WATCH, INFER, COORDINATE: INFERRING ROBOT PARTNER CONSTRAINTS FOR ZERO-SHOT COORDINATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robots operating in the physical world will increasingly need to coordinate with other robots, particularly in manipulation tasks where an object may be too large or heavy for a single robot to carry alone. Physical limitations caused by hardware degradation or actuator faults can restrict the actions a robot can reliably execute, yet these limitations may be unknown to its partner. We study whether a helper can infer a robot partner's physical constraints from observing it coordinate with another robot, then use the inferred capability to coordinate with the same partner on a new task. This is difficult because a demonstration shows what the constrained robot did, but not what it could have done. In physically coupled tasks, the other robot may also compensate for its limitations, making those limitations difficult to identify from the constrained robot's behavior alone. Our key insight is that these constraints shape the joint behavior of the team, making the actions of both robots informative about the constrained partner's capability. We introduce Watch, Infer, Coordinate, a benchmark spanning three physically coupled manipulation settings, together with an inference approach that scores candidate constraints using observed joint behavior. Across all three settings, our method substantially improves constraint inference and zero-shot coordination, approaching an oracle with access to the true constraints.

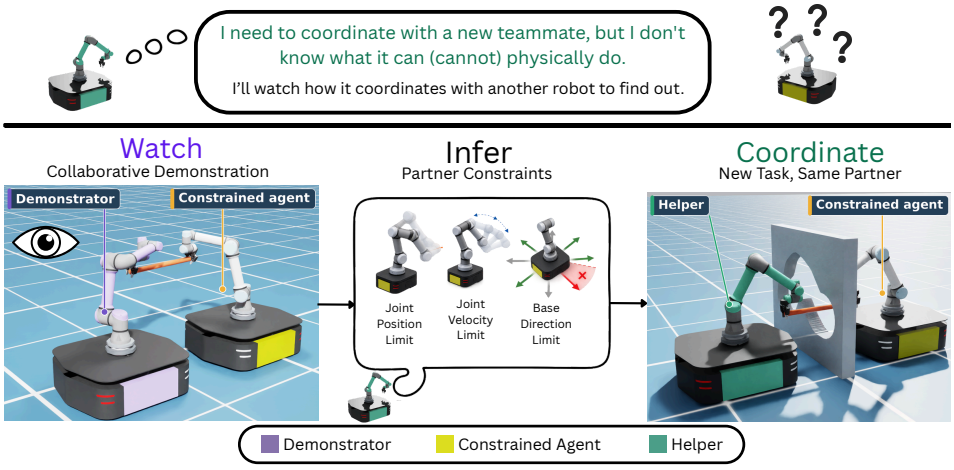

Figure 1: **Overview of our problem setting.** A helper must coordinate with a constrained partner whose physical capabilities are initially unknown. Rather than interacting with the partner directly, the helper first watches it coordinate with another robot and uses this demonstration to infer constraints on its low-level actions, such as joint position, joint velocity, or base-direction limits. The helper then replaces the demonstrator and must coordinate with the same constrained partner on a new task. Although the required coordination changes between tasks, the partner's physical constraints remain the same, allowing the helper to transfer what it learned about the partner's capabilities to the new task.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

As robots move from controlled laboratories into the physical world, they will increasingly need to not only adapt to their environment, but also to learn to coordinate with one another. Some manipulation tasks naturally require multiple robots, for example when transporting objects that are too large or heavy for a single robot to handle alone (Farivarnejad & Berman, 2022). In such physically coupled tasks, the robots' actions jointly determine the motion of the shared object, requiring closely coordinated movements, often in decentralized settings in which robots cannot access one another's plans (Pandit et al., 2025b; Doshi et al., 2026; Yang & Mavrogiannis, 2025; Shi et al., 2026). In practice, a robot's physical capabilities may also change over time due to hardware degradation or actuator faults. Such failures can reduce actuator effectiveness or restrict the motion of individual joints, changing the set of actions the robot can reliably execute (Lewis & Maciejewski, 1997; Tinós et al., 2007). In this work, we consider a helper that must coordinate with a robot, which we call the *constrained agent*, subject to unknown constraints on its low-level physical actions, such as position or velocity limits on individual joints.

Existing methods do not directly address how a helper could learn these constraints from observing prior coordination. Partner-aware assistance approaches have inferred properties such as goals or task-level capabilities (Puig et al., 2021; Cao et al., 2024; Du et al., 2024), while methods for physically coupled coordination typically adapt to a partner's behavior without explicitly inferring persistent constraints on the actions that partner can execute (Yang & Mavrogiannis, 2025; Pandit et al., 2025b; Doshi et al., 2026). We therefore ask whether a helper can infer a partner's physical constraints by observing it coordinate with another robot, and then use that knowledge to coordinate with the same partner on a new task. As illustrated in Figure 1, the helper first observes a demonstrator coordinate with the constrained agent, then takes the demonstrator's place on a different task. Because the helper has not observed how the constrained agent coordinates on this new task before deployment, we refer to this as *zero-shot coordination*.

Inferring these constraints from watching prior coordination is difficult because a demonstration shows what the constrained agent did, but not what it could have done. If the constrained agent does not take a particular action, that action may be physically infeasible or simply unnecessary for the task. Physical coupling makes this ambiguity more subtle, as the demonstrator may compensate for the constrained agent's limitations. This can make the constraint difficult to identify from the task outcome or the constrained agent's actions alone. Our key insight is that a partner's physical constraints shape the joint behavior of the team, so the actions of both agents can be used to reason about them. Because these low-level constraints belong to the agent rather than a particular task, they can also provide a capability model that remains useful when the task changes. We study these ideas in simulation settings with varying levels of complexity, allowing us to systematically vary physical constraints across different coordination problems.

We introduce **Watch, Infer, Coordinate**, a framework for studying coordination under unknown partner constraints. As shown in Figure 2, Watch, Infer, Coordinate includes a benchmark spanning three joint-carrying settings of increasing physical complexity, with obstacle-free demonstrations followed by a new task at test time in which obstacles block the direct path to the goal and require different coordinated motions. The benchmark includes 2D rod carrying, fixed-base dual-UR5 carrying, and mobile dual-UR5 carrying, spanning a low-dimensional abstraction to increasingly complex robotic manipulation. We also propose an inference approach that uses the observed actions of both agents to determine which candidate constraints are most consistent with their coordination. We use the resulting capability estimate to support zero-shot coordination with the constrained agent on the new task.

To summarize, our main contributions are fourfold. (1) We formulate the problem of inferring a robot partner's persistent low-level physical constraints from prior multi-agent coordination and using them for zero-shot coordination on a new task. (2) We introduce the Watch, Infer, Coordinate benchmark, spanning three joint-carrying settings of increasing physical complexity. (3) We propose an inference approach that uses the actions of both agents to infer the constrained agent's physical constraints from observed coordination. (4) We show that our approach improves both constraint inference and downstream task success across all three settings, approaching the performance of an oracle that is given the true constraints.

2 Related Works

Table 1: Comparison with related work. Prior-Trajectory Inference indicates that episode-specific information is inferred from trajectories observed before the downstream task. Multi-Agent Trajectories indicates that the trajectories used for inference contain behavior from multiple interacting agents. Low-Level Partner Constraints indicates that the inferred quantity specifies low-level constraints such as limits on the partner's joint positions, joint velocities, or allowable motion directions, rather than task-level capabilities such as reachable height or liftable weight. Continuous Control indicates continuous-valued physical control rather than symbolic or macro actions. Physically Coupled Manipulation indicates that multiple agents jointly manipulate the same object such that their actions mechanically combine to determine its motion. Transfer to New Task indicates that the inferred information is subsequently used on a distinct task or goal requiring different behavior.

