Optimizing Interaction Through Environment Design: Reward Alignment and Intent Prediction in Human–Robot Collaboration

by

Yi-Shiuan Tung

B.S., Massachusetts Institute of Technology, 2015

M.Eng., Massachusetts Institute of Technology, 2018

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2026

Committee Members:

Alessandro Roncone, Chair

Prof. Bradley Hayes

Prof. Christoffer Heckman

Prof. Nisar Ahmed

Prof. Stefanos Nikolaidis

Tung, Yi-Shiuan (Ph.D., Computer Science)

Optimizing Interaction Through Environment Design: Reward Alignment and Intent Prediction in Human–Robot Collaboration

Thesis directed by Prof. Alessandro Roncone

For humans and robots to collaborate safely and fluently, the robot must anticipate its partner across two levels of intent: (1) the human's high-level goals (task structure and reward preferences) and (2) their low-level motion. Prior work improves these predictions by developing better inference algorithms or collecting richer datasets while treating the environment as fixed. This thesis argues that the environment itself is a first-class design variable that the robot can actively configure to improve its understanding of the human across both levels of intent.

We develop novel algorithms for environment design in three human-robot collaboration settings. First, a bilevel optimization approach segments and schedules assembly tasks while arranging parts on a kitting tray, enabling just-in-time delivery that reduces makespan and improves user-perceived efficiency in a furniture-assembly user study and shop-flow simulations. Second, a bilevel active-preference-learning framework jointly optimizes environment parameters and counterfactual trajectory pairs to elicit the human's reward function; across navigation and tabletop domains, it achieves higher reward-estimation accuracy with fewer queries and lower reported cognitive workload than state-of-the-art baselines. Third, a quality-diversity search arranges physical objects and projects virtual obstacles via augmented reality to elicit more legible human motion; the resulting workspaces improve goal-prediction accuracy and data efficiency for both a Bayesian predictor with learned cost functions and a time-series multivariate Gaussian model.

Together, these contributions demonstrate that proactive environment design is a powerful and broadly applicable lever for fluent human-robot collaboration, complementing advances in human-intent inference algorithms. We conclude by discussing how environment design can be extended to improve not only the robot's understanding of the human, but also the human's

understanding of the robot, advancing toward truly bidirectional and trustworthy HRC.

Dedication

Acknowledgements

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

1.1 Motivation: Robot Chef Collaborator

Consider a robot assisting a person with meal preparation. The robot must infer the human's immediate motion to plan safe, collision-free paths in the shared workspace, while also predicting higher-level intentions to decide when to bring ingredients, utensils, or seasoning. Beyond task structure, individual preferences further shape the interaction: some users may prefer the robot to help stir the pot, while others might want it to handle cleanup. These considerations span three levels of intent: (1) **low-level motion**, (2)**high-level task structure**, and (3) **high-level preferences** and highlight a common requirement: effective collaboration depends on a shared mental model between human and robot [76]. Without such a model, even a technically capable robot may appear clumsy, uncooperative, or unsafe. Building this shared understanding requires the robot to continually infer what the human is doing, what they will do next, and why.

1.2 The Environment as a Design Variable

Most prior work in human-robot collaboration improves intent inference by developing more sophisticated models or collecting richer data: advances in motion prediction [68], preference learning [11], and human behavior modeling. These approaches assume that the **environment**, where the interaction takes place, is fixed.

This thesis challenges that assumption. We argue that the environment is itself a first-class design variable — one that the robot can actively configure to improve inference, coordination, and

preference alignment. We refer to this paradigm as **environment design** (ED).

Environment design has been extensively studied in purely robotic settings, where it is used to generate training curricula for reinforcement learning agents [16, 87], systematically evaluate policy robustness [23], and reduce sim-to-real discrepancies in manipulation. Extending ED to human-robot interaction, however, introduces fundamentally new challenges:

- Human intent is uncertain and context-dependent.
- Interaction is bidirectional: the environment influences the human's behavior.
- Environmental structure directly affects legibility, trust, and comfort in ways that are difficult to model explicitly yet critical for effective collaboration.

1.3 Research Questions

This thesis investigates how environment design can improve human-robot collaboration through two central research questions:

RQ1. What aspects of the environment can the robot modify to improve task performance and fluency in human-robot interaction?

RQ2. How can the robot efficiently search and optimize over the high-dimensional space of environment configurations?

RQ1 focuses on identifying which aspects of the environment, such as object placement, spatial constraints, and virtual guidance, effectively shape human behavior and reduce ambiguity. RQ2 addresses the computational challenge of selecting such environments, requiring scalable optimization methods that balance performance and diversity.

1.4 Two Levels of Human Intent

We organize the thesis around the two levels of intent that a robot collaborator must infer:

High-level intent: task structure and reward. What is the human trying to accomplish and how? Chapter 2 assumes known task structure and investigates how environment design through joint optimization of task segmentation, action scheduling, and kit layout can align the robot's assistance with the human's execution, ensuring timely, predictable, and fluent collaboration. Chapter 3 treats the human's preferences as a latent reward function and infers it via preference learning. In this setting, the robot leverages environment design to actively generate informative queries, enabling efficient inference of the human's underlying objectives.

Low-level intent: motion. What is the human physically doing? Chapter 4 shows that by arranging physical objects and projecting virtual obstacles via augmented reality, a robot can elicit more legible human motion, thereby reducing the uncertainty of learned goal-prediction models.

1.5 Contributions

The contributions of this thesis are:

(1) **A bilevel optimization framework for just-in-time robotic kitting** (Chapter 2). We formulate kitting as a joint environment design and coordination problem, where the upper level segments and schedules assembly tasks and the lower level optimizes the spatial layout of parts on a kitting tray. We evaluate the framework in an 18-participant flat-pack furniture assembly study and in shop-floor simulations under logistic delays, demonstrating reduced makespan and improved perceived efficiency compared to fixed kitting strategies.

(2) **A bilevel active-preference-learning framework with counterfactual reasoning and environment design** (Chapter 3). We introduce CRED, which jointly optimizes environment parameters (outer loop, Bayesian optimization) and informative trajectory queries generated from counterfactual reward samples (inner loop). Across simulation domains, including Lunar Lander, tabletop manipulation, and navigation, and in a 25-participant user study, CRED achieves higher reward accuracy with fewer queries and lower reported mental

workload than state-of-the-art baselines.

(3) **A quality-diversity approach to workspace optimization for human motion prediction** (Chapter 4). Using MAP-Elites, we design environments by rearranging physical objects and projecting virtual obstacles via augmented reality to elicit more legible human motion. We validate this approach in a 2D navigation study (20 participants) and a tabletop manipulation study (12 participants), showing improved goal-prediction accuracy and data efficiency across multiple classes of prediction models.

Chapter **??** synthesizes these results, examines the limitations of purely objective metrics for evaluating fluent human-robot collaboration, and outlines a path toward environment design that supports not only the robot's understanding of the human, but also the human's understanding of the robot.

Chapter 2

Just-in-Time Robotic Kitting via Bilevel Optimization

Chapter overview. We return to the meal-preparation scenario introduced in Chapter 1: before the chef begins, ingredients and tools can be organized to reflect the structure of the recipe, supporting a shared understanding of the task between the human and the robot. In a factory analogue (flat-pack furniture assembly), we show that a robot can take on this role, not by following a fixed script, but by adaptively segmenting the assembly into **just-in-time** kits [80].

We formulate this problem as a bilevel optimization, where the robot jointly decides which parts to deliver, when to deliver them, and how to arrange them spatially on the kitting tray. By designing both the task decomposition and the physical environment, the robot improves coordination and ensures that assistance is timely, predictable, and aligned with the human's workflow.

This bilevel structure serves as a template for later chapters. In Chapter 3, we apply a similar formulation to a different latent quantity—the human's reward function—using environment design to elicit preferences rather than to structure task execution.

2.1 Introduction

In the conventional or "single model" assembly line, only one product type is manufactured at a time. While this traditional manufacturing approach simplifies the responsibilities of line workers, it limits flexibility: making one product at a time means costly changeovers when switching product types, making it difficult to respond fluidly to changing dynamics of customer demand, upstream part shortages, and customizations. For this reason, manufacturers may employ *"mixed model"*

assembly, in which multiple products — with different parts and assembly steps — are assembled on the same line. This approach smooths upstream part demand and can lead to much greater overall productivity in the assembly line [98].

