ARCHIVE OF MECHANICAL ENGINEERING

DOI: 10.24425/ame.2026.158900 2026, Vol. 73, No. 2, pp. 217–242

Ilya KURINOV [1], Miroslav IVANOV [2], Grzegorz ORZECHOWSKI [1], Aki MIKKOLA [1]

Towards reinforcement learning based log loading automation

Received 14 February 2026, **Revised** 29 April 2026, **Accepted** 3 June 2026, **Published online** 11 June 2026

Keywords: reinforcement learning, heavy machinery, forestry machinery, reward function, observation vector, proximal policy optimization, simulation model

Forestry forwarders transport cut logs from the felling site to a landing area or a secondary transport vehicle, a task that is physically and mentally demanding for the operator. Even partial automation of the loading cycle can reduce workload and improve safety. This study extends prior reinforcement-learning (RL) work on grasping to the full pick–lift–transport–deliver sequence. We train an agent with Proximal Policy Optimization (PPO) and a two-stage curriculum in a GPU-accelerated simulator (NVIDIA Isaac Gym), using a trailer-type forwarder model with a hydraulic crane and grapple. The task is simplified to a single log and a fixed target location inside the bunk: the agent must reach a randomly placed log, grasp it, lift it above the bed guards, and deliver it with reduced vertical impact velocity. The reward is decomposed into three components, reaching (r_1), lifting/unloading (r_2), and stable target delivery (r_3), and we compare curriculum compositions. In an evaluation over 1,024 parallel episodes, the best two-stage configuration ($r_1 + r_2$, then $+r_3$) reaches a 94 % success rate, outperforming flat and fully staged alternatives. Generalization tests on unseen log sizes, elevated ground, and rough terrain show partial transfer with clear degradation under larger shifts. The study is limited to a highly simplified simulation setting (flat ground, fixed forwarder base, single log), and no sim-to-real transfer was attempted.

✉ Grzegorz ORZECHOWSKI, email: grzegorz.orzechowski@lut.fi

[1]LUT University, Yliopistonkatu 34, 53850 Lappeenranta, Finland.

[2]Lapland University of Applied Sciences, Jokiväylä 11, 96300 Rovaniemi, Finland.

1. Introduction

Forest forwarders are heavy-duty machines that perform tasks such as lifting, transferring, and positioning logs in complex forest environments. They combine articulated hydraulic cranes and grippers, which provide the necessary strength and dexterity required to manipulate logs of various sizes and weights with precision and stability [1]. Designed for rugged outdoor conditions, forest forwarders must perform reliably on rough terrain and through obstructions such as trees, bushes, or other equipment.

Currently, human operators are responsible for the control of these machines. They operate from raised cabins with clear views, or they control the equipment remotely using teleoperation technologies. The control interfaces of modern forwarders may include pedals and joysticks or state-of-the-art haptic or assistive feedback devices for precise control of the crane and grapple [2]. The operation of a forest forwarder is physically demanding, requiring operators to maintain a high level of focus and precision continuously for extended periods of time [3]. This is due to the combination of continuous fine motor control, repetitive movements, and the need for sustained concentration over long shifts. Operators must manipulate joysticks, pedals, and control panels with precision, often requiring small, frequent adjustments to the crane's hydraulic system. Maneuvering massive logs with a machine that has multiple degrees of freedom demands excellent spatial awareness, quick decision-making, and fine motor skills. The stress associated with crane operation is amplified by the potential risk of accidents, equipment malfunctions, and pressure to meet productivity targets.

As the mechanical complexity and operating demands of log handling increase, interest in automating forwarders using Artificial Intelligence (AI) and robotics is growing. Autonomous or semi-autonomous operation of such machines not only holds the promise of improving operational efficiency and lowering operator fatigue, but also of improving safety by limiting human exposure to hazardous environments. Among various approaches, reinforcement learning is particularly well-suited for training intelligent agents to execute control policies in complex systems such as forest forwarders via training in simulated environments.

This research aims to continue the exploration of the application of reinforcement learning algorithms in forestry forwarder automation, focusing on automating the full cycle of log picking. The previous studies discussed in the literature review focus mainly on automating forwarders only till grabbing stage or using conventional methods. Grabbing the log is a challenging task, but continuing the motion from grabbing is a substantially more complex task. Additional complexity of the log loading from the grabbing motion comes from a longer horizon of the task and shifting task objectives, such as grabbing, transporting, and loading to the bed. It requires maintaining a stable grasp while moving the payload over the bunk, avoiding contacts with the bed guards, and reducing impact velocity at the target, which increases the number of failure modes.

The key contributions of this paper are:

1. The study extends previous works in forestry forwarder automation by learning a policy that goes beyond grasping and performs lifting and transport to a fixed target drop location inside the forwarder bunk (single-log setting).
2. Demonstrating the feasibility of the reinforcement learning agent for tasks with a long horizon and identifying potential issues in the approach.
3. Defining the example action and observation spaces with a suitable reward function.
4. Identifying benefits and pitfalls of the approach, including observed failure modes and limitations that must be addressed before real-world deployment.

We emphasize that PPO and curriculum learning are standard tools; the contribution of this work is an application study in a simplified forwarder simulation, with an ablation of reward-composition curricula for a long-horizon loading sequence.

2. Literature review

The topic of forestry forwarder automation appears in multiple studies over the years. Studies relevant to the topic of this research and related to automation using artificial intelligence fall into categories of application of reinforcement learning and convolutional neural networks. Studies applying reinforcement learning methods focus on automating only the grasping motion. The research by [4] studied automation of the grasping motion using reinforcement learning with curriculum on a simulation model of an experimental forestry forwarder. The study applied a curriculum with distance from the initial position increase during the training and an energy optimization objective. The best agent was able to grab logs with 97% accuracy, while the energy optimization model was able to grasp logs with 93% success rate. The researchers propose that future studies focus on the application of the agent in a more unstructured environment. Another study by [5] applies reinforcement learning and virtual visual servoing for multi-log grasping. The study uses a reinforcement learning agent that takes an image from a virtual camera (with a CNN for identifying the target grasping position) and is trained on a curriculum with an energy optimization objective. The resulting agent is able to grasp logs with 95% overall success rate, 97% for three logs grasping and 91% for five logs grasping. The study by [6] investigated automation of the log grasping process using a convolutional neural network. The study trained an agent in the simulated environment and further tested it on a scaled forwarder crane in a laboratory environment. Two agents were trained to control the crane to follow the predefined path tracks. The study by [7] automates the forwarder's locomotion task using reinforcement learning. The study applies a forestry forwarder model and scans of a real harvesting site terrain for reconstructing the environment. The resulting agent is capable of traversing steep-angle terrain and applying different strategies for locomotion.

Some studies focus on forestry forwarder operation using methods other than reinforcement learning. The research by [8] studied the application of a CNN for grasp planning of a log. The grasp planning pipeline uses input from a depth camera to recreate the scene in a Gazebo simulation environment with a grasp map. The tests were conducted on a real forestry machine and demonstrated robust grasping with a maximum success rate of 98.3%. Another study by [9] shows the application of a single stereo camera for log detection and grasping. The system was able to recognize the grapple, the log, and its grasping point with further calculation of the boom motion vector. The resulting system was tested on a real forestry forwarder with a grasp success of 82%. The research by [10] developed a system for the autonomous operation of the forestry forwarder. The system is capable of navigating the job site, locating the log, loading it into the bed, and delivering it to the final destination. Mentioned capabilities were achieved by combining multiple methods in one system: the locomotion uses GPS and predefines the path tracking point of a mission, the log detection uses a depth camera and a DNN for tracking and estimating the position of the log. The log handling uses the Dynamic Movement Primitives algorithm, which is trained on demonstrations from experienced operators. The system was capable of fully autonomous operation in a repeatable and stable way.

3. Methods

Reinforcement learning (RL) is a type of machine learning and optimal control approach that addresses sequential decision-making problems. An agent learns a policy that maps states to actions by interacting with an environment to maximize cumulative rewards over time [11]. RL is widely used for game testing [12], robotics [13], autonomous vehicles [14], healthcare [15], and finance [16]. It is also applied in various mechanical engineering and heavy machinery control applications, including optimization of robotic arms' operation in manufacturing [17] and in the autonomous control of heavy machinery such as excavators [18] and bulldozers [19].