Work	Prior-Trajectory Inference	Multi-Agent Trajectories	Low-Level Partner Constraints	Continuous Control	Physically Coupled Manipulation	Transfer to New Task
CHORUS (Doshi et al., 2026)	✗	✗	✗	✓	✓	✗
Watch-and-Help (Puig et al., 2021)	✓	✗	✗	✗	✗	✗
TICC-POMDP (Lee et al., 2020)	✗	✗	✗	✗	✗	✓
Smart Help (Cao et al., 2024)	✗	✗	✗	✗	✗	✗
CHAIC (Du et al., 2024)	✗	✗	✗	✗	✓	✗
CE-CM-Div (Tisnikar et al., 2026)	✓	✓	✗	✗	✗	✓
CAMO (Tisnikar et al., 2024)	✓	✗	✓	✓	✗	✓
GRACE (Liu et al., 2025)	✗	✗	✓	✓	✗	✓
ICRL Benchmark (Liu et al., 2022)	✓	✗	✗	✓	✗	✗
MACL (Zhang et al., 2026)	✓	✓	✗	✓	✗	✓
Ours	✓	✓	✓	✓	✓	✓

Physically Coupled Robot Collaboration. Prior work has studied physically coupled robot collaboration in both human-robot and robot-robot teams. Human-robot methods adapt to the partner's current behavior, for example by inferring intent from joint motion (Yang & Mavrogiannis, 2025), using language to revise motion plans (Shi et al., 2026), or learning cooperative policies from human-human demonstrations (Ng et al., 2023). Multi-robot methods learn decentralized controllers using shared (Pandit et al., 2025a;b; Doshi et al., 2026), or agent-specific policies (He et al., 2025; Dong et al., 2025). These methods span model-based planning (Stouraitis et al., 2020; Kennel-Maushart & Coros, 2024; Jaafar et al., 2024; Yang & Mavrogiannis, 2025), reinforcement learning (Nachum et al., 2020; Zhang et al., 2020; Shibata et al., 2023; Aljalbout et al., 2023; Pandit et al., 2025a;b), diffusion policies (Dong et al., 2025; Ng et al., 2023; He et al., 2025), and VLM/VLA-based policies (Song et al., 2025; Doshi et al., 2026). However, they do not explicitly infer a persistent physical constraint on the actions that a particular partner can execute. Ability-aware controllers can account for differences in robots' actuation limits, but assume those limits are known (Yan et al., 2021). Our work instead infers the constrained agent's feasible action set from prior physically coupled demonstrations and uses the inferred model to coordinate with the same agent in a new environment.

Partner-Aware Assistance. The watch-and-help challenge proposed the problem of observing an agent, inferring its goal, and then helping it achieve that goal (Puig et al., 2021). Subsequent work has developed methods for inferring and updating the agent's goal online (Puig et al., 2023; Zhi-Xuan et al., 2020). More broadly, partner-aware assistance methods has modeled other properties that shape an agent's behavior, including its beliefs about environment dynamics (Reddy et al., 2018), and its latent strategy across repeated interactions (Xie et al., 2021). Capability-aware methods instead reason about what the partner can do, but often represent capability through task-level properties or high-level actions. TICC-POMDP models capability through action success probabilities (Lee et al., 2020). Smart Help estimates semantic capabilities such as reachable height and liftable weight (Cao et al., 2024), while CHAIC uses VLM and LLM based helpers to infer a partner's goals and physical limitations online and reasons about these limitations through task-level actions and assistance plans (Du et al., 2024). CE-CM-Div infers domain-specific capabilities defined over symbolic and macro actions (Tisnikar et al., 2026). Our work instead infers a partner's low-level physical action constraints, such as joint and velocity limits, from demonstrations of the partner collaborating with another agent. A new helper then uses this capability model to plan joint motion with the partner, zero-shot, on a new task.

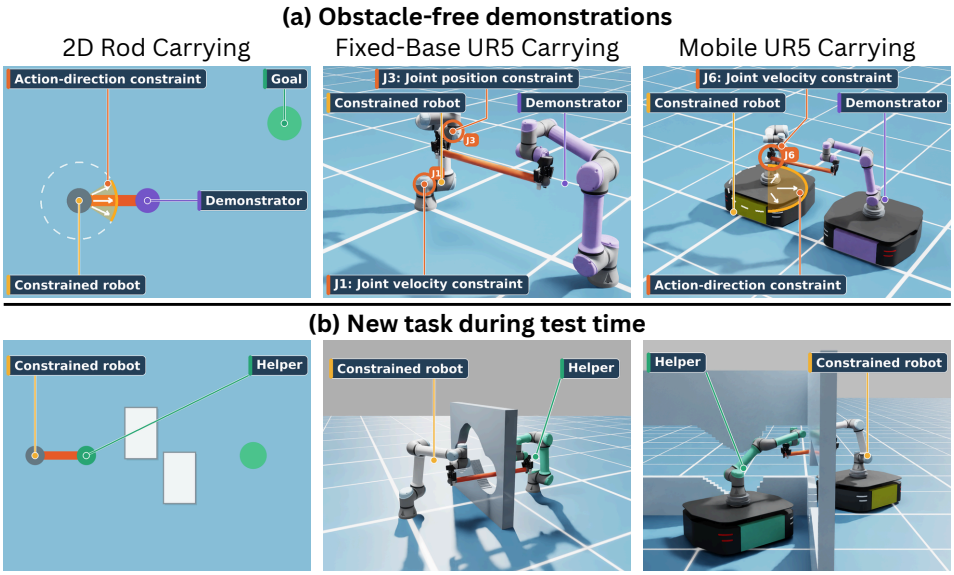

Figure 2: **Overview of our physically coupled partner-adaptation benchmark.** (A) During the observation stage, the helper watches a demonstrator coordinate with the constrained agent on an obstacle-free task. The labels show physical constraints on the action space of the constrained agent, which are dependent on the setting. (B) At test time, the helper replaces the demonstrator and coordinates zero-shot with the same constrained agent on a new task with obstacles. The benchmark includes three settings of increasing physical complexity: 2D rod carrying, fixed-base dual-UR5 carrying, and mobile dual-UR5 carrying. Across all settings, the constrained agent's physical constraints remain fixed while the task and helper change.

Constraint Inference from Demonstrations. The goal of constraint inference from demonstrations is to recover unknown constraints on an agent or its environment from observed behavior (Chou et al., 2018; Scobee & Sastry, 2019; Malik et al., 2021). Optimization-based methods search for lower-cost alternatives to the demonstrated trajectory, relying on the idea that if the demonstration is optimal, these alternatives must violate an unknown constraint (Chou et al., 2018; 2020). Other methods infer constraints by finding those that make the observed demonstrations most likely (Scobee & Sastry, 2019; Malik et al., 2021). However, much of this work focuses on environmental constraints, such as hidden obstacles and forbidden regions, that are directly reflected in a single agent's trajectory. Prior work on agent centric constraint inference either observes the agent acting on its own (Tisnikar et al., 2024; Liu et al., 2025) or models capability as domain-specific restrictions on which tools or task-level actions the agent can use (Tisnikar et al., 2026). Recent work has also inferred constraints from multi-agent demonstrations, but focuses on interaction constraints between agents rather than the physical limits of a particular partner (Zhang et al., 2026). Our work instead infers a partner's low-level physical limits from cooperative, physically coupled team demonstrations, where the helper may compensate for the constrained partner, and the resulting team motion can hide the partner's limitations.

3 Watch, Infer, Coordinate: Problem Setup and Benchmark

3.1 Problem Setup

We consider a helper that must coordinate with a robot whose low-level physical capabilities are initially unknown. We refer to this robot as the *constrained agent*. Its physical actions are restricted by constraints, such as joint position limits, velocity limits, or allowable motion directions. During the observation stage, the helper watches the constrained agent coordinate with another robot, which we call the *demonstrator*. The helper knows the goal of each demonstration and observes the states and actions of both agents, but not the constrained agent's physical constraints. At test time, the helper replaces the demonstrator and must coordinate with the same constrained agent on a new

task without first observing how it behaves on that task. The constraints remain fixed while the task and required coordinated motions change, requiring the helper to infer a transferable model of the constrained agent's physical capability from prior coordination.

