binocular vision is utilized to observe the environment [11]. As the picking robot, a 6-DOF robot arm is used. The manipulator and the barriers are appropriately simplified to produce a collision-free path. The RRT method is then used to plan the manipulator's picking path, and the concept of target gravity is incorporated to the RRT algorithm to speed up the path-finding process. The RRT algorithm's generated path is optimized using the GA algorithm and flattening approach to provide an optimal or nearly optimal path. Finally, real-world fruit-picking trials are used to validate the revised RRT algorithm before being applied to collaborative virtual simulation [12].

Identification and positioning alone cannot achieve fully automated fruit picking, so it is very critical to get the robotic arm to the location of the fruit that has been detected. And in recent years, some scholars have proposed to use the digital twin approach to plan the robot arm motion trajectory in the virtual world [13]. In [14], the authors used the Unity 3D engine for robotic arm path planning to ensure smooth trajectories of real-world industrial robots along specified paths. More work [15] subsequently confirmed that the Unity 3D engine provides a powerful ability to interact with virtual and real data, that is, the ability to pass data in both directions. Moreover, in [16], the authors propose that using ML-agents, an open-source package

in the Unity engine, it is possible to achieve simulation experiments in a virtual world using integrated deep learning algorithms and produce desired results. In [17], the authors used ML-agents as a tool and invoked the deep reinforcement learning tool in it to train the robotic arm and succeeded in precisely finding the motion points in the environment. And in [18], the authors have also successfully implemented the planning and optimization of paths in digital twins by means of genetic algorithms in the ML-agent tool. What we can determine through the literature [17] and [18] is that using deep learning algorithms, it is indeed possible to train and optimize the path of the robotic arm to reach the object in the digital twin. However, no scholars today have proposed how to train a robotic arm to perform specific actions such as picking fruits in a specific scenario, which is what we consider as a research gap.

3 Methodology

This chapter will be divided into different sections to describe the entire process of using the digital twin as a platform and using reinforcement learning as a framework to train the robotic arm, as well as the specific hands-on situation, in Figure 1, the general framework is presented.

We apply the Unity game engine, which enables the use of a machine learning (ML) toolkit to train agent (or agents) to connect with the digital world known as ml-agents [19]. By simultaneously training several

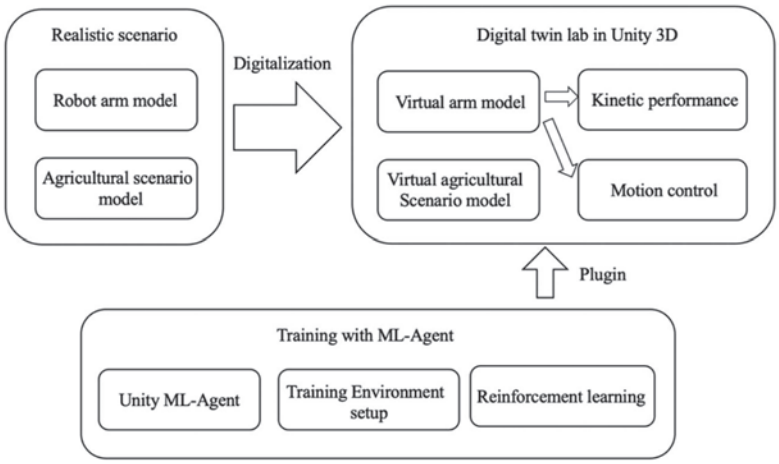


Figure 1 Overall framework schema.

agents with the same taught data at fast rates, training time can be reduced. This encourages simulation learning among the ML community. The Unity ML-Agents Toolkit is an open source project under the Apache 2.0 license. Therefore, you can modify and implement ML-Agents as needed. Using Unity and the ML-Agents toolkit, we can create physical, visual, and cognitively rich AI environments and use them for benchmarking and researching new algorithms and methods.

3.1 Virtual Twin for Robot Arm

In the process of building our digital twin simulation platform, we used an open source six-axis robotic arm model for importing into the Unity engine. The arm was placed on an automated guided vehicle (AGV) in order to simulate the working conditions of a fully automated fruit picking robot as closely as possible and to focus our attention on the "last mile" problem once the vehicle reaches the target place.