Formally, an RL task is often modeled as a Markov decision process defined by a tuple data structure (S, A, P, R, γ), where S is the state space, A is the action space, $P(s'|s, a)$ is the state transition probability to state s' given state s and action a, $R(s, a)$ is the reward function, and $0 \leq \gamma < 1$ is a discount factor weighting future rewards [11]. In the PPO experiments in this study, the discount factor was set to $\gamma = 0.98$. The agent's behavior is governed by a *policy* $\pi(a \mid s)$ that specifies the choice of action in each state.

Value functions are central in RL to estimate the expected discounted return under a policy and are used by many algorithms to guide policy improvement. The state-value $V(s)$ gives the expected total discounted reward from state s (following the policy thereafter). These functions satisfy recursive Bellman equations; in particular, the Bellman optimality equation for the optimal value function:

$$V^*(s) = \max_a \mathbb{E}\left[R(s,a) + \gamma V^*(s')\right], \tag{1}$$

and $V^*(s')$ is a value function for the next state. Multiple RL algorithms are designed to approximate optimal values, following the above equation. Common methods include advantage actor-critic [20], PPO [21], soft actor-critic [22], and deep Q-networks [23].

3.1. Curriculum Learning for Reinforcement Learning

Curriculum Learning (CL) is a training strategy where an agent learns through a sequence of tasks with increasing difficulty [24]. In reinforcement learning, it is used to improve learning efficiency, especially in problems with sparse rewards and long-horizon tasks. Rather than training an agent on the full task from the start, CL initially introduces simplified versions of the task and gradually increases complexity [24].

In sparse reward environments, agents often fail to reach the goal via random exploration. CL addresses this by structuring tasks so that agents first learn sub-skills with intermediate rewards [25]. For example, in robotic manipulation, agents might first learn to reach for an object, then grasp it, and finally place it at a target location [26]. This method improves credit assignment and accelerates learning.

CL is also used for tasks comprising multiple phases. By dividing the task into stages, each with its own reward, the agent can focus on one subtask at a time. In this project, for example, PPO was used to train a forest forwarder to perform a sequence of actions: locate a log, grasp, lift, place on a trailer, and return to the initial position. A simple curriculum was applied by assigning rewards to these subtasks, which guided the learning process and reduced the burden of exploration. See Section 4.3.3 for more details.

Various strategies exist to design curricula, including manually defined stages and automated methods [26]. Studies show that curriculum learning improves sample efficiency and leads to higher-quality policies. For robotic tasks specifically, it has been effective in enabling agents to learn complex behaviors with fewer training steps.

3.2. Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a popular reinforcement learning algorithm that balances simplicity and performance. PPO optimizes the policy directly and avoids large, disruptive changes to the policy by using a clipping mechanism, which helps maintain stability during training [21]. In mechanical engineering, particularly in heavy machinery control, PPO is highly effective due to its ability to handle complex, continuous control tasks. For example, PPO was used to optimize the operation of autonomous excavators or bulldozers [18]. We selected PPO also in this study for its stable on-policy updates and strong empirical performance

in continuous-control tasks in physics simulation; off-policy alternatives such as SAC [22] and TD3 [27] are common baselines and are left for future comparison.

3.3. Isaac Gym

Isaac Gym is state-of-the-art simulation software developed by NVIDIA. It was designed to efficiently train RL agents in virtual environments. Efficient training is achieved by leveraging the parallelization capabilities of Graphics Processing Units (GPUs) to simultaneously simulate thousands of environments and by close coupling with RL frameworks, often based on PyTorch [28]. This allows for large-scale training on the GPU alone and reduces the need for data movement between the GPU and central processing unit memory in the training cycles.

The Isaac Gym simulator sustains articulated and rigid-body systems with complex multibody dynamics support, contact simulation, and collision handling, features critical to simulating the real-world robotic system behaviors of, for example, manipulators, legged robots, or forestry equipment such as forest forwarders. Additionally, Isaac Gym provides domain randomization capabilities and configural flexibility, which support robust policy learning in a variety of dissimilar environments; exploiting these features for sim-to-real transfer requires additional system identification and domain randomization beyond what is applied in this work. Furthermore, the simulator's modularity enables the use of custom reward functions, multi-modal sensing, and a variety of control devices, which renders it a flexible platform for experimentation on a broad range of robotics and control problems. Isaac Gym Preview has since been succeeded by Isaac Sim / Isaac Lab; we use Isaac Gym in this work and consider migration to newer platforms as future work.

4. Problem statement

The study applies reinforcement learning control to solve a loading task. As shown in Figure 1, this task involves performing multiple nontrivial consecutive steps to load the log. The challenge is that successful loading requires coordinated reaching and grasping, lifting above the bed guards, and approaching the target drop point with reduced vertical velocity to avoid impacts. To enable these behaviors, the observation includes the forwarder joint states and the poses of the grapple and log, and the reward is shaped to provide intermediate feedback for reach, lift/unload, and final target delivery. This study introduced the involved stages by designing several separate rewards for grabbing, lifting, and dropping a log. The environment was designed and implemented in Isaac Gym [28], and control was implemented via a PPO agent.

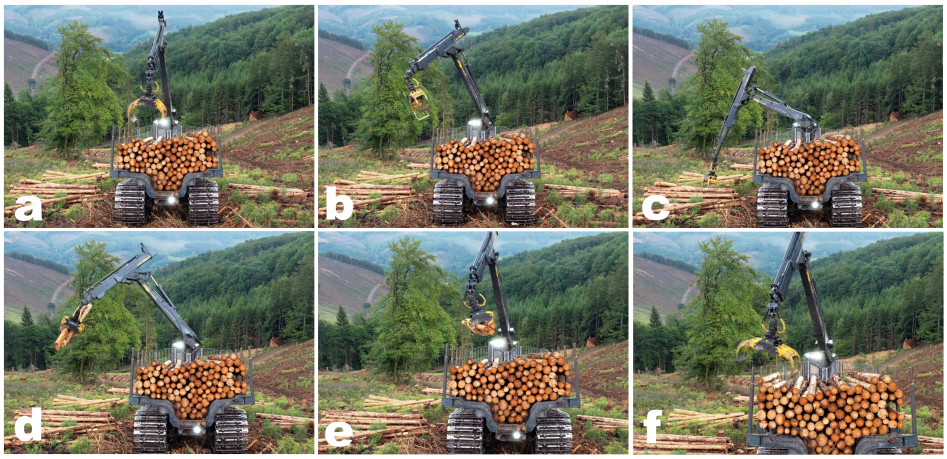

Fig. 1. Forwarder operation cycle: a) The grapple is at the initial position above the bed. b) The forwarder reaches the log pile position. c) The grapple closes around the logs. d) The crane lifts logs above the bed. e) The pile of wood is lowered onto the bed. f) The grapple releases.

4.1. Forestry forwarder operation

The typical forester forwarder operation sequence begins by traveling to the harvesting site. Depending on the type of forestry forwarder, it may be driven or transported to the site. An operator must consider the topology of the terrain, the type and size of the wood products, visibility, and weather conditions [29].

Once positioned near the log pile, the forwarder uses the hydraulic crane equipped with a grapple to pick up logs. See Figure 1. The grapple grasps a log, lifts it above the forwarder bed, and lowers it onto the bed. Proper log arrangement is critical to optimizing load capacity and stability. In this study, adaptive arrangement of multiple logs is not considered; the task involves a single log and a fixed target location for delivery inside the bunk. This process continues until the bed is fully loaded or all harvested logs have been handled.

Once the bed has been loaded, the forwarder transports the logs from the harvesting area to a piling area where the logs are loaded onto a truck or stored for further transport by truck. Similarly to the transportation step, a forwarder operator must consider terrain topology and weather conditions, and the operator must account for changes in the center of mass to handle this step safely and effectively.

