8 Appendix

8.1 Mathematical Details of the IFL Problem Formulation

Recall that ROHE takes the expectation over a distribution of trajectories, $p_{\omega,\theta_0}(\tau)$, where each trajectory $\tau = (\mathbf{s}^0, \mathbf{a}^0, ..., \mathbf{s}^T, \mathbf{a}^T)$ is composed of consecutive task episodes separated by resets and where the state-action tuples come from both π_θ and π_H. This distribution of trajectories is induced by ω and θ_0 because θ_0 parameterizes the initial robot policy π_{θ_0}, and ω affects the states that comprise D_H^t, which updates the robot policy π_θ for subsequent timesteps. In this section, we derive the mathematical relationship between the trajectory distribution $\tau \sim p_{\omega,\theta_0}(\tau)$ and the allocation policy ω.

Given an allocation policy ω, the human policy π_H, and the robot policy π_{θ_t} at each timestep t, the joint hybrid human-robot policy of all robots can be expressed as

$$\pi^t_{H\cup R}(\mathbf{s}) = \begin{bmatrix} \pi_{\theta_t}(s_1)(1 - \mathbb{1}_{\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)_1}) + \pi_H(s_1)\mathbb{1}_{\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)_1} \\ \vdots \\ \pi_{\theta_t}(s_N)(1 - \mathbb{1}_{\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)_N}) + \pi_H(s_N)\mathbb{1}_{\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)_N} \end{bmatrix}, \tag{4}$$

where $\mathbb{1}_{(\cdot)}$ is an indicator function that selects the robot policy π_{θ_t} if robot i is allocated to a human and selects the human policy $\pi_H(s)$ otherwise. For notational convenience, $\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)_i := \sum_{j=1}^M \boldsymbol{\alpha}^t_{ij} \in \{0, 1\}$.

The trajectory distribution $p_{\omega,\theta_0}(\tau)$ can then be expressed as

$$p_{\omega,\theta_0}(\tau) = p_{\omega,\theta_0}(\mathbf{s}^0, \mathbf{a}^0, ..., \mathbf{s}^T, \mathbf{a}^T) \tag{5}$$

$$= \bar{p}^0(\mathbf{s}^0) \prod_{t=0}^{T} \pi^t_{H\cup R}(\mathbf{a}^t|\mathbf{s}^t) \prod_{t=0}^{T-1} \bar{p}(\mathbf{s}^{t+1}|\mathbf{s}^t, \mathbf{a}^t). \tag{6}$$

We comment that the soft and hard reset can be easily incorporated into the transition dynamics $\bar{p}$ depending on the task. For example, for constraint violations (i.e., hard resets) $c(s_i^t) = 1$, we can set $p(s_i^{t+k}|s_i^t, a_i^t) = \delta(s_i^t)$ for $1 \leq k \leq t_R - 1$ and $p(s_i^{t+T_R}|s_i^t, a_i^t) = p_i^0(s^0)$ where $\delta(\cdot)$ is the Dirac delta function and t_R is the hard reset time. Similarly, for goal-conditioned tasks with goal g, soft resets after achieving the goal can be expressed through the transition dynamics $p(s_i^{t+1}|s_i^t, a_i^t) = p_i^0(s^0)$ if $s_i^t \subseteq g$. For MDPs with finite time horizon where the environment resets when the maximum time horizon is reached, we can augment the state with additional time information that keeps track of the timestep in each episode, and reset the state when it times out. In this case, the MDP transition dynamics will be time-dependent: $p_t(s_i^{t+1}|s_i^t, a_i^t)$.

8.2 Fleet-DAgger Algorithm Details

In this section, we provide a detailed algorithmic description of Fleet-DAgger.

Fleet-DAgger uses priority function $\hat{p}$ and t_T to define an allocation policy ω. Concretely, it can be interpreted as a function F where $F(\hat{p}) = \omega$, i.e., a "meta-algorithm" (algorithm that outputs another algorithm) akin to function composition. The pseudocode of Fleet-DAgger is provided in Algorithm 1.