The trade-off of mixed model assembly is that it requires assembly line workers to take on a greater range of tasks. Rather than constantly performing a rote series of assembly steps, workers now need to keep track of the various parts and steps for each product on the line. One approach to streamlining this step at the worker level is *kitting*. In manufacturing systems, kitting refers to the process of collecting components in a "kit" or container before feeding them to workstations where intermediate or end products are built. Kitting possesses a number of advantages: it 1) reduces storage space requirements at the workstations, 2) enables efficient product changeover (since common components are stored at a central location), and 3) increases worker productivity by reducing time spent gathering parts [6, 21].

Kitting is an excellent opportunity space for leveraging collaborative robotics to improve the efficiency of workers. While automation has seen great success in many areas of manufacturing, assembly tasks often remain too complex to automate completely: robots lack the fine-motor capabilities required for dexterous manipulation of small parts and tools. Kitting, however, is well within the mechanical capabilities of modern robots, and its automation would allow a robot to prepare kits while more dexterous agents assemble the product. This parallelism could greatly decrease the time and effort required to complete each product assembly.

In this work we propose and evaluate a dynamic robotic kitting planner that segments and

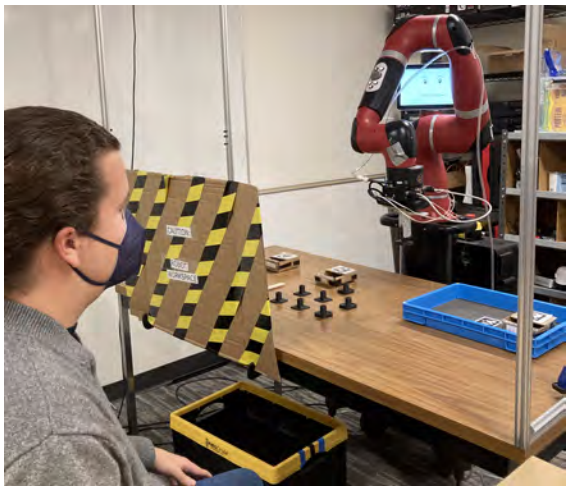


Figure 2.1: This work presents an optimization approach to automate just-in-time robot part delivery for collaborative assembly. A robot gathers parts required for an assembly task and places them on a kitting tray. The human then uses this "kit" to complete the next set of tasks.

schedules assembly tasks to minimize idle time and
reduce makespan. The primary contributions of this
chapter are: (a) a novel optimization algorithm for just-in-time robotic kitting, which incorporates a model of the assembly goal, estimates of human and robot task times, and estimated usability of the generated kit layout; (b) a demonstration of the optimization and its performance outcomes on an experimental human-robot collaboration scenario; and (c) an analysis of the optimization's robustness to logistic delays in simulation.

We formulate just-in-time kitting and delivery as a *bilevel* optimization, where the "upper-level" problem of task scheduling and segmentation is optimized in concert with the "lower-level" problem of part kitting. By performing this nested optimization, the robot adapts to its kitting conditions (e.g., kit arrangement and part availability) while syncing its workflow to that of the human.

To evaluate our kitting approach's ability to increase efficiency and improve users' experience in a human-robot collaborative assembly, we implement our system on a Sawyer robot (Figure 2.1) and conduct a within-subjects user study of collaborative furniture assembly. To evaluate the system's performance in scenarios with logistic delays, we use data from the user study to simulate the assembly process under varying conditions of part availability and throughput to explore our system's robustness.

2.2 Background and Related Work

One of the disadvantages of kitting is that the preparation of kits consumes worker time and effort that don't directly contribute to the assembly task. Kitting in mixed-model assembly lines also has higher rates of error [35] and assembly workers report higher cognitive loads [36, 71].

Due to the physical and cognitive demands of manual kit preparation, robotic kitting has been explored to increase efficiency and flexibility. Caputo et al. [9] developed an economic model to show that automated systems are more cost effective and efficient, except in low-production settings. Boudella et al. [5] developed a delivery-time-sensitive model to optimally distribute kitting

tasks between a human and robot in a mixed-model assembly line, and saw significant efficiency improvements over baselines where the human worker took on the majority of the tasks. A mixed-integer linear programming approach to online kit delivery scheduling is presented in [48], in which the robot takes over all kitting tasks and results in large productivity improvements over a manual kitting baseline.

Our work differs from the above examples in that it presents a fully-automated kitting system that achieves Just in Time (JIT) part feeding, a lean manufacturing principle designed to eliminate waste in production [21]. The goal of JIT manufacturing is to feed the required quantity of parts to the workstations when they are needed, rather than before or after. Traditionally, JIT kitting is accomplished by determining the required kits at each workstation and designing the assembly layouts and kit sizes accordingly [79]. Other production systems use material handling techniques like kitting trolleys [56] or automated delivery vehicles [97] to achieve JIT.

Several optimization approaches have been proposed to enable timely human-robot or robot-robot interactions. An auction algorithm for solving task allocation with temporal constraints is presented in [59], and a non-linear program for flexible scheduling of robot tasks is presented in [90]. In addition to optimizing for efficiency, temporal considerations have been shown to enable fluent human-robot teaming in [50]. Our work differs from the above by using bilevel optimization to minimize idle times and also optimize for part selection and kit arrangement.

In this work, we use task segmentation and scheduling of kitting tasks to achieve JIT delivery. By incorporating an awareness of human and robot task times, as well as part availability into kit layout and design, our system is more capable of adapting its kitting behavior to meet the changing demands of mixed-model assembly lines.

2.3 Problem Formulation

Consider a two-station, human-robot collaborative task where a human assembler works at an assembly workstation and a robot prepares kits at a preparation workstation. The robot receives

part deliveries and produces kits, while the human receives kits and produces product. Let:

$$\mathcal{A} = \{a_1, a_2, \ldots a_n\}$$

be the set of n assembly tasks that need to be performed to complete the assembly. Each task a_i is composed of the robot-dependent kitting task a_i^R, in which the robot gathers the necessary parts, and the human-dependent assembly task a_i^H. The robot has to complete preparation task a_i^R (i.e., kitting and delivering components) before the human can begin a_i^H (assembling). In order to schedule its actions, the robot's goal is to compute a *task ordering* over $\mathcal{A}$ for itself and the human to follow in order to finish the assembly as quickly as possible.

(a) Human task (b) Robot task

Figure 2.2: A task graph for assembly of a stool. The robot retrieves the parts required for the human to complete the assembly. The arrows indicate precedence constraints. The robot tasks are represented by $T^R = \{a_1^R, a_2^R, a_3^R\}$ and the human tasks are represented by $T^H = \{a_1^H, a_2^H, a_3^H\}$.

The tasks in $\mathcal{A}$ may not need to be completed in a strict order, but some tasks may have ordering constraints — for example, one may need to complete "insert screws to top" before one can perform "insert front leg" to build a table. Let us say that if task a_i must be completed before a_j, then a_i has a *precedence constraint* with a_j. Any task ordering the robot computes must satisfy these precedence constraints. See Figure 2.2 for an example task precedence graph, and the associated part retrievals the robot needs to perform. Our proposed approach is generalizable to tasks with precedence constraints as described above.

To create a *schedule* to complete each task given a task ordering, the robot *segments* tasks into groups. This means the robot can prepare a kit one segment at a time, rather than having to make one kit for each of the human's tasks. This also introduces the additional constraint that

a segment can only be valid if all the associated parts can fit in the kit. Let us represent a task segment at time step t as $T_t \subseteq \mathcal{A}$.

Once the tasks have been segmented into groups, the robot generates a schedule by determining when to start each segment. For this, it uses its prior knowledge for how long a given segment will take, both for itself and the human; let us represent this as $duration^H(T_t)$ and $duration^R(T_t)$ respectively. For the task in Figure 2.2, one possible segmentation is $T = \{T_1, T_2\} = \{\{a_1, a_2\}, \{a_3\}\}$ and a possible schedule is $S = \{0\text{s}, 35\text{s}\}$.

Lastly, the robot must arrange all of the parts for the current segment T_t into the kit — that is, all of the parts required for each task $a \in T_t$.