The entire cycle, from the trip to the cutting area, loading, transporting, unloading, and returning, repeats continuously throughout the timber harvesting process, allowing efficient flow of wood from the forest to the processing or transport sites. The productivity of the forwarder and its cycle time depend on factors such as terrain conditions, log arrangement, and the distance between the cutting area and the landing site [30]. Optimal operation requires skilled maneuvering, precise crane

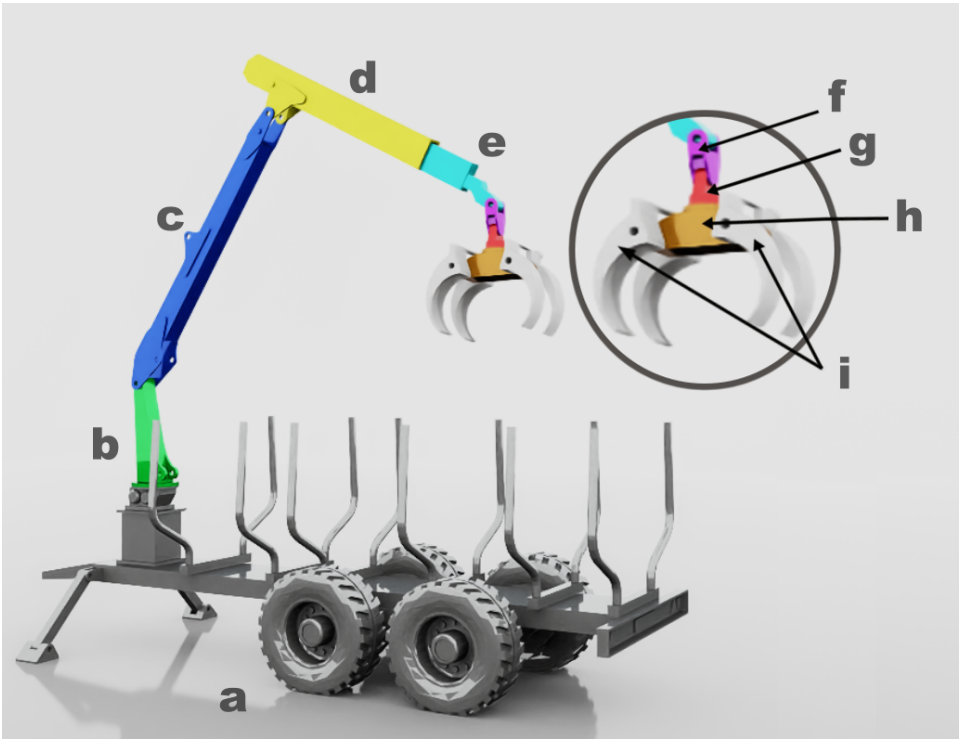

Fig. 2. Model topology of forwarder: a) base with trailer and bed, b) manipulator base, c) crane arm, d) extension arm, e) extension, f) intermediate hook, g) grapple rotator, h) grapple body, i) grapples.

control, and fluent loading/unloading techniques, all of which contribute to the total productivity of the forestry operation.

4.2. Forestry forwarder model description

The simulation model imitates a trailer-type forest forwarder equipped with a crane and a grapple attachment. It is an open-loop system with 9 degrees of freedom. The main parts of the machine are a four-wheel trailer used for the transportation and storing of logs and a hydraulic crane with a grapple for log handling. The list of bodies is presented in Table 2. The base of the trailer is the first body in the model, which contains the bed and four wheels. The wheels in the simulation serve a decorative function, and they are part of the base. In this simulation, therefore, no suspension model was implemented. In the simulation environment, the base is fixed to the ground plane. The bed guard collision geometry was approximated by a small set of box primitives representing the guard rails, which restricts the movement of the grapple between bed poles compared to real-world operation.

Contacts are simulated using the Isaac Gym rigid-body contact model. The default physics material parameters were static friction 0.1, dynamic friction 0.1, and restitution 0.0. Rolling friction is not explicitly modeled.

The manipulator crane comprises five bodies. The base of the crane is attached to the trailer with a rotational joint, which is allowed to move 140 degrees, i.e., 70 degrees to each side. The crane arm is responsible for vertical reach and lifting capabilities. The crane arm has a rotational joint with a limit of 70 degrees. The extension arm is a body with an extension boom attached. With motion limited to 70 degrees, it is responsible for the reaching and, partially, lifting capabilities. The extension boom attaches to the extension arm with a slider joint, which makes it possible to extend to 1.5 meters to assist in reaching a load. The intermediate hook is a body that attaches the grapple assembly to the crane. In the provided USD model, this part of the chain is represented by two serial revolute joints (Revolute_4 and Revolute_5) with limits $[-90, 120]$ and $[-90, 90]$ degrees, respectively (Table 1).

The grapple assembly consists of four bodies: the grapple hook (rotator), the grapple body, and two grapples. The grapple hook connects the grapple to the intermediate hook (Revolute_5) and to the grapple body (Revolute_6), while the two grapples connect to the grapple body via revolute joints with limits $[-45, 50]$ and $[-50, 45]$ degrees (Revolute_7 and Revolute_8) as summarized in Table 1. All of the aforementioned bodies have a simplified convex mesh, which was not altered. To obtain stable contacts during grasping, the collision geometry of the grapples was simplified using box primitives (instead of relying on complex convex meshes), which reduces simulation overhead at the cost of geometric fidelity.

Table 1. Names, types and limits of the forestry forwarder model joints

Joint Name	Symbol	Joint type	Min limit	Max limit	Units
Base – Manipulator base	j_1	Revolute	−70	70	Degrees
Manipulator Base – Crane arm	j_2	Revolute	−30	40	Degrees
Crane arm – Extension arm	j_3	Revolute	−30	40	Degrees
Extension arm – Extension	j_4	Slider	0	1.5	Meters
Extension – Intermediate hook	j_5	Revolute	−90	120	Degrees
Intermediate hook – Grapple hook	j_6	Revolute	−90	90	Degrees
Grapple hook – Grapple body	j_7	Revolute	0	180	Degrees
Grapple body – Grapple left	j_8	Revolute	−50	45	Degrees
Grapple body – Grapple right	j_9	Revolute	−45	50	Degrees

Table 1 represents a detailed description of the joints and their characteristics. Defining joint limits is important to approximate the behavior of the actual forestry forwarder and accurately replicate its motion constraints. Real machines often use hydraulic cylinders and motors for actuation. Therefore, due to the closed-loop structure of the crane, joint motions are limited. Because the simulation model has an open-loop structure, joint limits were introduced to imitate actual machine behavior. This makes it possible to simulate a machine with relatively accurate

Table 2. Names and masses of the forestry forwarder model

Body name	Mass,kg
Base	1500
Manipulator Base	158
Crane arm	171
Extension arm	132
Extension	75
Intermediate hook	16
Grapple hook	12
Grapple body	50
Grapple left	40
Grapple right	40

movements for the agent to solve a log handling task and limit the workspace to reduce the complexity of the task.

4.3. Task

The previously described model solves a simplification of log loading that is close to the real-world scenario. The operation involves a forestry forwarder simulation model, a ground plane, and a log model. The simulation environment is also simplified. The ground plane is flat, and the log parameters do not change during simulation. The agent's goal is to move the log from a random starting position onto the bed using the controls of the forwarder simulation model.

The simulation timestep was set to $dt = 0.03$ s with control frequency inverse 3 (policy action is applied every 0.09 s), and the episode length was 300 steps.

4.3.1. Action space

In the context of reinforcement learning, an action space is the complete set of possible actions that an agent can perform at any state in a particular environment. It specifies the output domain of the policy that maps states or observations into subsequent actions. Action spaces are generally classified into two categories: discrete and continuous. A discrete action space contains a finite set of different actions. It is typically used in settings such as games, where the agent selects among a pre-defined set of options. In a continuous action space, the agent can select from a range of values for each action dimension. It is frequently used in applications such as robotic control, where actions involve smooth motor commands.

The action space plays a crucial role in determining complexity with regard to the learning problem. Larger or more continuous spaces result in higher-dimensional search problems for the policy that require using more advanced exploration and optimization methods. The form and shape of the action space

are essential in determining the effectiveness of the agent to interact with and productively affect its environment.

