Can Policy Learning with Time Limits be used for Contact-rich Industrial Manipulation?

Bharat Singh*, Jack Kilpatrick, Sebastian Joya-Paez, Ryo Hanai,
Ixchel G. Ramirez-Alpizar, Natsuki Yamanobe, Yukiyasu Domae

***Abstract*— Contact-rich industrial manipulation poses a significant challenge for reinforcement learning policies, requiring dexterous interactions with objects exhibiting complex contact dynamics. Additionally, in industrial applications, completion deadlines are equally important to task success, for integration into wider processing pipelines. Further, in the standard reinforcement learning setting, failure to account for remaining time in episodic tasks can result in state aliasing or inconsistent temporal difference errors, therefore, this research work seeks to determine the most effective integration of time limits in policy learning. We propose that the remaining time be used as both an input and a scaler for the task success reward, demonstrating the effectiveness for the dexterous unscrewing of a nut from a bolt. The resulting time-based policy completes the unscrewing task with a success rate of 90% in 10 simulated trials, the highest of all approaches considered, including a standard baseline. It takes an average completion time of 21.67 seconds across the trials, given a 35 second time limit, which, while not the fasted method considered, may indicate more stable motion resulting from awareness of the time limit. Finally, the efficacy of the learned unscrewing policy is validated on a real UR5e manipulator for the nut-bolt disassembly task.**

I. Introduction

Contact-rich industrial manipulation, while of great interest in the re-manufacturing economy, is challenging to automate, particularly due to the complex interaction dynamics between the objects to be manipulated, which often requires considerable dexterity to navigate and significant computational resources to simulate. Despite these constraints, the dismantling of end-of-life products is of particular interest in the automotive sector, where an efficient, widely applicable method is essential for meeting sustainable development goals [1], [2]. In the literature [3]–[5], there are several existing frameworks for dismantling such products. These often separate the task into several simple, well-understood phases, such as the removal or reorganization of basic components, typically executed according to a high level planner which determines phase ordering and possible failure modes. Such systems often take the form of a Finite State Machine (FSM), operating a rule-based policy against live vision feedback. However, the design of general, robust rules for many parts of interest is a very cumbersome task and is very hard to scale to complex assemblies. Likewise, solely relying on vision feedback is infeasible in many environments, particularly where the robot performing the task can occlude the objects being interacted with due to a lack of on-board cameras. Where such line-of-sight issues can arise, systems that have the capability to augment known object states with robot kinematics and force feedback are of great interest [6], [7]. Given their success across a wide variety of fields, embodied AI agents have emerged as a viable option for learning the required behavior directly from interaction data. In particular, machine learning methods attempt to establish a control policy from demonstration data, an approach that has demonstrated broad effectiveness and generalization for many current disassembly tasks of interest [8]. However, data collection for contact-rich tasks that require dexterous motion alongside representative force and kinematic data is very time and labor intensive and is therefore often infeasible in many industrial settings. Several research works have proposed reinforcement learning methods as a potential solution, performing the combined role of path planning and task decision making optimized solely from a task reward using trial and error [7], [8] .

In the last decade, reinforcement learning algorithms have been successfully applied to many robotics control problems, including pick-and-place tasks, dexterous in-hand manipulation and contact-rich assembly and disassembly [9], [10]. Contact-rich disassembly tasks in particular have received comparatively less focus within the research community, partly owing to their complexity to complete and the cost of accurate simulation. More recently, the emergence of End-to-end GPU-vectorized simulators, such as IsaacLab, mitigate the computational cost of accurate dynamics simulation using parallel training environments [11]. This has allowed a renewed research focus on policy learning for assembly and disassembly applications. *Y. Narang et al.* have also presented IsaacLab's Factory simulation environment for the task of screwing a nut and bolt [12]. In addition, existing work has applied the infinitely rotating HEBI X-series gripper to train a policy for fast completion [13]. In addition, reinforcement learning for human-robot collaboration, applied to nut unscrewing on an electric vehicle battery, has also been explored, though still with an infinitely rotating gripper [14]. These approaches are effective, however many robotic manipulators used for assembly/disassembly tasks can only perform a fixed number of wrist rotations, requiring a more dexterous motion [9]. Additionally, this can introduce efficiency challenges, since the limits of the manipulator must be respected while the task objects are manipulated. This makes previous approaches less suited to autonomous disassembly, but further highlights the need for a reinforcement learning method which considers these

National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
*Corresponding author e-mail: bharat.singh@aist.go.jp

constraints.

This research work concerns the application of reinforcement learning for such disassembly tasks, where dexterous motion is required given finite wrist rotations, applied to the unscrewing of a nut in a nut-bolt assembly. This task has two basic phases: a) unscrewing the nut from the bolt through a combination of grasping, rotating and re-grasping, requiring both discontinuous contact and efficient handling of the end-effector, and b) a separation phase where the nut must be separated from the bolt with stability using the end-effector, then brought to a final target position.