2.4 Proposed Approach

We formulate our solution as a bilevel optimization, a hierarchical constrained optimization where one problem is embedded in another (Figure 2.3). The general formulation of a bilevel optimization problem is:

$$\min_{x \in X, y \in Y} F(x, y) \quad s.t. \quad G(x, y) \leq 0 \ , \tag{2.1}$$

$$y \in \arg\min_{z \in Y} \{f(x, z) \quad s.t. \quad g(x, z) \leq 0\} \tag{2.2}$$

where F is the upper-level objective function, G represents the upper-level constraints, f is the lower-level objective function, and g represents the lower-level constraints [13]. In our approach, the upper-level problem Eq. 2.1 determines the task segmentation and schedule, and the lower-level problem Eq. 2.2 determines the physical arrangement of parts in the kit.

2.4.1 Task Segmentation and Scheduling

We define our notation as follows: $K = \{a_1, a_2, ...\}$ is an ordered sequence of future tasks to execute (where a_1 represents both robot and human tasks a_1^R and a_1^H), i is an index partitioning K to determine the tasks to be included in the next kit, T is a time-ordered sequence of task segments as defined above, $t \in [1, |T|]$ is the current task segment being completed, P is a kit layout defined

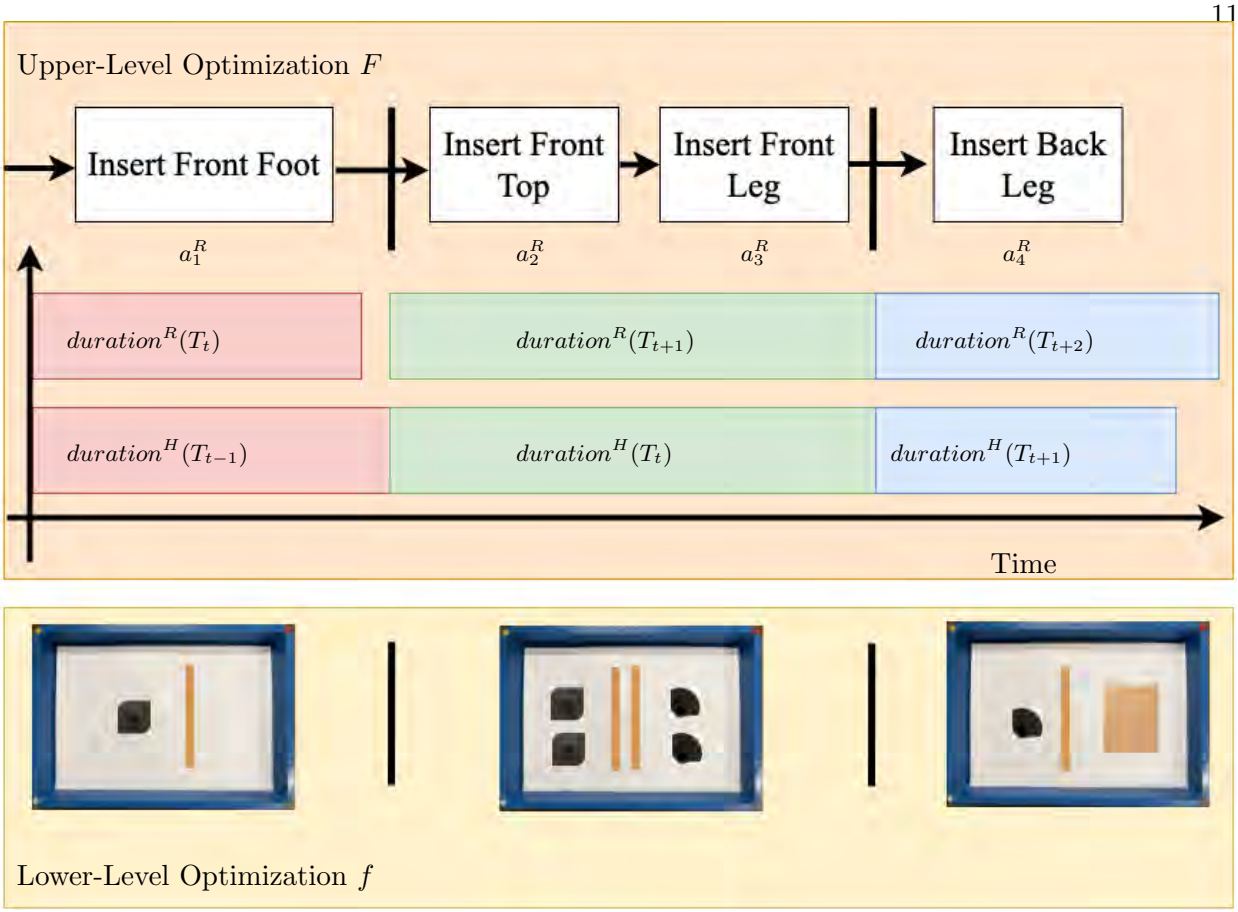


Figure 2.3: Overview of the bilevel optimization algorithm for just in time kitting. The upper level optimization segments and schedules tasks, and the lower level optimization arranges parts on the tray. By minimizing the differences between $duration^R(T_{t+1})$ and $duration^H(T_t)$ (shown in green) and the differences between $duration^R(T_{t+2})$ and $duration^H(T_{t+1})$ (shown in blue) as well as other terms in the objective function F, the robot determined the task segment T_{t+1} which corresponds to the set of tasks *Insert Front Top* (a_2^R) and *Insert Front Leg* (a_3^R). The task segment T_{t+1} also depended on the kit fitness function f which evaluates the kit layout given the parts required P for tasks in T_{t+1}. The robot's role is to prepare parts required for a_2^R and a_3^R in a kit according to the layout determined by f at timestep $t+1$.

by a set of part coordinates (part id, x, y) in a kitting tray, G is the task graph encoding precedence constraints for the overall task, d is a scalar constant indicating the time required for a kit delivery, and W_k is a scalar weight hyperparameter for the kth objective function component. We utilize two functions within the objective, **allowed(task, set of completed tasks, task precedence graph)** which returns 1 if **task** can be completed given the **set of completed tasks** and the **task precedence graph** (0 otherwise) and **kit_fitness(kit layout)** which evaluates a **kit layout** for

usability (returning a scalar where larger is better).

The optimization for timestep $t+1$ is defined as follows:

$$\min_{K,i,P} W_1 \sum_{j=1}^{|K|}(1 - \text{allowed}(K_j, \bigcup_{x=1}^{t} T_x \cup \bigcup_{x=1}^{j-1} K_x, G)) \tag{2.3}$$

$$- W_2 * i \tag{2.4}$$

$$+ W_3(d + \sum_{j=1}^{i} \text{duration}^R(K_j) - \text{duration}^H(T_t)) \tag{2.5}$$

$$+ W_4(d + \sum_{j=i+1}^{|K|} \text{duration}^R(K_j) - \sum_{j=1}^{i} \text{duration}^H(K_j)) \tag{2.6}$$

$$- W_5 * \text{kit\_fitness}(P) \tag{2.7}$$

The first term (2.3) allows us to heavily penalize proposed task sequences K that do not satisfy precedence constraints. The second term (2.4) can be used to override other terms to prioritize kits covering more tasks. The third term (2.5) penalizes differences between the human's remaining task time and the duration required to prepare and deliver the proposed next kit. The fourth term (2.6) is similar, penalizing partitions that are likely to lead to idle times after T_{t+1}. Finally, the fifth term (2.7) incentivizes kit layouts with high utility, which is solved for by the second level of optimization given a proposed K and i.

After solving for values of K, i, P, we set task segment $T_{t+1} = K_{1:i}$ and increment t. This optimization is run on a greedy basis at each timestep while the robot executes its kitting behaviors. Without loss of generality, i can be defined as a vector of indices that create $|i| + 1$ task segments $T_t = K_{0:i[0]}$, $T_{t+1} = K_{i[0]:i[1]}$, $\ldots$, $T_{t+|i|} = K_{i[|i|-1]:|K|}$. Terms (2.5, 2.6) would be updated to minimize the differences between human and robot task times for each segment.

The proposed task segmentation and scheduling solution for JIT kitting is an instance of flow-shop scheduling, which has been shown to be NP-hard [19, 28]. We approximate the optimal solution by only considering a limited horizon and partial task plans K of fixed length up to N ($N = 5$ in our experiments).

