frontiers | Frontiers in Manufacturing Technology

TYPE Original Research
PUBLISHED 16 March 2023
DOI 10.3389/fmtec.2023.1154263

OPEN ACCESS

EDITED BY
Pritesh Shah,
Symbiosis International University, India

REVIEWED BY
Harikrishnan R,
Symbiosis International University, India
Sushma Parihar,
Symbiosis International University, India
Pedro Neto,
University of Coimbra, Portugal

*CORRESPONDENCE
Sanjay Nambiar,
sanjay.nambiar@liu.se

SPECIALTY SECTION
This article was submitted to Digital Manufacturing,
a section of the journal
Frontiers in Manufacturing Technology

RECEIVED 30 January 2023
ACCEPTED 28 February 2023
PUBLISHED 16 March 2023

CITATION
Nambiar S, Wiberg A and Tarkian M (2023), Automation of unstructured production environment by applying reinforcement learning.
Front. Manuf. Technol. 3:1154263.
doi: 10.3389/fmtec.2023.1154263

Automation of unstructured production environment by applying reinforcement learning

Sanjay Nambiar*, Anton Wiberg and Mehdi Tarkian

Linköping University, Division of Product Realisation, Department of Management and Engineering, Linköping, Sweden

Implementation of Machine Learning (ML) to improve product and production development processes poses a significant opportunity for manufacturing industries. ML has the capability to calibrate models with considerable adaptability and high accuracy. This capability is specifically promising for applications where classical production automation is too expensive, e.g., for mass customization cases where the production environment is uncertain and unstructured. To cope with the diversity in production systems and working environments, Reinforcement Learning (RL) in combination with lightweight game engines can be used from initial stages of a product and production development process. However, there are multiple challenges such as collecting observations in a virtual environment which can interact similar to a physical environment. This project focuses on setting up RL methodologies to perform path-finding and collision detection in varying environments. One case study is human assembly evaluation method in the automobile industry which is currently manual intensive to investigate digitally. For this case, a mannequin is trained to perform pick and place operations in varying environments and thus automating assembly validation process in early design phases. The next application is path-finding of mobile robots including an articulated arm to perform pick and place operations. This application is expensive to setup with classical methods and thus RL enables an automated approach for this task as well.

KEYWORDS

reinforcement learning, unity game engine, mobile robot, mannequin, production environment, path-finding, design automation

1 Introduction

Product customisation based on unstructured and uncertain requirements has increased the challenges in developing product and production processes. Knowledge-based engineering and various design optimization methods have proved to be efficient over the years but only up to a certain extant before they become too expensive. With more demanding and complex market demands, a more flexible and robust approach is required to take the next leap in product customization. An approach utilizing Reinforcement Learning (RL) to perform preliminary exploration and validation is outlined in this paper. With the conventional knowledge- and rule-driven automation approaches, the models are created according to the provided parameters which are later optimized according to the design space. For an unstructured and uncertain environment however, this task becomes too time-consuming and expensive because the environment becomes difficult and sometimes impossible to model in a flexible and reusable way. The benefit of RL

in this regards is the possibility to automatically adapt to unstructured and uncertain environments in a dynamic fashion.

Utilizing game engines like Unity allows for fast training of agents, especially when compared to legacy CAD and CAE software. The Unity platform supports dynamic multi-agent interaction (Juliani et al., 2018), with the possibility of generating a virtual world with natural physical forces. With legacy software on the hand, many type of engineering evaluations are done intuitively and iterative by experienced expert users. This approach is computationally expensive and manually exhaustive, furthermore, it is not certain that optimal solutions can be achieved.

This study explores the use of RL, the agent setup, and its adaptability in an unstructured production setting using a game engine as the platform. One of the key contributions of this work to the field of study is its open-source nature, making all elements of the project freely accessible to the public. This paper outlines the use of RL in two separate industrial applications, which are discussed in detail in subsequent sections.

1.1 Mobile robot control

With advancements in automation, it is possible to program a robot to perform a predefined task. As to path-finding, the conventional method relies on the data from the surrounding environment, and the reduced efficiency of the algorithms in a dynamic environment restricts the development of a better approach in this field (Bakdi et al., 2017; Yu et al., 2020). In the case of a mobile robot, conventional path-finding algorithms like the Dijkstra algorithm and A* are effective for a static work environment (Karur et al., 2021). However, when it comes to stochastic and dynamic environments, there is a need to investigate a different approach which can validate the process/product development method. With the introduction of RL and a lightweight game engine, it is possible to perform the path-finding operation for a mobile robot platform integrated with a Robot Operating System (ROS). In real-life applications, sensors similar to industry tools are used to perform the simulation. Implementing RL for path planning and obstacle avoidance of a mobile robot not only improves the process planning efficiency but also can be further developed for a safe human-robot interaction. The second application presented is path-finding and pick-and-place operation of a mobile robot by utilizing Robot operating system (ROS) integrated with Unity.

1.2 Mannequin control

Assembly procedures must be validated when generating new designs. Legacy programs can evaluate these features, however there is a need to setup the context of the assembly and optimize the position of mannequin's limbs to validate whether the assembly operation is possible and ergonomically safe (Hanson et al., 2014). Needless to say, this is a repetitive and time consuming task since the design and environment will change repeatedly during the product development process, from early conceptual to later design phases. By combining RL and physics based game engines, it is possible to setup a framework which can automatically load new context and designs and then explore and validate these in an automated way. In order to benefit from such a setup, it is required that the automated process is able to mimic human motion and also evaluate ergonomic motions. This is the starting point for the presented mannequin application. In the paper, the mannequin performs tasks in unstructured environments in a automatic fashion.

2 State of the art

A challenge with unstructured production environments is fuzzy objectives and constraints. With the current development in the manufacturing, higher machine-human interaction is desired. Numerous research is being conducted in this field for faster production and customised production line. The challenge is accurate sensing, obstacle identification, and modelling of crowded unstructured environments. The concept of digital twins and simulation models have made it possible to program different robots or machines to work synchronously in a cluttered environment. But with the introduction of a dynamic environment, these models fail to achieve their objectives for undefined contexts. The models need to adapt to real-life conditions dynamically. With the technological transformation aligned with Industry 4.0, a smart factory should use advanced technologies such as AI and ML to automate, optimize and analyze the data in a dynamic a real time fashion. Modern industrial systems' data creation has grown rapidly, reaching about 1,000 Exabytes annually (Tao et al., 2018). The robustness of production planning control may be enhanced by ML since data-included knowledge may aid in handling both predictable and unforeseen situations (Cadavid et al., 2019). The conventional method of supervised learning has proven to be useful in a static environment, but as it comes to an entire factory floor with multiple mobile robots, automated guided vehicles and humans working simultaneously, the capability of predicting the uncertainties is arguably poor. The model needs to adapt to new situations rather than from the pre-created data sets and rules. Considering the classical method of controlling the robot motion, it is required to take into account all potential scenarios in the collaborative environment (Liu et al., 2021). However, research with a focus on deep learning where the robots can learn their control strategies on their own with minimal human interventions has opened up the real-world application of RL (Gu et al., 2017). Since RL and the application of production is still at the cutting edge of AI and product development research, there is a need for investigation on generalization of RL in complex scenarios, with real physics and enhanced sensors for smoother controls in customized industrial tasks.

2.1 Mobile robot control

Recent research has sought to use deep RL to address a range of continuous motor control issues, such as motion planning, object detection, and collision avoidance for industrial robots directly from sensory input (Mnih et al., 2016). However, these methods are computationally expensive and heavily time-consuming (Meyes et al., 2017). Tools for robot manipulations are usually built in-house and are often computation-intensive. The low-level mapping

of sensors to actuators must be frequently iterated to reach calibration. It can be challenging for a programmer to convert their understanding of how to perform a task into a language that the robot can comprehend (Smart and Kaelbling, 2002). In the case of a mobile robot, the challenging parts are the navigation of the robot platform towards a goal avoiding obstacles and performing the desired operations without any collision or singularities. Presented research that utilized Q-learning and PID for mobile robot trajectory generation was effective in simple cases, but performed poorly in more complicated scenarios. (Wang et al., 2020).