In the standard reinforcement learning setting, *F. Pardo et. al.* have presented the effective integration of time limits as part of the environment state, arguing that this mitigates state aliasing in finite-horizon Markov Decision Processes [15]. Alternatively, they suggest that when these time limits are imposed exclusively during training, they may not be considered part of the environment, and therefore value bootstrapping at states where the time limit is reached can prevent inconsistent temporal difference updates. This work evaluates the use of time limits across a broad set of standard continuous control tasks. However, these problems mainly possess smooth-continuous dynamics, with minimal or no objects to manipulate or contacts between them, as well as relatively few constraints on the feasible motion or allocated completion time. This is in sharp contrast to many industrial applications, such as part disassembly, where there are often many complex objects with interlocking non-convex geometries in the task environment, which must be separated dexterously, with minimal induced collisions, and ideally, within a known time limit for the purposes of efficiency, planning and cost evaluation. Given these constraints, we argue that effectively integrating time limits into policies for contact-rich robotic disassembly tasks is an essential and under-explored research area for industrial applications, which we pursue under the time limit framework presented by *F. Pardo et. al.,* in [15], as well as through the task reward.

Following the above discussion, this research work explores several methods of integrating time limits into reinforcement learning policies trained for contact-rich manipulation tasks, applied to the problem of dexterously unscrewing a nut and bolt with finite wrist joint rotations. The major contributions of this research work are as follows:

1) The integration of time limits into the reinforcement learning policy is explored as both an input to the agent and to the reward function, applied to the task of unscrewing a nut.
2) The reward is designed to enforce one overall behavior which matches realistic human wrist motions.
3) The efficacy of the learned policy is validated on a real UR5e manipulator for unscrewing a nut and bolt.

To the best of authors knowledge, no relevant work has fully addressed all these topics.

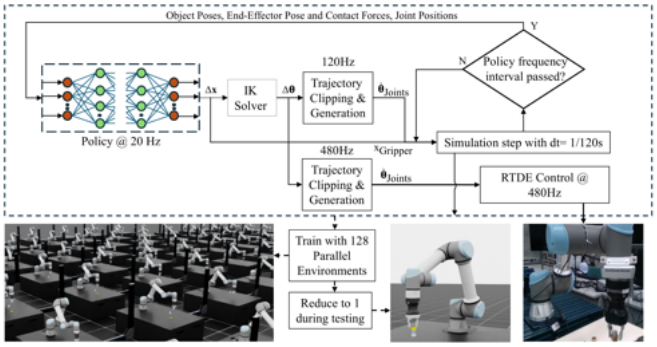


Fig. 1: System Overview

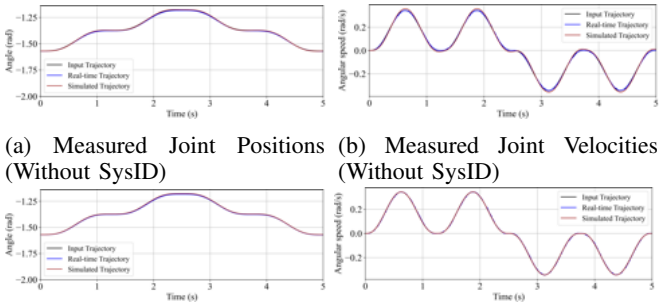
II. Methodology

A. System Overview

Fig. 1 shows a schematic overview of the proposed method applied to the unscrewing task using a UR5e manipulator with a Hand-E gripper. The control policy generates target cartesian displacements and axis-angle offsets for the end-effector, as well as target gripper position, according to the current object poses, joint positions, end-effector pose, and contact forces. The resulting displacements are transformed to joint space as a servo position using inverse kinematics. The target joint positions are clipped to satisfy velocity constraints and are fed to a trajectory generation block which attempts to minimize jerk. Using a GPU-vectorized IsaacLab environment built upon the provided Factory framework, the policy is trained for 24 hours across 128 parallel environments using a single NVIDIA RTX 6000 Ada GPU. The proposed system operates at multiple frequencies for control of the simulator, simulated robot and real robot. Each dynamics step of the simulator occupies $\frac{1}{120}$ seconds, with the control policy operating at 20Hz, or once per 6 simulation steps. The real UR5e manipulator is controlled through the RTDE protocol at 480 Hz. To reduce the Sim2Real gap, we have performed system identification of the physical parameters of the simulated robot, using a vectorized particle swarm optimizer, following the method proposed in the literature [16]. Fig. 2 shows the impact of this step on one sample joint position and velocity trajectory, with identification performed on the simulated joint stiffness, joint friction, and link masses. The joint stiffness and friction [K & F] were optimized in two groups. The first group consists of three joints, namely the shoulder, elbow, and forearm joints and the second group is made up of the last three remaining wrist joints. The optimized parameter values for both groups were found to be [24868, 2.0] and [5652.27, 0.1], respectively. The optimized link masses, in the order of the base joint to the gripper, are as follows: [m_1=3.768, m_2=7.958, m_3 =2.746, m_4=1.269, m_5 = 1.200, m_6 = 0.315, Gripper = 1.399].