2.4.2 Kit Arrangement

The nested lower-level objective function **kit_fitness** is defined to score the arrangement of parts for a given task segment on the tray used for kit delivery. This task segment T_{t+1} is given by K and i as defined in the previous section.

Let P denote the set of parts required for the tasks in T_{t+1} and their coordinates on the 2D plane of the kitting tray. Let $\mathbf{P_{j,x}}$ and $\mathbf{P_{j,y}}$ denote the x and y coordinates of the jth part's centroid in the kit. These coordinates are solved for with the lower optimization, with an objective function that maximizes kit fitness. We first define terms of the **kit_fitness** objective to prioritize logical grouping of items on the tray:

$$D_{diff}(P) = \sum_{k=1}^{|P|} \sum_{\substack{j=k+1: \\ Q(P_j) \neq Q(P_k)}}^{|P|} \sqrt{(P_{k,x} - P_{j,x})^2 + (P_{k,y} - P_{j,y})^2} \tag{2.8}$$

$$D_{same}(P) = \sum_{k=1}^{|P|} \sum_{\substack{j=k+1: \\ Q(P_j) = Q(P_k)}}^{|P|} \sqrt{(P_{k,x} - P_{j,x})^2 + (P_{k,y} - P_{j,y})^2} \tag{2.9}$$

where $Q(p)$ denotes the *part type* of part p, such as "M6 screw" or "connector." Equation 2.8 sums the Euclidean distance between all parts of different types, while 2.9 does the same for all parts of the same type.

We also define a term to penalize overlap between items on a tray, as we wish to discourage the robot from piling objects on top of one another. Let $BB(p, x, y)$ denote the *bounding box* of part p centered at point (x, y). We define Z as the total overlapping area between the bounding boxes of all parts:

$$Z(P) = \sum_{k=1}^{|P|} \sum_{j=k+1}^{|P|} BB(P_k, P_{k,x}, P_{k,y}) \cap BB(P_j, P_{j,x}, P_{j,y}) \tag{2.10}$$

Lastly, we use the terms described above to define our kit fitness objective function, with terms K, i defined by the upper-level optimization and W_6 a scalar hyperparameter to modulate the optimization's focus on discouraging part overlap:

$$\min_P D_{same}(P) - D_{diff}(P) - W_6 * Z(P) \tag{2.11}$$

$$\text{subject to: } \bigcup_{k=1}^{|P|} BB(P_k, P_{k,x}, P_{k,y}) \subset \text{area}_{kit}$$
$$\bigcup_{j=1}^{i} \text{parts\_in}(K_j) \in P$$

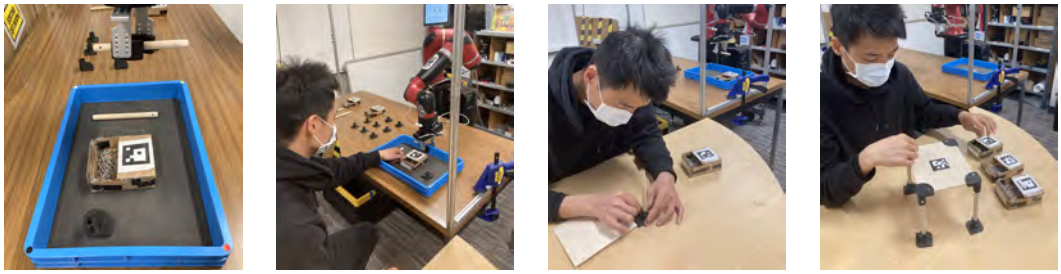
The objective function maximizes distances of parts with different types, minimizes distances of parts with the same type, and penalizes overlaps. The constraint ensures that the 2D space taken up by parts in the kit does not exceed or extend beyond the tray area and that all necessary parts are included in the kit.

We use the cross entropy (CE) method to solve the kit arrangement problem. CE is a stochastic algorithm that provides an adaptive procedure for solving combinatorial optimization problems and has asymptotic convergence properties [15]. Other stochastic methods such as simulated annealing [85] and genetic algorithms [30] can also be used.

Algorithm 1 Cross Entropy Method for kit arrangement

Require: list of parts P, sample count to keep c
  initialize $\mu(P_{k,x}, P_{k,y}, \theta_i \; \forall \; k \in 1 : |P|)$, Σ randomly
  while not converged and max iterations not reached **do**
    $samples \sim \mathcal{N}(\mu, \Sigma)$
    $scores \leftarrow sorted(score(samples))$ ▷ score samples and sort from largest to smallest
    $\mu \leftarrow mean(samples[:c])$
    $\Sigma \leftarrow covariance(samples[:c])$ ▷ recompute mean and covariance from new samples
  end while
  return μ

Algorithm 1 shows the implementation for CE. We first randomly initialize the mean vector and covariance matrix. The mean vector contains the position (x and y) and orientation (θ) of the parts that have to be placed on the tray. While the solution hasn't converged or the maximum number of iterations (100) has not been reached, we draw samples from a Gaussian distribution (200 in our experiments) with the current estimated mean and covariance. The samples are scored using Equation 2.11 and sorted from largest to smallest. Then we use the top c samples to recompute the mean and covariance ($c = 30$ in our experiments). The algorithm returns the estimated mean after convergence or after the maximum number of iterations.



(a) (b) (c) (d)

Figure 2.4: (a) The robot arranges parts on the tray according to the lower-level optimization. (b) The robot delivers a finished kit to the user. (c) The user begins the assembly task while the robot prepares the kit for the next tasks. (d) The user finishes assembling a flat-pack furniture table.

2.5 Experimental Validation

With the goal of evaluating our proposed system under a variety of logistic delay scenarios, we first perform a user study to collect user experience data. We then use this study as a basis to perform a series of simulated shop floor evaluations of the robot's kitting delivery performance under multiple conditions varying supply failure likelihood and delivery distance between kitting and assembly stations.

2.5.1 User Study

We had 19 people (10 males, 9 females) participate in our IRB-approved study, ranging in age from 22 to 31 ($M = 25.47$, $SD = 2.37$). One data point was discarded because the Sawyer robot lost connection to the study computer in the middle of the experiment. Participants were tasked with the construction of a flat-pack furniture table [93] shown in Figure 2.4. Table assembly involves connecting four feet, four legs, four connectors, and a flat surface plank by snapping the pieces together and securing with screws and nuts.

Experimental Design. We use a within-subjects experiment to evaluate our optimization approach and two baseline conditions: (a) *Whole Assembly* is a baseline which resembles traditional kitting strategies where all the required parts for a single unit are placed on the kitting tray prior to assembly, (b) *Single Task* is a human-designed JIT schedule where the robot delivers parts for a

single task at a time as segmented by the task graph itself (one kit per vertex), and (c) *Optimized* is the result of applying the proposed bilevel optimization algorithm to the assembly task. To minimize learning effects, we counterbalance the condition orderings and ask the participants to practice assembling part of the table before the experiment.

The assembly task is divided into a total of 12 tasks, i.e., 3 tasks per leg of the table. The precedence constraints dictate that the feet and connecting joint have to be secured to the leg before the leg can be secured to the plank. The robot's tasks include delivering the screws, nuts, four legs, four feet, and four joints to the human participant via a kitting tray. The human participant and the robot have separate workspaces, and the human participant is instructed to not enter the robot's workspace during the experiment. Figure 2.1 shows the experiment setup.

Metrics. The objective metrics used to assess the efficiency of our proposed approach are the total task time and the human idle time. The *total task time* is the total experiment time for assembling a single table. The *human idle time* is defined as the duration of the experiment in which the participant was not actively assembling parts. A post-experiment survey was used to assess the user experience: participants were asked to rank each trial based on the robot's Usefulness, Intuitiveness, and Efficiency during that trial.

Hypotheses.

- **H1:** The *Optimized* condition will have a makespan lower-bounding *Single Task* and *Whole Assembly* conditions with reduced idle time that is proportional to delivery duration.
- **H2:** Participants will perceive the *Optimized* condition more highly along subjective measures (usefulness, intuitiveness, and efficiency) than the *Single Task* and *Whole Assembly* conditions.