2.1.1 Path-finding and navigation

Creating a series of motion orders to transport a robot from its present start point to a specified target location in a predetermined environment is the classical mobile robot navigation problem (Xiao et al., 2022). One of the oldest methods used to find the shortest path in a known environment is by using Dijkstra's algorithm. This is a type of tree search algorithm, which creates nodes in a 2D environment and uses the weights of the edges to calculate and minimize the distance between the source node and all other nodes Dijkstra (1959). Another efficient method is the A* algorithm, which is an extension of Dijkstra's algorithm using heuristics preventing searching of unnecessary nodes from searching (Goyal et al., 2014). This is accomplished by weighting the cost values of each node's distance from the target endpoint by its euclidean distance. Therefore, only routes that are typically taking the right course will be assessed. Apart from these two methods, there are other methods using probabilistic methods like the rapidly-exploring random tree technique or by discretization of working space into small grids and using occupancy grid maps. However, the A* and D* algorithms failed to predict/plan a path in a dynamic environment, but the AD* algorithm was proposed which can work perfectly in both static and dynamic environments (Ng et al., 2020).

A pioneering proof of concept for totally replacing the conventional sense-plan-act architecture with a single learned policy was attempted using end-to-end ML for fixed goal navigation (Thrun, 1995). The first approach which learns from a supervised demonstration to navigate a robot platform through obstacle-cluttered environments to reach the provided target. In this method, a network is trained using imitation learning which mimics the ROS navigation package utilizing Dijkstra's algoritm (Pfeiffer et al., 2017). However, with low-dimensional LiDAR data, RL where the robot learns by trial-and-error proved to be working as efficiently as imitation learning, but without the need for demonstrations (Tai et al., 2017). Moreover, a DRL approach facilitated changing goals and environment using onboard sensors and without classical localization or mapping of the environments (Zhang et al., 2017). For a dynamic environment, the most suitable approach to perform navigation seems to be with the application of DRL. An unstructured production line will be a perfect application for DRL, but for faster training involving natural physical forces, a game engine platform is the desired solution.

2.1.2 Trajectory planning

In industrial robots, trajectory planning is moving the tool centre point from point A to point B over time while avoiding object collisions. The most prevalent kind of robot employed in an industrial setting is the one with rotating joints called articulated robots. The framework of an articulated robot might be as basic as two axes or as complicated as ten axes, while the most common are the ones with six axes. The relationship between a robot's links and its position, orientation, and acceleration comes under robot kinematics, which can be separated into forward kinematics (FK) and inverse kinematics (IK). In the forward kinematics method, the position of the end effector is computed from the kinematics equations and joint variables of each link. IK is the exact opposite of FK, where the end effector position is specified and the kinematics equation is used to calculate the joint variables for each link (Singh et al., 2017). With the development of AI and artificial neural networks along with matured sensors and improvement in image recognition and vision systems, the next step in the field of trajectory planning of the end effector of a robot arm is the implementation of ML in both path plannings as well as for validation. However, the requirement to properly translate the robot position in Cartesian space to the angular and/or translational displacements in joint space is one of the most important issues in motion planning and motion control (Li et al., 2021). Most of the research focused on the application of DRL in robotics is to control the position and/or orientation of end effector in the Cartesian space, and later use the IK method to translate this into joint space (Yang et al., 2021). There have been methods developed to approximate the IK solutions in terms of joint values (Phaniteja et al., 2017), although it is very time consuming, especially as the robot's degrees of freedom rise, it is challenging for DRL to obtain precise enough IK solutions through a finite number of excursions because of the randomly sampled experience updates (Li et al., 2021). To solve the problem of time consumption and visualization, the contemporary game engine (Unity) aids the creation of a virtual environment which allowed for simultaneous training of several robot arm instances sharing their experience as one brain. This is an important breakthrough in the automation of an unstructured production environment where the robots can learn and discover the most efficient and fault-tolerant approach to perform an operation (Matulis and Harvey, 2021).

2.2 Mannequin control

Since its inception, the manufacturing sector has undergone continual innovation, allowing it to achieve increasing levels of performance to accommodate more demanding clients. Nonetheless, despite rapid technical advancement, humans remain an important resource in most industrial settings. Humans are an important variable during the design, development and operation of a production environment. Commonly the human factor is related to ergonomic evaluations and it is a necessary link for worker wellbeing in monitoring, production planning and human-robot collaboration. Conventionally, all these evaluations are performed offline and the model cannot adapt to a dynamic environment (Montini et al., 2021). Human-robot collaboration and ergonomic assessments for adaptive

automation are two interesting subjects in the use of controlling mannequins.

2.2.1 Human-robot collaboration and adaptive automation

With an increase in automation and the use of robots in the manufacturing industry, human-robot collaboration is on the rise and research on the application of smoother and safe human-robot interaction is increasing. An approach using camera vision and wearable sensors and an artificial neural network for static, dynamic, and composed gesture classification of human upper body gestures in an environment with human and robot collaboration proposes a method to perform operations like holding an object to/for an assembly (Neto et al., 2019). Similarly, a human-robot collaboration framework that allows for real-time adaptation to dynamic human elements and intentions using DNN for image recognition improved the ergonomics and reconfigurability of a production process (Kim et al., 2019). The approach using DCNN for improved planning and control of robot motion, predicting the actions of a human using visual observation of worker's motion achieved a high recognition accuracy (Wang et al., 2018). The use of RL in human-robot collaboration to solve the problem of task allocation and sequence allocation allowed the model to cope with dynamic human model features without the requirement for pre-training data (Zhang et al., 2022). This study demonstrates that the RL technique has a lot of potential for dynamically dividing human-robot collaboration activities.

2.2.2 Ergonomic analysis

One of the primary characteristics of a production environment is the ergonomics of both humans and robots. Bio-mechanical stress is a major risk factor in the production environment and a potential cause of musculoskeletal disorders. Various numerical calculation methodologies have been developed over the years with a focus on an efficient and safe production environment. However, most of these methods are based on checklists and are subjective. Which often makes it difficult to include this factor while designing a production/process line. A tool called ema (editor for manual work activities) implements a modular method to create and define human work activities based on "complex operations" that reflect an aggregation of single elementary motions geared at carrying out a specific work activity (Fritzsche et al., 2011). This tool can convert typical job descriptions into a series of natural movements, which can later be used to perform ergonomic evaluations. After the outbreak of research in digital twins, a lot of research has been conducted to develop a digital twin of various machines. However, a methodological framework that uses digital twins of humans, aids in the monitoring and decision-making process regarding the ergonomic performance of manual production lines (Greco et al., 2020). ML approaches can be employed for process planning and system design in addition to traditional product design and job assignment. Anghel et al. (2019) offers experimental research on the ergonomic workstation redesign process for automobile assembly lines by merging neural networks with rapid upper limb assessment (RULA) analysis. Apart from the conventional approach of digital simulation, camera sensors and recorded videos are used to automatically compute RULA scores based on computer vision and ML (MassirisFernández et al., 2020).

2.3 Research gap

As depicted above, there has been a major leap in cutting edge research in the automation of machine elements in production processes. RL frameworks are currently either implemented in legacy CAD software or non-public in-house software. Creating a virtual environment and training in legacy software is computationally expensive which limits the performance of the RL. There is a need for an open source environment in order to achieve result validation. Moreover, the open source alternative should be fast and agile in order to instigate easy implementation of open source ML and RL algorithms. In this paper, a Unity implementation is presented and the data and models are shared with the research community.

3 Methodology

The conventional product development process (PDP) is a structured approach for creating and launching new products. This process typically begins with the concept development stage and culminates in the launch or production of the new product as shown in Figure 1. In a virtual setting, humanoids and robots can play a critical role in evaluating this process, specifically in the design and testing stages. The use of a humanoid mannequin, for example, can begin in the early stages of the PDP, such as the preliminary design stage. The mannequin can be used for visualization and validation purposes, including worker accessibility and ergonomics evaluations. On the other hand, the application of a path-finding mobile robot typically comes later in the PDP, mainly during the detailed design stage and process planning procedures.