The action space of the forest forwarder model constitutes joint position goals, which are incremented by a scaled action provided by the agent. Each active joint is scaled by the corresponding scaling factor and a timestep dt. This approach was taken to reduce the velocity of the base of the crane, the crane arm, the extension arm, and the extension. Bigger position increments result in erratic movements of the arm, unstable behavior of the grapple, and an inability to grab the log. Shorter joint increments allow more precise and safer movement. The resulting action vector is clipped to the joint limits discussed previously in the Forestry Forwarder Model Description. In the simulator, these targets are executed through the articulated rigid-body dynamics with USD-defined joint drives, limits, and finite effort/velocity constraints (i.e., the crane is not modeled as infinitely strong or as purely kinematic). The action scaling vector used in the experiments was $\mathbf{c} = [6, 7, 9, 27, 9, 64, 64]$. The action space is described as:

$$\mathbf{U} = \text{clip}(\mathbf{q} + \mathbf{A} \cdot \mathbf{c} \cdot dt, u_{min}, u_{max}), \tag{2}$$

where $\mathbf{U}$ is a joint position goals matrix of size $n_{joints} \times n_{envs}$, $\mathbf{q}$ is a current joints position matrix of size $n_{joints} \times n_{envs}$, $\mathbf{A}$ is an action provided by the agent, $\mathbf{c}$ is a scaling factor vector, and dt is a time step. The joint position matrix has the following form:

$$\mathbf{U} = \begin{bmatrix} u_{j11} & u_{j21} & u_{j31} & u_{j41} & u_{j71} & u_{j81} & u_{j91} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ u_{j1n} & u_{j2n} & u_{j3n} & u_{j4n} & u_{j7n} & u_{j8n} & u_{j9n} \end{bmatrix}, \tag{3}$$

where u_{j1n}, u_{j2n}, u_{j3n}, u_{j4n}, u_{j7n}, u_{j8n}, u_{j9n} are the joint positions, and n is the number of environments.

4.3.2. Observation space

The RL observation space refers to the set of all possible states or sensory inputs that an agent can perceive from the environment at any given time. It defines the structure, dimensionality, and boundaries of the data that the agent receives to make decisions. Formally, the observation space is a subset of the environment's state space, which contains complete information about the environment's current configuration.

However, the agent may only receive partial information, which constitutes its observation. The space can be continuous, as in physical attributes such as position or velocity, or discrete, as in categorical data such as different terrain types. The design and representation of the observation space are crucial because

they directly influence the agent's ability to learn and generalize effectively from the environment. A well-defined observation space ensures that the agent can extract relevant features to make informed decisions, which affects the overall performance of the RL algorithm. The observation space of the forestry forwarder model is represented by four main elements: a forwarder parameters vector $\mathbf{S}_{fwd}$, the grapple body parameters vector $\mathbf{S}_g$, a log parameters vector $\mathbf{S}_l$, and the task parameters vector $\mathbf{S}_{task}$.

The first element of the observation vector describes the state of the forwarder simulation model. The forwarder parameters vector includes joint positions and joint velocities. Joint positions describe rotational positions of the active joints in radians and the translational position of the extension (slider) joint in meters. Joint velocities describe the angular velocities of the forwarder's active joints in radian/sec and the translational velocity of the slider joint in m/s. These parameters provide vital spatial information for the agent, which helps to determine an optimal course of action. The resulting forwarder parameters vector has the following form:

$$\mathbf{S}_{fwd} = \begin{bmatrix} u_{j1} & \dots & u_{jm} & \dot{u_{j1}} & \dots & \dot{u_{jm}} \end{bmatrix}, \tag{4}$$

where u_{jm} and $\dot{u_{jm}}$ are joint positions and velocities of active degrees of freedom of the forwarder model, and m is the joint number. The active degrees of freedom of the forwarder model are $j_1, j_2, j_3, j_4, j_7, j_8, j_9$ as described in Table 1.

The second part of the observation vector describes the parameters of the grapple body. This vector provides information about the grapple body and the grapples. The vector contains parameters of the grapple body as x, y, and z position in world coordinates in meters, velocity in m/s, and rotation in quaternion form. This data helps to navigate and align the grapple body with the log. The velocity of the grapple body helps the agent decide the velocity of approach to the log or bed to avoid hard contact with the ground or machine. The left and right grapple positions provide world-coordinate data that further aid in achieving the accuracy of alignment of the grapple with either the log or load bed. The grapple body observation vector has the following form:

$$\mathbf{S}_g = \begin{bmatrix} p_{gb} & H_{gb} & \dot{p_{gb}} & p_{gl} & p_{gr} \end{bmatrix}, \tag{5}$$

where p_{gb}, p_{gl}, p_{gr} are xyz positions of the grapple body and left and right grapples in world coordinates, $\dot{p_{gb}}$ is the velocity of the grapple body, and H_{gb} is the rotation of the grapple body in quaternions.

The third part of the observation vector is the log parameters. This vector shows the position in world coordinates and the orientation of the log in quaternion form. Along with grapple body parameters, the log parameters are essential for accurate grapple aiming and log transport. The resulting vector is:

$$\mathbf{S}_l = \begin{bmatrix} p_l & H_l \end{bmatrix}, \tag{6}$$

where p_l is the position vector, and H_l is the rotation in quaternions of the log.

The last part of the observation vector describes important task parameters and the degree of completion. The p_{unl} and p_{tgt} parameters were introduced to evaluate completion of the log handling task. The p_{unl} parameter describes the position of the imaginary point located above the center of the bed and bed guards. Establishing this intermediate target point on the way to the target helps to navigate without following the direct path. The target point p_{tgt} signifies the log target position, which is located at the bottom of the bed. Terms $p_{gb} - p_l$, $p_l - p_{unl}$, and $p_l - p_{tgt}$ are the difference in positions along the x-y-z axes. They provide the agent with the relative position to the grapple body, the unloading point, and the target. The a_l parameter is a variable indicating completion of the task. It gives a score from 0–1 if the log is delivered to the target point:

$$\mathbf{S}_{task} = \begin{bmatrix} p_{unl} & p_{tgt} & p_{gb} - p_l & p_l - p_{unl} & p_l - p_{tgt} & a_l \end{bmatrix}, \tag{7}$$

where p_{unl} is the position of an unloading point situated above the bed, p_{tgt} is the position of a target point inside of the bed, a_l is a variable indicating completion of the log transport, and $p_{gb} - p_l$, $p_{unl} - p_l$, and $p_l - p_{tgt}$ are the position differences between the grapple body to the log, the log to the unloading point, and the log to the target, respectively. The a_l variable can be expressed by the following equation:

$$a_l = x_{ltgt} \cdot x_{lgb} \cdot x_{gbtgt}, \tag{8}$$

where x_{ltgt} is a score of reducing distance between the log and the target, x_{lgb} is a score of reducing distance between the log and the grapple body, and x_{gbtgt} is a score of reducing distance between the grapple body and the target point. These scores are calculated as follows:

$$x = \frac{1}{1 + d^2}, \tag{9}$$

where d is the Euclidean distance between two positions. Eventually, the resulting observation vector can be represented by the following equation:

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{fwd1} & \mathbf{S}_{g1} & \mathbf{S}_{l1} & \mathbf{S}_{task1} \\ \dots & \dots & \dots & \dots \\ \mathbf{S}_{fwdn} & \mathbf{S}_{gn} & \mathbf{S}_{ln} & \mathbf{S}_{taskn} \end{bmatrix}, \tag{10}$$

where n is the number of training environments. The observation vector provides the required sensor reading for an agent to decide on the course of action to achieve the goal. Several parameters used in the construction of the observation vector, such as the distance reduction scores denoted by x, were used to formulate the reward function.

4.3.3. Reward function

The reward function is used to provide feedback from the forwarder model on actions taken. The reward given offers insight into the efficiency of reaching, grabbing, lifting, and delivering the log. Introducing the multiplicative factor b allows the agent to focus on a particular task during the training process. Therefore, the agent's behavior is shaped not only by its grabbing and lifting performance, but also by its ability to move the log towards the bed and complete the unloading phase, which is critical to maximizing cumulative reward.

The reward function has three main elements: r_1, r_2, and r_3. The first element r_1 is responsible for guiding the grapple body towards the center of mass of the log. The r_1 reward is especially important at the beginning of the process because the grapple is at a random position and must navigate to the log position to grab it. r_1 takes the following form.