B. Concept of Time Limits

Reinforcement Learning is a problem setting in which an agent defined by a decision policy π must determine sequences of actions which maximize the expected cumulative



(a) Measured Joint Positions (Without SysID)

(b) Measured Joint Velocities (Without SysID)

(c) Measured Joint Positions (With SysID)

(d) Measured Joint Velocities (With SysID)

Fig. 2: System Identification

reward in a Markov Decision Process (MDP). An MDP expresses a task environment with sequences of states $s_t \in S$ and actions $a_t \in A$, a transition function $p(s_{t+1}|s_t, a_t)$ and a reward function $r(s_t, a_t)$. In robotics control tasks, typically $A \subseteq \mathbb{R}$. The environment may have an infinite time horizon or consist of fixed-length episodes. Additionally, a constant $\gamma \in [0, 1]$ is used for discounting the cumulative reward. An agent's decision policy is optimal when it maximizes $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$. *Time constraints* of a task can be specified in several ways in reinforcement learning problems. In the standard setting, the discount factor, γ, can be tuned to indicate the preference for immediate rewards in a task expressed as an MDP. An optimal policy π_* must maximize the expected discounted cumulative reward, and so an agent must learn to achieve greater rewards earlier. Task time limits can be enforced by defining an MDP with fixed-length episodes which are terminated by deterministically or stochastically resetting the environment state. In the literature [15], authors have argued that as termination happens according to the time limit, the remaining episode time is a state variable within the MDP. This means the environment is partially observed when the remaining time is not known to the agent, and the state with remaining time zero is considered a terminal state equivalent to any other state-based episode termination criterion such as task success or failure. This is treated as distinct from the case where time-limited episodes are imposed exclusively during training to accelerate learning. This can cause inconsistent temporal difference errors, since value bootstrapping is used for states only if they are reached within the time limit. The authors propose instead to bootstrap where a state is otherwise non-terminal. We argue that both cases are relevant and applicable to industrial manipulation tasks. Where a task completion deadline is imposed, a control policy which can trade off speed and stability based on the time limit, without additional reward engineered for these quantities directly, may be preferable. Particularly in the contact-rich setting, unnecessarily quick motions may cause undesirable robot-object or object-object collisions, which awareness of the remaining time may prevent. Further, as the complexity of the manipulation task increases, it is possible that fixed interval resets during training may be practical for learning the dexterous motion at each phase of the task more quickly, particularly where certain phases of the task may be more contact heavy, requiring more experience to learn due to object interaction dynamics. To thoroughly explore both scenarios, we train and evaluate our policy with the remaining time as input, as well as with value bootstrapping from states where the time limit is reached. We used an episode length of 35 seconds of simulation time, or 700 dynamics steps for the nut and bolt unscrewing task. When given as input to the policy, the remaining time is normalized to the $[0, 1]$ interval through division by the time limit. Aside from this limit, our task does not include any terminal states, meaning the policy can accumulate additional reward by completing the task earlier in the episode. Since the remaining time can be recognized as a valid state variable, we also evaluate the effect of explicitly rewarding the policy in proportion to the remaining time when the task is completed, for comparison with the standard setting where the discount factor is used for this purpose.

C. Algorithm

Proximal Policy Optimization is a model-free reinforcement learning algorithm which produces a stochastic policy, updated iteratively from the most recent environment transitions [17]. The policy is parametrized by an actor network, which predicts the mean and standard deviation of a Gaussian distribution from which the current action is sampled. A critic network predicts the value of states experienced by the current policy, forming a gradient ascent objective for the actor in proportion to the estimated advantage, and a clipped action probability ratio taken between updates. The critic is trained using its prediction error against the returns from the same sets of states. We use the PyTorch implementation provided by the SKRL library [18], which is GPU-vectorized, optimizing the actor and critic using samples collected from many parallel rollouts of the current policy, with actions selected for all environments in a single forward pass. Separate MLP networks are used for the actor and critic, with the default learning rate schedule provided. This decreases and increases the learning rate for both networks by a constant factor when the KL divergence of the policy is above and below a set threshold, respectively.