In order to effectively train RL agents, it is important to utilize dynamic digital environments that offer flexibility and versatility. Unity game engine is utilized to create an unstructured production environment for two use cases. This allows the use of an open-source project called *Unity machine learning agents toolkit (ML-Agents)* to train and interact with the agents in the virtual environment (Unity-Technologies, 2019). For RL, it is required to define three entities called the observations, actions, and reward signals in an environment. Observations in Unity can either be numeric or visual, where the numeric observations as the name suggests are numbers like the coordinates of objects, distance between objects, force, velocity, etc. While the visual observations are the images from the camera sensors attached to the agent. Similar to observations, actions are also divided into continuous and discrete. The reward signals are scalar values used to define the performance of an agent with positive and negative values for reward and punishment respectively. However, the training of neural networks is not done by Unity, as it is only a platform to observe and simulate the environment while data is transferred to python trainers by using a communicator. This communicator connects the environment in the Unity scene with the python low-level API which serves as an interface between the python trainer and the learning environment. The necessary RL algorithms are stored in the

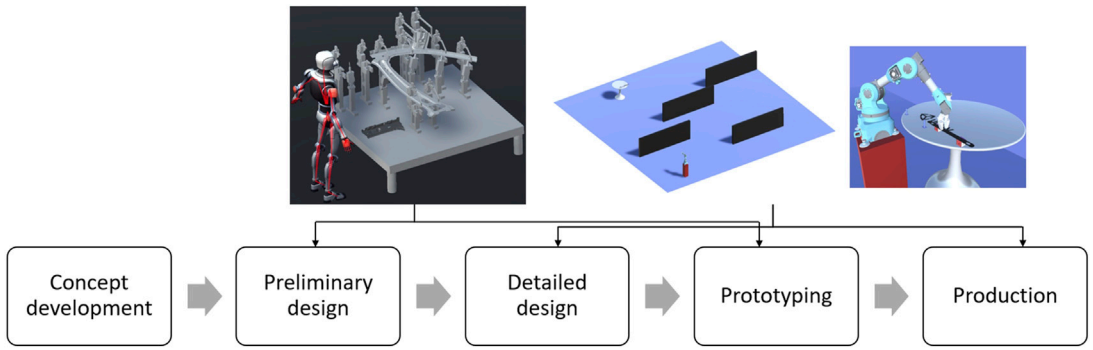

FIGURE 1
Possible application of trained mannequin and mobile robot using RL in a product development process.

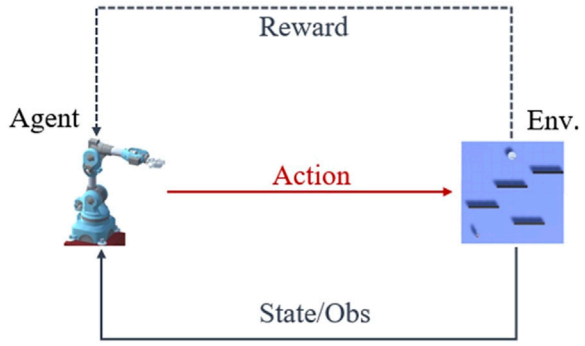

FIGURE 2
Basic structure of a RL training cycle.

python trainers as a python package. The training cycle of the RL method is shown in Figure 2, depicting the flow of information between an environment and an agent. This method is utilized to control the movement of a mannequin's hands and torso during assembly operations, as well as for path-finding and trajectory planning on a mobile robot in combination with the Robot Operating System (ROS).

3.1 Mobile robot control

In this test case, Niryo one robot running on full Niryo One ROS stack is used to perform a pick and place operation on a sample fixture. An URDF model of the robot is imported into the Unity scene and a two-way communication is set up with ROS for trajectory planning. Along with the robot, the part required for pick up and a fixture designed in CATIA is imported into the scene. The Niryo one robot is placed on a platform as its child game object. While the robot and fixture are at either end of a plane, there are multiple dynamic obstacle walls in the scene as shown in Figure 3. The main objective in this test case is for the robot to find a path towards the fixture without any collisions and perform a pick-and-place operation. While the trajectory of the robot is planned using ROS, the path-finding of the robot is achieved by using RL. While training, the robot and fixtures are spawned at random positions at either end of the plane. To replicate a dynamic production environment, four obstacle walls are instantiated between the robot platform and the fixture table. Similar to the robot the walls are also moved to a different position along one axis. Since Unity allows simultaneous training of multiple agents, 120 agents (robot platform) is set up in the scene training parallelly while sharing a single brain.

3.1.1 Observation space

For the agent to behave properly, it should be able to form a relation between the observations recorded and the reward signals for each action taken. Unity has the flexibility to use vector observation, ray-cast sensors and camera observations to record the state of the agent at each time step. In this environment, a single ray-perception sensor is used to detect the dynamic obstacles. Along with the ray cast, the distance between the robot and both the fixture and pick-up object is taken as observations. The velocity of the robot platform is also added as an observation to make sure the platform does not fall off the plane. The ray perception sensors script in the Unity has the provision to add the number of rays, and the angle between them along with the detectable tags. The tags of the walls are added as input to the ray-perception sensor. A total of four vector observations along with a ray cast observation as shown in Figure 4 is used to generate the actions for the agent.

3.1.2 Action space

The platform of the robot is considered a rigid body and the motion of this object is achieved by using force vectors. Since we are considering the plane as a flat surface, only the motion in the *X* and *Z* direction is taken into account. Y is not taken into account because the table height is fixed. A force vector of (X, 0, Z) with X and Z as continuous vector action is used to move the mobile robot platform along the plane.

3.1.3 Reward function

The PPO RL method optimizes an agent's decisions so that the agent obtains the maximum cumulative reward over time. The more effective your incentive mechanism, the more effectively your agent will learn. The rewards signals used in the mobile robot environment are shown in Table 1.

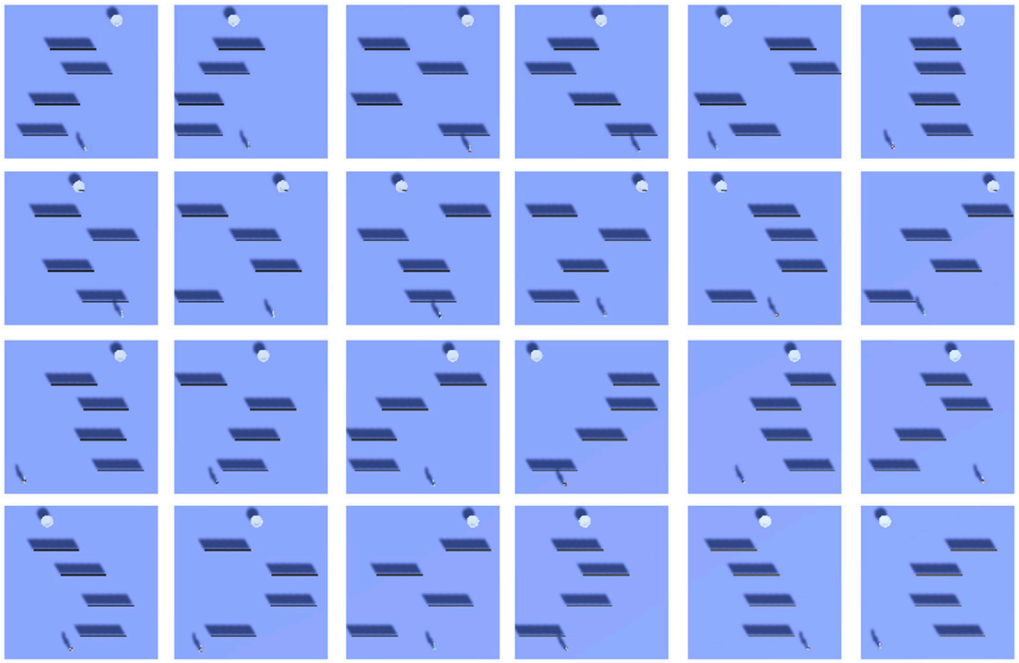

FIGURE 3
Multiple environment created with random element positions for parallel training.