In Unity, the robot arm is able to rotate as in the real world, but also as in the real situation, it must have certain angle restrictions, i.e., it does not allow some movements that defy the rules of physics. So, when we want to command a robotic arm to complete a movement in the digital twin, instead of having it rotate or move at an unreasonable angle, we should let the machine learning tool learn how to achieve optimal path planning by controlling the logic of the secondary arm rotation. For the robotic arm in the digital twin, we can still regard it through the same viewpoint as the physical robot, the cosine rule [20] can still be utilized to calculate the number of degrees each sub-arm must spin to reach a particular target location (X, Y) in the virtual robot system. The layout of the virtual arm is shown in Figure 2, name of each joint of the robot arm is also shown in Figure 2, and the following derivation is given:

$$\theta_1 = arccos\frac{L_1^2 + X^2 + Y^2 - L_2^2}{2L_1\sqrt{X^2 + Y^2}} + \theta_t$$

$$\theta_2 = arccos\frac{X^2 + Y^2 - L_1^2 - L_2^2}{2L_1L_2}$$

3.2 Unity ML-Agent Implementation

The Unity Editor is a graphical user interface that is part of the game engine that makes up the Unity game development platform. Unity was first

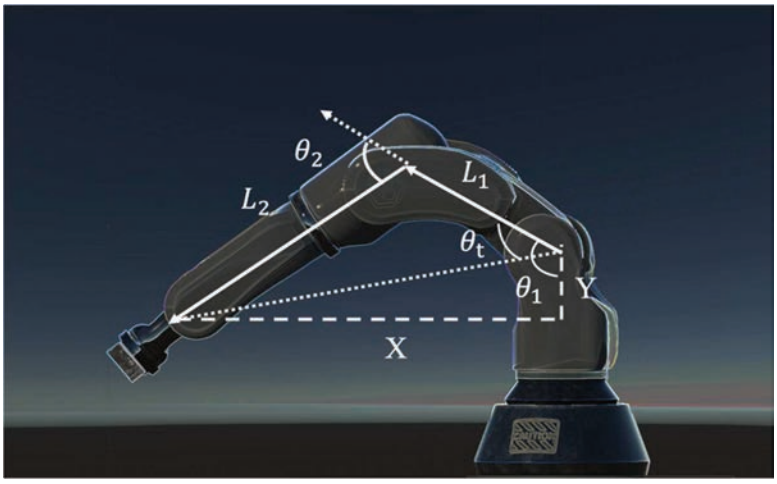


Figure 2 Angles to be solved for each arm constituent for a goal location are depicted over a virtual robot arm (x, y).

developed in 2005 and has since expanded. Currently, it is used by developers for a wide range of interactive simulations, such as high-end console games and AR/VR experiences [21], and mobile and browser-based games. Previously, agents had to be artificially programmed to exhibit the desired behaviours, but today, in a training setting, they are able to continuously learn. Techniques for instructing agents to improve the efficiency of development. An open source plugin called Unity ML-Agents enables the use of games and simulations as training grounds for intelligent agents. Agents can be trained using reinforcement learning, genetic algorithm, neuron evolution, or other machine learning techniques using the simple-to-use Python API.

3.3 Training Environment for Reinforcement Learning

The robotic arm's reinforcement learning training is put into practice using Unity ML-Agents. In the Unity reinforcement learning tool, the action performer is referred to as an agent, who is merged into the environment; the strategy is the objective of the action execution; the brain is in control of providing associated agents decision-making strategies to guide the execution of the action [22]; Before and after the action is taken, there will be two states, and the difference between the 2 states will produce a reward value that meets the conditions of the strategy, information exchange and general instruction. Figure 3 depicts the link between "Agent," "Brain," and "Academy":

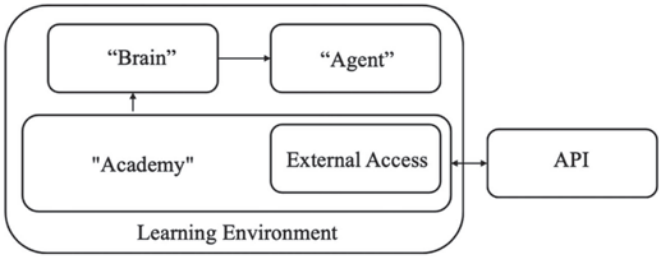


Figure 3 Relationship schema.

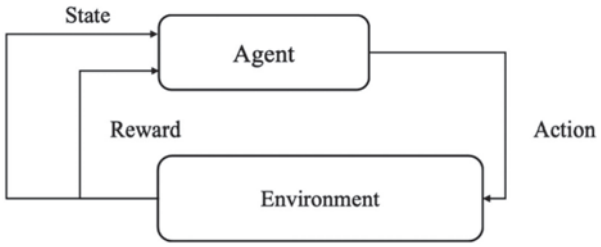


Figure 4 Connections and workflow between agent and environment.

Our digital twin training environment is built based on Anaconda Navigator, the officially recommended plugin for deep learning training on objects. And in the training process, TensorFlow was used as the tool for reinforcement learning. All the training process in based on the macOS Catalina 10.15.7, and in the Terminal, we installed "ml-agents-env" and "ml-agents".