Study Protocol. After a brief introduction to the robot's role in part delivery, the human participant was presented with a printed guide document, which included step-by-step instructions in both written and visual form. The participant was asked to practice constructing a single leg of the table before the experiment began.

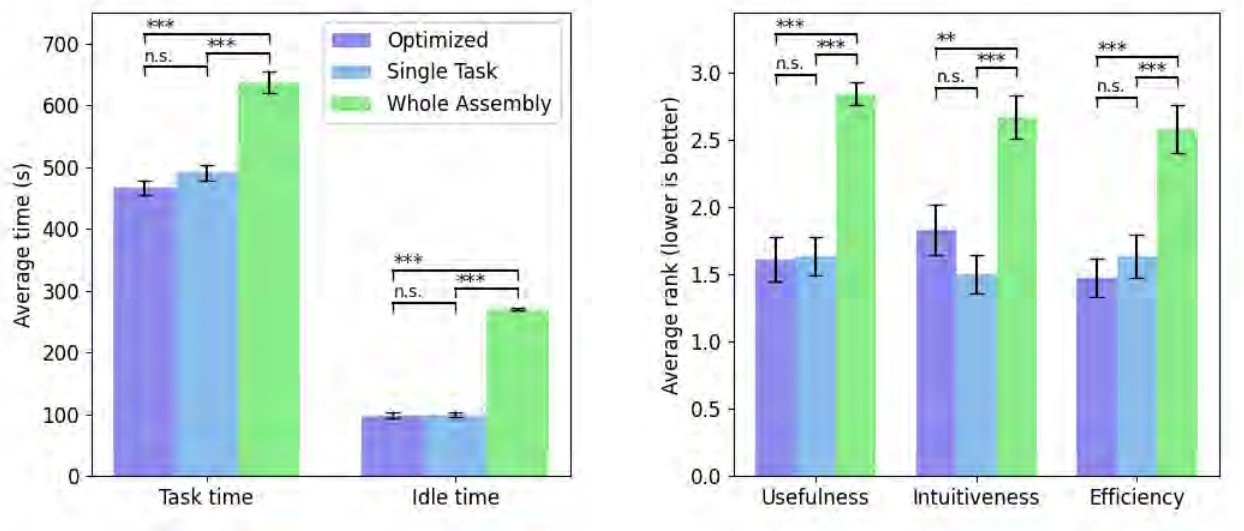


Figure 2.5: ***Left:*** Total task time and human idle time across all three conditions. Both the *Optimized* and *Single Task* approaches had significantly lower total times in both metrics than in the *Whole Assembly* condition. ***Right:*** Total user ranking scores for each condition across three subjective metrics: usefulness, intuitiveness, and efficiency. The *Whole Assembly* condition had significantly worse scores than the algorithmic conditions across all three rankings. The *Single Task* and *Optimized* conditions did not vary significantly on any of the above metrics, likely due to the proximity of the robot to the human (no delivery delays). ** indicates $p < .001$; *** indicates $p < .0001$.

During the experiment, the required parts were initially placed in the robot workspace. The robot kitted the assembly parts by placing some number of them onto a tray, and then pushing the tray towards the human. The participant was then notified to retrieve the parts on the tray. The robot returned to its kitting task once the participant retrieved the parts. The same assembly was repeated for all three conditions.

Implementation Details. We implemented the optimization framework on a Sawyer Research Robot, using the Robot Operating System (ROS) [39]. We installed an Intel RealSense D435 Depth Camera above the robot workspace and used the Facebook Detectron2 algorithm [92] for image segmentation of the assembly parts. We used the ArUco library [29] to detect markers placed on the screw boxes and tray to read their pose.

2.5.2 Shop Floor Simulation

While the user study provides useful insights for user preferences and task efficiency for a single assembly, we use discrete event simulation (DES) to evaluate our proposed approach when multiple assemblies of the furniture table are performed in sequence. We evaluate the framework under various assembly part arrival time distributions and part-feeding machine breakdown conditions (i.e., the part is not available to the robot until after the machine is repaired). DES is able to explicitly model events using probability distributions and answers questions such as the line throughput and utilization [65].

Experimental Design. We simulated the assembly of ten flat-pack furniture tables. For each assembly, we randomly sampled a participant's task times and the robot's task times as recorded from the user study detailed in Section 2.5.1. The arrival time of assembly parts to the robot's workspace is modeled by an exponential distribution with Mean Arrival Time denoted by MAT. In other words, the arrival of parts is a Poisson process with rate $1/MAT$. Since increasing the arrival times of all the parts linearly increases the total task times, we only vary the MAT of the leg and foot part types. We also model the machine breakdown for the feeding of the leg and foot parts with an exponential distribution. The mean time to failure of the machines is denoted by $MTTF$. Each machine has a fixed repair time of 30 seconds, after which it starts feeding parts again according to the part arrival process with rate $1/MAT$.

Hypothesis.

- **H3:** When simulating multiple assemblies of the task, the *Optimized* condition will significantly outperform baseline kitting strategies in both total task time and human idle time in the presence of logistical delays.

Implementation Details. The simulation is implemented in Python using the SimPy library. In order to account for assembly part shortages due to slow arrival times or machine

breakdowns, we add an additional term to the upper-level objective function:

$$W_7 \sum_{k=1}^{|P|} U(P_k) \tag{2.12}$$

where $U(p)$ returns 1 if part p is currently unavailable for the robot to place on the tray and 0 otherwise.

2.6 Results

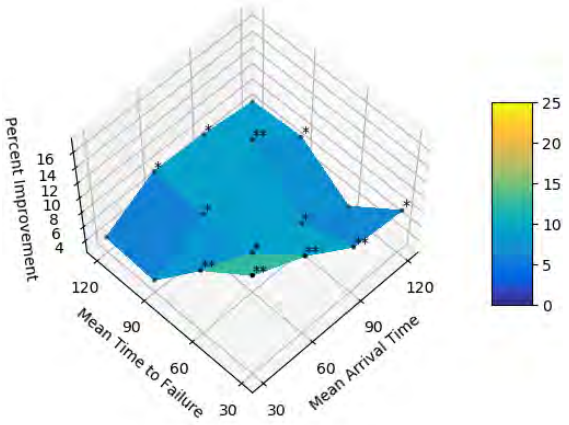
2.6.1 User Study

Objective measures. Total task time and human idle time measures were analyzed using the one-way analysis of variances (ANOVA) with experimental condition as an independent variable. Post-hoc analysis used Tukey's HSD test for multiple comparisons to test for effects between condition pairs. The results are summarized in Figure 2.5.

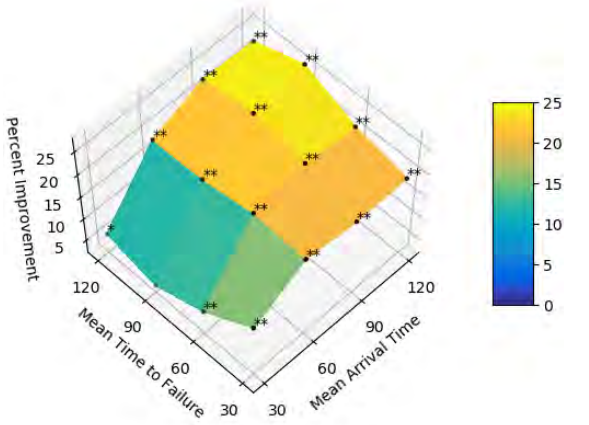
Total task time. The effect of condition significantly influenced total task time, $F(2, 34) = 57.60, p < .0001$, with a significant reduction in task time between the *Whole Assembly* condition and the *Single Task* condition ($p < .0001$, 95% C.I. $= [-188.211, -103.851]$), as well as between the *Whole Assembly* and *Optimized* conditions ($p < .0001$, 95% C.I. $= [-213.213, -128.853]$).

Human idle time. The effect of experimental condition significantly influenced human idle time, $F(2, 34) = 709.87, p < .0001$; there was a significant reduction in idle time between the *Whole Assembly* and *Single Task* conditions ($p < .0001$, 95% C.I. $= [-184.465, -158.586]$), as well as between the *Whole Assembly* and *Optimized* conditions ($p < .0001$, 95% C.I. $= [-186.020, -160.141]$).