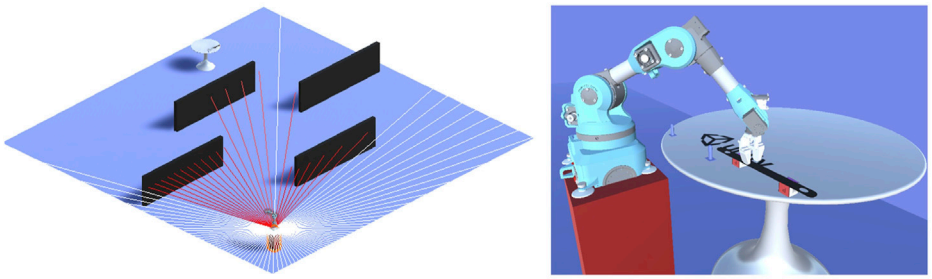

FIGURE 4
Observation sensors (ray perception sensors) detecting obstacles around the platform (left) and close up view of the robot along with pick-up object.

TABLE 1 Rewards signals and its description for mobile robot training environment.

Reward	Description
+1.0f	The maximum reward when the agent reaches near the fixture and pick-up object such that both the objects are within the robot's working envelop. The episode ends after this reward function
−1.0f	The maximum negative reward when the robot falls over the floor. The episode ends after this reward function
−0.1f	Collision with obstacle walls, but training continues without ending the episode
−1.0f/MaxStep	An increment at each step to promote faster or shorter episode (MaxStep = 5,000)

3.2 Mannequin control

The second test case is to control the hand and torso of a mannequin, which is an asset from the Unity asset store. The main objective is to train the mannequin's right hand and torso to pick a sheet metal and place it on a fixture. The environment setup for the training of this mannequin is as shown in Figure 5, where the black sheet metal on the table is the pick-up object

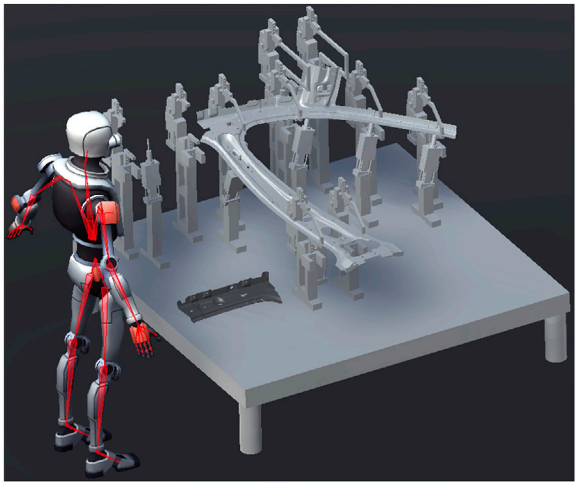

FIGURE 5
Virtual environment created in Unity with fixture and sheet metal for the training of mannequin's hand and torso control.

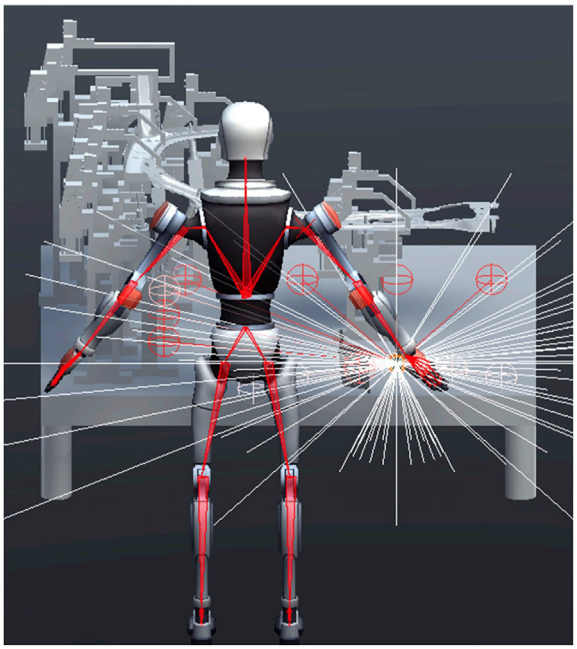

FIGURE 6
Multiple ray-perception sensors detecting obstacles, fixture, sheet metal and drop-off location.

which needs to be placed near the top part of the B-pillar. The approach presented involves the implementation of inverse kinematics (IK) and animation controllers. An animator controller is connected to the mannequin in order to assign a rig and Fast IK Fabric constraint. The Fast IK Fabric constraint in Unity controls the hierarchy of simple game objects, by specifying the chain length/number of limbs. The wrist, forearm, upper-arm, shoulder and rib are assigned as chain elements and a source game object to specify the location of the palm/tip (arm-mover_target).

The application of IK allows a human-like motion for heuristic control. However, to control the motion using a force vector similar to the previous test case, a rigid body is required. But Unity does not allow the elements connected using Fast IK Fabric constraints to be manipulated by physical forces. Hence a dummy game object with a rigid body component is created and it is connected to the arm-mover_target by equalising their transform position and rotation, allowing the wrist to follow the dummy game object. Afterwards, the agent tries to control the motion of this dummy game object. The environment is created such that the agent is intended to perform three operations sequentially.

- Reach the target by moving the hand and create a fixed joint.
- Move the hand along with the connected sheet metal to the position right on top of the fixture (drop-off location)
- Align the sheet metal in the correct position at the drop-off location.

3.2.1 Observation space

Unlike the first case of mobile robot path-finding, a mannequin's hand have multiple degrees of freedom and observations recorded must be robust enough to track all these motions. Adding a camera sensor as the source of observation will have a high resemblance to a real-life situation. However, when it comes to spaces where, there is no direct line of sight for the mannequin, this method for collecting observation fails. Hence a set of five ray-perception sensors are attached to the dummy game object as seen in Figure 6. The radius of the rays, ray length, and the vertical offset is set in such a way that it is possible to detect every object within the mannequin's arm's length. The colliders of the sheet metal, fixture and table are given different tags and layers to make it possible for the ray cast to identify individual elements separately.

In addition to observations of ray-casts, one conditional observation of distance and a Boolean value are added to the collection. In each episode, the distance between the palm and the sheet metal is recorded as an observation until the pick-up operation is completed. Afterwards, the distance between the sheet metal and the drop-off location is added. The Boolean observation will be set to false before the pick-up and true afterwards.

3.2.2 Action space

In this test case, the actions taken are discrete values which are used to create a force vector to apply force to the rigid body. Usage of discrete actions allows the agents to explicitly avoid actions that would result in a negative reward. The range of discrete actions varies from 1 to 6 (for 6 degrees of freedom) for each step. These actions specify the direction of motion of the palm, and the velocity of 0.1 is adjusted in that particular vector. Since the hand is connected to the dummy game object, this velocity change is applied to the rigid body of the dummy object. Additionally, the motion of the dummy game object can also be controlled using manual heuristics.

3.2.3 Reward function

The reward signals used for the training of mannequin's hand is presented in Table 2. As pointed out earlier, the agent has three

TABLE 2 Rewards signals and its description for mannequin hand and torso control for assembly operation.