8.3 Additional Experiment Details

8.3.1 Simulation Hyperparameters

Implementations of C.U.R. and baselines are available in the code supplement, where the scripts are configured to run with the same hyperparameters as the simulation experiments in the main text. Recall that $N = 100$ robots, $M = 10$ humans, minimum intervention time $t_T = 5$, hard reset time $t_R = 5$, and operation time $T = 10000$. For reference, additional parameters are in Table 1, where $|S|$ is the dimensionality of the (continuous) state space, $|A|$ is the dimensionality of the (continuous) action space, $\hat{r}$ is the risk threshold below which robots are assigned zero risk, $\hat{u}$ is the uncertainty threshold below which robots are assigned zero uncertainty, and t_I is the length of the initial C.U.R. period during which constraint violation is not prioritized.

Algorithm 1 Fleet-DAgger

Input: MDP $\mathcal{M}$, Number of robots N, Number of humans M, Priority function $\hat{p}$, Minimum teleoperation time t_T, Hard reset time t_R
Output: Allocation policy ω

function $\omega(\mathbf{s}^t, \pi_{\theta_t}, \boldsymbol{\alpha}^{t-1}, \mathbf{x}^t)$ `#The allocation policy` ω `returns a matrix` $\boldsymbol{\alpha}^t \in \{0,1\}^{N \times M}$
  Compute priority scores of each robot: $\hat{p}(s_i^t, \pi_{\theta_t}) \quad \forall i = 1, ..., N$
  Initialize $\boldsymbol{\alpha}_{ij}^t = 0 \quad \forall i, j$
  for $i \in \{1, \ldots, N\}$ **do**
    for $j \in \{1, \ldots, M\}$ **do**
      if $\boldsymbol{\alpha}_{ij}^{t-1} = 1$ **then** `#For robots that were receiving assistance during the last timestep, check whether the minimum intervention time has lapsed using auxiliary information` $\mathbf{x}^t$
        if Intervention type for robot i = `Hard reset` and Intervention duration $< t_R$ **then**
          $\boldsymbol{\alpha}_{ij}^t = 1$
        if Intervention type for robot i = `Teleop` and Intervention duration $< t_T$ **then**
          $\boldsymbol{\alpha}_{ij}^t = 1$
  Let $I = \{i : \sum_{j=1}^{M} \boldsymbol{\alpha}_{ij}^t = 1\}$ `#Set of robots that will continue with past assistance`
  Let $J = \{j : \sum_{i=1}^{N} \boldsymbol{\alpha}_{ij}^t = 1\}$ `#Set of humans that will continue with past assistance`
  Sort robot indices with positive priority scores that are not in I from highest to lowest, denoted as $\{i_1, i_2, ...\}$
  Let $k = 1$
  for $j \in \{1, \ldots, M\} \setminus J$ **do**
    $\boldsymbol{\alpha}_{i_k,j}^t = 1$
    $k = k + 1$
  return $\boldsymbol{\alpha}$

return ω

Environment	$\lvert S\rvert$	$\lvert A\rvert$	$\hat{r}$	$\hat{u}$	t_I
Humanoid	108	21	0.5	0.05	1000
Anymal	48	12	0.5	0.05	250
AllegroHand	88	16	0.5	0.15	2500

Table 1: Simulation environment hyperparameters.

8.3.2 Training Critic Q-Functions

Some IFL algorithms require pretraining a safety critic (C.U.R.) or goal critic (Fleet-ThriftyDAgger) to assist in supervisor allocation. Here we provide details on how we train these critics in practice. Additional details about training these critics in practice are also available in Recovery RL [67] and ThriftyDAgger [9] for the safety critic and goal critic respectively.

