IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Inverse Constraint Learning and Generalization by Transferable Reward Decomposition

Jaehwi Jang, Minjae Song, and Daehyung Park†

***Abstract*—We present the problem of inverse constraint learning (ICL), which recovers constraints from demonstrations to autonomously reproduce constrained skills in new scenarios. However, ICL suffers from an ill-posed nature, leading to inaccurate inference of constraints from demonstrations. To figure it out, we introduce a transferable constraint learning (TCL) algorithm that jointly infers a task-oriented reward and a task-agnostic constraint, enabling the generalization of learned skills. Our method TCL additively decomposes the overall reward into a task reward and its residual as soft constraints, maximizing policy divergence between task- and constraint-oriented policies to obtain a transferable constraint. Evaluating our method and five baselines in three simulated environments, we show TCL outperforms state-of-the-art IRL and ICL algorithms, achieving up to a 72% higher task-success rates with accurate decomposition compared to the next best approach in novel scenarios. Further, we demonstrate the robustness of TCL on two real-world robotic tasks.**

***Index Terms*—Learning from Demonstration, Constrained Motion Planning**

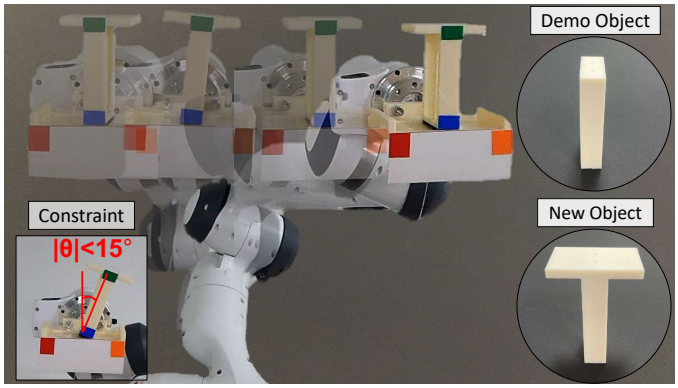

Fig. 1. Illustration of a tray-carrying task, which requires transporting an object on the serving tray while preventing the object from falling, particularly constraining the angular difference from the upright state in 15°. By learning a transferable constraint from demonstrations with a *demo object*, our method can generate demonstration-like constrained behaviors with novel dynamic properties of *new objects*.

I. Introduction

REINFORCEMENT learning (RL) has demonstrated its effectiveness in a variety of domains. However, applying these advances to real-world tasks can be challenging due to personal, task-related, and environmental restrictions in practice. Therefore, it is crucial to establish necessary constraints that impose strict limitations to ensure the validity of the task solution, independent of the optimization objective in RL.

In this context, we aim to address the problem of inverse constraint learning (ICL), a variant of inverse reinforcement learning (IRL), that recovers constraints encoded in demonstrations to autonomously define and reuse constraints. We particularly seek transferable constraints that enable agents to reproduce a family of constrained behaviors in new scenarios, maximizing new task objectives through constrained RL (CRL) [1], [2]. However, the ICL problem is ill-posed since there can be multiple possible reward-and-constraint pairs for which the reproduced behavior is optimal, leading to inaccurate inferences of underlying constraints.

To resolve the ill-posedness, researchers have often restricted the space of constraints [3], [4]. Conventional ICL methods often assume a parameterized form of constraints, such as linear (or nonlinear) equality or inequality constraints [3], [5], or a combination of constraint templates [6], [7]. Alternatively, recent approaches expand the form of constraints by adopting high expressiveness of neural networks over discrete state-action space [4], [8] or continuous state-action space [9], [10], [11]. However, these methods require predefining task rewards that cannot be explicitly given from demonstrations. Therefore, a desired method needs to recover constraints and task rewards while keeping the expressiveness of the constraint model.

We introduce a transferable-constraint learning (TCL) algorithm that jointly infers *task* reward and *residual* task-agnostic constraint pairs from demonstrations. The core idea is an expert's overall reward can be additively decomposed into a *task* reward and its *residual* part as a soft constraint (i.e., a negative reward with a Lagrange multiplier). The *residual* penalizes task-oriented behavior violating underlying constraints. To infer the *residual* reward without a priori known *task* reward, TCL's reward decomposition (RD) method 1) finds the largest *task* reward minimizing the divergence between expert and task-oriented behaviors given the space of *task* rewards, and 2) selects the complement of the *task* reward as the *residual* reward in the overall reward.

To expedite the RD process, we also introduce a com-

Manuscript received: May, 24, 2023; Revised August, 28, 2023; Accepted October, 16, 2023.

This paper was recommended for publication by Editor Hanna Kurniawati upon evaluation of the Associate Editor and Reviewers' comments. This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00311), National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (No.2021R1A4A3032834 and 2021R1C1C1004368), and the KAIST Convergence Research Institute Operation Program.

All authors are with Korea Advanced Institute of Science and Technology, Korea ({wognl0402, smj0398, daehyung}@kaist.ac.kr). †D. Park is the corresponding author.

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

putationally efficient approximated solution measuring the divergence between energy-based policies [12]. Adopting an off-policy learning scheme, TCL efficiently infers a constraint-based transferable reward function from demonstrations. We evaluate the proposed method and five baseline methods in a total of 6000 simulated tray-carrying, wiping, and wall-following environments. TCL outperforms ICL and IRL algorithms and shows higher task-success rates in novel scenarios. Its RD also shows superior reward decomposition, giving higher accuracy. We also demonstrate the generalization capability of transferable constraints through real tray-carrying scenarios as shown in Fig. 1.

Our main contributions are as follows:

- We propose a novel transferable-constraint learning method that resolves the ill-posedness issue through additive reward decomposition in ICL.
- We introduce a more accessible ICL scheme that jointly infers reward and constraint functions without requiring a predefined *task* reward.
- We statistically evaluate the generalization performance of learned transferable constraints through diverse simulation settings. Then, we demonstrate real-world performance using a real manipulator, Panda from Franka Emika, through tray-carrying and wall-following tasks.

II. Related Work

Constraints play a crucial role in enabling robust manipulation and allowing for more complex behaviors in real-world environments [13], [6], [14]. Researchers have solved constrained control problems by modeling a Markov decision process (MDP) with additional constraints, resulting in a constrained Markov decision process (CMDP) [1], in which Safe RL (SRL) [15] and constrained RL [16] are representative solution frameworks.

Most SRL/CRL approaches aim to find an optimal policy maximizing the expected return while restricting the cumulative constraint cost below a threshold based on CMDP [17], [2]. The difference between approaches depends on their optimization methods, such as constrained policy optimization (CPO) that defines a trust region [17], or Lagrangian relaxation of constrained optimization [18], [2], [19], [20]. Following the Lagrangian relaxation, TCL also constrains state-action costs within a margin while solving a relaxed optimization problem.