3.2 Why Coordinated Behavior is Informative

A demonstration shows what the constrained agent did, but not what it could have done. If it does not take a particular action, that action may be physically infeasible or simply unnecessary for the task. Inferring its constraints therefore requires reasoning about how the constraints affect the observed behavior. In physically coupled tasks, the demonstrator can help resolve this ambiguity. A constraint on the constrained agent may require the demonstrator to compensate so that the object they are jointly carrying still follows the intended trajectory. This compensation can make the constraint difficult to identify from the constrained agent's actions alone, while the demonstrator's actions provide information about the underlying limitation. Our key insight is that the constrained agent's physical capability shapes the joint behavior of both robots, so both agents' actions provide information for capability inference.

3.3 Watch, Infer, Coordinate Benchmark

To evaluate this problem, we construct a benchmark with three joint-carrying settings of increasing physical complexity: 2D rod carrying, fixed-base dual-UR5 carrying, and mobile dual-UR5 carrying. In each setting, the two agents must jointly transport an object from an initial position to a goal. As shown in Figure 2, demonstrations are collected on obstacle-free tasks requiring different coordinated motions. At test time, obstacles block the direct path to the goal, requiring motions that may not have appeared in the demonstrations. For each setting, we define a fixed set of demonstration tasks and use the same tasks regardless of the constrained agent's physical constraints. The tasks require a range of coordinated motions so that different constraints can affect the observed behavior.

Because physical capability cannot be inferred from arbitrary behavior, we assume a model of how the agents choose their actions. We model the demonstrator and constrained agent as boundedly rational agents coordinating toward a shared task objective (McKelvey & Palfrey, 1995; Lindig-León et al., 2021). Formally, let D denote the demonstrator, P the constrained agent, and c its capability. At state s, the agents choose actions a^D and a^P. Each setting defines an action-pair score $Q_c(s, a^D, a^P)$, computed using the known transition model to score how good the two agents' actions are for completing the carrying task from state s.

We model the agents using the regularized shared objective

$$\mathcal{J}_c(\pi_D, \pi_P; s) = \mathbb{E}_{\substack{a^D \sim \pi_D \\ a^P \sim \pi_P}} \left[Q_c(s, a^D, a^P) \right] - \tau_D D_{\mathrm{KL}}(\pi_D \| u_D) - \tau_P D_{\mathrm{KL}}(\pi_P \| u_P), \tag{1}$$

where D_{KL} denotes the Kullback-Leibler divergence (Kullback & Leibler, 1951), u_D and u_P are uniform over each agent's available actions, and τ_D and τ_P control how strongly the agents favor higher-valued actions. This objective gives the mutual soft responses

$$\pi_D^\star(a^D \mid s, c) \propto u_D(a^D) \exp\left(\frac{\mathbb{E}_{a^P \sim \pi_P^\star}[Q_c(s, a^D, a^P)]}{\tau_D} \right), \qquad \pi_P^\star(a^P \mid s, c) \propto u_P(a^P) \exp\left(\frac{\mathbb{E}_{a^D \sim \pi_D^\star}[Q_c(s, a^D, a^P)]}{\tau_P} \right) \tag{2}$$

The two policies depend on one another. Restricting the constrained agent's feasible actions changes both its own behavior and how the demonstrator is expected to respond, capturing the compensatory behavior described above. Together with the physical feasibility rules of each setting, these policies define a distribution $\rho_c(a^D, a^P \mid s)$ over action pairs, which we use to generate demonstrations under the true capability.

At test time, the helper replaces the demonstrator while the constrained agent retains the same capability and behavioral model. The agents choose their actions simultaneously, and the resulting action pair is executed according to the dynamics and feasibility rules of the setting. We provide details on computing the mutual responses in Appendix A.1.1.

The benchmark spans three joint-carrying settings of increasing physical complexity. We begin with 2D rod carrying, where two agents apply forces at opposite ends of a rigid rod and jointly determine its translation and rotation. The constrained agent is limited to a continuous range of

Algorithm 1 Watch, Infer, Coordinate overview

Input: demonstrations $\mathcal{D}$, candidate capabilities $\mathcal{C}$, prior $p(c)$, test task
for each capability $c \in \mathcal{C}$ **do**
    Compute $S(c, \mathcal{D})$ from the observed actions of both agents
    $L(c) \leftarrow \log p(c) + S(c, \mathcal{D})$
end for
$b(c \mid \mathcal{D}) \leftarrow \mathrm{softmax}_{c \in \mathcal{C}} L(c)$
Plan with the constrained agent using $b(c \mid \mathcal{D})$

force directions, and we consider 2,665 candidate force-direction constraints. We then move to fixed-base dual-UR5 carrying, where two robot arms grasp opposite ends of a rigid beam and the constrained agent is subject to joint position and joint velocity limits. Because the joint motion required to move the beam depends on the robot's posture, the same beam motion may be feasible in one configuration and infeasible in another. This setting contains 172 candidate capabilities over individual and paired joint limitations. Finally, mobile dual-UR5 carrying adds mobile bases to the two arms, introducing constraints on both arm and base motion. Combining the possible arm and base constraints gives 475 candidate capabilities. In each setting, we use three fixed physical-constraint cases for final evaluation. Additional benchmark specifications and evaluation details are provided in Appendix A.2. We will release the benchmark code, demonstration data, and evaluation scripts upon acceptance.

4 METHOD

4.1 OVERVIEW

We infer the constrained agent's capability by treating each candidate capability as a hypothesis about which actions the agent can execute. At each observed state, a candidate changes the constrained agent's feasible actions and, in turn, how the demonstrator is expected to respond. Using the boundedly rational behavioral model, we predict the joint behavior of the two agents under each candidate and score how well it matches the observed actions across demonstrations. These scores are used to form a belief over the constrained agent's capability. In the low-dimensional setting, we can compute this score explicitly. In the more complex settings, evaluating the behavioral model for every observed state and candidate becomes too expensive, so we use learned models to estimate how well each candidate explains the observed coordination.

Formally, let $\mathcal{D}$ denote the observed demonstrations and $\mathcal{C}$ the set of candidate capabilities. We define $S(c, \mathcal{D})$ as the score measuring how well candidate c explains the observed coordination across all demonstrations. Let $p(c)$ denote the prior probability assigned to candidate capability c before observing the demonstrations. We combine this prior with the demonstration score to obtain

$$L(c) = \log p(c) + S(c, \mathcal{D}), \qquad b(c \mid \mathcal{D}) = \mathrm{softmax}_{c \in \mathcal{C}} L(c), \tag{3}$$

where $b(c \mid \mathcal{D})$ is the resulting belief over the constrained agent's capability. Algorithm 1 summarizes the overall procedure.

To compute $S(c, \mathcal{D})$, we measure how likely the observed actions of both agents are under candidate capability c. At each observed state, the behavioral model defines a distribution $\rho_{c,\tau}(a^D, a^P \mid s)$ over their joint actions, where $\tau = (\tau_D, \tau_P)$ contains the agents' behavior temperatures. We evaluate the probability assigned to each observed action pair and combine these probabilities across all states and demonstrations:

$$S(c, \mathcal{D}) = \log \left(\sum_{\tau} \exp \left[\sum_{d \in \mathcal{D}} \sum_{t \in d} \log \rho_{c,\tau} \left(a_t^D, a_t^P \mid s_t \right) + \log p(\tau) \right] \right). \tag{4}$$

A candidate therefore receives a high score when it predicts both the constrained agent's actions and the demonstrator's response well.