Reward	Description
+2.0f	Adds the reward when the hand touches the sheet metal on the table
+2.5f	Adds a reward when the sheet metal is at the drop-off location. The episode ends after this reward function with a maximum cumulative reward
−1.0f	Too much force added to the hands causing the Inverse Kinematics connection element to break. The episode ends after this reward function
−0.1f	Collision of hand with anything other than the target
−5.0f/MaxStep	An increment at each step to promote faster or shorter episode (MaxStep = 5,000)

```
MobileRobot:
    trainer_type: ppo
    hyperparameters:
      batch_size: 512
      beta: 1.0e-2
      buffer_size: 102400
      epsilon: 0.2
      lambd: 0.93
      learning_rate: 1e-3
      learning_rate_schedule: linear
      num_epoch: 3
    network_settings:
      normalize: false
      num_layers: 3
      hidden_units: 256
      use_recurrent: false
      memory_size: 128
    reward_signals:
      extrinsic:
        strength: 1.0
        gamma: 0.99
    time_horizon: 1024
    summary_freq: 50000
    max_steps: 2550000
```

FIGURE 7
Hyper parameters for mobile robot path-finding training.

```
MannequinControl:
    trainer_type: ppo
    hyperparameters:
      batch_size: 1024
      beta: 1.0e-2
      buffer_size: 102400
      epsilon: 0.2
      lambd: 0.93
      learning_rate: 1e-5
      learning_rate_schedule: linear
      num_epoch: 3
    network_settings:
      normalize: false
      num_layers: 3
      hidden_units: 256
      use_recurrent: false
      memory_size: 128
    reward_signals:
      extrinsic:
        strength: 1.0
        gamma: 0.99
    time_horizon: 1024
    summary_freq: 50000
    max_steps: 30000000
```

FIGURE 8
Hyper parameters for mannequin hand and torso control training.

objectives in a single episode hence an increment in the magnitude of the reward signals are required for the second and third task. Balancing positive and negative rewards is important for consistent training because, like in this case, if an agent receives several rewards, the cumulative sum of all rewards is applied for that step.

3.3 Training configuration

Figures 7, 8 show the training configuration file of the mobile robot and mannequin control respectively. Two main trainer type provided in the ML-agent package is Proximal policy optimization (PPO) and Soft actor-critic (SAC). SAC instigates an experience replay mechanism which will result in confusing behaviour as both environments have non-stationary dynamic objects. Hence a policy gradient method for RL called proximal policy optimization (PPO) trainer is used for both of the environments. Compared to the mobile robot environment, the mannequin control environment has a higher number of observations and multiple rewards within an episode. So, a higher buffer size is used to collect all the experiences before updating the policy. The mobile robot have only one objective, hence a smaller training step of 2.55 million is used, while the mannequin has three sequential objectives to finish an episode and the training is done for 30 million iterations.

4 Result

4.1 Mobile robot control

The navigation system in Unity supports the creation of navigation meshes on the scene elements and plans a path depending on the dynamic obstacles according to the designed environment. This method is sufficient when the only objective is to reach a known target location. However, in the case of a complex environment with multiple obstacles and elevations, the generated mesh creates many non-walkable areas. When iterating between environments, the mesh generated for Navmesh agent needs to be baked separately for each case. The agent is trained to perform a task

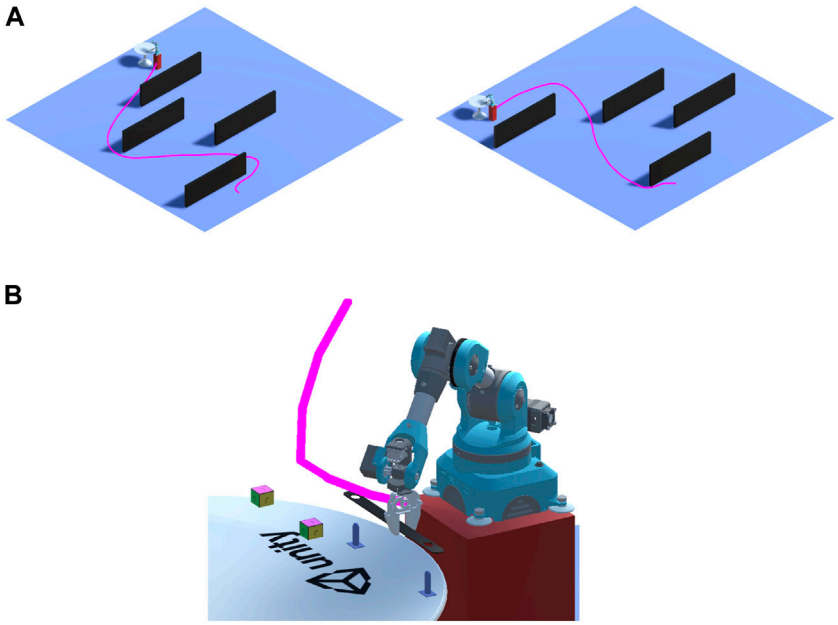

FIGURE 9
Path-finding of mobile platform using RL in a dynamic environment **(A)** and trajectory planned by ROS and executed in Unity after path-finding sequence **(B)**.

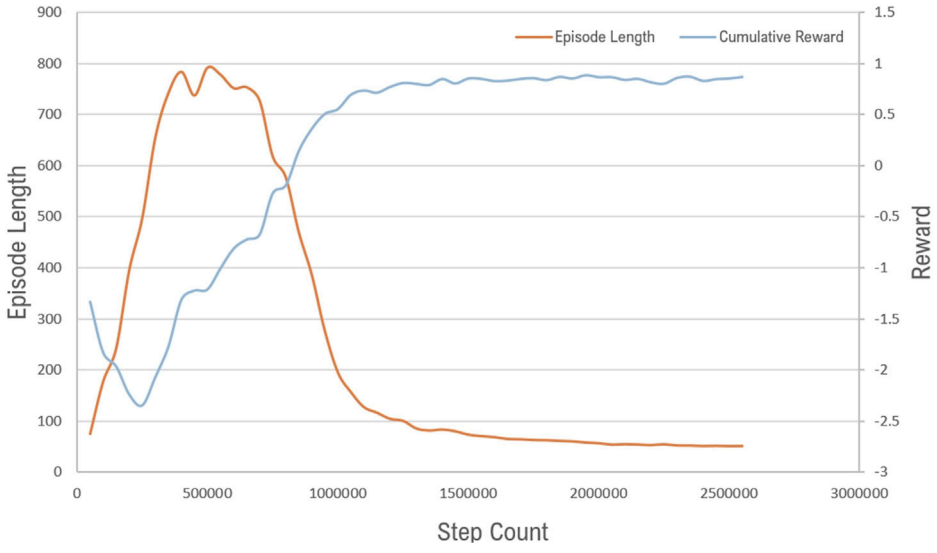

FIGURE 10
Cumulative reward and episode length of the training of mobile robot for path-finding.

in an unknown dynamic environment making it more independent compared to the navmesh approach in Unity.

Following the environment setup as shown in Figure 3, the trained model with a trail renderer for two different environments are presented in Figure 9A. Since we are considering the location of the targets (Fixture and pick-up object) unknown, the platform is required to search the design space and move towards the table with the correct orientation such that the pick-up and drop-off locations are within the working envelope of the robot. The cumulative rewards and episode length of the training process are shown in Figure 10. From the graph, the convergence is visible after 1.5 million steps

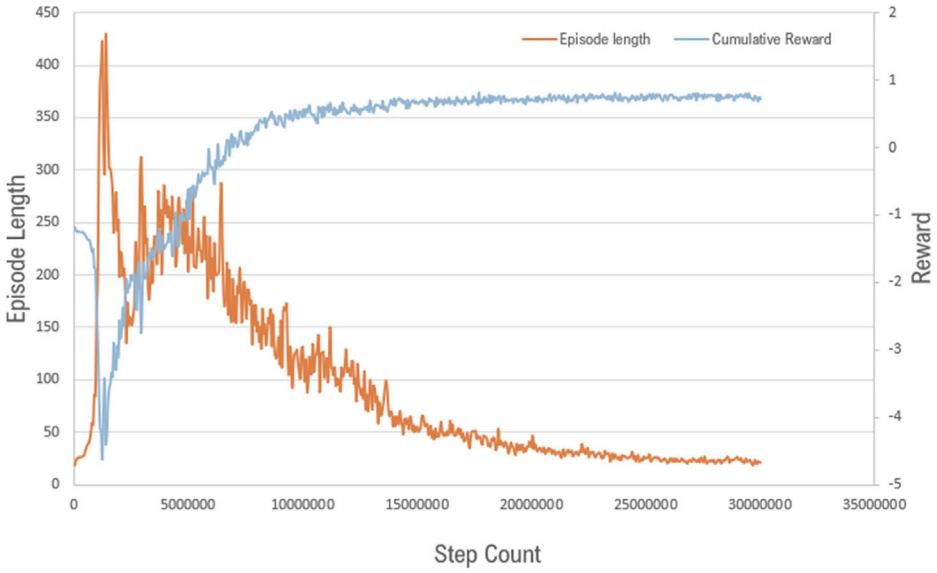

FIGURE 11
Cumulative reward and episode length of the training of mannequin's right hand and torso for assembly operation.

accompanied by a reduced episode length. Manually iterating the training configurations improved the total training time (~ 1 h) and faster convergence in the cumulative rewards. Once the model is trained in the dynamic environment, it is possible to use the trained neural network (brain) on agents in environments with topographical or morphological changes.