On the other hand, ICL is an extension of IRL where the objective is, opposite to the forward SRL/CRL, recovering the constraint function from demonstrations instead of inferring the reward function [4], [9], [8], [21]. Conventional methods often learn the parameters of geometric constraints where a linear form of constraints is given in advance [3], [6]. Chou et al. [5] introduce another model-based method of learning non-linear constraints. Recent approaches show model-free ICL that captures arbitrary Markovian feature constraints where state-action features are constrained in a set [4]. Likewise, Malik et al. [9] show a neural network-based ICL that approximates a constraint restricting a state-action set from expert demonstrations. Papadimitriou et al. extend the aforementioned method by inferring a Bayesian posterior distribution over possible constraint sets [21]. These methods require not only demonstrations but also unobservable *task* (i.e., nominal) rewards. However, our method assumes only knowledge of the space of *task* reward, not the reward itself. Constrained IRL is another IRL extension that identifies rewards from demonstrations in a constrained MDP, but requires a priori known constraints [22], [23].

The goal of learning constraints is to generalize expert skills across a wide range of tasks and environments [24]. To achieve transferable constraints, researchers use environment-agnostic representations. Park et al. [7] jointly recover feature constraints as well as simple *task* rewards via Bayesian non-parametric IRL. Willibald and Lee then extend the work by using multimodal features [25] to generalize the learned constraints across geometries. Alternatively, abstract representation in semantic constraints helps encode simple or complex constraints in a unified format (e.g., $\square\neg$*unsafe* [26]). Our method differs in that it recovers task-agnostic constraints from an overall reward by maximizing the divergence of policies between the *task* reward and the constraints.

III. Preliminaries

An MDP is a tuple, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$, where the elements represent a set of states, a set of actions, a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow Pr(\mathcal{S})$ where $Pr(\mathcal{S})$ defines a set of all probability distributions over $\mathcal{S}$, a reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and a discount factor $\gamma \in [0, 1]$, respectively. For the MDP, a policy $\pi \in \Pi$ gives an action for each state, where Π is a set of all stationary stochastic policies. The goal of RL is to find an optimal policy π^* that generates a trajectory τ with its maximum cumulative discounted rewards: $\pi^* = \arg\max_\pi \ \mathbb{E}_{\tau\sim\pi}[\sum_{t=1}^{T} \gamma^t r(s_t, a_t)]$. In this work, we use $R(\tau)$ to denote the sum of discounted rewards over the trajectory $\tau = (s_1, a_1, ..., a_{T-1}, s_T, a_T)$ with a finite horizon, T.

In contrast, the goal of IRL is to infer a reward r^* that best explains an expert policy π_E for demonstrated behaviors, given an MDP, $\mathcal{M}^- = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma)$, with no reward function. We can solve the inference problem by adopting maximum causal entropy IRL [27]:

$$r^* = \arg\max_{r\in\mathcal{R}} \big(\min_{\pi\in\Pi} -H(\pi) - \mathbb{E}_\pi[r(s,a)] \big) + \mathbb{E}_{\pi_E}[r(s,a)], \tag{1}$$

where $H(\pi) \triangleq \mathbb{E}_\pi[-\log \pi(a \mid s)]$ is a causal-entropy term [28] to regularize the policy search. The rollout traces of the expert policy π_E correspond to demonstrated trajectories $\mathcal{D}_E = (\tau_1, \tau_2, ...)$.

On the other hand, a CMDP [1] is another tuple with additional constraint, $\mathcal{M}_c = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, c, \gamma)$, where c is a constraint-cost function $c : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_0^+$, where $\mathbb{R}_0^+$ is a space of non-negative real numbers. The forward RL methods with CMDP find the optimal policy π^*, which maximizes the expected return while the expectation of cumulative constraint cost $C(\tau) = \Sigma_{t=1}^{T} c(s_t, a_t)$ over trajectory τ is less than or equal to a threshold $\xi_\tau \in \mathbb{R}$: $\pi^* = \arg\max_\pi \ \mathbb{E}_{\tau\sim\pi}[R(\tau)]$ subject to $\mathbb{E}_{\tau\sim\pi}[C(\tau)] \leq \xi_\tau$ [17], [2]. To relax the optimiza-

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

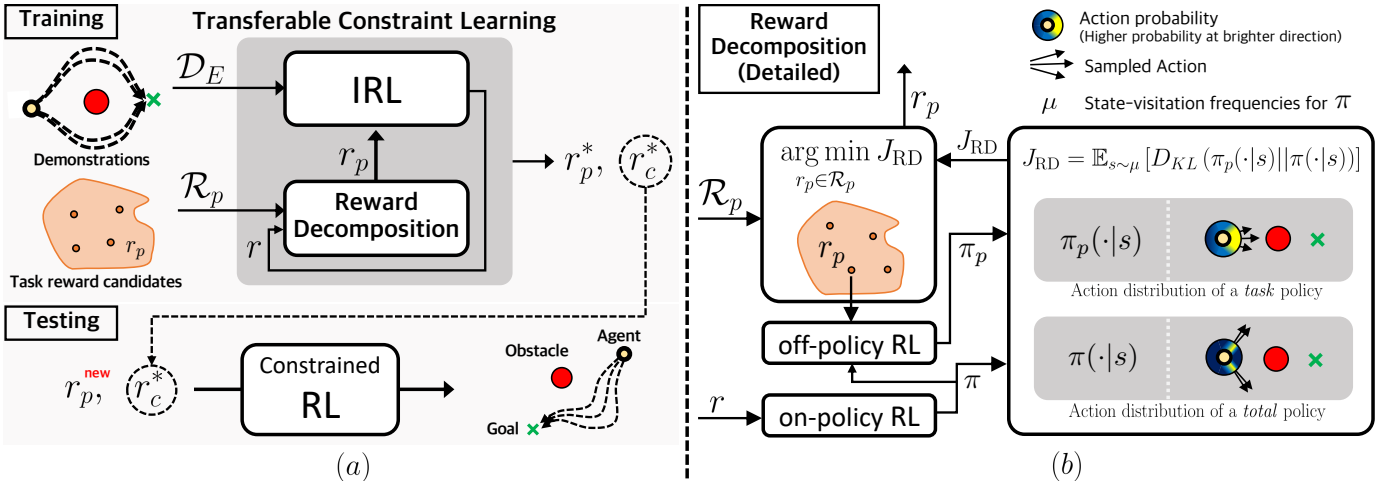

Fig. 2. The pipeline of transferable constraint learning (TCL) framework. (a) The overview of the training and testing process of TCL. In the training process, TCL takes demonstrations $\mathcal{D}_E$ and space of *task* rewards $\mathcal{R}_p$, outputs a pair of *task*-oriented and -agnostic rewards (r_p^*, r_c^*). In the testing process, constrained reinforcement learning reproduces a demonstration-like constrained behavior using the learned *task*-agnostic reward r_c^* and a new task reward r_p^{new}. (b) The details of the reward decomposition (RD) module. In detail, an inverse reinforcement learning module infers an overall reward r while the RD module decomposes r into two rewards (r_p, r_c). To find the task-agnostic reward r_c, the RD module finds r_c as a complement of r_p minimizing the divergence between π_p and π.

tion, we can reformulate an unconstrained dual problem [2] softening the constraint:

$$\min_{\lambda \geq 0} \max_{\pi} \ \mathbb{E}_{\tau \sim \pi}[R(\tau) - \lambda(C(\tau) - \xi)], \tag{2}$$

where λ is a Lagrange multiplier. As λ increases, the result converges to the solution of the constrained optimization problem. In this problem, $C(\tau)$ is to restrict the cumulative cost over the entire trajectory τ but does not guarantee each state is valid. Instead, we constrain an agent under $\mathcal{M}_c$ to be always in a valid state by regulating the immediate cost $c(s_t, a_t)$ at each time step t with $c(s_t, a_t) \leq \xi$ at $t \in \mathbb{R}$, where $\xi \in \mathbb{R}$.