3.4 Reinforcement Learning

In this section, we talk about the reinforcement learning content that is based on the ML-agent tool, which corresponds to the robotic arm in this paper. Reinforcement learning is a branch of machine learning that emphasizes how to act based on the environment to maximize the expected benefit, inspired by the behaviourist theory in psychology of how organisms gradually develop expectations of stimuli in response to rewarding or punishing stimuli from the environment, producing habitual behaviours that maximize the benefit [23]. As shown in Figure 4, Agent represents the robot arm and Environment represents the environment [24]. Reinforcement learning is actually an interaction between the environment and the robot, where the environment stimulates the robot to produce the next action by generating a reward for the robot, and so on and so forth. This is very similar to conditioned reflexes.

By using the logic of mathematics to explain this process, we describe the process by establishing the following variables:

Reward R_t: The solution to all issues revolves around maximizing the accumulated rewards. It should be a scalar function that can be viewed as an accumulation of points corresponding to the degree of completion in the training robotic arm motion path.

History H_t: It should be possible to record all actions and rewards of past behaviours, i.e., a sequence containing information on observation, reward, and behaviour that satisfies $H_T = O_1, R_1, a_1, \ldots, O_t, R_t, a_t$.

Observation O_t: observation value at the time step.

State S_t: Is a function of determining the future, the information already available, about the history $S_t = f(H_t)$.

Action a_t: Record what the action is at the current moment.

In our experiment, Objective is the score obtained from the simulation; State is the angle of the sub-arm at each moment in the movement of the robot arm; Action is the decision to control the movement of the robot arm up and down, left and right; Reward is the scoring of whether the object is accurately touched in real time.

The Markov Decision Process (MDP), which is important in reinforcement learning, is then introduced [25]. At the moment $t = 0$, the random initialization state $s_0 \sim P(s_0)$, when the robot chooses the action plan a_t according to s_t, based on which the environment gives the reward $r_t \sim R(\cdot|s_t, a_t)$. The environment gives the state of the next moment $s_{t+1} = P(\cdot|s_t, a_t)$, then the robot receives r_t, s_{t+1} and so on again. On this basis will produce new R_{t+1}, we can denote this variation by (s_t, a_t, s_{t+1}). As a result, a Markov reward process tuple is produced, with S being a finite set of states $<S, A, P, \gamma>$. A is a limited number of possible actions, and [0, 1] is a discount factor γ that gives priority to immediate benefits.

Technically, P is the state transition probability matrix, which offers a means of representing the chances of a transition for an agent a from any state s to a future state s':

$$P^a_{ss'} = \mathbb{P}[s_{t+1} = s'|S_t = s, A_t = a]$$

The reward function, r, encodes the anticipated reward following a change from P:

$$r^a_s = \mathbb{E}[r_{t+1}|S_t = s, A_t = a]$$

The assessed reward is defined as the total of all potential discounted rewards for a particular time step:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Then, $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$ is the expected reward evaluation for a specific state, s, and exploration policy, is the state value function of the reward:

$$V_\pi(S) = \mathbb{E}[R_t|S_t = s, \pi]$$

This valuation function *V* informs the agent of the anticipated total of potential rewards for a specific policy for a state *S* as well as the time step *t*. As a result, the agent is able to select the course of action that will maximize the total rewards [26]. The agent has liberty over how to proceed from the most recent state visited. The agent's prime goal in this process is to obtain the greatest reward in the fewest number of time steps. In order to choose an action in simulation space, we generally need to describe an exploration policy. To choose the kind of response an agent will get for each action, we need a reward function. A learning rule enables the policy for path planning to change in response to feedback and reward. Last but not least, a discount factor permits the value of short-term benefit over long-term reward in accordance with achieving the maximum reward in the shortest amount of time.

3.5 Path Smoothing

In our training process, there are inevitable obstacles such as tree branches encountered during the movement from the starting point of the robot arm to the target point. Although reinforcement learning can help us optimize the path for robot arm so as to reach the target point, path smoothing is still one of the problems we need to solve. For example: Two obstacles need to be avoided in order to go from 1→8, as indicated in Figure 5. The un-smoothed path is shown in black and goes as follows: 1→2→3→4→5→6→7→8. But according to normal human logic, to move from 1 to 4 while avoiding obstacle 1, the shortest path is directly from 1 to 4, and passing through 2 and 3 on the way is unnecessary. All superfluous locations on the pathway are eliminated using the smooth approach. So according to this method, the result is obtained as shown in the red line, it saves two unnecessary paths, 2→3 and 5→6→7. Path smoothing is used to eliminate superfluous points and make the path more streamlined.