The Unity have a version of ROS melodic that is available in a docker image. This version has all the necessary packages and modules to set up a ROS workspace. A ROSConnection component in Unity provides the functionality required to publish, subscribe, or execute a service utilizing the TCP endpoint ROS node. Once the robot reaches close enough (within the working envelope) to the target, the location of the pick-up and drop-off location along with joint configurations is passed as input to ROS. An array of trajectory coordinates for a pre-grasp pose, grasp pose, pick pose and place pose is sent back from the ROS side to a trajectory-planner script. This array of trajectories is executed to perform a pick-and-place operation on a fixture as shown in Figure 9B.

4.2 Mannequin control

With the help of the animation rigging feature in Unity, it is possible to rig the limbs of the mannequin and apply predefined constraint components to these limbs. One bone is created for each body part of the mannequin and is displayed as red bones in Figure 5. In this test case, the Fast IK Fabric constraint is applied, which controls the forward kinematic hierarchy of five bones into an IK expression. This expression is attached to an empty game object, and the manipulation of its transform values created human-like motion. Since the transform values of this empty object are changed to move the right hand and torso, it is impossible to move it using rigid-body forces. As specified in the methodology, attaching this empty game object to another rigid body and controlling the motion of the rigid body resulted in a smooth human-like motion.

Training the hand of a mannequin to reach a location is a relatively easy task for the agent. The presented test case comprised multiple objectives however, each of which had to be completed sequentially. Hence training the hand to move towards the correct location after picking up increased the complexity and required calibration of the training parameters. The cumulative reward and the episode length of the training process of the mannequin's right hand are shown in Figure 11. A convergence is reached for the cumulative rewards after 20 million iterations. However, the episode length kept decreasing until 28 million iterations. Even though 100 agents are trained in parallel, the entire training process is comparatively more time consuming than the previous test case because of multiple objectives. On average, it takes approximately 5.5 h for the training to complete on a computer with 16 GB of RAM and a RTX3080 graphics card.

The trained agent in an episode performs a sequence of actions: target detection, approach, pick-up, and drop-off. This is shown in Figure 12. The agent is trained in a dynamic environment where the location of the sheet metal changes. However, the performance of the agent is not optimal in terms of the number of steps taken per episode. To improve performance, the model was retrained in a static environment, where the sheet metals position is fixed. This resulted in shorter, faster episodes. The improvement was demonstrated through comparison of the dynamic and retrained static models, using three test cases as shown in Figure 13. The step count for 20 episodes was recorded for both models. The re-initialization of the static models took less than 10 min.

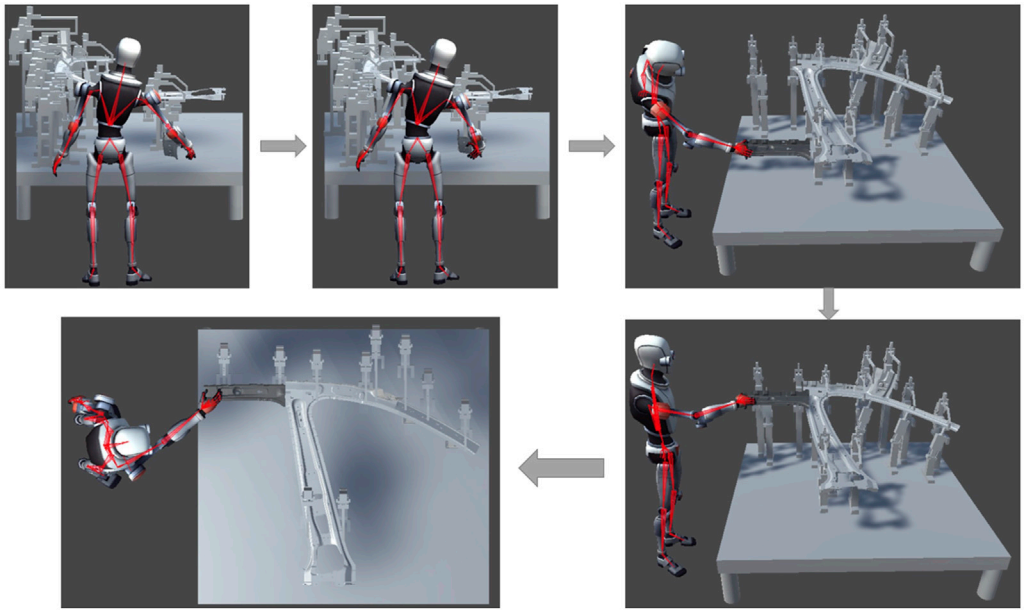

FIGURE 12
Sequence of operations performed by a trained mannequin, including approaching an object, picking it up, transporting it to a designated drop-off location, positioning itself at the drop-off location, and finally releasing the object at the final location.

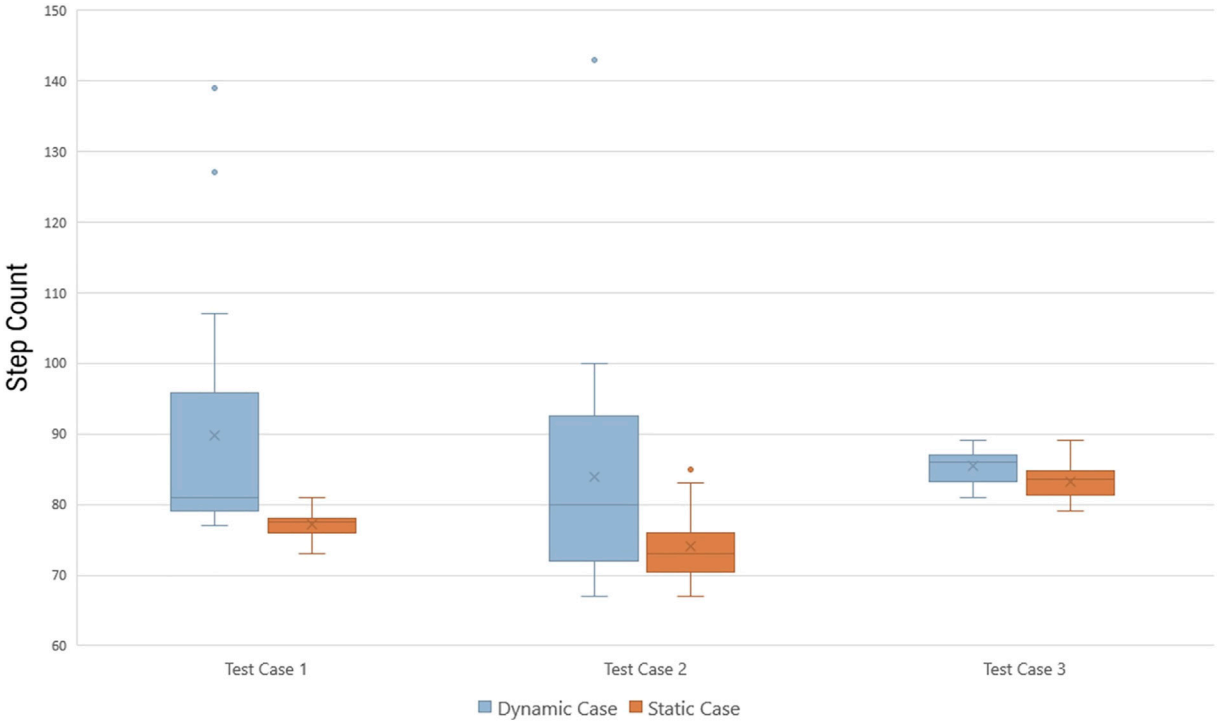

FIGURE 13
Comparison of steps taken in each episode for model trained in dynamic environment and re-trained model for static environment in three different test cases.