Inverse constraint learning (ICL) is then an IRL extension that seeks either a constraint or a reward-and-constraint pair from $\mathcal{M}_c^- = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma)$. ICL also finds an optimal policy π^* where the expectation of constraint costs $c(s,a)$ over state-action pairs visited by the policy is less than or equal to a threshold ξ: $\pi^* = \arg\max_\pi \ \mathbb{E}_\pi[r(s,a)]$ subject to $\mathbb{E}_\pi[c(s,a)] \leq \xi$. We will describe the detail in Sec. IV.

IV. Transferable-Constraint Learning

We describe the transferable-constraint learning method shown in Fig. 2. For notational convenience, we drop the arguments of $r(s,a)$ and $c(s,a)$. We also call a reward function as 'reward' or a cost function as 'cost.'

A. Problem Formulation

The objective of TCL is to jointly infer the *task* reward $r_p \in \mathcal{R}_p$ and the constraint-related *residual* reward $r_c \in \mathcal{R}$ given globally constrained demonstrations $\mathcal{D}_E$ and the space of *task* rewards $\mathcal{R}_p \subseteq \mathcal{R}$. Here, r_p is to produce an expert-like but unconstrained behavior, restricted in the *task*-reward space $\mathcal{R}_p$. We define r_p using predefined task-relevant features. r_c is a negative constraint-cost function, $r_c = -c(s,a)$, which is task-agnostic and transferable to other tasks producing constrained behaviors. Note that we assume constraints are encoded in demonstrations $\mathcal{D}_E = \{\tau_1, \tau_2, ...\tau_N\}$, where each demonstration τ is a state-action trajectory and N is the number of trajectories.

The core idea behind TCL is the expert's total reward r can be additively decomposed into a pair of rewards, r_p and r_c, based on Q-decomposition [29]. Then, we can formulate an optimization problem that not only learns an overall reward r from demonstrations $\mathcal{D}_E$ via IRL but also decomposes r to an optimal reward pair (r_p^*, r_c^*) via RD:

$$\begin{aligned}(r_p^*, r_c^*) = \arg\max_{\substack{r_p \in \mathcal{R}_p \\ r_c \in \mathcal{R}}} \big(\min_{\pi \in \Pi} J_{\text{IRL}}(r, \pi; \mathcal{D}_E) \big) - J_{\text{RD}}(r_p, \pi; \mathcal{R}_p) \\ \text{s.t.} \quad r = r_p + r_c,\end{aligned} \tag{3}$$

where J_{IRL} and J_{RD} are objective functions (see details in Sec. IV-B). Note that π is an output policy associated with the overall reward r.

B. Objective Functions

We further describe objective functions used in Eq. (3).

1) J_{IRL} objective function: The IRL objective function J_{IRL} is to infer a total reward r and its associated policy π that can produce behaviors similar to demonstrations $\mathcal{D}_E$. Adopting the total reward r on maximum casual entropy IRL [27], we design J_{IRL} as a max-min optimization objective (i.e., $\max_r \min_\pi J_{\text{IRL}}(r, \pi; \mathcal{D}_E)$) that maximizes the reward expectation of the policy π while minimizing the difference between the expectation and the empirical expectation around the demonstrations $\mathcal{D}_E$:

$$\begin{aligned}J_{IRL}(r, \pi; \mathcal{D}_E) =& \mathbb{E}_{s,a \sim \mathcal{D}_E}[r(s,a)] \\ & - \mathbb{E}_{s,a \sim \pi}[r(s,a)] - H(\pi),\end{aligned} \tag{4}$$

where $H(\pi) = \mathbb{E}_\pi[-\log \pi(a|s)]$ is the entropy regularization term to reduce the ill-posedness of multiple optimal policies in IRL. In this work, we use parameterized reward models, such

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

as a linear combination of features or neural networks, that are flexible enough to represent not only the *task* reward but also the *residual* reward. Therefore, we can find an optimal reward r^* and policy π^* given $\mathcal{D}_E$.

2) $J_{\mathbf{RD}}$ objective function: The reward-decomposition function J_{RD} is to select r_c given the overall reward r, assuming the additive decomposition $r = r_p + r_c$. We use r_p to define a *task* policy π_p, where the execution of π_p governs all the task-relevant but unconstrained actions. By finding the most extensive r_p close to r, we can also obtain a task-agnostic *residual* reward r_c where its corresponding policy can be transferred to other tasks. To find such r_p, we design an objective function J_{RD} minimizing the action divergence between π_p and π:

$$J_{\text{RD}}(r_p, \pi; \mathcal{R}_p) = \mathbb{E}_{s\sim\mu}\big[D_{KL}(\pi_p(\cdot|s)||\pi(\cdot|s))\big], \tag{5}$$

where μ is the state-visitation frequencies for π and $D_{KL}(\cdot)$ represents a Kullback-Leibler (KL) divergence between two given policies. Note that π_p optimizes the sum of r_p, $\pi_p = f(r_p)$. In this work, r_p returns a sparse non-negative value or the highest value at the goal state to guide the agent.

In this work, we manually choose task-relevant features, which serve as the basis vectors of the task space. In cases where task-relevant features are not readily available, we can use features suitable for sparse rewards or binary indicators.

C. Generalization of Learned Constraints

Given the learned constraint-cost function $c = -r_c$, we want to reproduce the demonstration-like constrained behaviors in unforeseen environments by inferring a new optimal policy π^*_{new}. Defining a new *task* reward r_p^{new} and a cost threshold ξ of transferred constraint, $c \leq \xi$, we formulate a Lagrangian dual problem for CRL similar to Eq. (2):

$$\min_{\lambda\geq 0}\max_{\pi} \mathbb{E}_\pi[\underbrace{r_p^{\text{new}}(s,a)}_{\substack{\text{user-defined}\\ \text{task reward}}} + \underbrace{\lambda r_c(s,a) + \lambda\xi}_{\substack{\text{transferred constraint}\\ \text{-related reward}}}], \tag{6}$$

where $r_p^{\text{new}} \in \mathcal{R}_p$ is a user-defined *task* reward for the new setup. The CRL procedure finds a high-entropy policy that maximizes the new cumulative reward. In this work, we use the inequality threshold ξ that validates all samples in the expert demonstrations: $\xi = \max_{(s,a)\in\mathcal{D}_E} -r_c(s,a)$.