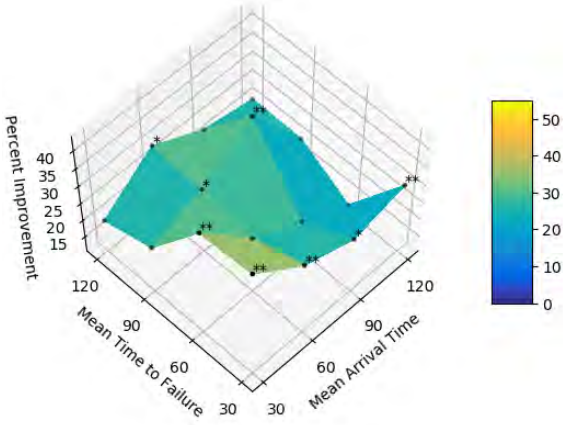
Subjective measures. We analyzed rank scale data on our post-experiment survey using a nonparametric Kruskal-Wallis Test with experimental condition as a fixed effect. Post-hoc comparisons used the Wilcoxon method for analyzing rank significance between condition pairs. The results are summarized in Figure 2.5.



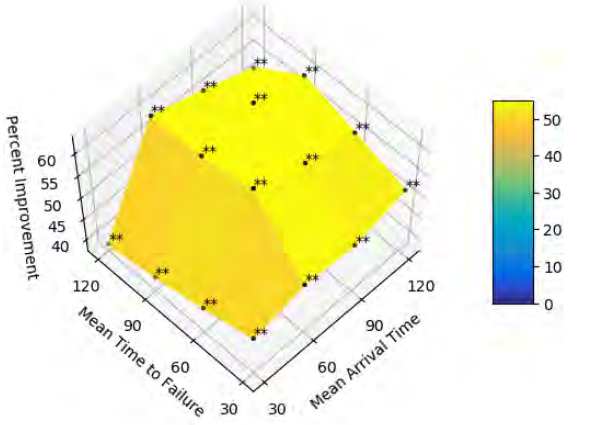
(a) Percent improvement in total task time of *Optimized* over *Single Task*.



(b) Percent improvement in total task time of *Optimized* over *Whole Assembly*.



(c) Percent improvement in human idle time of *Optimized* over *Single Task*.



(d) Percent improvement in human idle time of *Optimized* over *Whole Assembly*.

Figure 2.6: Comparisons of Optimized approach with baselines in simulation. (a) and (b) show the percent improvement in total task time of the *Optimized* approach over the *Single Task* and *Whole Assembly* conditions respectively. With low part shortage (i.e., low *MAT*), *Optimized* is most advantageous over *Single Task* because *Single Task* has many more tray deliveries which increase the total robot task time. With high part shortage (i.e., high *MAT*), *Optimized* is most advantageous over *Whole Assembly* because *Optimized* adapts the kitting strategy according to which parts are available. (c) and (d) show the percent improvement in human idle time of the *Optimized* approach over the *Single Task* and *Whole Assembly* conditions respectively. The trend for human idle time is similar to that of total task time. * denotes $p < .05$ while ** denotes $p < .001$.

2.6.2 Simulation

In simulation, we compare the task time and human idle time outcomes between *fixed* strategies (*Single Task* and *Whole Assembly*) and the *Optimized* strategy. Percent improvements in task time and human idle time between conditions are summarized in Figure 2.6. Detailed tables appear below (Tables 2.1 and 2.2). Results were analyzed using the same methods as in Section 2.6.1.

2.7 Discussion

Hypothesis **H1** regarding task time and idle time is supported by the user study results. We suspect that *Optimized* is similar to *Single Task* due to the high variance in the participants' task times and the absence of delivery distance (the human and robot were co-located across a table). The average human task time (i.e., non-idle time) for a single assembly and across all conditions was 374.78 seconds with a standard deviation of 61.14 seconds.

Hypothesis **H2** is supported; the *Optimized* condition had significantly lower ranking scores (i.e., were ranked better) in usefulness, intuitiveness, and efficiency than *Whole Assembly*. However, there was no significant difference in rankings of *Single Task* and *Optimized* on any of the three subjective metrics (all $p \geq .05$). From the post-experiment interviews, participants found the *Optimized* or *Single Task* conditions most efficient because *"it gave me the parts in a timely manner"* and *"[it has] less waiting time."* This suggests that participants preferred just-in-time kitting strategies.

Some participants indicated they found the just-in-time approaches less cognitively demanding: *"[The robot is] guiding me through the most efficient way to complete the assembly job . . . the robot lifted part of my pressure to plan it."*

Hypothesis **H3** is supported; the total task time and human idle time are significantly shorter for *Optimized* than either *Single Task* or *Whole Assembly* for most of the scenarios with logistic delays. *Optimized* is most advantageous over *Whole Assembly* when there is a high part shortage (high *MAT*) because *Whole Assembly* waits for all the parts to be kitted before delivering the kit

to the assembly station. When a part is delayed, *Whole Assembly* waits until the part becomes available. While *Single Task* also suffers from logistic delays, the effect is less prominent because kits are delivered with partial assembly parts. The *Optimized* approach is able to dynamically determine which tasks to deliver parts for in order to minimize idle times and maximize efficiency.

2.8 Chapter Summary and Future Work

This chapter introduced a bilevel optimization approach for robot kitting and demonstrated its ability to reduce both overall task time and human and robot idle times for a furniture assembly task. In a user study, we evaluated this approach against a generic whole-kit assembly (*Whole Assembly*) and a human-designed just in time approach (*Single Task*), and found that just-in-time kitting had quicker task completion and was rated more highly by users on a number of subjective metrics. Simulating longer and more varied task environments revealed that the online optimized approach demonstrates significant performance improvement over the fixed kitting strategies in more realistic settings that include logistic delays.

The user study suggested two additional avenues of future research: online estimates of human task time to better minimize idle times, and the addition of human factors into the optimization framework. Lastly, we plan to investigate which factors best improve the efficiency and intuitiveness of kitting tray designs. One limitation of this work is the limited horizon used, which may not generate the most optimal kitting strategy with respect to the objective function; with more compute power or longer planning times, our approach can generate more optimal kits.

Transition to Chapter 3. Chapter 2 showed that environment design pays off when the robot knows what the human is trying to do but not how to schedule around them. In the next chapter, we tackle the harder case: the robot does not know the human's preferences at all, and must design environments that help the human **teach** their reward function efficiently. The bilevel structure we introduced here — outer loop over environment parameters, inner loop over a low-level object — carries over directly; only the variables change.

Table 2.1: Total task time of *Optimized* (top, **bold**) versus *Single Task* (middle) and *Whole Assembly* (bottom). Asterisks indicate significant differences between the indicated condition and the Optimized condition; * denotes $p < .05$ while ** denotes $p < .001$. Left half: total task time (s). Right half: human idle time (s).

		Mean Time to Failure				Mean Time to Failure			
		30	60	90	120	30	60	90	120
Mean Arrival Time	30	**4372.70 (129.04)**	**4105.70 (141.02)**	**4152.80 (151.47)**	**4022.70 (144.17)**	**1035.20 (128.66)**	**716.10 (70.19)**	**681.60 (79.10)**	**621.40 (130.06)**
		5257.40** (174.46)	4638.10** (164.43)	4331.30 (172.11)	4212.20 (111.98)	1780.10** (216.83)	1211.50** (273.77)	897.70 (227.64)	776.80 (102.62)
		5079.60** (173.86)	4433.30** (142.38)	4313.40 (164.69)	4300.70** (226.32)	1931.80** (288.81)	1262.10** (143.52)	1149.60** (240.44)	1016.40** (229.18)
	60	**4609.00 (124.66)**	**4251.00 (135.65)**	**4289.40 (190.34)**	**4282.80 (152.39)**	**1289.50 (128.77)**	**861.40 (157.48)**	**862.20 (239.49)**	**854.90 (170.41)**
		5326.20** (241.09)	4627.70** (135.26)	4651.20** (115.46)	4664.60** (192.96)	1881.50** (234.90)	1179.00 (168.68)	1217.70** (237.81)	1222.50** (249.50)
		5762.20** (205.53)	5370.30** (324.03)	5301.90** (295.83)	5283.00** (314.16)	2536.50** (226.92)	2251.50** (412.17)	2134.90** (301.54)	2107.70** (300.58)
	90	**4851.60 (277.47)**	**4611.10 (170.93)**	**4455.00 (216.52)**	**4587.60 (191.00)**	**1402.10 (321.81)**	**1144.30 (224.20)**	**1027.30 (253.25)**	**1202.50 (246.25)**
		5328.50** (182.15)	4929.50** (142.42)	5083.40** (152.22)	4980.40** (237.39)	1914.10** (276.84)	1425.50 (143.63)	1662.80** (188.22)	1571.10 (307.27)
		5981.70** (221.53)	5959.10** (353.97)	5897.30** (253.95)	5976.00** (364.24)	2728.20** (317.71)	2723.10** (370.88)	2707.20** (330.04)	2744.00** (513.12)
	120	**4839.30 (176.64)**	**4733.50 (212.87)**	**4764.10 (196.58)**	**5010.40 (294.95)**	**1396.20 (271.32)**	**1265.30 (215.11)**	**1321.50 (297.61)**	**1550.70 (329.21)**
		5275.50** (188.07)	4896.00 (237.56)	5155.00** (306.22)	5399.40 (365.42)	2005.80** (138.29)	1449.10 (246.05)	1670.90 (308.25)	1970.50 (412.39)
		5994.30** (332.51)	6043.90** (218.83)	6462.30** (226.59)	6510.90** (413.18)	2905.50** (418.61)	2859.30** (273.99)	3255.70** (341.55)	3279.80** (451.55)