5 Discussion

This paper presents two RL test cases utilizing a physics-based game engine. Using Unity it is possible to train multiple agents simultaneously in a virtual environment. Unlike legacy CAD software, a game engine is fast and agile in for the proposed RL applications. Time taken for training the models depends on the complexity of the environment and objectives. But increased processing/computing power is reducing this negative aspect of data-intensive methods.

One of the most challenging parts in the application of RL is that the agent becomes reward greedy and finds a shortcut in acquiring more reward rather than performing the desired task. To counter this effect, with higher punishments/negative rewards, also has downsides since the agent ceases to explore the design space and settles for less optimal episodes and with the least punishment. To effectively address this challenge, an appropriately weighted reward system must be implemented to nudge the agent towards a desired behavior. Along with a logical reward system, the training parameters are required to be calibrated depending on the environment for the training results to converge in a reasonable time. With these two components adjusted, the agent is promoted to take risks and explore the design space. The observations regarding two industrial test cases showcased in this paper are discussed in the following sections.

5.1 Mobile robot control

Unlike conventional methods of automating a robot's motion/operation using legacy software and methods, the RL approach using a game engine proves effective in dynamic and unstructured environments. To perform a path-finding operation, the traditional approach using algorithms like A* and the Dijkstra algorithm works elegantly in static and less complex environments. Most of these approaches convert the working environment into a graph and estimate the shortest distance between two coordinates. While for a production environment with automated guided vehicles, humans and other dynamic entities working synchronously, the algorithms need to be adaptive. In this paper, the approach using RL for path-finding requires an initial setup time which will be compensated by a net gain in efficiency over time during application in different design concepts. Currently, mobile robot systems rely on explicit domain knowledge which are based on rules, ontologies, and decision trees. RL algorithms, on the other hand, learn from experiences without prior knowledge of the domain, increasing the reusability and flexibility of the models. This approach can greatly reduce the amount of manual model calibration when deploying it in different environments. Since the pick and place operation in the test case is done by integrating ROS with Unity, the possibility of singularity in the robot is not prioritized. In spite of that, the singularity problem was never encountered while testing the trained neural network.

The ROS integration successfully generates a trajectory between the pick-up and drop-off location using the robot's coordinate system. However, it does not account for obstacles in the environment. To overcome this limitation, two solutions can be explored: 1) incorporating multiple way-points within the working envelope to generate a more robust trajectory, or 2) using RL to control the robot arm as an agent, resulting in a more flexible pick-and-drop operation without relying solely on ROS. The trained neural network demonstrates robustness in handling dynamic components within the environment. However, the addition of new obstacles or elements may affect its success rate. To address this limitation, re-training the neural network with updated environmental parameters can lead to improved performance and a higher success rate. The developed framework will be useful to validate the performance of a new product or a production line depending on the existing observations and experiences. Implementing RL to control the motion of the robot arm by avoiding obstacles in the working envelope will be a valuable extension of this framework.

5.2 Mannequin control

Mannequin control is critical in the manufacturing industry because of human participation in the process/production line. Manual exploration and validation of a production line including ergonomic assessments, human-robot collaboration, and other elements are today performed utilizing software where extensive manual manipulation is required. RL implementation for mannequin control can have several applications, such as increasing worker efficiency by optimizing processes, determining optimal workstation placements in a production setting and simulating worker interactions in a dynamic environment. In RL, the agents can adapt to changes such as addition of new equipment or alterations to the workspace layout. Legacy simulation-based models require either manual input or pre-defined rules for specific pre-set environments. This setup necessitates significant calibration for any environment change. Multiple non-public in-house software has been developed over the years to automate and optimize this process. The methodology presented in this paper uses Unity and RL approach to control a mannequin's right hand and torso to perform an assembly operation. Instead of managing the joint values in each of the mannequin's limbs, the IK package provided by Unity allows the mimicking of human motion. The aforementioned method is effective in the digital simulation of a human in an unstructured environment using RL. The same approach can be applied to perform a virtual simulation of a humanoid.

The definition of observation sensors and the balancing of reward signals are the most difficult aspects of this test case. In a production environment, humans mainly rely on visual inputs from the eyes. Fixing a camera sensor in the eyes of the mannequin model will have a close resemblance to a real-life scenario. In this case, the motion control of the limbs, depending on the observation from it is head was inconsistent. Moreover, this method fails when the objective/target is outside the field of vision of the camera sensor. Using a ray perception sensor attached to the hand instead of the head overcame the shortcomings of camera sensors. If the targets are smaller than the diameter of the rays, then there is a high possibility for the targets to be undetectable for the agent.

Evaluating the effectiveness of this framework for a complex environment is crucial. The approach entails incrementally adding more detail and re-initializing the training from the previous neural

network. A solution to overcome this limitation could be to repeatedly implement this method to enhance complexity and ultimately apply it in a real-world scenario. Even though the intended outcome of this framework is to perform an ergonomic evaluation on a mannequin, the ergonomic calculation is not performed on the presented work. To perform a RULA assessment on the mannequin, it is possible in Unity to extract the adjustments made in the limbs relative to the force acting on the palm/wrist. Taking the cumulative score from the RULA assessment and adding it as a reward function to the currently trained mannequin model is the main factor in the future development of this framework.

One of the shortcomings of the current model is its inability to coordinate the use of both hands for tasks such as picking up objects. To address this limitation, a potential solution is to treat both hands as separate entities/agents and develop an algorithm to determine the optimal hand combination (i.e., left hand only, right hand only, or both hands together) based on factors such as the weight of the object, its position relative to the center of gravity of the mannequin, *etc.* The current framework has a limitation in terms of lower body control below the hips, which limits its use for Rapid Entire Body Assessment (REBA). Although the model can be used for RULA evaluation in the future, the inability to control the lower body hinders the ability to assess the risks associated with certain job operations. A potential solution to this limitation is to link an IK model for the lower body to the torso and hands, thus enabling a comprehensive assessment. The use of an agent that can mimic appropriate posture and motion in a production setting can significantly reduce the time required for traditional ergonomic assessments through repetitive actions.

6 Conclusion

This paper presents two applications of RL setup in a lightweight game engine; 1. A Mobile Robot application which is used to plan and control a robot in an unstructured industrial environment and 2. a Mannequin application used as a support tool for automated simulation of unstructured assembly processes. The presented work is an initial study but still manages to show that RL is a promising approach to be used in a dynamic and unstructured product development and production.

The RL approach of training an agent does not require a pre-training dataset and only requires the agent's starting state as input. This improves the possibility of automation in unstructured and dynamic production environments, enabling faster product/process line generation and evaluation. A lightweight game engine is used as a platform to setup a virtual environment and to perform the training. The applications are shared with the research community.

For future work, the Mobile Robot application needs to be applied in a complex industrial environment with exhaustive test series to achieve validation. The Mannequin application can be improved with, e.g., integrated ergonomic calculations and validated by implementing it inside a design process to test usability and benefits over existing approaches.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://github.com/DesignAutomationLaboratory/Mannequin-Control https://github.com/DesignAutomationLaboratory/Robot_Pick_and_Place.

Author contributions

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

Funding

This research has been made possible by the financial support received from Vinnova-FFI AutoPack 2.0 - Automatic packing and preparation of electrical wiring based on optimization and machine learning project (2020-05173).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Anghel, D.-C., Nițu, E.-L., Rizea, A.-D., Gavriluță, A., Gavriluță, A., and Belu, N. (2019). "Ergonomics study on an assembly line used in the automotive industry," in *MATEC web of conferences* (Les Ulis, France: EDP Sciences), 290, 12001.