In the implementation used for this study, the scalar b corresponds to a constant reward scaling factor (reward_scale) set to $b = 10$, used to scale the unloading and target-reaching terms in r_2 and r_3. The curriculum is implemented by enabling r_3 only in the second training stage (rather than by varying b over time). The value $b = 10$ was selected through short preliminary sweeps to keep r_2 and r_3 numerically comparable to r_1 and to stabilize learning, and it was kept fixed across all reported experiments.

We did not include an explicit reward term for aligning the grapple orientation with the log, and we did not include an explicit penalty for collisions with the bed guards. In this simplified setup, the proximity-based terms together with the contact dynamics were sufficient to learn feasible grasp-and-transport behavior; collisions are instead indirectly discouraged because they disrupt progress toward the unloading/target points and reduce the achievable reward:

$$r_1 = x_{lgb}, \tag{11}$$

where x_{lgb} is the score of the distance reduction between the center of mass of the log and the grapple body (see Eq. (8)).

Part r_2 of the reward function is responsible for manipulating the log before loading it onto the bed of the forwarder. Scalar r_2 identifies movement in the z-axis and towards the unloading point located above the bed. Scalar r_2 can be described by the following equation:

$$r_2 = p_{lz} + x_{lunl} \cdot b, \tag{12}$$

where p_{lz} is the z-axis position of the log, x_{lunl} is the score of the distance reduction between the log and the unloading point, and b is the weighting factor. The weighting factor b plays an important role in the movement sequence, making the agent focus on moving towards the unloading point.

Part r_3 of the reward function manages the final movement of the grapple towards the target. This part emphasizes the importance of two aspects of movement: bringing the log to the target point p_{tgt} and reducing the velocity of the log in the z-axis. The velocity reduction is critical, since it prevents the agent from dropping or hitting the log hard, which could lead to equipment damage. The scaling factor b presented in Eq. (12) is raised to the power of 3, emphasizing the importance of the "current" over the "previous". r_3 can be expressed with the following equation:

$$r_3 = \frac{x_{ltgt}^2}{\dot{p}_{lz}} b^3, \tag{13}$$

where x_{ltgt} is the score for reducing the distance between the log and the target point, $\dot{p}_{lz}$ is the velocity of the log in the z-axis, and b is the scaling factor.

The resulting reward function depends on the chosen hierarchical learning configuration. Rewards can be arranged in four ways. The first requires r_1, r_2, and r_3 to be a separate stage of the hierarchical learning, adding subsequent reward after completing the stage. The second and third combine two of the reward elements into one stage. The fourth option combines all of the rewards into one stage. The most promising training combination is $r_1 r_2 + r_3$. This implies that the model is pretrained on a reward function that consists of the sum of r_1 and r_2. The second half of the training process continues from the best model of the previous step.

5. Results

Subsequent paragraphs describe a PPO agent trained on two stages of curriculum training. The reward for the first training stage is the sum of r_1 and r_2. The second stage's reward is the sum of the stage one reward and r_3. In all the experiments, PPO used $\gamma = 0.98$, $\tau = 0.95$, learning rate $5 \cdot 10^{-4}$, horizon length 128, minibatch size 2048, and 8 mini-epochs per update.

The resulting agent was able to learn the optimal policy to operate the forwarder. It adapts to environment changes such as random positions of the forwarder joints and random positions of the log. It is capable of aligning the orientation of the grapple body to the log, enabling log grasping. In cases where the log is outside of the pick-up script, such as behind the bed or places out of reach, the agent struggles to get a hold. In rare cases, the agent may push the log out of reach.

In general, however, the agent is capable of moving the log and positioning it without touching the side guards of the bed. Depending on the grasping position of the log, the agent may touch the bed guards, which is an undesired behavior in a real machine and can induce large contact forces and oscillations; in our simplified setup, it is not explicitly penalized and therefore remains a limitation.

Figure 3 shows a sequential visual representation of the learned behavior of the reinforcement learning agent for executing the log picking task with a simulated forestry forwarder in Isaac Gym. The sequence illustrates the successful execution

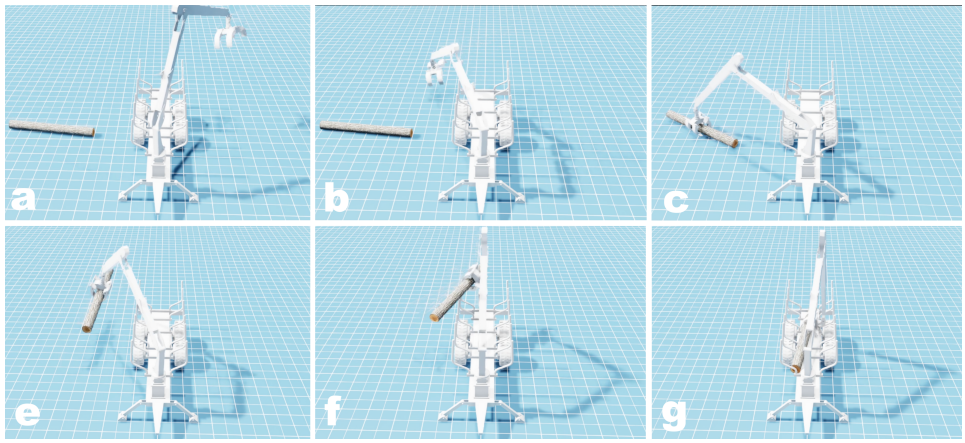

Fig. 3. Result of training: The Forest forwarder agent successfully delivers the log to the bed. a) The agent starts with random joint and log positions. b) The agent navigates the grapple to the log. c) The grapple attachment reaches and grasps the log. d) The agent lifts the log and begins to move to the unloading point position. e) The log is transferred to the unloading position. f) The log is delivered to the target point located inside the bed. g) The manipulator places the log into the bed.

of the task, decomposing it into a number of key subtasks executed by the agent, which demonstrates the success of the learned policy to integrate multiple-degree-of-freedom arm control with object manipulation.

In the first example 3.a, the forwarder is presented in its default initial state with the manipulator arm retracted and a target log set on the ground in front of the vehicle. In 3.b, the agent begins by extending the manipulator arm towards the log. The motion reflects that the policy uses the state-based log and robot observations to generate a smooth approach that typically avoids hard contacts in the simplified environment. The next frame 3.c depicts the gripper over the log, with fine adjustments being performed by the manipulator in order to acquire a good orientation for grasping. This stage demands good coordination among arm articulation and end-effector placement.

After alignment with the log, the agent performs a grasping action from 3.c to 3.e, followed by a lifting stage. As indicated by 3.e, the log has been successfully grasped and lifted off the floor, which indicates that the agent has learned not just to reach out and grasp but also to ensure grip stability. Frame 3.f illustrates the transport phase that lifts the log over the cargo space of the forwarder. Finally, as illustrated in 3.g, the manipulator places the log into the bed.

This completes the entire process of approach, manipulation, and placement, and these actions demonstrate the agent's ability to relocate the object without losing grip in the nominal setting; occasional contacts with the bed guards can still occur. Successful execution of this sequence proves that the policy employs a coherent strategy to complete a complex, multi-step log loading task. The success confirms the reward shaping and curriculum learning methods implemented during

training and demonstrates their contribution to policy generalization and stability. In our evaluation, an episode is considered successful when the log reaches the target region with low vertical velocity; the reward does not explicitly require opening the grapple or releasing the log.

5.1. Reward progression

Figure 4 illustrates the progression of cumulative rewards over the course of the first phase of curriculum learning, where the agent is trained only on using the main reward component ($r_1 + r_2$). This phase plays a vital role. The intention is to allow the agent to establish core skills such as grasping, lifting, and moving logs into proximity and above the bed. The plot presents the results of five separate training trials (Experiments 0–4) with different random seeds. The dashed line shows the average performance over all trials. All early trials exhibit minimal reward accumulation; however, they consistently increase as training continues, suggesting effective skill development. Here, the dashed line denotes the mean across the five seeded runs (Experiments 0–4).