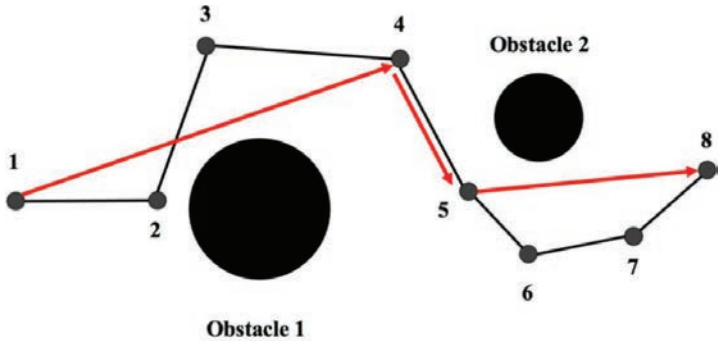


Figure 5 Path smoothing.

The path smoothing procedure starts with path point a and its initial value is 1, as illustrated in Figure 6. A local path that needs to be found consists of path points a, a + 1, and a + 2. First, the distances between the path's points are calculated. The expression $d_{a,a+1}$ denotes the separation between route site a and route site a + 1. Point a and point a + 2 are directly connected if the length between path points a and a + 2 (abbreviated as $d_{a,a+2}$) is smaller than the total of $d_{a,a+1}$ and $d_{a+1,a+2}$. The next step is to insert detection points between point a and point a + 2 to see if a and a + 2 collide, the length between point a and point a + 2 determines the total number of detection points. The space between detection locations needs to be closer than the path's step size, the local route between route site a and site a + 2 is pedestrian if none of the detection points run into any barriers, then the route site a + 1 is deleted. In the new path, route sites a and a + 2 are saved, and site a + 3 is added to make up the local route to be identified (route sites a, a + 2, and a + 3 are set as locations to be recognized). The next headers (i.e., a + 3 and a + 4) are added to make up the local path to be identified (i.e., a + 2, a + 3, and a + 4 are set as recognition points) if there is a conflict between recognition marks and obstacles. Route site a, a + 1, and a + 2 are stored in the positive trajectory. Up until the last route site is found, the aforementioned procedure is repeated.

4 Experiment and Result

In this section, we will talk in detail about how to use the Unity engine to put the robot arm 3D model into the digital twin for training to get a better path planning.