V. Approximation of TCL

The optimization problem of TCL, particularly its reward decomposition, requires restricting $r_p \in \mathcal{R}_p$. In detail, Eq. (5) needs to constrain $\pi_p \in \Pi_p$ where Π_p is a space of policies from $r_p \in \mathcal{R}_p$. However, the condition $\pi_p \in \Pi_p$ requires non-trivial computation to obtain action-value functions $Q_p \in \mathcal{Q}_p$, where $\mathcal{Q}_p$ is a space of action-value functions given $\mathcal{R}_p$.

To reduce the computational complexity, we simplify the reward decomposition by approximating the KL divergence and its condition. Adopting energy-based policies [12] that $\pi_p(\cdot|s) \propto \exp(Q_p(s,\cdot))$, we formulate a new minimization problem for the reward decomposition as follows:

$$\min_{Q_p\in\mathcal{Q}_p} D_{KL}\Big(\frac{\exp Q_p(s,\cdot)}{Z_p}||\frac{\exp Q(s,\cdot)}{Z}\Big), \tag{7}$$

where Z_p and Z are normalization factors, $Z_p = \Sigma_a \exp Q_p(s,a)$ and $Z = \Sigma_a \exp Q(s,a)$, respectively. In this work, we train the state-action value functions, Q and Q_p given rollouts from π, following the Bellman equations:

$$Q(s,a) \coloneqq r(s,a) + \gamma V(s') \tag{8}$$

$$Q_p(s,a) \coloneqq r_p(s,a) + \gamma V_p(s'), \tag{9}$$

where $V(s)$ and $V_p(s)$ are state-value functions given rewards r and r_p, respectively. Note that we compute $V_p(s) = \mathbb{E}_{a\sim\pi(\cdot,s)}[Q_p(s,a)]$ assuming the total policy π.

Finally, we formulate a new approximated J_{RD} function by projecting the condition $Q_p \in \mathcal{Q}_p$ in Eq. (7) onto the space of rewards to reduce computational cost and also to use the existing condition $r_p \in \mathcal{R}_p$:

$$\begin{aligned} J_{\text{RD}}(r_p, \pi; \mathcal{R}_p) = D_{KL}\Big(\frac{\exp Q_p(s,\cdot)}{Z_p}||\frac{\exp Q(s,\cdot)}{Z}\Big) \\ + \alpha\cdot||Q_p(s,a) - \big(r_p(s,a) + V_p(s')\big)||^2, \end{aligned} \tag{10}$$

where $\alpha \in \mathbb{R}^+$ is an adjustable constant, which is 1 in this work.

VI. Experimental Setup

A. Quantitative Evaluation through Simulation

We assess the learning and reproduction capabilities of TCL and baseline methods. Fig. 3 presents the simulated environments: *tray-carrying*, *wall-following*, and *wiping* created using a 2-dimensional physics simulator engine, called Box2D [30]. For each environment, we collected expert demonstrations $\mathcal{D}_E$ and trained a rollout policy using reward-constrained policy optimization (RCPO) [2] with manually designed constraints. Subsequently, we evaluated each method by training with $\mathcal{D}_E$ and predefined $\mathcal{R}_p$. We describe evaluation setups below:

1) *Tray-carrying* environment: Fig. 3 (Left) shows the environment, which requires a tray agent (black) to move an object (red) from a start position (green) to a goal position (purple) without tilting the object more than 15°. Using a bar object, we trained a rollout policy network with a size of $(64, 64)$ using multi-layer perceptrons (MLP) to define a CMDP with the following specifications:

- $\mathcal{S} = \{s|s = [\mathbf{x}_{\text{tray}}, \mathbf{x}_{\text{obj}}, x_{\text{goal}}] \in \mathbb{R}^{15}\}$ where $\mathbf{x}_{\text{tray}}$ and $\mathbf{x}_{\text{obj}}$ are the tray and object states, respectively. Each state includes horizontal position and velocity ($[x_{\text{tray}}, \dot{x}_{\text{tray}}] \in \mathbb{R}^2$), an angular velocity ($\dot{\theta}_{\text{tray}} \in \mathbb{R}$), and inclination angle encodings ($[\sin(\theta_{\text{tray}}), \cos(\theta_{\text{tray}}), \sin(2\theta_{\text{tray}}), \cos(2\theta_{\text{tray}})] \in \mathbb{R}^4$). $x_{\text{goal}} \in [-0.6, 0.6]$ is the desired tray position.
- $\mathcal{A} = \{a|a = [f_x, \tau_\theta] \in \mathbb{R}^2\}$ where $f_x \in [-0.5, 0.5]$ and $\tau_\theta \in [-0.05, 0.05]$ are the horizontal force and tilting torque, respectively.
- $r_E = \mathbb{1}^{d\leq 0.01} - 0.1\cdot d\cdot \mathbb{1}^{d>0.01}$ where $d = ||x_{\text{tray}} - x_{\text{goal}}||$
- $c_E = \mathbb{1}^{|\theta_{\text{obj}}|>15^\circ}$

where r_E and c_E are the expert reward and constraint, respectively. We collected 32 demonstrations ($T = 100$) for training.

For testing, we generated 6000 environments with three types of objects, where 3000 used demonstration-like start-and-goal positions, while the remaining 3000 had uniform-randomly sampled positions. Note that we defined $\mathcal{R}_p$ based on two features: x_{tray} and x_{goal}.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