4.2 COMPUTING CAPABILITY SCORES

Computing the score in Eq. 4 exactly requires solving the behavioral model at every observed state for every candidate capability. This becomes prohibitively expensive as the state and action spaces

grow. We therefore train learned models offline on simulated coordination where the constrained agent's capability is known. At inference time, the true capability is hidden, and the models receive the observed coordination together with a candidate capability to estimate how well that candidate explains the demonstrations.

Across all three settings, we use the same capability-scoring idea, but vary how much of the computation is approximated by a learned model. In the 2D rod setting, we learn the action values predicted by the behavioral model for both agents, convert them into action distributions, and compute the capability score explicitly from the likelihood of the observed actions. In the fixed-base dual-UR5 setting, we predict each observed step's contribution to the same capability score directly and combine these predictions across the demonstration. The mobile dual-UR5 setting further increases the complexity and introduces both arm and base constraints, so we predict scores at the demonstration level and combine the contributions of the arm and base components when evaluating a candidate capability. The resulting scores are combined into the capability belief used for test-time planning.

4.3 Planning with the Inferred Capability

At test time, the helper must coordinate with the constrained agent on a new task that may require motions not seen in the demonstrations. Rather than transferring a demonstrated coordination strategy, we use the inferred capability to constrain the partner behavior considered during planning. Within a model predictive control (MPC) framework, we use the cross-entropy method (CEM) to optimize the helper's action sequence (Rubinstein, 1999; Rubinstein & Kroese, 2004). For each candidate helper sequence, the planner rolls out how the constrained agent is expected to respond under its inferred capability. During each rollout, the constrained agent is limited to actions that are feasible under the inferred capability. The planner therefore searches over coordination strategies that are compatible with what the constrained agent can physically execute, allowing the helper to plan for a new task without observing prior coordination on that task. We provide additional derivations and implementation details in Appendix A.1.

5 Experiments and Results

We evaluate whether our method can accurately infer the constrained agent's physical constraints from prior coordination and whether the inferred capability supports zero-shot coordination on a new task. We study both questions across the 2D rod, fixed-base dual-UR5, and mobile dual-UR5 settings.

5.1 Experimental Setup

For capability inference, we vary the number of observed demonstrations over $N \in \{1, 2, 4, 8\}$ and measure normalized Hamming distance between the inferred and true constraints, where lower is better. For zero-shot coordination, all inference-based methods use $N = 8$ demonstrations. We evaluate performance on the new tasks with obstacles and additionally report results on obstacle-free control tasks to separate failures in basic coordination from failures that arise when the new task requires different motions.

We compare against four baselines that isolate different sources of performance. CE-CM-Div (Tisnikar et al., 2026) is a prior capability-inference method that estimates task-invariant partner capabilities from multi-agent trajectories and serves as our primary inference baseline. Capacity-blind CEM uses the same planner as our method but receives no information about the constrained agent's capability, testing whether capability information is necessary for planning. Oracle CEM receives the true physical constraints and provides a reference for performance with perfect capability information. Behavioral cloning directly learns from the observed demonstrations, allowing us to test whether reproducing demonstrated coordination is sufficient when the task changes.

For planning, we report task success and Success weighted by Number of Actions (SNA) (Chen et al., 2021). Failed episodes receive an SNA of zero. For successful episodes, SNA is the ratio between the reference number of actions and the larger of the executed and reference numbers of actions, so higher SNA indicates more efficient successful coordination. Success and SNA are reported as percentages, with higher values indicating better performance. All results are reported as mean

and standard error over three random seeds. All CEM-based methods use the same planner and differ only in the capability information provided to it. Additional details on baseline implementations, evaluation tasks, training, and planner hyperparameters are provided in Appendix A.2.

5.2 Constraint Inference

We first measure constraint-inference accuracy from prior coordination across all three settings and demonstration budgets. Table 2 shows that our method outperforms CE-CM-Div across all three settings and demonstration budgets. Additional demonstrations are especially helpful in the mobile dual-UR5 setting, where error decreases from 0.190 at $N = 1$ to 0.009 at $N = 8$.

We also perform an ablation to test whether observing the demonstrator improves capability inference. For this ablation, we retrain the inference model using only the constrained agent's behavior. As shown in Table 2, using both agents' behavior roughly halves inference error in the fixed-base dual-UR5 setting across all demonstration budgets and reduces error in the mobile dual-UR5 setting at every budget. In the 2D rod setting, the demonstrator's behavior substantially improves inference when only one or two demonstrations are available. These results support our key insight that physical coupling makes the demonstrator's response informative about constraints that may be difficult to identify from the constrained agent's behavior alone.

Table 2: Constraint-inference accuracy at $N \in \{1, 2, 4, 8\}$ demonstrations. We report normalized Hamming distance between the inferred and true constraints. "Ours, partner only" uses only the constrained agent's behavior for inference and excludes the demonstrator's behavior. Values are mean $\pm$ standard error; lower is better. Bold indicates the best result within each robot setting and row, including ties.

	2D Rod			Fixed-Base Dual-UR5			Mobile Dual-UR5		
N	CE-CM-Div	Ours, partner only	Ours	CE-CM-Div	Ours, partner only	Ours	CE-CM-Div	Ours, partner only	Ours
1	0.304 ± 0.038	0.104 ± 0.015	**0.058 ± 0.017**	0.074 ± 0.000	0.068 ± 0.006	**0.031 ± 0.012**	0.306 ± 0.028	0.201 ± 0.045	**0.190 ± 0.060**
2	0.340 ± 0.041	0.031 ± 0.009	**0.003 ± 0.002**	0.074 ± 0.000	0.037 ± 0.000	**0.019 ± 0.000**	0.278 ± 0.028	0.148 ± 0.042	**0.130 ± 0.060**
4	0.348 ± 0.031	**0.002 ± 0.002**	**0.002 ± 0.002**	0.074 ± 0.000	0.037 ± 0.000	**0.019 ± 0.000**	0.250 ± 0.000	0.090 ± 0.006	**0.069 ± 0.005**
8	0.302 ± 0.011	**0.000 ± 0.000**	0.002 ± 0.002	0.074 ± 0.000	0.043 ± 0.006	**0.019 ± 0.011**	0.250 ± 0.000	0.042 ± 0.000	**0.009 ± 0.000**

5.3 Zero-Shot Coordination

We next evaluate whether the inferred capability supports coordination on the new tasks with obstacles. As shown in Table 3, our method achieves success rates of 92.6%, 85.2%, and 63.0% in the 2D rod, fixed-base dual-UR5, and mobile dual-UR5 settings, respectively, approaching Oracle CEM at 96.3%, 88.9%, and 70.4%. CE-CM-Div performs substantially worse across all three settings, showing that more accurate capability inference translates into better coordination when the task changes.

The remaining baselines highlight why capability information must transfer beyond the demonstrated behavior. Capacity-blind CEM performs worse when successful planning depends on accounting for the constrained agent's limitations, while behavioral cloning fails on all three new tasks despite nonzero performance on the obstacle-free controls. On the obstacle-free control tasks, our method closely matches Oracle CEM across all three settings.

5.4 Qualitative Examples

Figure 3 provides qualitative examples of how the inferred capability affects test-time coordination. Panel A shows successful trajectories across all three benchmark settings, where the helper coordinates with the constrained agent to navigate the jointly carried object around obstacles. Panel B illustrates how the inferred constraint changes the planner's behavior. When the partner's constraint is ignored, the planner selects a motion that requires the constrained agent to reach a natural but infeasible pose. Using the inferred constraint, the planner instead selects a feasible alternative that respects the partner's physical limitations.

Table 3: Planning performance on the obstacle-free control and new tasks with obstacles. Our method, CE-CM-Div, and behavioral cloning use $N = 8$ demonstrations. Capacity-blind CEM uses no demonstrations, while Oracle CEM receives the true constraints. Success and SNA are percentages reported as mean ± standard error over three seeds; higher is better. Bold indicates the best non-oracle result for each task and metric, including ties.