To collect a dataset of constraint violations, we simply run Behavior Cloning for a fixed amount of timesteps. Intuitively, since the initial robot policy π_{θ_0} is not highly performant, the robot should expect to encounter constraint violations, and these violations will occur within the state distribution visited by the robot fleet in the initial stages of online training. One could also inject noise into the BC policy to induce more constraint violations, or explicitly solicit human demonstrations of constraint violations as noted in [67]. For Humanoid, Anymal, and AllegroHand, we collect a dataset of 19625 transitions with 376 constraint violations, 19938 transitions with 63 constraint violations, and 19954 transitions with 47 constraint violations, respectively. The safety critic is then trained via Q-learning for 3000 gradient steps where a constraint-violating transition can be interpreted as incurring sparse reward $r = 1$ and all other transitions have reward $r = 0$. To reduce class imbalance issues, transitions are sampled from the replay buffer such that constraint-violating samples constitute 25% of the minibatch, which was found to be useful in practice in [67].

To collect a dataset of successes to pretrain the goal critic, we instead run the expert policy π_H, which is more likely to reach the goal. For AllegroHand, we collect a dataset of 19994 transitions with 489 successes. We then pretrain the goal critic in the same manner as the safety critic. Both the safety and goal critic continue to update during online training with the additional constraint violation and success data encountered.

Environment	Algorithm	Successes (↑)	Hard Resets (↓)	Idle Time (↓)
Humanoid	BC	0.0 ± 0.0	11925.3 ± 118.8	62473.7 ± 869.1
	Random	746.3 ± 40.5	340.0 ± 59.2	1700.0 ± 296.1
	Fleet-ED	617.7 ± 66.3	570.3 ± 139.0	2851.7 ± 694.8
	C.U.R.	$\mathbf{771.0 \pm 25.5}$	$\mathbf{289.3 \pm 21.1}$	$\mathbf{1446.7 \pm 105.3}$
	Expert	894	115	575
Anymal	BC	32.7 ± 0.5	1134.3 ± 33.9	5669.7 ± 170.0
	Random	207.3 ± 27.2	232.3 ± 89.6	1162.3 ± 449.1
	Fleet-ED	$\mathbf{257.3 \pm 1.2}$	109.0 ± 16.9	545.0 ± 84.4
	C.U.R.	$\mathbf{257.0 \pm 8.8}$	$\mathbf{64.3 \pm 8.6}$	$\mathbf{321.7 \pm 42.9}$
	Expert	293	1	5
AllegroHand	BC	70.0 ± 5.0	$\mathbf{523.0 \pm 9.4}$	$\mathbf{2614.3 \pm 46.4}$
	Random	$\mathbf{7360.3 \pm 231.0}$	2954.3 ± 131.0	14767.7 ± 653.2
	Fleet-ED	3032.0 ± 191.2	1764.7 ± 225.4	8818.0 ± 1129.7
	Fleet-TD	4296.3 ± 161.7	1397.7 ± 112.6	6988.0 ± 562.9
	C.U.R.	6032.7 ± 236.2	2343.7 ± 82.5	11715.0 ± 411.9
	Expert	21609	1202	6013

Table 2: Robot policy performance for each of the algorithms in Section 6.2. We report cumulative successes, hard resets, and idle time. We do not report return on human effort as teleoperation is not performed for this experiment.

8.3.3 Physical Experiment Protocol

We execute 3 trials of each of 4 algorithms (Behavior Cloning, Random, Fleet-EnsembleDAgger, C.U.R.) on the fleet of 4 robot arms for 250 timesteps each, for a total of $3 \times 4 \times 4 \times 250 = 12000$ individual pushing actions. Human teleoperation and hard resets were performed by the authors, where teleoperation is collected through an OpenCV (`https://opencv.org/`) graphical user interface and hard resets are physical adjustments of the cube toward the center of the workspace. Each of the 2 ABB YuMi robots is connected via Ethernet to a Linux machine on its local area network; the driver program connects to each machine over the Internet via the Secure Shell Protocol (SSH) to send robot actions and receive camera observations. The 2 YuMis are in different physical locations 0.5 miles apart, and all 4 arms execute actions concurrently with multiprocessing. $10\times$ data augmentation is performed on the initial offline dataset of 375 state-action pairs as well as the online data collected during execution as follows:

- Linear contrast uniformly sampled between 85% and 115%
- Add values uniformly sampled between -10 and 10 to each pixel value per channel
- Gamma contrast uniformly sampled between 90% and 110%
- Gaussian blur with σ uniformly sampled between 0.0 and 0.3
- Saturation uniformly sampled between 95% and 105%
- Additive Gaussian noise with σ uniformly sampled between 0 and $\frac{1}{80} \times 255$

8.3.4 Evaluating Trained Simulation Policies

Here we execute all policies from Section 6.2 after the 10,000 timesteps of online training for an additional 10,000 timesteps *without additional human teleoperation* to evaluate the quality of the learned robot policies in isolation. As in Section 6.2, $N = 100$ robots, $M = 10$ humans (for hard resets only), t_R = 5, and $T = 10000$, where allocation is performed by C-prioritization. We also include the expert policy performance (all $N = 100$ robots teleoperated by the trained PPO supervisor for $T = 10000$ steps) in the table for reference.

Results are in Table 2. We find that the policy learned via C.U.R. outperforms baselines and approaches expert-level performance for Humanoid and Anymal, but is second most performant and significantly below expert-level performance for AllegroHand, indicating $T = 10,000$ timesteps is insufficient for learning a highly performant robot policy in this challenging environment.

8.4 Hyperparameter Sensitivity and Ablation Studies

In this section, we run additional simulation experiments in the IFL benchmark to study (1) ablations of the components of the C.U.R. algorithm (Figure 6), (2) sensitivity to the ratio of number of robots

N to number of humans M (Figure 7), (3) sensitivity to minimum intervention time t_T (Figure 8), and (4) sensitivity to hard reset time t_R (Figure 9). All runs are averaged over 3 random seeds, where shading indicates 1 standard deviation.

Ablations: We test C.U.R.(-i), the C.U.R. algorithm without the initial period during which constraint violation is not prioritized. We also test all subsets of the C.U.R. priority function without the initial period. For example, U. indicates only prioritizing by uncertainty, and C.R. indicates prioritizing by constraint violations followed by risk (no uncertainty). Results suggest that C.U.R. outperforms all ablations in all environments in terms of ROHE and cumulative successes and is competitive in terms of hard resets and idle time. However, as in the main text, C.U.R. and C. incur more hard resets in AllegroHand than alternatives, as again, prioritizing constraint violations for a hard environment where learning has not converged may ironically enable more opportunities for hard resets. Interestingly, while C.U.R. outperforms ablations in ROHE in AllegroHand for large T, U-prioritization's ROHE is significantly higher for small values of T. We observe that since U. achieves very low cumulative successes in the same time period, U. must be requesting an extremely small amount of human time early in operation, resulting in erratic ratio calculations.

Number of Humans: While keeping N fixed to 100 robots, we run C.U.R. with default hyperparameters and vary M to be 1, 5, 10, 25, and 50 humans. In the Humanoid and Anymal environment, as expected, cumulative successes increases with the number of humans. The performance boost gets smaller as M increases: runs with 25 and 50 humans have very similar performance. Despite lower cumulative successes, $M = 10$ achieves the highest ROHE, suggesting a larger set of humans provides superfluous interventions. We also observe that with only 1 human, the number of hard resets and idle time is very large, as the human is constantly occupied with resetting constraint-violating robots, which fail at a faster rate than the human can reset them. Finally, in the AllegroHand environment, the number of humans when $M \geq 5$ does not make much of a visible difference, perhaps due to the relatively high number of cumulative successes.