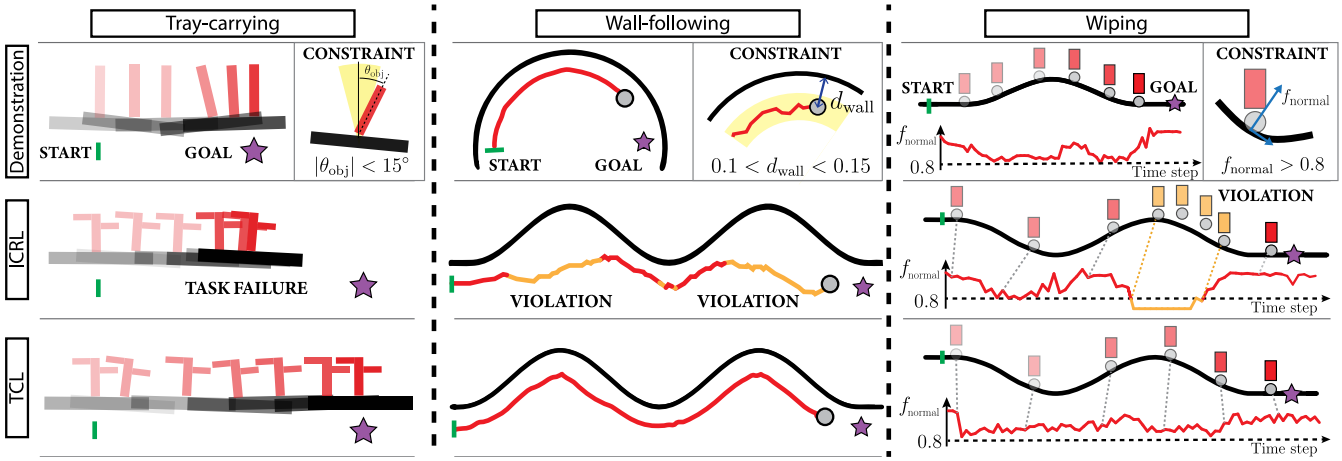

Fig. 3. Comparison of constraint-transfer performance in simulated environments. In each task (column), ICRL (the second row) and TCL (the third row) first learn constraints from demonstrations (the first row). We then make a CRL-based policy trained with the learned constraints to reproduce demonstration-like behaviors in novel environments. The green and purple shapes represent start and goal locations (or configurations), respectively. The red and orange trajectories (or boxes) represent a sequence of constrained and constraint-violated behaviors, respectively.

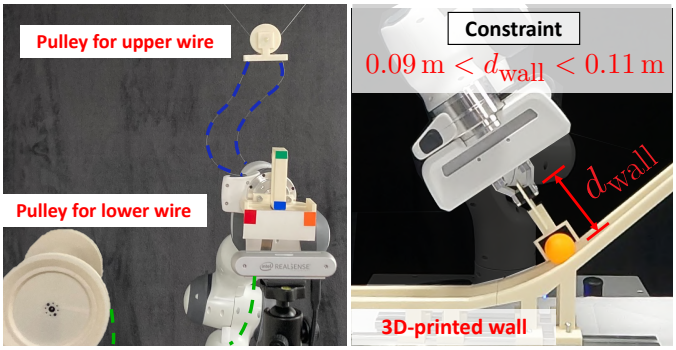

Fig. 4. Two real-world environments. (**Left**) A capture of the *tray-carrying* environment with a motorized reset mechanism. A Panda arm with an attached tray moves the bar-shaped object following a policy, where an RGB camera with an image-based pose detector provides the tray and an object poses in real-time. When the object falls over a certain angle, the reset mechanism pulls up and down wires to restore the object on the tray. (**Right**) A capture of the *wall-following* environment with a 3D-printed wall. The Panda arm moves along the curved wall without dropping the ball and hitting the wall.

2) *Wall-following* environment: Fig. 3 (Middle) shows the environment, which requires a point agent (gray) to reach a goal (green) while maintaining a certain distance from a wall (black). The specifications are as follows:

- $\mathcal{S} = \{s | s = [x_{\text{rel}}, y_{\text{rel}}, d_{\text{wall}}] \in \mathbb{R}^3\}$ where $[x_{\text{rel}}, y_{\text{rel}}]$ is the agent's displacement from a goal and d_{wall} is the agent's displacement from the wall.
- $\mathcal{A} = \{a | a = [\Delta x, \Delta y] \in \mathbb{R}^2\}$ where Δx and Δy are horizontal and vertical displacements ($\in [-1, 1]$), respectively.
- $r_E = \mathbb{1}^{d \leq 0.01} - 0.1 \cdot d \cdot \mathbb{1}^{d > 0.01}$ where $d = ||(x_{\text{rel}}, y_{\text{rel}})||_2$
- $c_E = \mathbb{1}^{d_{\text{wall}} < 0.1 \vee d_{\text{wall}} > 0.15}$

We sampled 32 demonstrations. For testing, we randomly generated 100 sinusoidal curve environments using the function: $y = A \sin^2(\pi x) + B \cos(\frac{\pi}{2} x)$ where A and B are randomly selected constants ($\in [0.1, 0.3]$). Note that we defined $\mathcal{R}_p$ with a feature d only.

3) *Wiping*: Fig. 3 (Right) shows the environment, which requires a bar agent (gray) to reach a goal (purple) while maintaining a certain contact force from the curved ground (black). The specifications are as follows:

- $\mathcal{S} = \{s | s = [x_{\text{rel}}, y_{\text{rel}}, \dot{x}_{\text{rel}}, \dot{y}_{\text{rel}}, f_{\text{normal}}] \in \mathbb{R}^5\}$ where $[x_{\text{rel}}, y_{\text{rel}}]$ and $[\dot{x}_{\text{rel}}, \dot{y}_{\text{rel}}]$ are the agent's displacement and velocity from the goal, respectively. f_{normal} is the normal contact force from the curve.
- $\mathcal{A} = \{a | a = [f_x, f_y] \in \mathbb{R}^2\}$, where f_x and f_y are horizontal and vertical forces ($\in [-1, 1]$).
- $r_E = \mathbb{1}^{d \leq 0.01} - 0.1 \cdot d \cdot \mathbb{1}^{d > 0.01}$ where $d = ||x_{\text{rel}}||$
- $c_E = \mathbb{1}^{f_{\text{normal}} < 0.8}$

We sampled 32 demonstrations. For testing, we randomly generated 100 sinusoidal curves using the function: $y = A \sin(\pi x)$ where $A \in [-0.3, 0.3]$. Note that we defined $\mathcal{R}_p$ with a feature d only.

Through evaluations, we measured the constraint-violation rate as $\sum_{i=1}^{N} \sum_{(s,a) \in \tau_i} c_E(s) / \sum_{i=1}^{N} \|\tau_i\|$, where τ_i represents a test trajectory and N is the number of trajectories. We also measure the task-success rate, classifying an episode as a success when the agent reaches the goal without any violation. For computation, we used Intel i9-10900K CPU processor and an NVIDIA RTX 2060 Super GPU.

B. Baseline Methods

For evaluation, we employed five baseline methods:

- GAIL-Constraint (GC): An imitation learning method, GAIL [31], associated with constraints [9]. GC finds r_c given a user-defined *task* reward r_p.
- AIRL-Constraint (AC): A variant of adversarial IRL method [32] associated with constraints. AC infers r that satisfies all constraints assuming a known *task* reward.
- ICL-Soft[1]: An ICL method that finds cumulative soft constraints assuming a known reward [10].
- Feature-Constraint (FC): A distribution-based constraint learner used in CBN-IRL [7]. FC uses a conjunction of feature-wise constraints that restrict current features within the observed range in demonstrations.

[1]We changed the original acronym to ICL-Soft to distinguish it from inverse constraint learning (ICL).

- ICRL: An ICL method that finds a neural-network based r_c from demonstrations and combines it with a predefined *task* reward r_p to obtain r [9].

C. Qualitative Evaluations with a Real Robot

We demonstrate our proposed method in two real-world manipulation environments using a 7 degree-of-freedom robotic arm, Panda from Franka Emika, as shown in Fig. 4.