		2D Rod		Fixed-Base Dual-UR5		Mobile Dual-UR5	
Task	Method	Success ↑	SNA ↑	Success ↑	SNA ↑	Success ↑	SNA ↑
Obstacle-free control	Capacity-blind + CEM	**100.0 ± 0.0**	91.4 ± 0.9	0.0 ± 0.0	0.0 ± 0.0	66.7 ± 0.0	55.2 ± 0.0
	CE-CM-Div + CEM	**100.0 ± 0.0**	90.4 ± 1.6	55.6 ± 11.1	30.5 ± 8.3	66.7 ± 0.0	54.3 ± 0.0
	Behavioral cloning	66.7 ± 0.0	52.5 ± 6.1	22.2 ± 11.1	7.5 ± 3.8	66.7 ± 0.0	31.1 ± 0.0
	Oracle + CEM	100.0 ± 0.0	91.4 ± 0.2	100.0 ± 0.0	44.8 ± 1.8	100.0 ± 0.0	80.2 ± 0.0
	Ours + CEM	**100.0 ± 0.0**	**92.4 ± 0.4**	**100.0 ± 0.0**	**43.9 ± 0.8**	**100.0 ± 0.0**	**80.2 ± 0.0**
New tasks with obstacles	Capacity-blind + CEM	51.9 ± 3.7	44.4 ± 2.8	0.0 ± 0.0	0.0 ± 0.0	25.9 ± 9.8	14.2 ± 5.4
	CE-CM-Div + CEM	55.6 ± 0.0	46.1 ± 0.7	44.4 ± 0.0	16.9 ± 1.5	37.0 ± 3.7	19.7 ± 1.8
	Behavioral cloning	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
	Oracle + CEM	96.3 ± 3.7	76.0 ± 2.0	88.9 ± 6.4	31.3 ± 1.9	70.4 ± 3.7	38.4 ± 2.5
	Ours + CEM	**92.6 ± 3.7**	**72.0 ± 3.3**	**85.2 ± 3.7**	**31.1 ± 0.8**	**63.0 ± 7.4**	**34.5 ± 4.4**

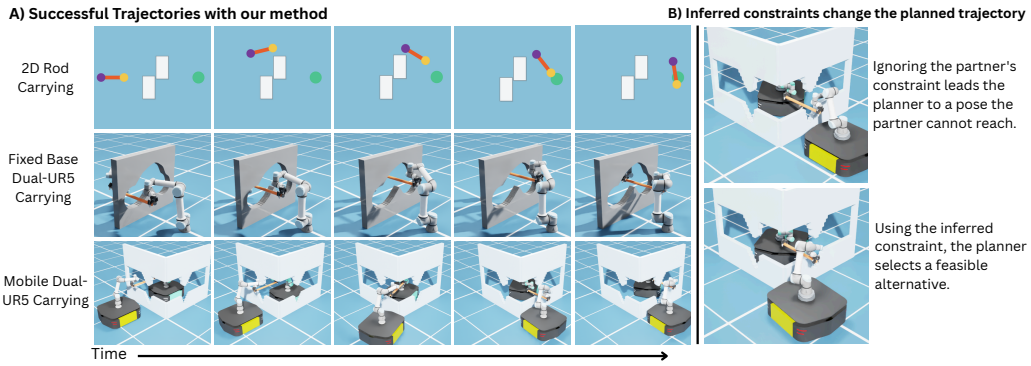

Figure 3: **Qualitative examples of zero-shot coordination using our method.** (A) Representative successful trajectories for 2D rod carrying, fixed-base dual-UR5 carrying, and mobile dual-UR5 carrying, shown from left to right over time. (B) Example of how an inferred physical constraint changes test-time planning. Ignoring the constrained agent's limitation leads to a pose it cannot reach, while incorporating the inferred constraint causes the planner to select a feasible alternative.

6 Conclusion and Limitations

In summary, we formulate the problem of inferring a robot partner's low-level physical constraints from prior multi-agent coordination and using them for zero-shot coordination on a new task. We introduce Watch, Infer, Coordinate, a benchmark spanning three physically coupled carrying settings, and show how the joint behavior of both agents can be used to infer a partner's physical constraints. Our experiments show that the inferred constraints improve zero-shot coordination and achieve performance close to an oracle with access to the true constraints.

As a first step, we intentionally assume a known behavioral model, goals and dynamics, full observations, and a finite constraint set to isolate the core challenge of constraint inference and coordination without introducing additional sources of uncertainty. Relaxing these assumptions is an important next step toward broader generalization. Moreover, directly comparable baselines are limited, as existing approaches use different capability representations or observations. We will release benchmark code and evaluation scripts to support broader evaluations across tasks, constraints, and partner behaviors, and future extensions to heterogeneous teams, including human-robot coordination.

7 AI USE DISCLOSURE

In this work, we used generative AI tools to write code to generate synthetic datasets, implement methods, as well as supporting qualitative data analysis. We have not used generative AI tools for developing conceptual frameworks, proposing or refining hypothesis, providing feedback on research methodology, or interpreting results. Formulating mathematical claims, writing of proofs, assisting with translation, cleaning and reformatting datasets are not applicable to this work. We have reviewed all AI-assisted work. For example, LLM-generated code was verified and tested for correctness by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

8 REPRODUCIBILITY STATEMENT

Sections 3.1 and 3.3 describe the problem setup, benchmark, and demonstration generation. Sections 4.1–4.3 describe capability inference and test-time planning. Sections 5.1–5.2 describe the experimental setup, baselines, evaluation metrics, and partner-only ablation. Appendix A.1 details the behavioral model, capability-scoring procedures, physical-consistency checks, and test-time planning. Appendix A.2's environment-specific subsections report benchmark specifications, demonstration configurations, model architectures, and training details. Appendix A.2's "Baseline Implementations" describes the baselines and ablation, while its "Evaluation Protocol" specifies evaluation metrics, trial counts, random seeds, aggregation procedures, and CEM-MPC hyperparameters. We will release the benchmark code, demonstration data, and evaluation scripts upon acceptance.

REFERENCES

Elie Aljalbout, Maximilian Karl, and Patrick van der Smagt. Clas: Coordinating multi-robot manipulation with central latent action spaces. In *Learning for Dynamics and Control Conference*, pp. 1152–1166. PMLR, 2023.

Zhihao Cao, Zidong Wang, Siwen Xie, Anji Liu, and Lifeng Fan. Smart help: Strategic opponent modeling for proactive and adaptive robot assistance in households. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18091–18101. IEEE, 2024.

Changan Chen, Sagnik Majumder, Al-Halah Ziad, Ruohan Gao, Santhosh Kumar Ramakrishnan, and Kristen Grauman. Learning to set waypoints for audio-visual navigation. In *ICLR*, 2021.

Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations. In *International Workshop on the Algorithmic Foundations of Robotics*, pp. 228–245. Springer, 2018.

Glen Chou, Necmiye Ozay, and Dmitry Berenson. Learning parametric constraints in high dimensions from demonstrations. In *Conference on robot learning*, pp. 1211–1230. PMLR, 2020.

Dayi Dong, Maulik Bhatt, Seoyeon Choi, and Negar Mehr. Mimic-d: Multi-modal imitation for multi-agent coordination with decentralized diffusion policies. *arXiv preprint arXiv:2509.14159*, 2025.

Ria Doshi, Tian Gao, Annie Chen, Chelsea Finn, and Jeannette Bohg. Chorus: Decentralized multi-embodiment collaboration with one vla policy. *arXiv preprint arXiv:2606.12352*, 2026.

Weihua Du, Qiushi Lyu, Jiaming Shan, Zhenting Qi, Hongxin Zhang, Sunli Chen, Andi Peng, Tianmin Shu, Kwonjoon Lee, Behzad Dariush, et al. Constrained human-ai cooperation: An inclusive embodied social intelligence challenge. *Advances in neural information processing systems*, 37: 44526–44553, 2024.