Table 2.2: Human idle time of *Optimized* (top, **bold**) versus *Single Task* (middle) and *Whole Assembly* (bottom). Asterisks indicate significant differences between the indicated condition and the Optimized condition; * denotes $p < .05$ while ** denotes $p < .001$. Columns are Mean Time to Failure. Rows are grouped by Mean Arrival Time (30, 60, 90, 120).

Mean Time to Failure			
30	60	90	120
1035.20 (128.66)	**716.10 (70.19)**	**681.60 (79.10)**	**621.40 (130.06)**
1780.10** (216.83)	1211.50** (273.77)	897.70 (227.64)	776.80 (102.62)
1931.80** (288.81)	1262.10** (143.52)	1149.60** (240.44)	1016.40** (229.18)
1289.50 (128.77)	**861.40 (157.48)**	**862.20 (239.49)**	**854.90 (170.41)**
1881.50** (234.90)	1179.00 (168.68)	1217.70** (237.81)	1222.50** (249.50)
2536.50** (226.92)	2251.50** (412.17)	2134.90** (301.54)	2107.70** (300.58)
1402.10 (321.81)	**1144.30 (224.20)**	**1027.30 (253.25)**	**1202.50 (246.25)**
1914.10** (276.84)	1425.50 (143.63)	1662.80** (188.22)	1571.10 (307.27)
2728.20** (317.71)	2723.10** (370.88)	2707.20** (330.04)	2744.00** (513.12)
1396.20 (271.32)	**1265.30 (215.11)**	**1321.50 (297.61)**	**1550.70 (329.21)**
2005.80** (138.29)	1449.10 (246.05)	1670.90 (308.25)	1970.50 (412.39)
2905.50** (418.61)	2859.30** (273.99)	3255.70** (341.55)	3279.80** (451.55)

Chapter 3

Environment Design for Active Preference Learning

Chapter overview. In Chapter 2, the human's high-level goal was assumed to be known (i.e. assembling a table according to a given precedence graph). In practice, however, a robot must collaborate with users whose preferences are not specified in advance: one user may prefer assistance with stirring, while another may expect help with cleanup. In this chapter, we model the human's preferences as a latent reward function and learn it through active preference learning [83]. We show that the bilevel optimization structure introduced in Chapter 2 extends naturally to this setting. The outer loop designs environments that elicit informative feedback, while the inner loop generates counterfactual trajectory pairs under sampled reward hypotheses. Together, this enables the robot to efficiently infer the human's underlying objectives.

3.1 Introduction

Robots deployed in the real world must operate in diverse, unpredictable environments and interact with users whose preferences and expectations vary widely. Predefining appropriate behaviors for all possible scenarios is infeasible — robot behavior must instead be fine-tuned through human feedback to ensure alignment with user expectations. For example, in autonomous delivery, different robot embodiments (e.g., wheeled vs. legged robots) may need to learn which terrains are acceptable to traverse based on both physical capabilities and user-specified trade-offs between efficiency and risk [51]. In domestic settings, users may prefer that household robots complete tasks more quickly, even if it means compromising slightly on task quality, such as folding laundry

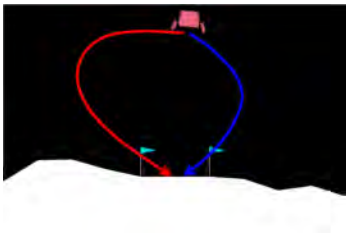
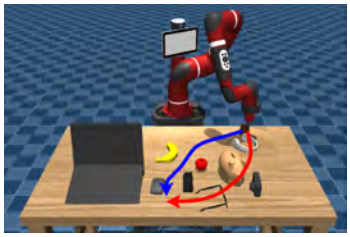
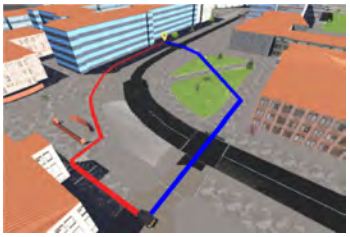
faster at the expense of producing perfectly neat folds. These types of trade-offs are often subtle, context-dependent, and difficult to encode manually, underscoring the need for systems that can efficiently learn from user feedback during deployment.

Preference learning (PL) infers a reward function based on human-provided rankings of trajectories, eliminating the need to manually define rewards and allowing for personalization. Unlike inverse reinforcement learning, which depends on demonstration quality, PL allows non-expert users to provide input in complex domains such as Atari games and robot navigation [11, 94]. A key challenge, however, is that accurate reward learning often requires numerous preferences, limiting scalability to high-dimensional problems. To address this, active preference learning (APL) finds preference queries — sets of robot trajectories presented to a human — that maximize information gain [69, 3]. Prior work makes this optimization tractable by restricting the query set to pre-generated trajectories [4] or replay buffers [43], but this confines search to previously observed behaviors and may miss rare but informative scenarios needed to identify the true reward function.

We introduce CRED, a novel and efficient query generation method for APL that learns reward functions which generalize to different scenarios by using 1) **C**ounterfactual **R**easoning and 2) **E**nvironment **D**esign. CRED's counterfactual reasoning generates queries that reflect different hypothesized human preferences instead of generating queries through random rollouts [4]. Assuming a linear human reward model $R(\xi) = w^T \Phi(\xi)$ (the dot product of reward weights w and trajectory features $\Phi(\xi)$), learning the reward function simplifies to learning w. We use Bayesian inference [67] to maintain a belief over w, updated with each human query. By sampling diverse w values (based on cosine similarity) from the current belief, CRED directly evaluates different human preferences and performs counterfactual reasoning, effectively asking "what if w_i or w_j were the true reward weight?" Our second key insight is that the environment itself strongly influences the generated trajectories and thus can unnecessarily bound the information value of queries. CRED employs Bayesian Optimization [58] to efficiently find **environment parameters** that yield the most informative preference queries.

The main contributions of this chapter include a novel approach to preference learning,

CRED, that leverages environment design and counterfactual reasoning. Empirical evaluations across close proximity (tabletop handover) and large length-scale (delivery navigation) tasks demonstrate that CRED's queries achieve higher reward accuracy and help users converge faster to the true reward function than prior methods. Moreover, user studies indicate that participants report lower mental workload and express stronger preference when interacting with CRED.

	Lunar Lander	**Tabletop**	**Navigation**
Features Φ	vertical speed, horizontal position	objects that the end effector hovers over and the trajectory length	path length, terrain traversed (paved, grass, asphalt and concrete)
Env params θ	wind power, ranging from 1 to 20	object positions on the table	terrain type of road network
Preferences	approach direction, landing speed	objects to avoid hovering over	terrains to avoid and path length

Figure 3.1: Examples of preference queries with details and visualization across the three simulation domains.