1) *tray-carrying*: This is a real-world *tray-carrying* environment using a robotic arm-actuated serving tray, a 3D printed object, and an RGB camera. Placing a bar-shaped object on the tray, we train a rollout policy via RCPO and then collect 16 demonstrations ($T = 30$) with randomly selected start-and-goal locations. After obtaining constraints using TCL with demonstrations in the environment, the robot agent learns and reproduces a demonstration-like behavior in novel random environments with an unforeseen T-shaped object.

To show the generalization capability, we use a bar-shaped object for training and a T-shaped object for testing. To detect tray and object poses, we use a 2D-pose estimator with a 60 Hz blob-detection module from OpenCV2 [33]. In operation, our RL-based control system applies horizontal force f_x and tilt torque τ_θ to the tray using a 1 kHz low-level torque controller and a 10 Hz high-level operational-space force/torque controller with the Jacobian transpose method. To expedite the learning process, we use a motorized reset mechanism that automatically restores the target object within 17 s in reset conditions such as failure or completion. We particularly reset when the object is tilted more than 90° or placed outside the workspace.

2) *wall-following*: This is a real-world *wall-following* environment in which a robot-attached tool moves along the curved wall without dropping the ball and hitting the wall. For training, we collect expert demonstrations maintaining the end-effector displacement from a wall, $c_E = \mathbb{1}^{d_{\text{wall}}<0.11\,\text{m} \vee d_{\text{wall}}>0.09\,\text{m}}$, via a Box2D simulator. Then, inferring constraints from demonstrations, we enable TCL to generate a CRL-based policy that reproduces the constrained behavior when given a new real-world wall. To expedite the learning process, we use a simulated *wall-following* environment. We then transfer the learned policy to the Panda arm running in the new environment.

For these evaluations, we used an Intel i9 CPU and a NVIDIA RTX 3070 GPU.

VII. Evaluation Results

We analyze the constraint learning-and-transfer performance of our proposed method, TCL, and baselines in simulated environments. The qualitative study in Fig. 3 demonstrates that TCL outperforms a representative baseline, ICRL, in transferring constraints to novel environments. TCL successfully learns constraints from demonstrations and reproduces demonstration-like constrained behaviors in novel environments, as shown in the first and third rows of Fig. 3. In the *wall-following* task that requires accurate decomposition of *task* and constraint-relevant *residual* rewards, TCL enables the agent to reach the goal without violations, tightly following

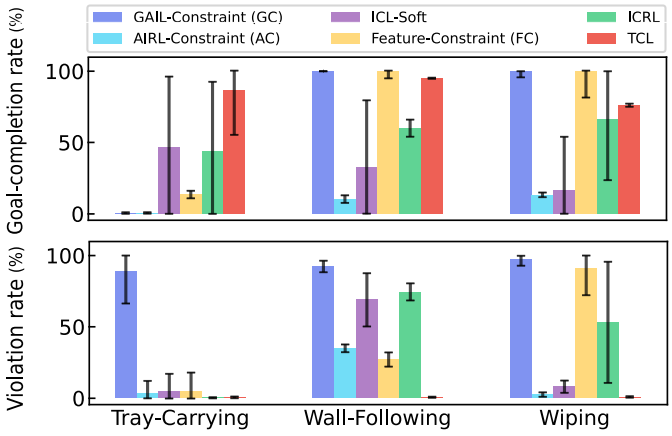

Fig. 5. Comparison of goal-completion and constraint-violation rates between TCL and baselines in simulated environments. Note that we determine the goal-completion rate regardless of any constraint violations.

the wall against the goal-directed behavior. On the other hand, ICRL either fails to reach the goal (i.e., *tray-carrying*) or violates constraints (i.e., *wall-following* and *wiping*) more frequently than TCL, since ICRL cannot distinguish between task rewards and task-agnostic constraints that minimally affect a new task objective in a novel environment.

Our quantitative study in Fig. 5 further demonstrates the superior generalization performance of TCL across diverse simulations. TCL consistently exhibits the lowest constraint-violation rate in all the novel test environments, while achieving high goal-completion rates with different task configurations and objects. In contrast, the other baselines show either low goal-completion rates or high violation rates. Note that the goal-completion rates of GC and FC in *wall-following* and *wiping* environments may appear high, but most of the results are not valid due to the high rate of constraint violations. Additionally, TCL showcases the capabilities of ICL by learning not only angle and distance constraints but also force constraints defined in feature space.

We conducted a more detailed statistical evaluation to measure task-success rates in randomly generated *tray-carrying* environments (see Fig. 6) The results demonstrate that TCL consistently outperforms the other baselines, achieving the highest success rate (up to 89.1%) with a maximum 72% difference from the next best approach, ICRL, when confronted two novel objects. This indicates that TCL discovers task-agnostic constraints that can be transferred to novel environments regardless of initial configurations and object dynamics. In contrast, ICRL, ICL-Soft, and FC exhibit similar performance when given demonstration-like start and goal configurations. However, when faced with new objects, the success rates of ICRL, ICL-Soft, and FC drastically decrease, suggesting that their extracted constraints still contain task-specific components that may conflict with reproducing properly constrained behaviors in a new setup. Furthermore, GC and AC consistently show the lowest performance since these methods excessively constrain state-action pairs [9]. AC shows similar or higher success rates than GC since AIRL can directly infer a reward. For this study, we determine a task to be successful when both the tray and the object are on the

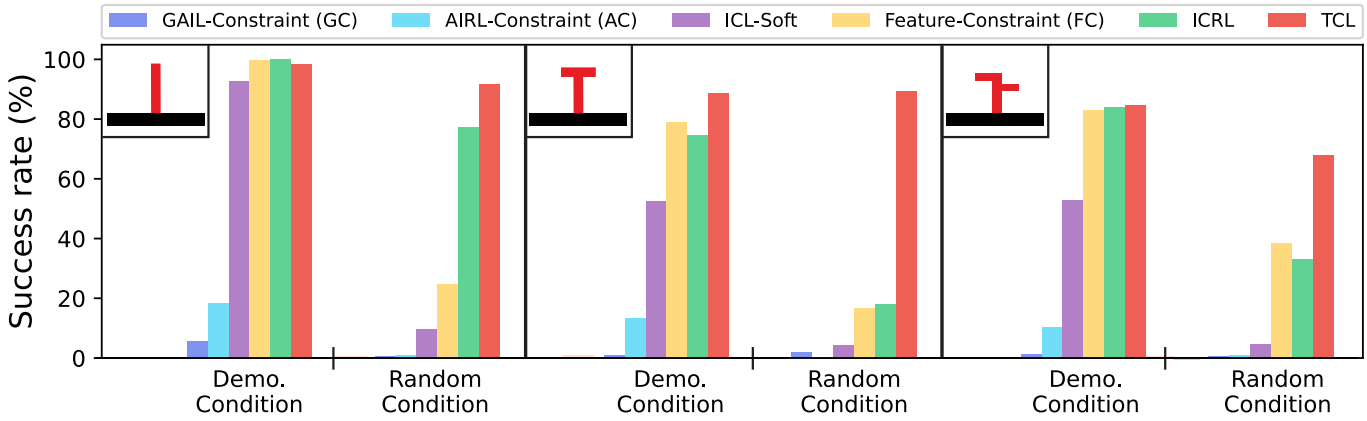

Fig. 6. Comparison of task-success rates in simulated *tray-carrying* environments, where we collected 32 demonstrations with a bar-shaped object (**Left**) and reproduced demonstration-like constrained behaviors training our proposed TCL and five baseline methods. The *demonstration condition* represents the use of demonstration-like start-and goal-locations with small variations. The *random condition* represents the use of uniform-randomly sampled start-and-goal locations.