Minimum Intervention Time: We run C.U.R. with default hyperparameters but vary t_T to be 1, 5, 20, 50, 100, and 500 timesteps. We observe that both decreasing t_T from 5 to 1 and increasing t_T to 20 and beyond have a negative impact on the ROHE due to ceding control prematurely (in the former case) and superfluous intervention length (in the latter). Hard resets are low and idle time is high for large t_T as the humans are occupied providing long teleoperation interventions. This also negatively affects throughput, as cumulative successes falls for very large t_T. Long interventions may also be less useful training data, as in the limit these interventions reduce to more offline data (i.e., labels for states encountered under the human policy rather than that of the robot).

Hard Reset Time: Finally, we run C.U.R. with default hyperparameters but vary t_R to be 1, 5, 20, 50, 100, and 500 timesteps. As expected, the ROHE decreases as t_R increases, as more human effort is required to achieve the same return. The other metrics follow similar intuitive trends: increasing t_R results in a decrease in cumulative successes, decrease in hard resets, and increase in idle time.

Other Parameters: We also found that the batch size (256 in our experiments) and number of gradient steps per experiment timestep (1 in our experiments) significantly impact the policy learning speed as well as computation time, and the effect varies with the size of the fleet (N) and set of human supervisors (M). This is because N and M (and ω) determine how much new data is available at each time t, the batch size and number of gradient steps determine how much data to update the policy with at each time t, and updating the policy with backpropagation is the computational bottleneck in the IFLB simulation.

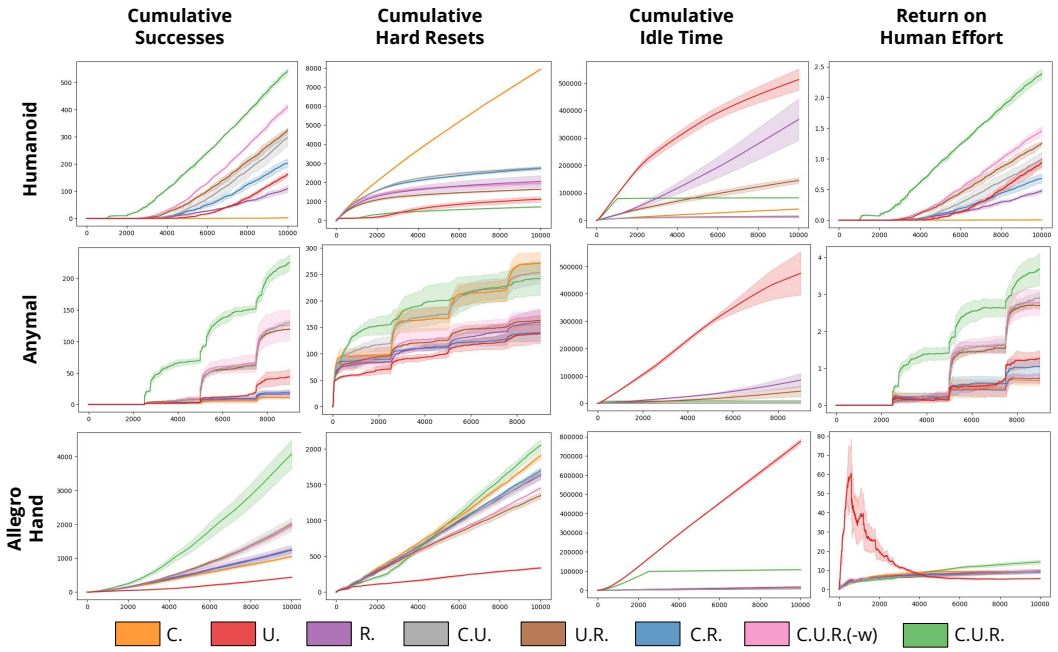

Figure 6: **Ablations:** Simulation results in the Isaac Gym benchmark tasks with ablations of C.U.R., where the x-axis is timesteps from 0 to $T = 10,000$. We plot the metrics described in 6.1. The C.U.R. algorithm outperforms all ablations on all environments in terms of ROHE and cumulative successes (except AllegroHand ROHE for low T values) and is competitive with ablations for cumulative hard resets and idle time.