D. Reward

The key inputs of the task reward function are illustrated in Figures 3 (I), 3 (II), and 3 (III). In Figure 3(III), this includes the distance d from the nut A to a final target position, C, at a fixed height above the bolt. Until the nut reaches the top of the bolt, which is defined as the separation point, we also reward the policy in proportion to the distance from the tool-center-point of the end-effector, B, to the nut A, encouraging stable grasping. Once the separation point is reached, we provide the maximum of this term per timestep, to reduce excessive penalization when the policy unscrews the nut but drops it afterwards. To further ease simultaneous learning of the contact-rich unscrewing careful lifting phases, we add additional reward terms based on the ideal motion at each phase, altered when the nut reaches

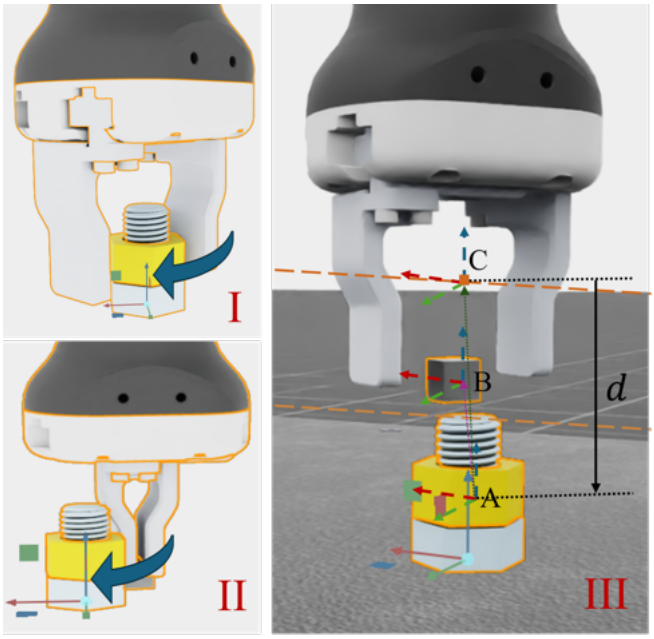


Fig. 3: Key components for proposed reward for the nut-bolt disassembly task. I and II show multiple possible learned behaviors. In I, the policy grasps and rotates with the end-effector, whereas in II, the gripper exterior is used to push. III shows the key objects used in the reward function. The two dotted lines represent target points for the nut to reach, one for separation from the bolt as well as a final target position. The distance, d, to the final target point is taken from the nut center where $d = 0.0360\ m$.

the separation point. For the nut and bolt disassembly task, there are multiple viable unscrewing strategies given finite wrist rotations, two of which are shown in Figures 3 (I) and 3 (II). We found that when using rewards solely related to the task, the policy would typically learn the behavior shown in Figure 3(II), owing to the fast completion time achievable by circumventing the limited wrist rotation of the Hand-E and therefore the time taken to reset the wrist position and re-grasp the nut at each rotation. While this behavior may be faster, a control policy which induces high-speed robot-object collisions is more likely to trigger a safety stop when deployed on real hardware, or cause object damage where part materials have limited impact strength. We assume that the behavior in Figure 3(I) is more desirable, at a completion speed cost, and incentivize this dexterous motion through bonus reward terms based on the end-effector to nut proximity, and the ideal wrist rotation and gripper motion. The result is a continuous, non-sparse, reward which increases monotonically with respect to reach phase of the task, and provides bonuses for the ideal motion. The combined reward function is given by Equation (1),

$$r_t = r_{BB1} + r_{BB2} + r_{IM} + r_{NT1} + r_{NT2} + r_B + r_S + r_{TL} \quad (1)$$

Equations (2-11) present the different components of the reward function, expressed through a concise notation detailed in the appendix. r_{BB1} and r_{BB2} are active when the end effector is within predefined boundaries surrounding the nut, which is determined using the end effector-nut distances d_{xy}^{AB} and d_z^{AB}. r_{IM} incentivizes the ideal motion as illustrated in Fig. 3(I), providing a bonus when the target gripper positions and wrist rotations, defined by $\mathbb{JT}$, are followed, and penalizing otherwise. J_{TH} is a signed value indicating the direction of motion, flipped when the wrist joint limit is reached. Additionally, this term is only active when the nut is unscrewed from the bolt, represented by the condition NU. Prior to separation, r_{NT1} increases the reward logarithmically in the z-distance from the nut to its target position, d_z^{AC}. Once separation occurs and the nut must be lifted, r_{NT1} is replaced by r_{NT2}, which incentivizes reaching the final target position with a fixed linear scale. A fixed additional bonus, r_B is also provided to offset penalization if the nut is dropped. Further, r_S incentivizes the control policy to reduce end-effector rotations and maintain hold of the nut as the final target is approached. Finally, a success bonus r_{TL} is given, scaled by the remaining allocated time when the time limit is considered as part of the environment. The task has been completed successfully when the nut reaches target position C in Fig. 3(III), within a tolerance of 0.00075m in the z-axis and 0.0025m in the xy-axes. The remaining time is calculated by subtracting the current discrete timestep CT from the episode length EL and dividing by EL to obtain a value in the [0,1] range.