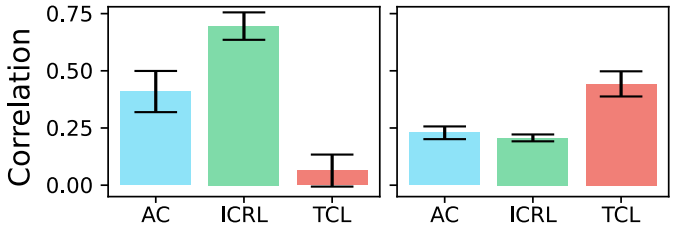

Fig. 7. Comparison of reward-decomposition performance over 100 2D reaching environments. (**Left**) The correlations between the ground-truth task-oriented reward r_p and the recovered constraint-relevant reward r_c, and (**Right**) The correlations between the ground truth and the inferred constraint-relevant rewards. Note that we use a negative cost as a constraint-relevant reward. The video description can be found here: https://youtu.be/SD9HV1z5owk

goal while the object remains upright.

We also investigated the decomposition performance of our method and the baselines by measuring the sample correlation coefficient between obtained rewards in 2D reaching environments, as shown in Fig. 2. To compute the correlation, we first inferred reward and constraint functions per algorithm from 100 expert demonstrations. We then sample 1000 constraint-relevant rewards at each of the 100 randomly generated environments. Given two sets of reward samples, we calculated the sample correlation coefficient. Fig. 7 (**Left**) shows TCL resulted in the lowest correlation between ground-truth task-oriented and inferred task-agnostic rewards, where the near-zero correlation bar indicates two decomposed rewards are independent. In contrast, the other baselines show exceedingly high correlations; their constraint-based rewards still contain task-dependent and non-transferable components. We also measured the correlation between the ground-truth constraint-based reward and the inferred constraint-based reward. Fig. 7 (**Right**) shows TCL precisely inferred constraints from demonstrations with a higher correlation than the others.

Finally, we demonstrated the generalization performance of TCL in real-world *tray-carrying* and *wall-following* environments. Fig. 8 (Top) shows the Panda arm with TCL successfully transported both known bar-shaped and unforeseen T-shape objects without large inclinations at every time step. Fig. 8 (Bottom) shows TCL successfully delivered the ball moving it along the curved walls while maintaining the precise displacement constraints. These demonstrations show TCL can robustly learn and generalize task-agnostic constraints in the real world. In this experiment, to obtain expert demonstrations satisfying the predefined constraint, we optimized an RCPO-based policy for 6 hours with a 30000 step length and trained each algorithm with the real robot for 2 hours. With the predicted constraint, our method could robustly and successfully reproduce the demonstration-like safe behavior.

VIII. Conclusion

We introduced a transferable constraint learning (TCL) algorithm that jointly infers a task-oriented reward and a task-agnostic constraint reward, as a soft constraint, from demonstrations. To accurately infer the task-agnostic reward resolving the ill-posedness of ICL, our method decomposes an overall reward to a *task* and its *residual* rewards following the idea of additive reward decomposition. The statistical evaluation shows our method outperforms baseline methods in terms of the task-success rate, the constraint-violation rate, the accuracy of decomposition, and the transferability to novel environments. We also demonstrate the proposed TCL can be deployed to real-world manipulation scenarios.

References

[1] E. Altman, *Constrained Markov decision processes: stochastic modeling*. Boca Raton, FL, USA: CRC, 1999, vol. 7.

[2] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," in *Proc. Int'l. Conf. on Learning Representations*, 2019.

[3] L. Armesto, J. Bosga, V. Ivan, and S. Vijayakumar, "Efficient learning of constraints and generic null space policies," in *Proc. Int'l Conf. on Robotics and Automation*, 2017, pp. 1520–1526.

[4] D. R. Scobee and S. S. Sastry, "Maximum likelihood constraint inference for inverse reinforcement learning," in *Proc. Int'l. Conf. on Learning Representations*, 2020.

[5] G. Chou, D. Berenson, and N. Ozay, "Learning constraints from demonstrations with grid and parametric representations," *Int'l J. of Robotics Research*, vol. 40, pp. 1255 – 1283, 2018.

[6] C. Pérez-D'Arpino and J. A. Shah, "C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *Proc. Int'l Conf. on Robotics and Automation*, 2017, pp. 4058–4065.