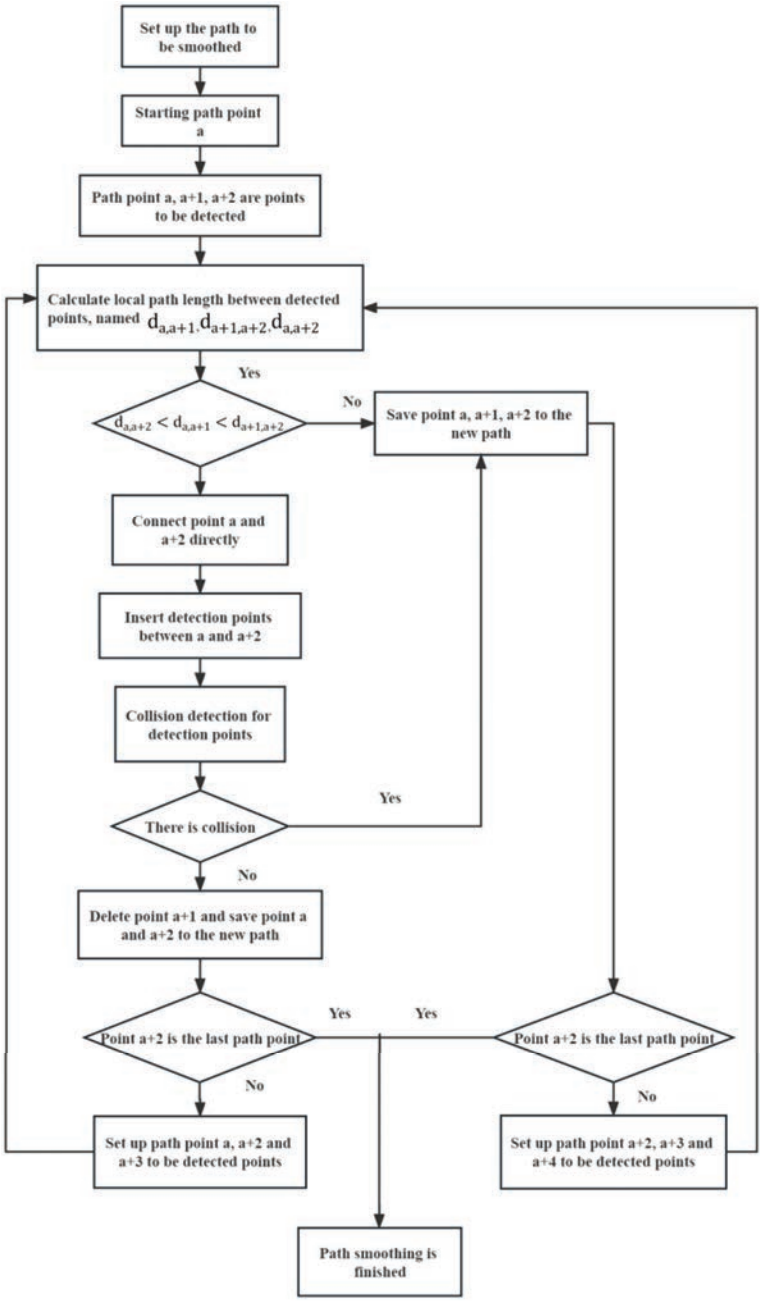


Figure 6 Path smoothing flow chart.

4.1 Scenario and Case Description

The Unity we use is version 2022.1.16f1c1 for macOS-based systems. The use of robotic arms for picking ripe fruit is a very large application scenario, as there may be hundreds of fruit trees in the orchard, or more than one cart can be used for picking. However, we cannot expect to achieve such a daunting goal at once, so we have selected only one of the work areas for primary testing. We assume that the fruit to be picked is the apple, which is the most prevalent and common hard fruit, and is of moderate size, making it a perfect candidate for studying the control of an automatic picking robot arm. And as for the robotic arm, that is, the AGV with mechanical gripper we mentioned before.

4.2 Unity Scene Modelling

Throughout the work area, we placed an apple tree, corresponding to the tree with several ripe apples, which is our target. In addition to this, there is the robotic arm that we use for picking. According to reality, the tree and the cart should be placed in the same plane, so in the digital twin we did the same. The schematic diagram of the working scenario is shown in Figure 7.

The robotic arm was accused in our experiment that after the AGV moved to the specified position (roughly, thus enabling the robotic arm to reach the apple), the robotic arm was considered to be able to pick the apple when it successfully touched it within the kinetic range. The picked apples will be put back into the basket nested on the AGV according to the set route (not

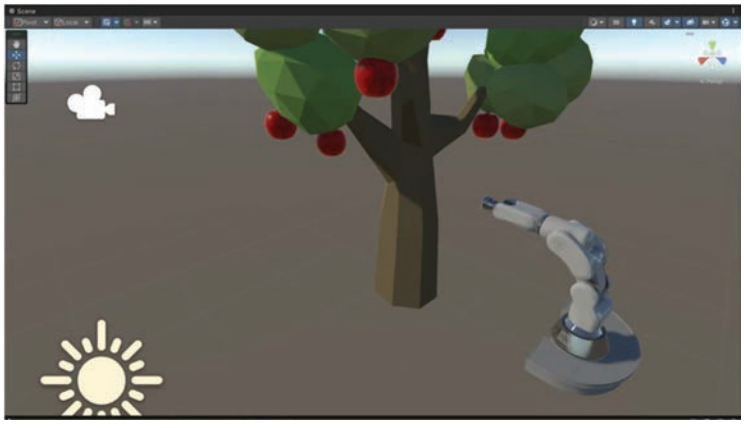


Figure 7 The schematic diagram of the working scenario.

Table 1 Reinforcement learning main parameters configuration

Parameter Name	Parameter Value
batch_size	1024
buffer_size	20480
learning_rate	0.0003
beta	0.001
epsilon	0.2

shown in the figure). A successful picking task is considered to have been accomplished by completing the above.