$$\begin{aligned} r_{BB1} = \{d_{xy}^{AB}/0.707 < 0.3 \longrightarrow \{-d_{xy}^{AB}/0.707 \;||\; -1.0\}\} \\ \oplus \{d_z^{AB}/0.05 < 0.15 \longrightarrow \{-d_z^{AB}/0.05 \;||\; -1.0\}\} \end{aligned} \quad (2)$$

$$\begin{aligned} r_{BB2} = \{d_{xy}^{AB}/0.707 < 0.15 \longrightarrow \{1.0 \;||0.0\}\} \\ \odot \{d_z^{AB}/0.05 < 0.1 \longrightarrow \{1.0 \;||0.0\}\} \end{aligned} \quad (3)$$

$$\begin{aligned} r_{IM} = \{\{J_{TH} \odot \mathcal{R}_z > 0.0 \longrightarrow \{3.0 \;||\; -1.0\}\} \\ \oplus \{J_{TH} \odot \mathcal{A}_g < 0.0 \longrightarrow \{3.0 \;||\; -1.0\}\}\} \odot NU \end{aligned} \quad (4)$$

$$r_{NT1} = \left(-2 \leq \frac{2\log_{10}(1e^{-5} + 10|d_z^{AC}|))}{0.55} \leq 5\right) \odot NU \quad (5)$$

$$\begin{aligned} r_{NT2} = \{d_z^{AC} < 0.0130 \\ \longrightarrow \left(50.0 * \frac{|0.013 - d_z^{AC}|}{0.0130} \;||\; 0.0\right)\} \end{aligned} \quad (6)$$

$$r_B = \{d_z^{AC} < 0.0130 \longrightarrow \{9.0 \;\; || \;\; 0.0\}\} \quad (7)$$

$$\begin{aligned} r_S = \{d_z^{AC} < 0.0130 \longrightarrow \{\{|\mathcal{R}_z| < 0.0 \longrightarrow \{6.0||0.0\}\} \\ \oplus \{\mathcal{A}_g < 0.0 \longrightarrow \{3.0||0.0\}\}|| \; 0.0\}\} \end{aligned} \quad (8)$$

$$r_{TL} = \{d_z^{AC} < 0.005 \longrightarrow \left(50.0 * \frac{EL - CT}{EL} \;||\; 0.0\right)\} \tag{9}$$

$$J_{TH} = \{\text{sgn}\{\mathbb{JT}\} == -1 \rightarrow \{1.0 \;||\; -1.0\}\} \tag{10}$$

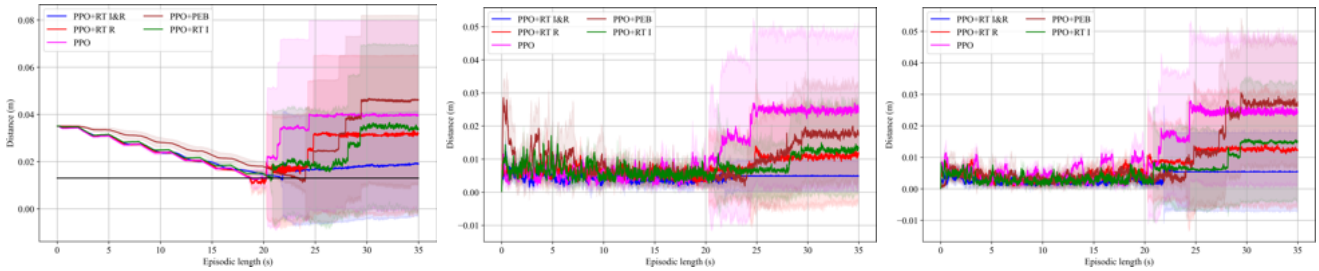
$$NU = \{d_z^{AC} < 0.0130 \longrightarrow 1.0 \;||\; 0.0\} \tag{11}$$