Hamed Farivarnejad and Spring Berman. Multirobot control strategies for collective transport. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):205–219, 2022.

Chengyang He, Gadiel Mark Sznaier Camps, Xu Liu, Mac Schwager, and Guillaume Adrien Sartoretti. Latent theory of mind: A decentralized diffusion architecture for cooperative manipulation. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp.

392–405. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/he25a.html.

Hussein Ali Jaafar, Cheng-Hao Kao, and Sajad Saeedi. Mr. cap: Multi-robot joint control and planning for object transport. *IEEE Control Systems Letters*, 8:139–144, 2024.

Florian Kennel-Maushart and Stelian Coros. Payload-aware trajectory optimisation for non-holonomic mobile multi-robot manipulation with tip-over avoidance. *IEEE Robotics and Automation Letters*, 9(9):7669–7676, 2024.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Joshua Lee, Jeffrey Fong, Bing Cai Kok, and Harold Soh. Getting to know one another: Calibrating intent, capabilities and trust for human-robot collaboration. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6296–6303. IEEE, 2020.

Christopher L Lewis and Anthony A Maciejewski. Fault tolerant operation of kinematically redundant manipulators for locked joint failures. *IEEE Transactions on Robotics and Automation*, 13 (4):622–629, 1997.

Cecilia Lindig-León, Gerrit Schmid, and Daniel A Braun. Bounded rational response equilibria in human sensorimotor interactions. *Proceedings of the Royal Society B: Biological Sciences*, 288 (1962):20212094, 2021.

Guiliang Liu, Yudong Luo, Ashish Gaurav, Kasra Rezaee, and Pascal Poupart. Benchmarking constraint inference in inverse reinforcement learning. *arXiv preprint arXiv:2206.09670*, 2022.

Ziang Liu, Yuanchen Ju, Yu Da, Tom Silver, Pranav N Thakkar, Jenna Li, Justin Guo, Katherine Dimitropoulou, and Tapomayukh Bhattacharjee. Grace: Generalizing robot-assisted caregiving with user functionality embeddings. In *2025 20th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 686–695. IEEE, 2025.

Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. Inverse constrained reinforcement learning. In *International conference on machine learning*, pp. 7390–7399. PMLR, 2021.

Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995.

Ofir Nachum, Michael Ahn, Hugo Ponte, Shixiang (Shane) Gu, and Vikash Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 110–121. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/nachum20a.html.

Eley Ng, Ziang Liu, and Monroe Kennedy. Diffusion co-policy for synergistic human-robot collaborative tasks. *IEEE Robotics and Automation Letters*, 9(1):215–222, 2023.

Bikram Pandit, Ashutosh Gupta, Mohitvishnu S. Gadde, Addison Johnson, Aayam Kumar Shrestha, Helei Duan, Jeremy Dao, and Alan Fern. Learning decentralized multi-biped control for payload transport. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1021–1034. PMLR, 06–09 Nov 2025a. URL https://proceedings.mlr.press/v270/pandit25a.html.

Bikram Pandit, Aayam Kumar Shrestha, and Alan Fern. Multi-quadruped cooperative object transport: Learning decentralized pinch-lift-move. *arXiv preprint arXiv:2509.14342*, 2025b.

Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B. Tenenbaum, Sanja Fidler, and Antonio Torralba. Watch-and-help: A challenge for social perception and human-{ai} collaboration. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=w_7JMpGZRh0.

Xavier Puig, Tianmin Shu, Joshua B. Tenenbaum, and Antonio Torralba. Nopa: Neurally-guided online probabilistic assistance for building socially intelligent home assistants. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7628–7634, 2023. doi: 10.1109/ICRA48891.2023.10161352.

Sid Reddy, Anca Dragan, and Sergey Levine. Where do you think you're going?: Inferring beliefs about dynamics from behavior. *Advances in Neural Information Processing Systems*, 31, 2018.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

Dexter RR Scobee and S Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. *arXiv preprint arXiv:1909.05477*, 2019.

Haojun Shi, Suyu Ye, Katherine M Guerrerio, Jianzhi Shen, Yifan Yin, Daniel Khashabi, Chien-Ming Huang, and Tianmin Shu. Safe and interpretable multimodal path planning for multi-agent cooperation. *arXiv preprint arXiv:2602.19304*, 2026.

Kazuki Shibata, Tomohiko Jimbo, and Takamitsu Matsubara. Deep reinforcement learning of event-triggered communication and consensus-based control for distributed cooperative transport. *Robotics and Autonomous Systems*, 159:104307, 2023.

Kun Song, Gaoming Chen, Shentao Ma, Ninglong Jin, Guangbao Zhao, Mingyu Ding, Zhenhua Xiong, and Jia Pan. Collabot: Vision-language guided simultaneous collaborative manipulation. *arXiv preprint arXiv:2508.03526*, 2025.

Theodoros Stouraitis, Iordanis Chatzinikolaidis, Michael Gienger, and Sethu Vijayakumar. Online hybrid motion planning for dyadic collaborative manipulation via bilevel optimization. *IEEE Transactions on Robotics*, 36(5):1452–1471, 2020.

Renato Tinós, Marco Henrique Terra, and Marcel Bergerman. A fault tolerance framework for cooperative robotic manipulators. *Control Engineering Practice*, 15(5):615–625, 2007.

Peter Tisnikar, Gerard Canal, and Matteo Leonetti. Probabilistic inference of human capabilities from passive observations. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8779–8785. IEEE, 2024.

Peter Tisnikar, Maja Swieczkowska, Benteng Ma, Gerard Canal, and Matteo Leonetti. Partner capability estimation for task-agnostic adaptation in ad-hoc teamwork. *arXiv preprint arXiv:2607.27177*, 2026.

Annie Xie, Dylan Losey, Ryan Tolsma, Chelsea Finn, and Dorsa Sadigh. Learning latent representations to influence multi-agent interaction. In *Conference on robot learning*, pp. 575–588. PMLR, 2021.

Lei Yan, Theodoros Stouraitis, and Sethu Vijayakumar. Decentralized ability-aware adaptive control for multi-robot collaborative manipulation. *IEEE Robotics and Automation Letters*, 6(2):2311–2318, 2021.

Elvin Yang and Christoforos Mavrogiannis. Implicit communication in human-robot collaborative transport. In *2025 20th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 23–33. IEEE, 2025.

Lin Zhang, Yufeng Sun, Andrew Barth, and Ou Ma. Decentralized control of multi-robot system in cooperative object transportation using deep reinforcement learning. *IEEE access*, 8:184109–184119, 2020.

Zhouyu Zhang, Chih-Yuan Chiu, and Glen Chou. Constraint learning in multi-agent dynamic games from demonstrations of local nash interactions. *IEEE Robotics and Automation Letters*, 11(6):6696–6703, 2026.

Tan Zhi-Xuan, Jordyn Mann, Tom Silver, Josh Tenenbaum, and Vikash Mansinghka. Online bayesian goal inference for boundedly rational planning agents. *Advances in neural information processing systems*, 33:19238–19250, 2020.

A Appendix

A.1 Method Details

A.1.1 Behavioral Model

The capability score in Eq. 4 requires predicting how the two agents would behave under a candidate capability c. We now describe the behavioral model used to construct this prediction. Intuitively, a candidate capability determines which actions the constrained agent can take. Given these feasible actions, each agent favors actions that lead to good joint outcomes, without assuming perfectly optimal action selection. The agents' predicted actions are interdependent: the demonstrator chooses actions based on what the constrained agent can do, while the constrained agent's preferred actions depend on how the demonstrator is expected to respond. We therefore update their predicted policies iteratively to obtain a model of their joint behavior.

Let D denote the demonstrator and P the constrained agent. At state s, the agents choose actions a^D and a^P. Each setting defines a shared action-pair value

$$Q_c(s, a^D, a^P),$$

which measures the value of taking the joint action (a^D, a^P) and continuing from the resulting state. The candidate capability c determines the feasible action set $\mathcal{A}_P(s, c)$ for the constrained agent. Thus, different candidate capabilities can produce different predicted behavior even at the same state.