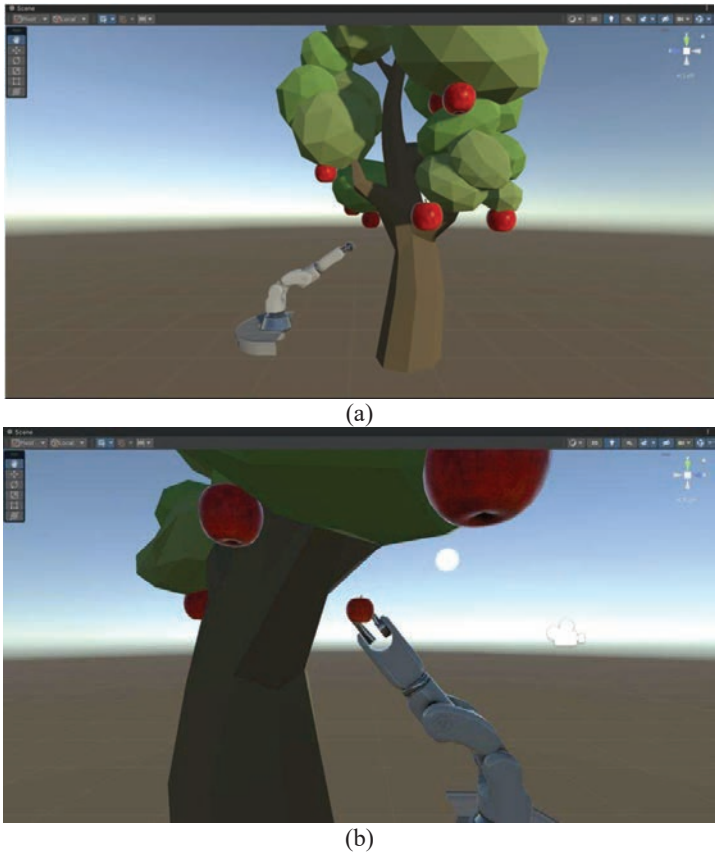
4.3 Task and Parameter Setting

The grabbing target is an single apple, and the constraints are as follows:

1. The initial position of the target to be grasped is 15 cm (distance data of the virtual environment in the twin system) from the origin of the robot arm, and the position is changed randomly when the target to be grasped is touched by the end of the robot arm, and the range of the changed position is limited to a rectangle of 10×5 cm^2 cantered on the initial position.
2. Set the joints of the robot arm to be connected by configurable hinges, and limit the rotation range of each joint with reference to the motion law of the physical model.
3. Based on the above steps, and then based on continuous tuning, we finally determined the main initial parameters, and finally based on this for reinforcement learning training, as shown in Table 1.

4.4 Training Process

The computer used for the training process was a MacBook Pro (13-inch, 2018) with an 8^{th} generation 2.3 GHz quad-core Intel Core i5 processor with 8 GB of RAM and an Intel Iris Plus Graphics 655 1536 MB graphics card. The digital twin system is built on Unity software version 2022.1.16f1c1, and the reinforcement learning training platform is Anaconda. During the training process, the robot arm moves randomly in space until it touches the apple, and once the apple is touched, the target apple will change its position after being touched. The total training duration was about 2.5 hours, and the training process is shown schematically in Figures 8(a) and 8(b).



(a)

(b)

Figure 8 (a) The robot arm moves randomly in the air until it touches the apple, (b) Robotic arm successfully picks up an apple.

4.5 Result

TensorBoard may be used to monitor the training process, and it is discovered that as the training goes on, the estimated value and cumulative reward both increase steadily, while the speed losses and various vectors decrease. The visualization images of training results is shown in Figure 9. Limited by the performance of the computer, in our training, only one agent was used. The plug-in supports the use of multiple agents for training at the same time, which will multiply the efficiency of the training and will achieve better performance to some extent.

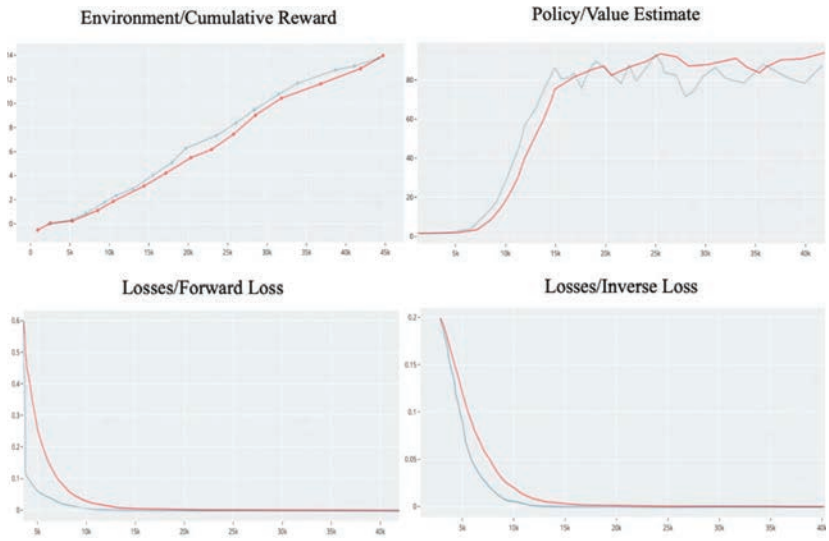


Figure 9 The visualization images of training results.

The data in the image shows that training the robot arm with reinforcement learning algorithms can be very effective, and the results are binary files that can be loaded in the Unity engine. The training enabled the virtual arm in the digital twin to precisely reach the apple and to find the next target when the apple was touched (considered picked and randomly moved to indicate the next apple).

5 Discussion

We consider it is very feasible to use Unity 3D as the basis for this type of digital twin research, and it can meet the current industrial needs. At the same time, we are looking forward to exploring more intelligent situational awareness and interaction platforms. This section consists of a summary of our work and an outlook for future work, which will be presented in two separate sections.