E. Evaluation Experiments

To analyze both task performance and possible changes in policy behavior resulting from our imposed time limit, we train five separate unscrewing policies for 250k environment steps with a constant time limit of 700 steps or 35 seconds per episode, utilizing the remaining time to varying extents. We include a baseline PPO agent with no additions, a policy with the remaining time as input (PPO+RT I), then as a scaler multiplying the task success reward (PPO+RT R), as well as a combination of these (PPO+RT I&R) and finally a policy trained with Partial Episode Bootstrapping (PPO+PEB). All policies were evaluated across 10 trials, each with a fixed random seed. We evaluate the task performance and stability of motion using the unscrewing success rate, nut to target distance per timestep, and the distance between the nut and robot tool center point per timestep, examining the xy-axes and z-axis separately. For application to the hardware, we re-trained the best performing policy across all evaluation trials for 500k environment steps, clipping high changes in joint angle to minimize path deviation safety stops resulting from robot-object collisions, which are not enforced in simulation. We apply the resulting policy without utilizing real inputs, sending the clipped target trajectories produced in simulation, up-sampled to 480Hz.

III. Results & Discussion

The results of all 10 evaluation trials are shown in Table I. The policy with the remaining time as input and with a time-scaled success reward achieves the highest success rate of 90%. This suggests that, as the time-limit framework proposed by [15] recognizes the remaining time as a valid state variable within episodic MDPs, explicitly rewarding for the resulting states in combination with a time-aware policy may provide additional benefit. Beyond the standard setting with only discounted cumulative reward, where any policy will seek to maximize value regardless of any set time limit, explicit time-awareness and direct association with the task reward may ease the learning of actions which consistently reach greater per timestep reward that can be achieved within the specified time limit. Providing the remaining time only as a scaler for the task success reward, with no input changes, resulted in the fastest policy in all evaluation trials, with a mean nut-bolt separation time of 18.67 seconds, but a success rate of only 70%. While this partially observed case may further incentivize fast completion, it is possible that the policy cannot as easily and as directly learn action sequences which are slower, but consistently result in success given sufficient remaining time, which we argue is a more useful trade-off when task time limits are imposed, particularly as all policies already separated the nut and bolt with more than 10 seconds of the allocated 35 seconds remaining. The resulting difference in behavior when the time-scaled success reward is used, with both the presence and absence of the remaining time input, can be observed in Figures 4(a), 4(b) and 4(c). In Fig. 4(a), all policies separate the nut and bolt within approximately 20-25 seconds, however in Fig. 4(b) and 4(c), it is clear that only the policy with both the remaining time as input and as a reward scaler achieves a relatively stable mean fingertip to nut distance in the post-separation phase, as well as the lowest post separation means of 0.0051m and 0.0060m in the xy-axes and z-axis respectively, as shown in Table I. Combined with the slightly slower unscrewing time compared to the reward-only policy, and lower fingertip-nut and nut-target distances pre/post-separation, this suggests that the time-aware and time-rewarded policy produces more stable unscrewing and grasping motions to achieve more consistent success, at the cost of completion speed. In contrast, the baseline PPO agent achieves a relatively quick separation time of 18.8 seconds, but frequently drops the nut post-separation, as shown by the large step change in the evaluated nut-to-object distances and the second highest mean post-separation nut-target distance of 0.0383m across all cases. This indicates that without use of the time limit, the policy attempts to reach the final target as quickly as possible and may be more unstable, leading to a lower success rate of 60%. Using the remaining time as input with no reward changes leads to no change in success rate, but a slower completion time of 20.69 seconds, as well as lower mean post-separation nut-target, fingertip-nut xy and fingertip-nut z distances of 0.0271m, 0.0101m and 0.0178m respectively. This could suggest that time-awareness may lead to slower, more steady motions by itself, but that applying the remaining time as a reward scaler of the task completion bonus may also be needed to incentivize steady motions which result in success, however further evaluation trials would be essential in confirming this difference. For completeness, we also examined the impact of Partial Episode Bootstrapping, where value errors are bootstrapped from states where the time limit is reached. Interestingly, this lead to a degradation in performance from the baseline, with the resulting policy achieving a success rate of 50%, an average completion time of 23.45 seconds. While bootstrapping at time limits should reduce inconsistencies in temporal difference updates, this may indicate that they are adding noise to the learning process. This may be a particular risk for the nut-bolt disassembly task, since the time-limited state value can vary significantly depending on whether the nut is unscrewed, dropped, or successfully lifted within the remaining time available, resulting in erroneous policy updates which are not grounded in any observed rewards after the time limit expires. Using a longer training horizon, and more closely observing differences in the value prediction errors, depending on the task progress achieved within the



(a) Nut to target distance (b) Fingertip to nut xy distance (c) Fingertip to nut z distance

Fig. 4: Impact analysis of time limits on the policy performance

TABLE I: Evaluation trials of policies with varying time-awareness in simulation

Overall

	PPO+RT I&R	**PPO+RT R**	**PPO**	**PPO+PEB**	**PPO+RT I**
Task success rate (%)	**90%**	70%	60%	50%	60%
Separation time (s)	21.67	**18.67**	18.80	23.45	20.69