Bakdi, A., Hentout, A., Boutami, H., Maoudj, A., Hachour, O., and Bouzouia, B. (2017). Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control. *Robotics Aut. Syst.* 89, 95–109. doi:10.1016/j.robot.2016.12.008

Cadavid, J. P. U., Lamouri, S., Grabot, B., and Fortin, A. (2019). Machine learning in production planning and control: A review of empirical literature. *IFAC-PapersOnLine* 52, 385–390. doi:10.1016/j.ifacol.2019.11.155

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numer. Math.* 1, 269–271. doi:10.1007/bf01386390

Fritzsche, L., Jendrusch, R., Leidholdt, W., Bauer, S., Jäckel, T., and Pirger, A. (2011). "Introducing ema (editor for manual work activities)–a new tool for enhancing accuracy

and efficiency of human simulations in digital production planning," in *International conference on digital human modeling* (Berlin, Germany: Springer), 272–281.

Goyal, A., Mogha, P., Luthra, R., and Sangwan, N. (2014). Path finding: A* or dijkstra's? *Int. J. IT Eng.* 2, 1–15.

Greco, A., Caterino, M., Fera, M., and Gerbino, S. (2020). Digital twin for monitoring ergonomics during manufacturing production. *Appl. Sci.* 10, 7758. doi:10.3390/app10217758

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)* (Singapore: IEEE), 3389–3396.

Hanson, L., Högberg, D., Carlson, J. S., Bohlin, R., Brolin, E., Delfs, N., et al. (2014). "Imma–intelligently moving manikins in automotive applications," in *Third international summit on human simulation (ISHS2014)*. Tokyo, Japan: doi:10.13140/2.1.4225.9203

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., et al. (2018). *Unity: A general platform for intelligent agents*. San Fransisco, United States: arXiv. doi:10.48550/arXiv.1809.02627

Karur, K., Sharma, N., Dharmatti, C., and Siegel, J. E. (2021). A survey of path planning algorithms for mobile robots. *Vehicles* 3, 448–468. doi:10.3390/vehicles3030027

Kim, W., Lorenzini, M., Balatti, P., Nguyen, P., Pattacini, U., Tikhanoff, V., et al. (2019). "A reconfigurable and adaptive human-robot collaboration framework for improving worker ergonomics and productivity," in *IEEE robotics and automation magazine* IEEE. doi:10.1109/MRA.2018.2890460

Li, X., Liu, H., and Dong, M. (2021). A general framework of motion planning for redundant robot manipulator based on deep reinforcement learning. *IEEE Trans. Industrial Inf.* 18, 5253–5263. doi:10.1109/tii.2021.3125447

Liu, Q., Liu, Z., Xiong, B., Xu, W., and Liu, Y. (2021). Deep reinforcement learning-based safe interaction for industrial human-robot collaboration using intrinsic reward function. *Adv. Eng. Inf.* 49, 101360. doi:10.1016/j.aei.2021.101360

MassirisFernández, M., Fernández, J. Á., Bajo, J. M., and Delrieux, C. A. (2020). Ergonomic risk assessment based on computer vision and machine learning. *Comput. Industrial Eng.* 149, 106816. doi:10.1016/j.cie.2020.106816

Matulis, M., and Harvey, C. (2021). A robot arm digital twin utilising reinforcement learning. *Comput. Graph.* 95, 106–114. doi:10.1016/j.cag.2021.01.011

Meyes, R., Tercan, H., Roggendorf, S., Thiele, T., Büscher, C., Obdenbusch, M., et al. (2017). Motion planning for industrial robots using reinforcement learning. *Procedia CIRP* 63, 107–112. doi:10.1016/j.procir.2017.03.095

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning (PMLR)* New York, United States: arXiv, 1928–1937. doi:10.48550/arXiv.1602.01783

Montini, E., Bonomi, N., Daniele, F., Bettoni, A., Pedrazzoli, P., Carpanzano, E., et al. (2021). "The human-digital twin in the manufacturing industry: Current perspectives and a glimpse of future," in *Trusted artificial intelligence in manufacturing: A review of the emerging wave of ethical and human centric AI technologies for smart production*, 132–147. doi:10.1561/9781680838770.ch7

Neto, P., Simão, M., Mendes, N., and Safeea, M. (2019). Gesture-based human-robot interaction for human assistance in manufacturing. *Int. J. Adv. Manuf. Technol.* 101, 119–135. doi:10.1007/s00170-018-2788-x

Ng, M.-K., Chong, Y.-W., Ko, K.-m., Park, Y.-H., and Leau, Y.-B. (2020). Adaptive path finding algorithm in dynamic environment for warehouse robot. *Neural Comput. Appl.* 32, 13155–13171. doi:10.1007/s00521-020-04764-3

Pfeiffer, M., Schaeuble, M., Nieto, J., Siegwart, R., and Cadena, C. (2017). "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *2017 IEEE international conference on robotics and automation (ICRA)* (Manhattan, New York: IEEE), 1527–1533.

Phaniteja, S., Dewangan, P., Guhan, P., Sarkar, A., and Krishna, K. M. (2017). "A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots," in *2017 IEEE international conference on robotics and biomimetics (ROBIO)* (Manhattan, New York: IEEE), 1818–1823.

Singh, T. P., Suresh, P., and Chandan, S. (2017). Forward and inverse kinematic analysis of robotic manipulators. *Int. Res. J. Eng. Technol. (IRJET)* 4, 1459–1468.

Smart, W. D., and Kaelbling, L. P. (2002). "Effective reinforcement learning for mobile robots,", Washington, DC, USA, 11-15 May 2002, 3404–3410. *Proceedings 2002 IEEE international conference on robotics and automation (cat. No. 02CH37292)*

Tai, L., Paolo, G., and Liu, M. (2017). "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (Manhattan, New York: IEEE), 31–36.

Tao, F., Qi, Q., Liu, A., and Kusiak, A. (2018). Data-driven smart manufacturing. *J. Manuf. Syst.* 48, 157–169. doi:10.1016/j.jmsy.2018.01.006

Thrun, S. (1995). An approach to learning mobile robot navigation. *Robotics Aut. Syst.* 15, 301–319. doi:10.1016/0921-8890(95)00022-8

Unity-Technologies (2019). *Unity-technologies/ml-agents* GitHub.

Wang, P., Liu, H., Wang, L., and Gao, R. X. (2018). Deep learning-based human motion recognition for predictive context-aware human-robot collaboration. *CIRP Ann.* 67, 17–20. doi:10.1016/j.cirp.2018.04.066

Wang, S., Yin, X., Li, P., Zhang, M., and Wang, X. (2020). Trajectory tracking control for mobile robots using reinforcement learning and pid. *Iran. J. Sci. Technol.* 44, 1059–1068. doi:10.1007/s40998-019-00286-4

Xiao, X., Liu, B., Warnell, G., and Stone, P. (2022). Motion planning and control for mobile robot navigation using machine learning: A survey. *Aut. Robots* 1–29, 569–597. doi:10.1007/s10514-022-10039-8

Yang, X., Ji, Z., Wu, J., Lai, Y.-K., Wei, C., Liu, G., et al. (2021). Hierarchical reinforcement learning with universal policies for multistep robotic manipulation. *IEEE Trans. Neural Netw. Learn. Syst.* 33, 4727–4741. doi:10.1109/tnnls.2021.3059912

Yu, J., Su, Y., and Liao, Y. (2020). The path planning of mobile robot by neural networks and hierarchical reinforcement learning. *Front. Neurorobotics* 14, 63. doi:10.3389/fnbot.2020.00063

Zhang, J., Springenberg, J. T., Boedecker, J., and Burgard, W. (2017). "Deep reinforcement learning with successor features for navigation across similar environments," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24-28 September 2017, 2371–2378.

Zhang, R., Lv, Q., Li, J., Bao, J., Liu, T., and Liu, S. (2022). A reinforcement learning method for human-robot collaboration in assembly tasks. *Robotics Computer-Integrated Manuf.* 73, 102227. doi:10.1016/j.rcim.2021.102227