5.1 Conclusion

Our theoretical analysis as well as our experiments show that the creation of a virtual lab through the digital twin is indeed able to train robotic arms in a

virtual engine to work on the precise picking of produce. Thanks to the powerful support of Unity 3D software and the diverse deep learning algorithms provided by the ML-Agent plugin, we succeeded in using reinforcement learning to achieve the desired goal. Base on the low-cost operation of digital scenes, the application of training robotic arms and even robots for complex scenes will become more common as we can make the physical training digitally through simulation by the established digital twin. In addition, the time cost of training will be greatly compressed and more budget will be cut. The use of smart agriculture can still be expanded to other industries, and more digital twins will be available in the future to face a more intelligent and digital future.

5.2 Improvements

Our work is not perfect, and there are more comprehensive issues to be studied. And for future research directions, we have the following suggestions. First, we continue to go deeper into the digital twin of the digital agriculture scene, upgrading from an apple tree to an orchard with more robotic arm AGV carts. This can be more in line with the industrial reality, while forming a more complete digital twin scene. Second, more comprehensive digital twin training for individual robotic arms, such as getting the AGV to move, rather than setting it right at a given location and just moving the arm. This will provide a more comprehensive training and solution, and is also more capable of meeting realistic scenarios. Third, we continue to delve into how the movements and effects generated in the digital twin can be transferred to the real scene and have the robotic arm make the corresponding movements. This is a problem of interaction between the real and virtual worlds, but once the research is successful, the models in the digital twin can be transferred in real time through industrial cameras and simulated through the digital twin to derive the best path optimization, thus forming a more complete solution for automatic picking path optimization based on digital agriculture.

Acknowledgements

This work was supported in part by Science and Technology development fund (FDCT), Macau SAR (File Nos. 0003/2021/ITP) and Science and Technology Planning Project of Zhanjiang (File Nos. 2021A05038).

References

[1] Rose, D. C., and Chilvers, J. (2018). Agriculture 4.0: Broadening responsible innovation in an era of smart farming. Frontiers in Sustainable Food Systems, 2, 87.

[2] Smith, M. J. (2018). Getting value from artificial intelligence in agriculture. Animal Production Science, 60(1), 46–54.

[3] Rotz, S., Duncan, E., Small, M., Botschner, J., Dara, R., Mosby, I., ... and Fraser, E. D. (2019). The politics of digital agricultural technologies: a preliminary review. Sociologia Ruralis, 59(2), 203–229.

[4] Inshakova, A. O., Frolova, E. E., Rusakova, E. P., and Kovalev, S. I. (2020). The model of distribution of human and machine labor at intellectual production in industry 4.0. Journal of Intellectual Capital, 21(4), 601–622.

[5] Daudelin, J., Jing, G., Tosun, T., Yim, M., Kress-Gazit, H., and Campbell, M. (2018). An integrated system for perception-driven autonomy with modular robots. Science Robotics, 3(23), eaat4983.

[6] Zhang, Y., Li, M., Qiao, J., and Liu, G. (2008). A segmentation algorithm for apple fruit recognition using artificial neural network. Aktualni zadaci mehanizacije poljoprivrede. Zbornik radova, 35. međunarodnog simpozija iz podruèja mehanizacije poljoprivrede, Opatija, Croatia, 11-15 veljaèe 2008., 359–367.

[7] Wang, Z. (2018, September). Robot obstacle avoidance and navigation control algorithm research based on multi-sensor information fusion. In 2018 11th International Conference on Intelligent Computation Technology and Automation (ICICTA) (pp. 351–354). IEEE.

[8] Au, W., Chung, H., and Chen, C. (2016). Path planning and assembly mode-changes of 6-DOF Stewart-Gough-type parallel manipulators. Mechanism and Machine Theory, 106, 30–49.

[9] Gómez-Bravo, F., Carbone, G., and Fortes, J. C. (2012). Collision free trajectory planning for hybrid manipulators. Mechatronics, 22(6), 836–851.

[10] Yang, H., Li, L., and Gao, Z. (2017). Obstacle avoidance path planning of hybrid harvesting manipulator based on joint configuration space. Transactions of the Chinese Society of Agricultural Engineering, 33(4), 55–62.

[11] Wang, C., Tang, Y., Zou, X., SiTu, W., and Feng, W. (2017). A robust fruit image segmentation algorithm against varying illumination for vision system of fruit harvesting robot. Optik, 131, 626–631.

[12] Cao, X., Zou, X., Jia, C., Chen, M., and Zeng, Z. (2019). RRT-based path planning for an intelligent litchi-picking manipulator. Computers and electronics in agriculture, 156, 105–118.

[13] Garg, G., Kuts, V., and Anbarjafari, G. (2021). Digital twin for fanuc robots: Industrial robot programming and simulation using virtual reality. Sustainability, 13(18), 10336.