Pre-separation

	Mean	*Max.*	*Min.*	*Mean*	*Max.*	*Min.*	*Mean*	*Max.*	*Min.*	*Mean*	*Max.*	*Min.*	*Mean*	*Max.*	*Min.*
Fingertip midpoint-nut xy distance (m)	**0.0044**	0.0121	0.0001	0.0059	0.0101	0.0001	0.0055	0.0112	0.0001	0.0074	0.0283	0.0001	0.0063	0.0171	0.0001
Fingertip midpoint-nut z distance (m)	**0.0024**	0.0056	0.00004	0.0039	0.0092	0.00004	0.0046	0.0114	0.00004	0.0037	0.0070	0.00004	0.0037	0.0070	0.00004

Post-separation

	Mean	*Max.*	*Min.*	*Mean*	*Max.*	*Min.*	*Mean*	*Max.*	*Min.*	*Mean*	*Max.*	*Min.*	*Mean*	*Max.*	*Min.*
Nut-target distance (m)	**0.0178**	0.0194	0.0160	0.0291	0.0332	0.0148	0.0383	0.0405	0.0270	0.0400	0.0468	0.0240	0.0271	0.0366	0.0157
Fingertip midpoint-nut xy distance (m)	**0.0051**	0.0054	0.0050	0.0126	0.0168	0.0055	0.0185	0.0216	0.0125	0.0139	0.0177	0.0082	0.0101	0.0151	0.0052
Fingertip midpoint-nut z distance (m)	**0.0060**	0.0062	0.0058	0.0149	0.0181	0.0068	0.0193	0.0222	0.0111	0.0204	0.0269	0.010	0.0178	0.0269	0.0020

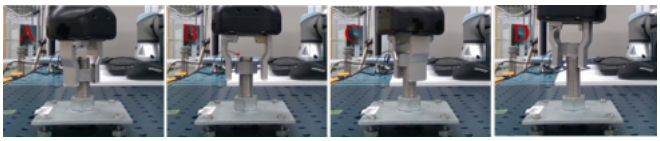


Fig. 5: Validation of the trained policy on real hardware

time limit, would be a key area for further exploration. Finally, the resulting motion for hardware validation is shown at four evenly-spaced time intervals in Fig. 5. Despite the apparently increased stability of the time-aware policy, we observed safety stops during the unscrewing phase, induced by path deviations when the real robot attempted to push the nut and bolt, or its grasp was misaligned during unscrewing. To prevent this, we further clipped the target trajectories on the real hardware, wherever significant deviations in the joint positions were observed. By further adjusting the targets and not utilizing the real inputs, this limits the applicability of the policy in the presence of variations in the object poses, sizes, and interaction forces, and prevents recovery from errors or potential failures. We also assume the object poses are always known to the policy, which may not be the case when visual feedback is limited. The removal of this assumption, as well as the addition of real world inputs alongside a trajectory generation method which respects contact-forces to mitigate path-deviations, such as an impedance-based method, are therefore essential for future work.

IV. Conclusion

This work proposes the combination of time-limit-based reward and input for reinforcement learning policies trained for contact-rich industrial manipulation, applied to the task of nut and bolt disassembly with a finitely rotating wrist. Using the GPU-vectorized IsaacLab framework, we train a time-aware policy with a time limit of 35 seconds per attempt for 250k environment steps across 128 parallel environments, achieving an unscrewing success rate of 90% and average completion time of 21.67 seconds across 10 simulation trials. We further validate our approach on a real UR5e manipulator. Our method achieves the highest task

success rate in simulation when compared to a baseline PPO agent, an agent with only a remaining time input or a time-based success reward, and with partial episode bootstrapping. While the proposed combination is not the fastest, we argue it's high success rate may be indicative of a learned trade-off between stability against efficiency given sufficient remaining time, which is particularly beneficial for industrial contact-rich manipulation where consistent operation success is desirable. The proposed approach has several drawbacks. All experiments were conducted using fixed length episodes with a single time-limit, however, this means generalization of the trained policy to variable episode lengths, in absence of training schemes such as domain randomization, is unclear. In addition, generalization of the proposed approach to longer or shorter horizon problems, such as industrial tasks of differing complexity, requires further exploration. Further, time-based rewards are only given for successful attempts, limiting potential benefits early in training. Additionally, we did not explore the impact of varying γ, the training horizon, or which reward terms are scaled by the remaining time. In addition, we does not examine the impact of partial episode bootstrapping on the value error directly, or the use of a recurrent architecture in the partially observed case. Further, we did not utilize the real environment inputs or a control method which minimizes path deviations based on the contact forces, and always assume the ground truth object poses are known to the policy without noise, limiting the deployability, reactivity and robustness of the policy on real hardware, where real world dynamics and uncertainty are both present. The removal of the current object poses and the incorporation of real inputs, either through a visual or force-based control method, taking into account uncertainty and safe compliance, followed by a thorough investigation of the outstanding areas identified, as well as more extensive evaluation trials concerning the desired stability vs. efficiency trade-off, are critical directions for future work.