We model both agents as boundedly rational under the shared regularized objective

$$\mathcal{J}_c(\pi_D, \pi_P; s) = \mathbb{E}_{a^D \sim \pi_D,\, a^P \sim \pi_P}\left[Q_c(s, a^D, a^P)\right] - \tau_D D_{\mathrm{KL}}(\pi_D \| u_D) - \tau_P D_{\mathrm{KL}}(\pi_P \| u_{P,c}), \quad (5)$$

where u_D is the uniform distribution over the demonstrator's available actions and $u_{P,c}$ is the uniform distribution over the constrained agent's actions that are feasible under capability c. The temperatures τ_D and τ_P determine how strongly the agents favor high-value actions. Smaller temperatures place more probability on the highest-value actions, while larger temperatures allow a wider range of actions.

The two agents' policies are coupled because whether an action is useful for one agent depends on what the other agent is expected to do. We therefore compute the expected value of each action by averaging over the other agent's current policy:

$$\bar{Q}_D(s, a^D) = \mathbb{E}_{a^P \sim \pi_P}\left[Q_c(s, a^D, a^P)\right], \qquad \bar{Q}_P(s, a^P) = \mathbb{E}_{a^D \sim \pi_D}\left[Q_c(s, a^D, a^P)\right]. \quad (6)$$

For example, $\bar{Q}_D(s, a^D)$ measures how good demonstrator action a^D is on average given what we currently expect the constrained agent to do.

We convert these expected action values into soft action distributions:

$$\widetilde{\pi}_D(a^D) = \frac{u_D(a^D)\exp\left(\bar{Q}_D(s, a^D)/\tau_D\right)}{\sum_{a^{D\prime}} u_D(a^{D\prime})\exp\left(\bar{Q}_D(s, a^{D\prime})/\tau_D\right)}, \quad (7)$$

and

$$\widetilde{\pi}_P(a^P) = \frac{u_{P,c}(a^P)\exp\left(\bar{Q}_P(s, a^P)/\tau_P\right)}{\sum_{a^{P\prime}} u_{P,c}(a^{P\prime})\exp\left(\bar{Q}_P(s, a^{P\prime})/\tau_P\right)}. \quad (8)$$

These distributions assign greater probability to actions with higher expected joint value. Because $u_{P,c}$ places probability only on actions that are feasible under capability c, infeasible constrained-agent actions receive no probability.

The two soft responses depend on one another. Changing the constrained agent's policy changes the demonstrator's expected action values, which can change the demonstrator's policy, and vice versa.

We therefore compute the policies iteratively. Starting from uniform policies, we first update the demonstrator based on the constrained agent's current policy and then update the constrained agent based on the newly updated demonstrator policy. To make these updates more stable, we move each policy partway toward its newly computed soft response. For the demonstrator, this update is

$$\log \pi_D^{(m)} = \text{lognormalize}\left((1-\eta)\log \pi_D^{(m-1)} + \eta \log \widetilde{\pi}_D^{(m)}\right), \tag{9}$$

and the constrained agent is updated analogously:

$$\log \pi_P^{(m)} = \text{lognormalize}\left((1-\eta)\log \pi_P^{(m-1)} + \eta \log \widetilde{\pi}_P^{(m)}\right), \tag{10}$$

where $\eta = 0.5$ and $\text{lognormalize}(x) = x - \text{logsumexp}(x)$. Algorithm 2 summarizes the complete procedure.

Algorithm 2 Mutual soft-response computation

Input: state s, candidate capability c, joint-action values Q_c, temperatures τ_D, τ_P

Initialize $\pi_D^{(0)} \leftarrow u_D$ and $\pi_P^{(0)} \leftarrow u_{P,c}$
for $m = 1, \ldots, M$ **do**
  Compute $\bar{Q}_D^{(m)}$ using $\pi_P^{(m-1)}$ and Eq. 6
  Compute $\widetilde{\pi}_D^{(m)}$ using Eq. 7
  Update $\pi_D^{(m)}$ using Eq. 9
  Compute $\bar{Q}_P^{(m)}$ using $\pi_D^{(m)}$ and Eq. 6
  Compute $\widetilde{\pi}_P^{(m)}$ using Eq. 8
  Update $\pi_P^{(m)}$ using Eq. 10
end for
Compute a final soft response π_D using $\pi_P^{(M)}$
Compute a final soft response π_P using $\pi_D^{(M)}$
Return: π_D, π_P

After the iterative updates, we compute one final soft response for each agent using the final policy of the other agent. We denote these resulting policies by π_D and π_P. Together, they define the capability-conditioned joint-action distribution

$$\rho_{c,\tau}(a^D, a^P \mid s) = \pi_D(a^D \mid s, c, \tau)\, \pi_P(a^P \mid s, c, \tau), \tag{11}$$

where $\tau = (\tau_D, \tau_P)$. Changing the candidate capability c therefore changes the constrained agent's feasible actions and, through the coupled policies, can also change the demonstrator's predicted behavior.

A.1.2 Capability Inference Details

Across all three settings, capability inference follows the same basic procedure. For each candidate capability, we compute a score measuring how well that candidate explains the observed coordination. Candidates that are physically inconsistent with the demonstrations are excluded, and the remaining scores are combined with a capability prior and normalized to form the belief used for planning. The settings differ only in how we compute the score.

In the 2D rod setting, the action space is small enough to score candidates from the predicted actions directly. For each observed state and candidate capability, the learned critic predicts values for the 64 possible force directions of both the demonstrator and constrained agent. We convert these values into action distributions and evaluate how much probability they assign to the two actions that were actually observed. Because the recorded actions are continuous directions, we interpret each discrete action probability as a density over its corresponding angular interval. We additionally mix the density with a small uniform component of weight 0.02 so that a single unusual action does not assign zero likelihood to an otherwise plausible candidate.

We sum the log probabilities of both agents' observed actions over all steps and demonstrations to obtain a score for each candidate. The behavioral temperatures of the two agents are not assumed to be known. We therefore evaluate seven possible temperatures for each agent,

$$\{0.001,\ 0.002,\ 0.005,\ 0.01,\ 0.02,\ 0.05,\ 0.1\},$$

giving 49 temperature pairs. A single temperature pair is shared across all selected demonstrations, and we marginalize over these 49 possibilities when computing the final candidate score. The resulting scores are combined with the capability prior and normalized across candidates.

For the UR5 settings, explicitly constructing these action distributions at every observed state for every candidate would require substantially more computation. We therefore learn the candidate score directly. In the fixed-base setting, a candidate-conditioned critic assigns a scalar score to each observed step using the state, the actions of both agents, their resulting motion, and the candidate capability. We sum these step scores over each demonstration and then combine them across all selected demonstrations. A separate physical-consistency check removes candidates whose joint position or velocity limits are incompatible with the observed motion. We then add a sparsity prior that favors capabilities with fewer active joint constraints and normalize the resulting scores over the 172 candidates.

The mobile setting uses the same candidate-scoring idea, but scores complete observed sequences rather than individual steps. A recurrent critic processes each demonstration and produces a score from the resulting sequence representation. Because a mobile capability contains both an arm constraint and a base-motion constraint, the critic scores these two components separately and then combines their contributions when evaluating a complete candidate. As in the fixed-base setting, an independent physical-consistency check removes candidates that cannot reproduce the observed motion before the learned scores are normalized. This produces a distribution over the 475 composite capabilities.

A.1.3 Physical Consistency and Test-Time Planning

At test time, the inferred capability is used inside CEM-MPC to predict which coordinated motions are physically possible for the constrained agent. For each candidate helper action sequence, the planner predicts the constrained agent's response while restricting it to actions that are feasible under the capability being considered. The resulting joint motion is rolled forward and scored for progress toward the test-task goal. In the rod and fixed-base settings, planning accounts for uncertainty over multiple candidate capabilities from the inferred belief, while the mobile setting plans with the highest-scoring capability. The helper executes the first action of the selected sequence and replans from the resulting observed state.