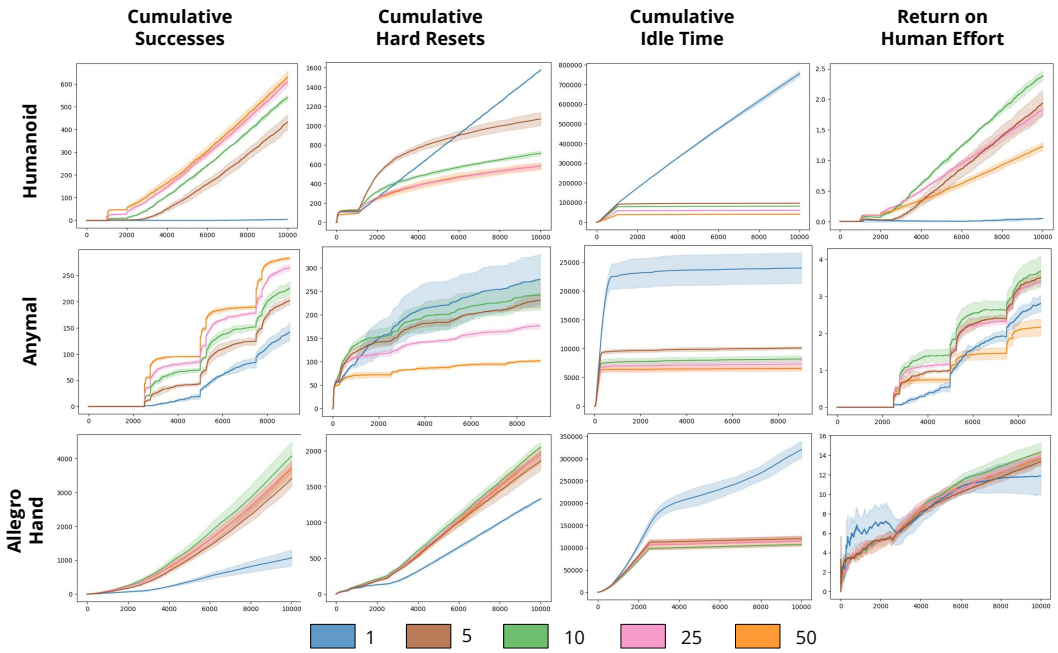

Figure 7: **Number of Humans:** Simulation results in the Isaac Gym benchmark tasks with $N = 100$ robots and M human supervisors, where M varies and the x-axis is timesteps from 0 to $T = 10,000$.