ACKNOWLEDGMENT

This paper is based on the results obtained from a project, JPNP14004, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

[1] A. R. Das and M. Koskinopoulou, "Towards sustainable manufacturing: A review on innovations in robotic assembly and disassembly," *IEEE Access*, 2025.

[2] J. Hathaway, C. A. Contreras, M. E. Asif, R. Stolkin, and A. Rastegarpanah, "Technoeconomic assessment of electric vehicle battery disassembly–challenges and opportunities from a robotics perspective," *IEEE Access*, 2024.

[3] K. Wegener, S. Andrew, A. Raatz, K. Dröder, and C. Herrmann, "Disassembly of electric vehicle batteries using the example of the audi q5 hybrid system," *Procedia Cirp*, vol. 23, pp. 155–160, 2014.

[4] L. Xia, Y. Hu, J. Pang, X. Zhang, and C. Liu, "Leveraging large language models to empower bayesian networks for reliable human-robot collaborative disassembly sequence planning in remanufacturing," *IEEE Transactions on Industrial Informatics*, 2025.

[5] L. Qi, W. Xu, K. Wang, J. Liu, X. Ye, H. Yang, and Y. Zhong, "Mixed-model product disassembly sequence optimization based on cognitive digital twin," *Journal of Manufacturing Systems*, vol. 82, pp. 497–508, 2025.

[6] T. Migimatsu, W. Lian, J. Bohg, and S. Schaal, "Symbolic state estimation with predicates for contact-rich manipulation tasks," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1702–1709.

[7] Z. He, H. Fang, J. Chen, H.-S. Fang, and C. Lu, "Foar: Force-aware reactive policy for contact-rich robotic manipulation," *IEEE Robotics and Automation Letters*, 2025.

[8] V. Galaiya, R. Masinjila, S. Khatibi, T. E. A. de Oliveira, V. P. da Fonseca, and X. Jiang, "Extraction of non-regular pegs using tactile sensing and reinforcement learning from demonstrations," in *2025 IEEE International systems Conference (SysCon)*. IEEE, 2025, pp. 1–6.

[9] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, T. Nishi, S. Kikuchi, T. Matsubara, and K. Harada, "Learning force control for contact-rich manipulation tasks with rigid position-controlled robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5709–5716, 2020.

[10] B. Singh, R. Kumar, and V. P. Singh, "Reinforcement learning in robotic applications: a comprehensive survey," *Artificial Intelligence Review*, vol. 55, no. 2, pp. 945–990, 2022.

[11] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[12] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu *et al.*, "Factory: Fast contact for robotic assembly," *arXiv preprint arXiv:2205.03532*, 2022.

[13] B. Son, H. Choi, J. Yoon, and D. Lee, "A learning-based estimation and control framework for contact-intensive tight-tolerance tasks," *arXiv preprint arXiv:2210.05524*, 2022.

[14] J. Gao, G. Wang, J. Xiao, P. Zheng, and E. Pei, "Partially observable deep reinforcement learning for multi-agent strategy optimization of human-robot collaborative disassembly: A case of retired electric vehicle battery," *Robotics and Computer-Integrated Manufacturing*, vol. 89, p. 102775, 2024.

[15] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev, "Time limits in reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 4045–4054.

[16] D. Baek, B. Peng, S. Gupta, and J. Ramos, "Online learning-based inertial parameter identification of unknown object for model-based control of wheeled humanoids," *IEEE Robotics and Automation Letters*, 2024.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[18] A. Serrano-Munoz, D. Chrysostomou, S. Bøgh, and N. Arana-Arexolaleiba, "skrl: Modular and flexible library for reinforcement learning," *Journal of Machine Learning Research*, vol. 24, no. 254, pp. 1–9, 2023.

APPENDIX

To concisely express our task reward function, we define equations (2-11) using additional operators. Conditional assignments are indicated using $\longrightarrow$, with each branch separated by $||$ and enclosed with braces $\{\}$. The addition and multiplication of the value of two conditional expressions is indicated by $\oplus$ and $\odot$, respectively. Finally, $==$ indicates an equality check.

As an illustrative example, consider the following equation (12),

$$R = \{Condition \longrightarrow \{O_1 \ ||O_2\}\} \tag{12}$$

Here, if the condition is true, R is equal to O_1 otherwise O_2.