[14] Xue, C., Qiao, Y., and Murray, N. (2020, August). Enabling human-robot-interaction for remote robotic operation via augmented reality. In 2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM) (pp. 194–196). IEEE.

[15] Li, Y., Zhang, Q., Xu, H., Lim, E., and Sun, J. (2022). Virtual monitoring system for a robotic manufacturing station in intelligent manufacturing based on Unity 3D and ROS. Materials Today: Proceedings.

[16] Juliani, A., Berges, V. P., Teng, E., Cohen, A., Harper, J., Elion, C., … and Lange, D. (2018). Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627.

[17] Lin, M., Shan, L., and Zhang, Y. (2020, September). Research on robot arm control based on Unity3D machine learning. In Journal of Physics: Conference Series (Vol. 1633, No. 1, p. 012007). IOP Publishing.

[18] Liu, X., Jiang, D., Tao, B., Jiang, G., Sun, Y., Kong, J., … and Chen, B. (2021). Genetic algorithm-based trajectory optimization for digital twin robots. Frontiers in Bioengineering and Biotechnology, 9.

[19] Juliani, A., Berges, V. P., Teng, E., Cohen, A., Harper, J., Elion, C., … and Lange, D. (2018). Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627.

[20] Akimov, S. S. (2022, May). Solution of the inverse kinematic problem for the KUKA KR AGILUS robotic arm. In Computer Applications for Management and Sustainable Development of Production and Industry (CMSD2021) (Vol. 12251, pp. 115–119). SPIE.

[21] Hung, P. T., Truong, M. D. D., and Hung, P. D. (2022). Tuning Proximal Policy Optimization Algorithm in Maze Solving with ML-Agents. In International Conference on Advances in Computing and Data Sciences (pp. 248–262). Springer, Cham.

[22] Urmanov, M., and Alimanova, M. (2020). Training a single Machine Learning Agent using Reinforcement Learning and Imitation Learning methods in Unity environment. Suleyman Demirel University Bulletin: Natural and Technical Sciences, 52(1).

[23] Recht, B. (2019). A tour of reinforcement learning: The view from continuous control. Annual Review of Control, Robotics, and Autonomous Systems, 2, 253–279.
[24] Terry, J., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sullivan, R., … and Ravi, P. (2021). Pettingzoo: Gym for multi-agent reinforcement learning. Advances in Neural Information Processing Systems, 34, 15032–15043.
[25] Ding, T., Zeng, Z., Bai, J., Qin, B., Yang, Y., and Shahidehpour, M. (2020). Optimal electric vehicle charging strategy with Markov decision process and reinforcement learning technique. IEEE Transactions on Industry Applications, 56(5), 5811–5823.
[26] Puterman ML. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons; 2014 Aug 28.

Biographies

Xinyuan Tian received his B.S. degree in Information Security in 2020 from the School of cyber science and engineering of Wuhan University, Hubei, China. He received his M.S. degree in Information Security in 2022 from the University of Glasgow. He is currently a Ph.D. student at the Faculty of Innovation Engineering of Macau University of Science and Technology in Macau, China. His research interests are related to Artificial Intelligence, Pattern Recognition, Intelligent Science and Systems and information security.

Bingqin Pan, graduated from the National University of Singapore with a Master's Degree majoring in Industry 4.0, and graduated from Wuhan University with a Bachelor's Degree majoring in Electrical Engineering and Automation. Now he working in the Institute of Future Transportation Engineering in Sichuan Digital Transportation Technology Co., Ltd, working as an engineer in Technology and Innovation Department. His research interests include Big data, digital twin and intelligent transportation.

Liping Bai is a researcher at Macau Institute of Systems Engineering in the Macau University of Science and Technology, China and she is currently an Associate Professor. Her research interests are related to Industry Engineering, Operations Research, Production Planning and Control, Information system, E-Commerce. She has published research papers at national and international journals, conference proceedings as well as chapters of books.

Guangbin Wang is a professor of Lingnan Normal University, a member of the Academic Committee of Lingnan Normal University and a leader in robotics. His research direction mainly focuses on high-end equipment such as wind turbines and aviation engines, conducting research on complex electromechanical system dynamics, modern sensing and detection, equipment health monitoring and fault diagnosis, and mechanical optimization design and remanufacturing for equipment maintenance.

Deyun Mo received his M.S. degree in Mechanical Engineering in 2022 from the Guangdong University of Technology. He is currently a Ph.D. student at the Faculty of Innovation Engineering of Macau University of Science and Technology in Macau, China. He is a senior experimentalist in School of Mechanical and Electrical Engineering at Lingnan Normal University. His research interests include high speed machining equipment, machine learning, and artificial intelligence.