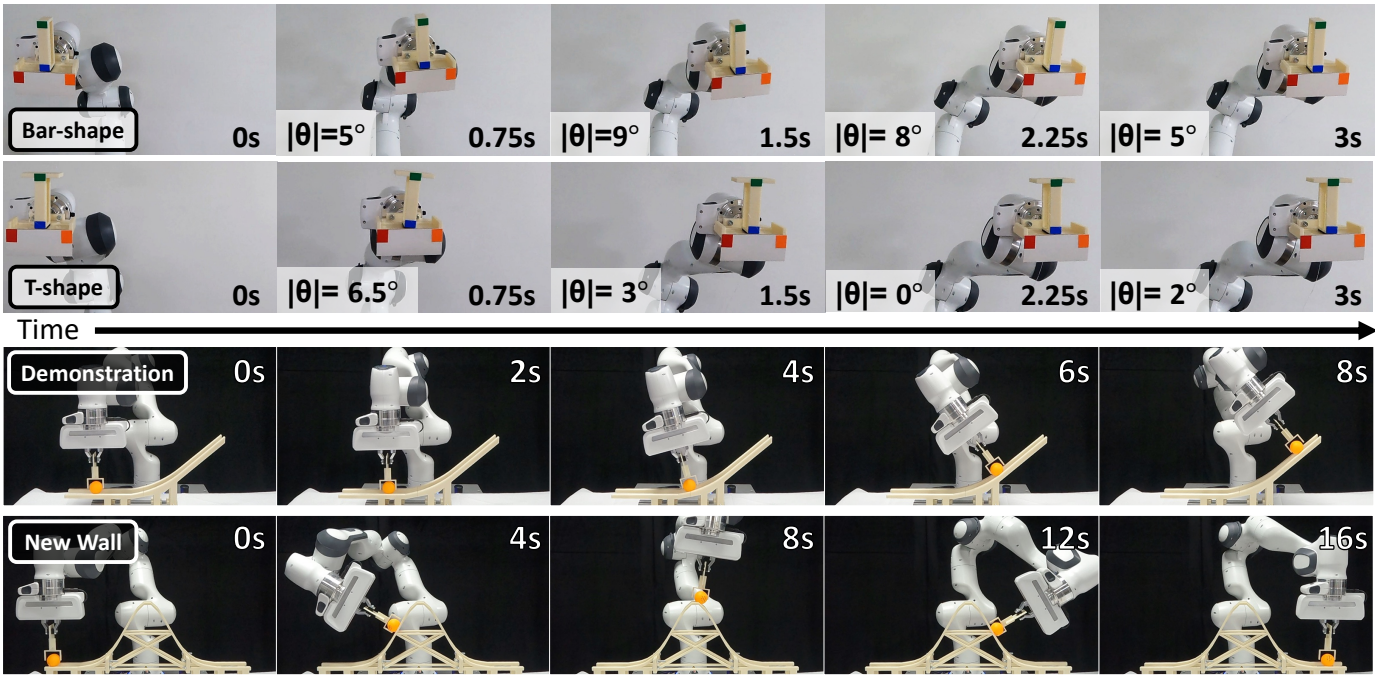

Fig. 8. Demonstration of TCL's constraint-transfer capability in the *tray-carrying* and *wall-following* environments. (**Top**) In *tray-carrying* environment, TCL learns a constraint from demonstrations in the bar-shaped environment. Transferring the learned constraint, TCL then enables the robot to reproduce a demonstration-like safe carrying behavior in the environments with an unforeseen T-shaped object. (**Bottom**) In *wall-following* environment, TCL learns the transferable constraint of maintaining the distance from the wall and successfully transfers its precise collision-free tracking behavior to the new wall environment.

[7] D. Park, M. Noseworthy, R. Paul, S. Roy, and N. Roy, "Inferring task goals and constraints using bayesian nonparametric inverse reinforcement learning," in *Proc. Conf. on robot learning*, vol. 100, 2020, pp. 1005–1014.

[8] D. L. McPherson, K. C. Stocking, and S. S. Sastry, "Maximum likelihood constraint inference from stochastic demonstrations," in *Proc. Conf. on Control Technology and Applications*, 2021, pp. 1208–1213.

[9] S. Malik, U. Anwar, A. Aghasi, and A. Ahmed, "Inverse constrained reinforcement learning," in *Proc. Int'l Conf. on Machine Learning*, vol. 139, 2021, pp. 7390–7399.

[10] A. Gaurav, K. Rezaee, G. Liu, and P. Poupart, "Learning soft constraints from constrained expert demonstrations," in *Proc. Int'l. Conf. on Learning Representations*, 2023.

[11] G. Liu, Y. Luo, A. Gaurav, K. Rezaee, and P. Poupart, "Benchmarking constraint inference in inverse reinforcement learning," in *Proc. Int'l. Conf. on Learning Representations*, 2023.

[12] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proc. Int'l Conf. on Machine Learning*, vol. 70, 2017, pp. 1352–1361.

[13] N. Somani, M. Rickert, A. Gaschler, C. Cai, A. Perzylo, and A. Knoll, "Task level robot programming using prioritized non-linear inequality constraints," in *Proc. RSJ Int'l Conf. on Intelligent Robots and Systems*, 2016, pp. 430–437.

[14] A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip, "Neural manipulation planning on constraint manifolds," *IEEE Robotics and Automation Letters*, vol. 5, pp. 6089–6096, 2020.

[15] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, pp. 1437–1480, 2015.

[16] Z. Qin, Y. Chen, and C. Fan, "Density constrained reinforcement learning," in *Proc. Int'l Conf. on Machine Learning*, vol. 139, 2021, pp. 8682–8692.

[17] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. Int'l Conf. on Machine Learning*, vol. 70, 2017, pp. 22–31.

[18] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *Journal of Machine Learning Research*, vol. 18, pp. 1–51, 2018.

[19] S. Huang, A. Abdolmaleki, G. Vezzani, P. Brakel, D. J. Mankowitz, M. Neunert, S. Bohez, Y. Tassa, N. Heess, M. Riedmiller, and R. Hadsell, "A constrained multi-objective reinforcement learning framework," in *Proc. Conf. on robot learning*, vol. 164, 2022, pp. 883–893.

[20] J. Lee, C. Paduraru, D. J. Mankowitz, N. Heess, D. Precup, K.-E. Kim, and A. Guez, "Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation," in *Proc. Int'l. Conf. on Learning Representations*, 2022.

[21] D. Papadimitriou, U. Anwar, and D. S. Brown, "Bayesian inverse constrained reinforcement learning," in *Workshop on Safe and Robust Control of Uncertain Systems (NeurIPS)*, 2021.

[22] J. Fischer, C. Eyberg, M. Werling, and M. Lauer, "Sampling-based inverse reinforcement learning algorithms with safety constraints," in *Proc. RSJ Int'l Conf. on Intelligent Robots and Systems*, 2021, pp. 791–798.

[23] A. Schlaginhaufen and M. Kamgarpour, "Identifiability and generalizability in constrained inverse reinforcement learning," in *Proc. Int'l Conf. on Machine Learning*, vol. 202, 23–29 Jul 2023, pp. 30 224–30 251.

[24] G. Subramani, M. Zinn, and M. Gleicher, "Inferring geometric constraints in human demonstrations," in *Proc. Conf. on robot learning*, vol. 87, 2018, pp. 223–236.

[25] C. Willibald and D. Lee, "Multi-level task learning based on intention and constraint inference for autonomous robotic manipulation," in *Proc. RSJ Int'l Conf. on Intelligent Robots and Systems*, 2022, pp. 7688–7695.

[26] M. Hasanbeig, D. Kroening, and A. Abate, "LCRL: Certified policy synthesis via logically-constrained reinforcement learning," in *Proc. Int'l Conf. on Quantitative Evaluation of Systems*, 2022, pp. 217–231.

[27] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling interaction via the principle of maximum causal entropy," in *Proc. Int'l Conf. on Machine Learning*, 2010, p. 1255–1262.

[28] M. Bloem and N. Bambos, "Infinite time horizon maximum causal entropy inverse reinforcement learning," in *Proc. Conf. on decision and control*, 2014, pp. 4911–4916.

[29] S. J. Russell and A. Zimdars, "Q-decomposition for reinforcement learning agents," in *Proc. Int'l Conf. on Machine Learning*, 2003, p. 656–663.

[30] E. Catto. Box2d: A 2d physics engine for games. [Online]. Available: http://www.box2d.org

[31] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Conf. on Neural Information Processing Systems*, vol. 29, 2016.

[32] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *Proc. Int'l. Conf. on Learning Representations*, 2018.

[33] G. Bradski. Open computer vision library (OpenCV). [Online]. Available: http://opencv.org/