There is a noticeable difference in learning rate and final performance across the experiments. For instance, experiments 1 and 3 achieve higher cumulative rewards sooner, converging towards approximately 120–140 training steps, which indicates quicker convergence. Experiment 0 displays a more gradual increase, achieving similar performance only after reaching approximately 220 steps. This kind of inconsistency is common in deep reinforcement learning, and it is attributed to the randomness present in policy updates, initializations, and exploration mechanisms. Nonetheless, all experimental outcomes converge to the same cumulative rewards in the end, therefore illustrating the robustness and consistency of the learning process in this initial phase.

The average reward curve (the dashed line) smooths out oscillations and gives a good overview of the overall learning trends. It features a steep rise between the 60 and the 140 step markers, marking the crucial period during which the agent evolves from random or useless behaviors to purposive control. After 160 steps, the curve levels off, reflecting that the agent has maximized the learning potential for the first-stage reward. This leveling off validates the curriculum design. After the basic skill set is learned by the agent, the reward can be fine-tuned to move the log into the bed.

Figure 5 illustrates the cumulative reward trajectories for the second half of the curriculum learning process, i.e., with regards to the training duration from 300 to 1000 steps. During this phase, the reward function is augmented with the introduction of a novel component r_3 that promotes stable target reaching by penalizing vertical log velocity. Building on the skills learned in phase one, this phase teaches the agent to carry out more intricate manipulations such as transporting the log toward the target region with stability constraints. The five curves correspond to five independent runs (Experiments 0–4) with different random seeds.

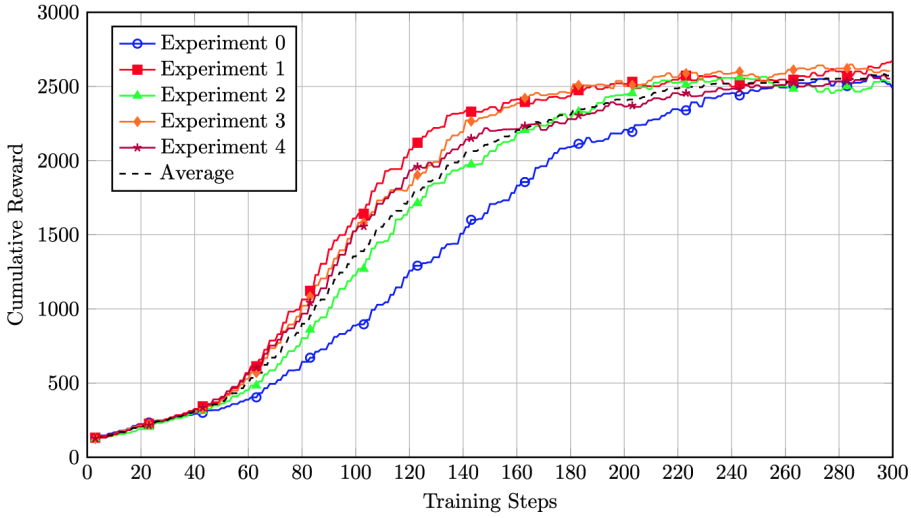

Fig. 4. Stage 1 cumulative reward: Comparison of cumulative reward over training steps for different experiments, including the average curve. The dashed line denotes the mean across the five seeded runs (Experiments 0–4).

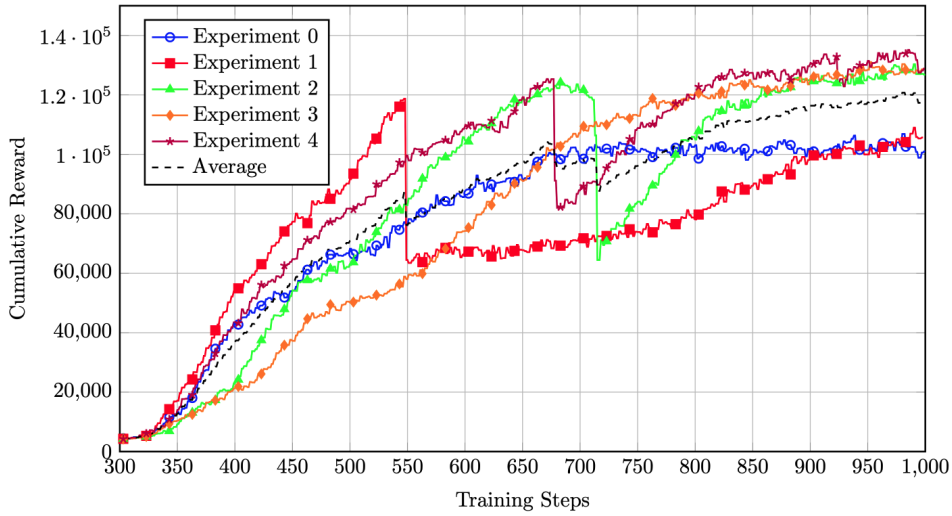

Fig. 5. Stage 2 cumulative reward: Comparison of cumulative reward over training steps for different experiments, including the average curve. The dashed line denotes the mean across the five seeded runs (Experiments 0–4).

The figure highlights significant differences in how various experiments adapt to the newly established reward goal. Experiment 2 has the fastest performance increase, with a cumulative reward of over $1.2 \cdot 10^5$ at step 700, although it suffers from a sudden drop caused by policy instability or exploration noise. Experiment

3 shows the same trend, with good early learning and maintained performance. Experiments 0 and 1 demonstrate more modest but steady growth. The variation between runs captures the difficulty of scaling task complexity. The agent must recall not only previously acquired habits, but it also must modify them to facilitate successful log lifting and unloading under a novel reward regime.

The average reward curve, indicated by the dashed line, reflects the general trend observed across all runs. The dashed line denotes the mean across the five seeded runs (Experiments 0–4). It shows steady and significant progress with diminishing returns after step 800. This trend suggests that the agents benefit from a curriculum of rewards that add complexity incrementally so that behavior can be refined step-by-step without overwhelming the learning process. This changeover between stages would appear to be successful since most agents utilize the first successful log handling policy to ramp to task completion that entails vertical positioning. The results confirm that curriculum learning enables more seamless adaptation to new reward objectives by promoting behavioral continuity and policy reuse.

The average reward curve, indicated by the dashed line, reflects the general trend observed across all runs. The dashed line denotes the mean across the five seeded runs (Experiments 0–4). It shows steady and significant progress with diminishing returns after step 800. This trend suggests that the agents benefit from a curriculum of rewards that add complexity incrementally so that behavior can be refined step-by-step without overwhelming the learning process. This changeover between stages would appear to be successful since most agents utilize the first successful log handling policy to ramp to task completion that entails vertical positioning. The results confirm that curriculum learning enables more seamless adaptation to new reward objectives by promoting behavioral continuity and policy reuse.

5.2. Evaluation

Figure 6 is a heatmap of the cumulative reward earned by agents with four curriculum configurations. Each curriculum configuration was trained with five independent runs (Experiments 0–4) using different random seeds. These configurations demonstrate methods of composition for the reward functions, with each including three distinct components: r_1, r_2, and r_3. Specifically, r_1 incentivizes successful location of the log, r_2 incentivizes lifting and reaching the unloading point above the bed, and r_3 incentivizes reaching the bottom of the bed with motion stability. It accomplishes this by penalizing vertical log velocity with strongly weighted impact through cubic weighting.

The most rewarding environment is $r_1 r_2 + r_3$ with the weighting factor w equal to 10. Here, w denotes the scalar reward scaling factor applied to the unloading/target-reaching terms (i.e., $w = b$ in the reward definition in Section 4.3.3), and the heatmap varies w to study the sensitivity to this scaling. This two-stage curriculum

first trains the agent using a combination of grasp, lift, and moving to the point above the bed rewards, so that it can learn robust manipulation skills. Once stable behavior has been learned, the full reward with the term focused on finishing the unloading r_3 is added in the second stage to fine-tune. The increased final reward in this setup illustrates the value of a progressive reward curriculum. Basic abilities are acquired prior to the introduction of higher-level refinement goals to avoid early penalization or gratification during initial exploration.

The arrangement labeled $r_1+r_2+r_3$, in which the agent learns in three sequential phases beginning with r_1, followed by r_2, and concluding with r_3, results in reduced cumulative rewards. This implies that learning in excessively separated phases can discourage the agent from developing cohesive strategies that encompass several objectives. For example, training to reach the log without the simultaneous context of grasping (or vice versa) could generate poor motor patterns that fail to generalize when all the elements are later reunited.