Proposed motions must also satisfy the physical constraints of the setting. In the UR5 settings, the planner checks whether a motion is compatible with the candidate capability as well as the robot kinematics, workspace, and collision constraints. In the rod setting, proposed motions are similarly checked for collisions. Motions that violate these constraints are treated as infeasible rather than modified into a different executable motion.

Within each setting, all CEM-based methods use the same planner and planning budget. They differ only in the capability information supplied to the planner: our method uses its inferred capability, CE-CM-Div uses its inferred capability, Capacity-blind CEM receives no demonstration-based capability information, and Oracle CEM receives the true capability.

A.2 Experiment Implementation Details

Across all three settings, we evaluate capability inference using $N \in \{1, 2, 4, 8\}$ obstacle-free demonstrations and use $N = 8$ demonstrations for zero-shot coordination. Each setting contains three fixed physical-constraint cases for final evaluation. Planning is evaluated on three new tasks with obstacles and one obstacle-free control task, using three evaluation seeds.

A.2.1 2D Rod Carrying

In the 2D rod setting, the state consists of the rod center and orientation, (x, y, θ). The two agents apply unit forces at opposite ends of the rod, jointly determining its translation and rotation. Each agent acts over 64 force directions. The demonstrator can apply force in any direction, while the constrained agent is limited to a continuous angular range specified by its capability. The candidate set contains 2,665 such constraints, and the three evaluation cases use angular ranges centered at 0° with widths of 30°, 45°, and 65°. Demonstrations are generated on obstacle-free goal-reaching tasks using the behavioral model described above. For $N = 1, 2, 4, 8$, we use increasingly larger subsets

of the same fixed demonstration sequence, so additional demonstrations are added without changing the earlier ones. To make inference efficient, we train a neural network to predict the action values produced by the behavioral model for both agents. The model uses three hidden layers of width 256 and is trained on 98,304 simulated examples pairing rod states and candidate capabilities with action-value targets from the behavioral model.

A.2.2 Fixed-Base Dual-UR5 Carrying

The fixed-base dual-UR5 setting is simulated in Isaac Sim with two UR5e arms grasping opposite ends of a rigid beam. Demonstrations represent the coordinated end-effector motion using the beam's y and z position and yaw orientation. Each agent selects from 257 possible three-dimensional beam-velocity commands, and the two proposals are combined into a shared beam motion. A capability describes possible restrictions on the six robot joints: each joint can have a reduced velocity limit, a tighter lower position limit, or a tighter upper position limit. This gives 18 possible individual restrictions. The 172 candidate capabilities include the unrestricted robot, each of the 18 individual restrictions, and every pair of restrictions. The three evaluation cases use a velocity restriction on the first joint, a lower position restriction on the third joint, and the combination of those two restrictions. The obstacle-free demonstrations cover six different beam motions, with $N = 8$ using eight demonstrations drawn from these motions. Capability scoring uses a neural network conditioned on both the observed coordination and a candidate capability. It produces a score for each observed step, and these scores are combined across the demonstrations. The model is trained with AdamW for 12,000 updates using demonstrations generated under known capabilities.

A.2.3 Mobile Dual-UR5 Carrying

The mobile dual-UR5 setting extends the same manipulation task with planar mobile bases. The state has 22 dimensions and contains the beam pose, both base poses, and both six-joint arm configurations. Each agent selects a four-dimensional motion command controlling translation and yaw from a set of 1,281 possible actions. A capability combines a constraint on the directions in which the base can move with a constraint on arm motion. We consider 25 base constraints and 19 arm constraints, producing 475 possible combinations. Each evaluation case is paired with eight fixed obstacle-free demonstrations totaling 100 observed steps. Six demonstrations emphasize arm motion while the bases remain stationary, and two emphasize base motion while the arms are held fixed. For $N = 1, 2, 4$, we use fixed subsets of these eight demonstrations. Capability scoring uses a recurrent neural network that processes an entire demonstration at once. It scores the arm and base components of each candidate and combines them to obtain a score for the complete capability. The model uses a one-layer GRU with hidden width 48 and is trained on 25 capability cases, with three additional cases used for validation.

A.2.4 Baseline Implementations

CE-CM-Div is our primary capability-inference baseline. It receives the same obstacle-free demonstrations as our method but infers capability from the observed state trajectories rather than the actions of both agents. For each candidate capability, it generates possible trajectories and compares them with the observed trajectory, then projects the resulting estimate onto the same candidate set used by our method. The inferred capability is passed to the same setting-specific CEM-MPC planner used by our method. In the mobile setting, the arm and base components are inferred separately and then combined into a complete capability.

Capacity-blind CEM uses the same planner but receives no information from the demonstrations about the constrained agent's capability. It therefore plans from a fixed, task-independent capability assumption defined for each setting. This corresponds to an unrestricted force-direction model in the rod setting, the fixed capability prior in the fixed-base setting, and the nominal composite capability in the mobile setting. Oracle CEM instead receives the true capability while using the same planner and planning budget, providing a reference for performance with perfect capability information.

Behavioral cloning directly predicts the helper's action from the $N = 8$ obstacle-free demonstrations. For each setting, we train a three-layer width-256 policy to predict the helper action in that setting's action space. At test time, the policy acts directly without capability inference or CEM-MPC planning.

For the partner-only comparison, we train separate inference models using only the constrained agent's behavior. These models use the same capability candidates and evaluation demonstrations as the full method but exclude the demonstrator's behavior when inferring the constraint.

A.2.5 Evaluation Protocol

Capability inference is evaluated using normalized Hamming distance between the inferred and true constraint representations. For binary representations $\widehat{z}, z \in \{0,1\}^m$, we compute

$$d_H(\widehat{z}, z) = \frac{1}{m} \sum_{j=1}^{m} \mathbf{1}[\widehat{z}_j \neq z_j]. \tag{12}$$

For the 2D rod, the representation indicates feasible motion over 72 fixed force directions. For the fixed-base dual-UR5, it indicates which of the 18 possible joint restrictions are active. For the mobile dual-UR5, we compute Hamming distance separately for the base-direction and arm constraints and give the two components equal weight. Lower values indicate more accurate constraint inference.

Zero-shot coordination is evaluated using task success and Success weighted by Number of Actions (SNA). A trial is successful if the robots reach the goal within the setting's action limit. If n is the number of actions executed and n_{ref} is the reference action count for the same task and physical constraint, we compute

$$\text{SNA} = \mathbf{1}[\text{success}] \frac{n_{\text{ref}}}{\max(n, n_{\text{ref}})}. \tag{13}$$

Failed trials therefore receive an SNA of zero, while successful trials receive higher scores when they complete the task using fewer actions. The reference trajectories use the true physical constraint and are used only for evaluation.

Table 4 summarizes the CEM-MPC parameters used in each setting. All CEM-based methods within a setting use the same planner configuration. The setting-specific CEM hyperparameters were chosen to keep planning runtime approximately comparable across the three environments.

Table 4: CEM-MPC hyperparameters used for zero-shot coordination.

Setting	Horizon	Population	Elites	Iterations
2D Rod	32	64	8	3
Fixed-Base Dual-UR5	4	16	2	2
Mobile Dual-UR5	4	4	1	1

Each setting uses three fixed physical-constraint cases for final evaluation. Planning evaluates each constraint on three new tasks with obstacles and one obstacle-free control task using seeds 42, 43, and 44. This gives 27 obstacle-task trials and 9 obstacle-free trials per method and setting. We first average performance across the constraint cases and tasks within each seed, then report the mean and standard error across the three seed-level results. Every scheduled evaluation trial is retained after one attempt, without outcome-dependent filtering or replacement.