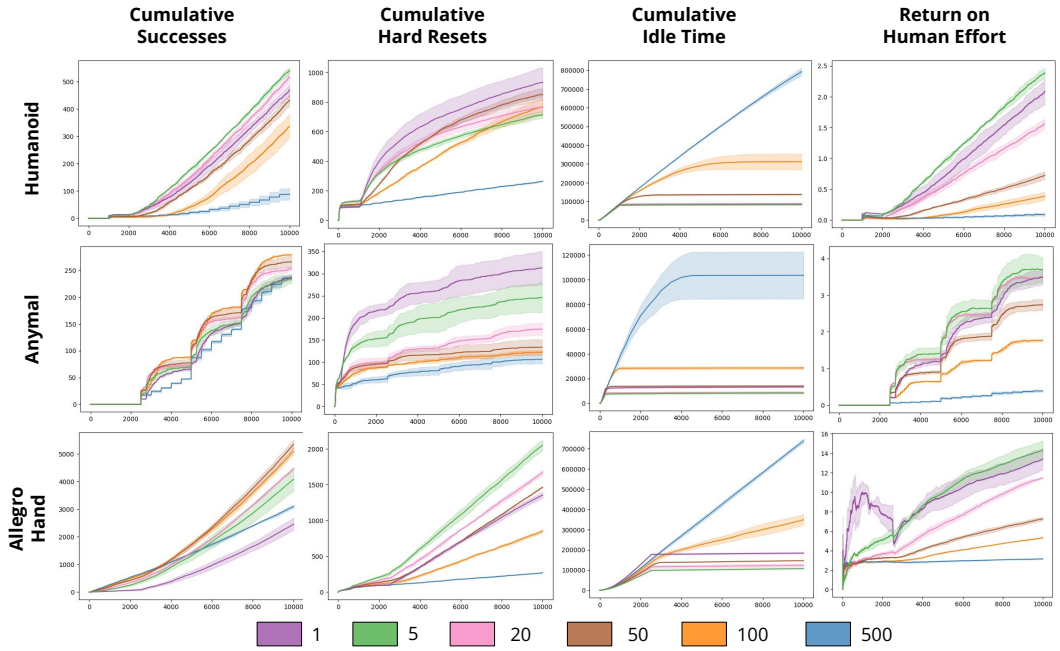

Figure 8: **Minimum Intervention Time:** Simulation results in the Isaac Gym benchmark tasks for variations in minimum intervention time t_T, where the x-axis is timesteps from 0 to $T = 10,000$.

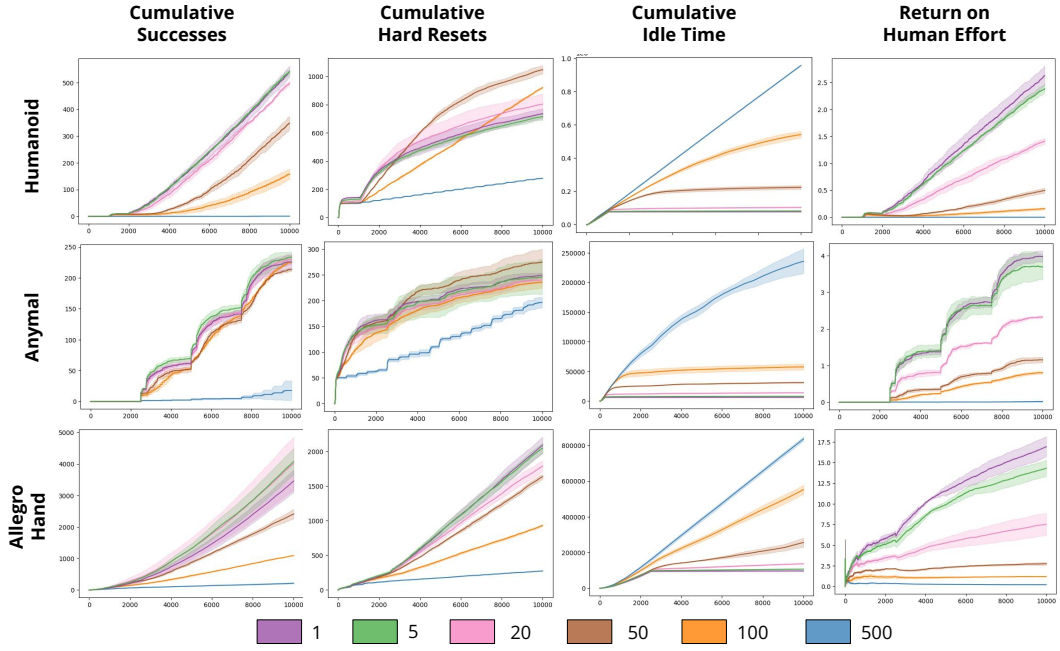

Figure 9: **Hard Reset Time:** Simulation results in the Isaac Gym benchmark tasks for variations in hard reset time t_R, where the x-axis is timesteps from 0 to $T = 10,000$.