The $r_1r_2r_3$ configuration, in which the agent receives the complete compound reward from the start, performs worse than stage-wise curricula. This outcome demonstrates the challenge of optimizing an extremely nonlinear reward structure from the ground up. The large effect of the stability term r_3, cubically scaled and rewarding stable vertical movement, can dominate early training stages and lead the learning process off course prior to building underlying control policies. Without a preliminary stage of organized supervision, the agent might not be able to discover purposeful behavior sequences.

Additionally, in all the explored configurations, for the two values of w tested ($w \in \{1, 10\}$), an increase in the reward scaling factor w results in better cumulative rewards. This indicates that stronger reinforcement signals enhance the effect of well-organized reward elements, especially in curriculum-based learning. However, the gap between curriculum-based training and flat training (e.g., $r_1r_2r_3$) remains considerable even with heavier weights, demonstrating the inherent strengths of progressive learning methods.

The success rate of the best-performing agent with 1024 parallel environments in the Isaac Gym simulator was measured as part of the final evaluation. Success in an episode was defined as the agent successfully performing the entire log loading task: pickup, lift, and placing down the log appropriately within the environment's time constraints. Episodes that did not satisfy the success condition before the episode time limit were counted as failures (timeout). The agent accomplished 966 out of 1,024 trials, which is equivalent to a 94% success rate. This superior performance demonstrates the stability of the policy acquired under a wide variety of initial conditions, which validates the curriculum-guided reward shaping approach used during training.

We note that this success rate is measured from a single evaluation run, and we do not report confidence intervals or repeated evaluations across multiple random seeds; future work should include statistical reporting (e.g., mean $\pm$ std across repeated runs) and comparisons to classical or hand-crafted baselines.

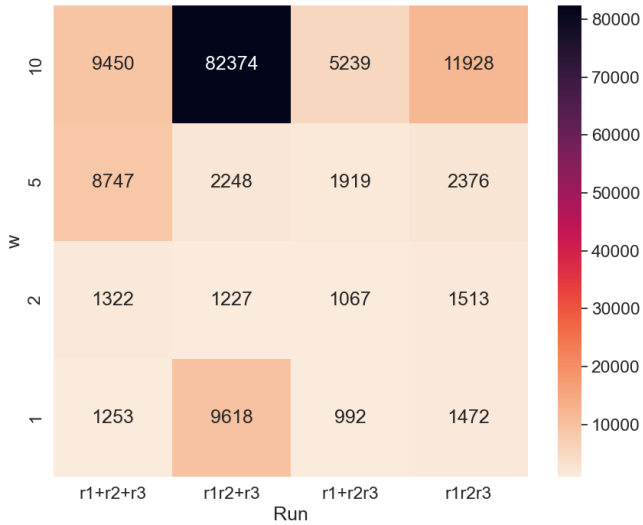

Fig. 6. Returns for various versions of curriculum: Results after 600 combined epochs. The color encodes cumulative episode return (sum of per-step rewards; unitless), where higher values indicate better task performance.

To improve reproducibility and transparency, we will provide the simulation model/assets, the training code and configuration files[31], and the trained policy checkpoints, together with a supplementary video demonstrating the learned policy in the simulator.

For the evaluation of the generalization abilities of the agent, a series of tests was performed. The tests were designed to test the agent's response to unseen states in the training. The tests involved elevated ground planes, variations of wood sizes, inclinations of the forestry forwarder, and rough terrain. These tests were not part of training; they provide an indication of generalization outside the nominal training distribution and, as expected, performance degrades in more challenging conditions.

Table 3 summarizes the reported generalization results.

6. Discussion

Most of the trained agents discussed in the evaluation chapter are capable of transferring the log, but the difference comes from performance and the occurrence of failure behaviors. The most common failure behaviors, which contribute to a decrease in success rate, are associated with grasping behavior when the agent is not capable of grasping the log. The failures with grasping often come from the forwarder trying to reach logs located at positions close to the ends of the loading

Table 3. Generalization tests for the best-performing agent under conditions not used during training. Success rate is reported over evaluation episodes, as in the nominal evaluation.

Condition	Success rate / trend
Log radius scale −10%	80%
Log radius scale −25%	30%
Log radius scale −50%	18%
Log radius scale +25%	78%
Log radius scale +50%	51%
Elevated ground (step size 0.2 m up to 1 m)	~13% success-rate drop per +0.2 m
Rough terrain (cubes with varying elevation)	76%

area, when the grapple approaches the log from the side, leading to pushing the log to an area off-limits for the arm. Some of the agents learned a physics exploitative behavior, when the log is pushed (not lifted) to the side of the machine and uses the collision to lift the log by pushing it up the machine side, which is consistent with the broader phenomenon of specification gaming/reward hacking in reinforcement learning [32].

The generalization experiments are summarized in Section 5.2 (Table 3). Overall, the results indicate partial generalization: moderate changes can be handled, while larger distribution shifts (e.g., large log-scale changes or substantial elevation changes) lead to significant degradation, which motivates domain randomization and/or higher-fidelity modeling in future work.

The possibility of transferring the agent to the real-world machine can be judged by the feasibility of obtaining the required parameters. Most of the parameters used for creating actions, observations, and rewards consist of positions of the log and parts of the machine. The positions of the log center, grapple body, and grapples may be obtained with stereo cameras or lidar, combined with segmentation or object detection [9]. Joint positions and velocities may be collected by joint position sensors and accelerometers. The remaining parameters in the reward or observation are derived from these positions. However, obtaining observations is not sufficient for sim-to-real transfer: the simulation must also capture the real dynamics with sufficient fidelity. In particular, real forwarder cranes are hydraulically actuated with compliance, friction, and valve dynamics, and they operate on deformable/rough terrain with sensor noise and unmodeled disturbances. These effects are only coarsely represented in the current simplified rigid-body simulation, which can lead to policies that do not transfer without additional system identification, domain randomization, and/or training in a higher-fidelity simulator that models hydraulic actuation. The sim-to-real transfer may start from further training of the agent on a high-fidelity simulation model with hydraulics. After training and validating the agent on a high-fidelity simulation model, transfer may be done with the virtual environment presented in [8], where the log and surrounding environment are recreated virtually. The virtual environment will allow the

agent to make several attempts before manipulating the real machine and to select the best-performing action sequence.

7. Conclusion

The study resulted in an autonomous forestry forwarder trained to handle logs within the Isaac Gym simulation environment. The best performing agent is capable of locating, grasping, transporting, and delivering the log to the target location inside the bed with a 94% success rate (where success is defined as reaching the target region with low vertical velocity, without requiring grapple release). The integral part of the approach used is curriculum and reward shaping that decomposes the reward into three components: moving towards the log (r_1), lifting and moving it to the unloading point above the bed (r_2), and unloading it with stability constraints (r_3). Each of these components corresponded to a subtask of the log handling process. Introducing each progressively through curriculum learning helped to reduce the complexity of the task for the agent. This structured reward design simplified the learning problem by guiding the agent through manageable subtasks and focusing it on one objective at a time.

The experiments suggest that curriculum-based sequencing of reward components improves training stability and final performance. Agents trained with the progressive curriculum achieved higher success rates in handling logs than those trained with the full composite reward from the outset. The phased introduction of objectives enables the agent to master each subtask sequentially, which reduces training oscillations and accelerates convergence. However, these conclusions are based on a limited evaluation protocol (single evaluation run without confidence intervals), and the generalization tests in Table 3 show notable degradation under distribution shifts. The findings confirm that curriculum learning is an effective strategy to decompose complex tasks and optimize reinforcement learning outcomes. The best performing resulting agent is able to locate, grasp, lift, and transport the log to the bed. The study showed a potential set of parameters, action, and observation spaces for automating the log loading process.

Even though the experiments showed the potential for automation of the forestry forwarder with reinforcement learning, the study has a number of factors that limit the end performance of the agent. Despite these limitations, the agent shows partial generalization ability, with performance degradation in unseen scenarios. Addressing the limiting factors of the training process may result in higher generalization capabilities and enable smoother transfer to real-world applications. The study uses an old version of the Omniverse Isaac Gym, which has limited capability for randomization of the environment and lacks hydraulic simulation capabilities; consequently, the agent was trained in a simplified environment. For future studies, continuation of training or using the current agent on a high-fidelity simulation model, e.g. in a Mevea simulator, is suggested. One of the areas that

future studies may explore is using multiple agents for each of the tasks, switching after successful completion of the previous tasks.

References

[1] D. Shevchuk, I. Malysheva, M. Alizadeh, and H. Handroos. Simulated and experimental analysis of a log crane with conventional and direct driven hydraulics. In *ASME/BATH 2021 Symposium on Fluid Power and Motion Control*, Fluid Power Systems Technology, page V001T01A044, 10 2021. 10.1115/FPMC2021-68939.

[2] V. Zhidchenko, E. Startcev, and H. Handroos. Blindfolded Operation as a Method of Haptic Feedback Design for Mobile Machinery. In Hiroyuki Kajimoto, Pedro Lopes, Claudio Pacchierotti, Cagatay Basdogan, Monica Gori, Betty Lemaire-Semail, and Maud Marchal, editors, *Haptics: Understanding Touch; Technology and Systems; Applications and Interaction*, pages 323–337, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-70058-3. 10.1007/978-3-031-70058-3_27.

[3] A. Hekmatmanesh, V. Zhidchenko, K. Kauranen, K. Siitonen, H. Handroos, S. Soutukorva, and A. Kilpeläinen. Biosignals in human factors research for heavy equipment operators: A review of available methods and their feasibility in laboratory and ambulatory studies. *IEEE Access*, 9:97466–97482, 2021. 10.1109/ACCESS.2021.3092516.

[4] J. Andersson, K. Bodin, D. M. Lindmark, M. Servin, and E. Wallin. Reinforcement learning control of a forestry crane manipulator. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2121–2126, 2021. 10.1109/IROS51168.2021.9636219.

[5] E. Wallin, V. Wiberg, and M. Servin. Multi-log grasping using reinforcement learning and virtual visual servoing. *Robotics*, 13(1), 2024. ISSN 2218-6581. 10.3390/robotics13010003.

[6] R. Dhakate, Ch. Brommer, Ch. Bohm, H. Gietler, S. Weiss, and J. Steinbrener. Autonomous control of redundant hydraulic manipulator using reinforcement learning with action feedback. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7036–7043, 2022. 10.1109/IROS47612.2022.9981425.

[7] V. Wiberg, E. Wallin, T. Nordfjell, and M. Servin. Control of rough terrain vehicles using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 7(1):390–397, 2022. 10.1109/LRA.2021.3126904.

[8] E. Ayoub, P. Levesque, and I. Sharf. Grasp planning with CNN for log-loading forestry machine. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11802–11808, 2023. 10.1109/ICRA48891.2023.10161562.

[9] T. Semberg, A. Nilsson, R. Björheden, and L. Hansson. Real-time target point identification and automated log grasping by a forwarder, using a single stereo camera for both object detection and boom-tip control. *Silva Fennica*, 58(1), 2024. 10.14214/sf.23062.

[10] P. La Hera, O. Mendoza-Trejo, O. Lindroos, H. Lideskog, T. Lindbäck, S. Latif, S. Li, and M. Karlberg. Exploring the feasibility of autonomous forestry operations: Results from the first experimental unmanned machine. *Journal of Field Robotics*, 41(4):942–965, 2024. https://doi.org/10.1002/rob.22300.

[11] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, Massachusetts, second edition edition, November 2018. ISBN 978-0-262-03924-6. 10.1109/TNN.1998.712192.

[12] R. Tufano, S. Scalabrino, L. Pascarella, E. Aghajani, R. Oliveto, and G. Bavota. Using reinforcement learning for load testing of video games. In *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, pages 2303–2314, 2022. 10.1145/3510003.3510625.

[13] Z. Chen. Research on autonomous navigation and control of unmanned surface vehicles based on reinforcement learning algorithms and multi-objective optimization models. In *Proceedings of the 4th International Conference on Computer, Artificial Intelligence and Control Engineering*, CAICE '25, page 365–372, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712647. 10.1145/3727648.3727709.

[14] K.G. Jash, J. Patel, R. Gupta, N.K. Jadav, S. Tanwar, and S. Desai. Reinforcement learning-based optimized driving behaviour framework for autonomous vehicles in intelligent transportation system. In *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, volume 1, pages 1–6, 2024. 10.1109/ICEECT61758.2024.10739324.

[15] P. Agrawal, S.iand Mitra. *Deep Reinforcement Learning in Healthcare and Biomedical Research*, pages 179–205. 2024. 10.1002/9781394272587.ch9.

[16] Y. Bai, Y. Gao, R. Wan, S. Zhang, and R. Song. A review of reinforcement learning in financial applications. *Annual Review of Statistics and Its Application*, 12(Volume 12, 2025):209–232, 2025. ISSN 2326-831X. 10.1146/annurev-statistics-112723-034423.

[17] H.-H. Huang, C.-K. Cheng, Y.-H. Chen, and H.-Y. Tsai. The Robotic Arm Velocity Planning Based on Reinforcement Learning. *International Journal of Precision Engineering and Manufacturing*, 24(9):1707–1721, September 2023. ISSN 2005-4602. 10.1007/s12541-023-00880-x.

[18] I. Kurinov, G. Orzechowski, P. Hämäläinen, and A. Mikkola. Automated Excavator Based on Reinforcement Learning and Multibody System Dynamics. *IEEE Access*, 8:213998–214006, 2020. ISSN 2169-3536. 10.1109/ACCESS.2020.3040246.

[19] J. Huh, J. Bae, D. Lee, J. Kwak, C. Moon, C. Im, Y. Ko, T.K. Kang, and D. Hong. Deep learning-based autonomous excavation: A bucket-trajectory planning algorithm. *IEEE Access*, 11:38047–38060, 2023. 10.1109/ACCESS.2023.3267120.

[20] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T.P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. June 2016. 10.48550/arXiv.1602.01783. arXiv:1602.01783 [cs].

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. August 2017. 10.48550/arXiv.1707.06347. arXiv:1707.06347 [cs].

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 2018. 10.48550/arXiv.1801.01290.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. (arXiv:1312.5602), December 2013. 10.48550/arXiv.1312.5602.

[24] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, Montreal Quebec Canada, June 2009. ACM. ISBN 978-1-60558-516-1. 10.1145/1553374.1553380.

[25] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M.E. Taylor, and P. Stone. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020. ISSN 1533-7928. 10.48550/arXiv.2003.04960.

[26] C. Florensa, D. Held, M. Wulfmeier, and P. Abbeel. Reverse curriculum generation for reinforcement learning. *CoRR*, abs/1707.05300, 2017. 10.48550/arXiv.1707.05300.

[27] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477, 2018. 10.48550/arXiv.1802.09477.

[28] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance GPU-based physics simulation for robot learning. *CoRR*, abs/2108.10470, 2021. 10.48550/arXiv.2108.10470.

[29] K. Donald, B. Boswell, D. Amishev, and J. Hunt. *Winch-Assist Forwarder: Best Practice Manual*. FPInnovations, Pointe-Claire, QC, Canada, 2018. ISBN 978-0-86488-583-8. URL https://www.bcforestsafe.org/wp-content/uploads/2021/03/WinchAssistForwarderBMP_compressed-compressed.pdf.

[30] M. Strandgard, R. Mitchell, and M. Acuna. Time consumption and productivity of a forwarder operating on a slope in a cut-to-length harvest system in a pinus radiata d. don pine plantation. *Journal of Forest Science*, 63:324–330, 09 2017. 10.17221/10/2017-JFS.

[31] Lappeenranta-Lahti University of Technology. Towards reinforcement learning based log loading automation files. April 2026. 10.5281/zenodo.19660824.

[32] V. Krakovna, J. Uesato, V. Mikulik, M. Rahtz, T. Everitt, R. Kumar, Z. Kenton, J. Leike, and S. Legg. Specification gaming: The flip side of AI ingenuity. DeepMind Blog, 2020. URL https://www.deepmind.com/blog/specification-gaming-the-flip-side-of-ai-ingenuity.