3.2 Related Work

Learning from Preferences. Preference learning infers a human's reward function by iteratively asking them to choose between robot trajectories [4]. In active preference learning (APL), the objective is to identify the most informative query by selecting the one that maximizes the expected difference between the prior and the posterior belief distributions over reward functions [69]. Subsequent work explores different query selection objectives: maximize mutual information of the query and the estimated weights [4], maximize regret [91], and maximize uncertainty in reward predictions [43]. For example, Lee et al. [43] proposes ensemble-based sampling, which selects trajectory pairs with high disagreement across an ensemble of learned reward models, and entropy-based sampling, which maximizes the entropy of the preference distribution. In this chapter,

we focus on maximizing mutual information, as it directly quantifies the expected reduction in uncertainty over the reward weights and aligns well with Bayesian preference learning. However, our framework is general and can be extended to support other objectives.

A major challenge of APL is generating trajectory pairs that effectively optimize the chosen query selection objective. Prior methods such as [4] and [43] address this by selecting queries from a fixed dataset or replay buffer, but this restricts the diversity of queries and does not scale well to long-horizon tasks. Moreover, existing approaches typically generate queries within a single, fixed environment, limiting their ability to explore informative regions of the feature space. In contrast, our method leverages counterfactual reasoning to synthesize trajectories from a belief distribution over reward weights and jointly optimizes environment parameters. This combination enables the generation of more informative and diverse preference queries.

Terrain Preferences for Navigation. Prior work has incorporated human terrain preferences into robot navigation tasks by learning which surfaces to traverse or avoid. Karnan et al. [37] learn a visual representation space to extrapolate preferences to out-of-distribution terrains, Mao et al. [51] train a bird's-eye view costmap from human feedback, and Zhang et al. [94] extend this by using demonstrations and counterfactuals to solicit additional human annotations. We adopt this domain for evaluation, but unlike these approaches which emphasize representation and costmap learning, our method targets the active selection of informative preference queries. As such, our framework can serve as a complementary front-end to generate the initial preference data that these methods require.

Environment Design in RL and Robotics. Environment design treats environment parameters as optimizable variables. In RL, it has been leveraged for curriculum learning to improve generalization and convergence, for instance through co-evolution of agents and environment difficulty [87] or by modifying parameters to maximize an agent's learning potential [16]. In human-robot interaction, environment modification has been used to generate interpretable robot behaviors [40], legible human motion [82, 81] (discussed in detail in Chapter 4), and to support collaborative teaming in settings like warehouse design [96] or tabletop reorganization [2]. Most relevant to our

setting is the work of [7], which formulates environment design for inverse reinforcement learning (IRL) as a bilevel optimization problem over transition dynamics to elicit more informative demonstrations from experts. Our work adapts this idea to preference learning, where feedback comes from comparisons rather than demonstrations.

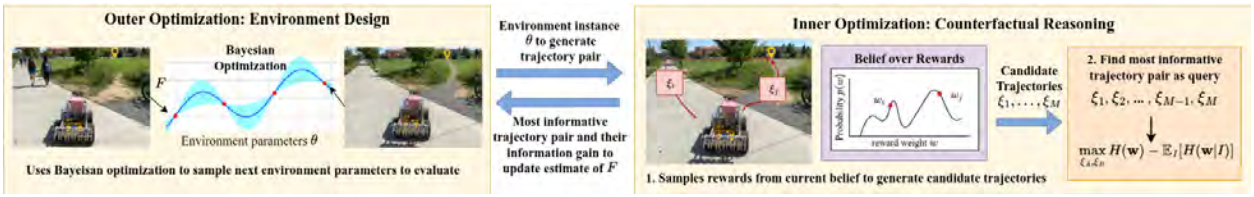


Figure 3.2: System overview of CRED as a bilevel optimization problem. **Outer optimization (Environment Design):** Bayesian optimization selects environment parameters θ to evaluate, seeking those that maximize estimates of query informativeness F. **Inner optimization (Counterfactual Reasoning):** Given θ, the system samples reward weights from the current belief to generate candidate trajectories $\{\xi_1, \ldots, \xi_M\}$. The most informative trajectory pair is returned to the outer optimization.

3.3 Preliminaries

Model. We consider a fully observable environment modeled as a Markov decision process (MDP) consisting of $\{S, A, T, R, \gamma, S_0\}$, where S is the set of states, A is the set of actions, $T : S \times A \times S \to [0, 1]$ is the transition function, $R : S \times A \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and S_0 is the initial state distribution. Let $s_t \in S$ and $a_t \in A$ denote the state and action at time step t. A trajectory $\xi \in \Xi$ is a finite sequence of state-action pairs: $\xi = ((s_0, a_0), (s_1, a_1), \ldots, (s_H, a_H))$ where H is the planning horizon. We overload the notation R and define the total reward of a trajectory as the sum of discounted per-step rewards: $R(\xi) = \sum_{t=0}^{H} \gamma^t R(s_t, a_t)$.

We assume a linear reward model over trajectory features: $R(\xi) = w^T \Phi(\xi)$ where $w \in \mathbb{R}^d$ are the unknown reward weights, and $\Phi(\xi) \in \mathbb{R}^d$ is the feature vector for trajectory ξ. To construct $\Phi(\xi)$, we define features over individual state-action pairs: let $\phi(s_t, a_t) \in \mathbb{R}^d$ be the feature vector at timestep t. The trajectory-level feature vector is then the discounted sum of per-step features: $\Phi(\xi) = \sum_{t=0}^{H} \gamma^t \phi(s_t, a_t)$. Once the reward weights w are learned from human feedback, we can

optimize trajectories with respect to the learned reward function using reinforcement learning [75] or trajectory optimization methods such as CHOMP [100].

Preference learning. The objective of preference learning is to learn w by querying a human for their preferences between pairs of trajectories. A preference query typically asks "Do you prefer trajectory ξ_A or ξ_B?" [3]. If a human prefers ξ_A over ξ_B, it implies $R(\xi_A) > R(\xi_B)$, or equivalently $w^T\Phi(\xi_A) > w^T\Phi(\xi_B)$. From this strict inequality, we can derive that $w^T(\Phi(\xi_A) - \Phi(\xi_B)) > 0$. The human's preference I can be encoded by $I = \text{sign}(w^T(\Phi(\xi_A) - \Phi(\xi_B)))$.

Human input can exhibit variability due to uncertainty in expressed preferences, which can be modeled using Boltzmann rationality. Under this model, the likelihood of a preference (Eqn. 3.1) is given by a softmax function over the reward values of the queried trajectories.

$$P(I \mid \mathbf{w}) = \begin{cases} \frac{\exp(R(\xi_A))}{\exp(R(\xi_A))+\exp(R(\xi_B))} & \text{if } I = +1 \\ \frac{\exp(R(\xi_B))}{\exp(R(\xi_A))+\exp(R(\xi_B))} & \text{if } I = -1 \end{cases} \tag{3.1}$$

Let $p(w)$ be our current belief distribution of the reward weights. We can perform a Bayesian update to compute the posterior given human input I, $p(w|I) \propto p(I|w)p(w)$. For uniqueness, we constrain the norm of the reward weights such that $\|w\|_2 \leq 1$. Since $p(w)$ can have arbitrary shapes, we use an adaptive Metropolis algorithm [32] to learn the posterior distribution. While we can use domain knowledge to initialize a non-uniform prior over reward weights, we adopt a uniform prior in all of our experiments for generality. Based on [4], the algorithm presents the human with a preference query and updates the belief distribution of w until a fixed number of iterations is reached.

Active Synthesis of Preference Queries. To learn w efficiently using minimal queries, active learning methods select preference queries (ξ_A, ξ_B) that maximize information gain. This is equivalent to maximizing the mutual information between the query and the estimated weights w [4]. Our objective function f is

$$\max_{\xi_A,\xi_B} f(\xi_A, \xi_B) = \max_{\xi_A,\xi_B} H(\mathbf{w}) - \mathbb{E}_I[H(\mathbf{w}|I)] \tag{3.2}$$

where $H(w) = -\mathbb{E}_w[\log(p(w))]$ is the information entropy of the belief $p(w)$. This objective finds preference queries such that the difference between the entropy of the prior and the posterior is maximized. However, directly optimizing this objective is challenging in practice: the search space is non-convex, and optimization often converges to local maxima, resulting in poor sample efficiency. Section 3.4 discusses our approach of using counterfactual reasoning and environment design to more effectively generate trajectories that optimize this information gain objective.

Environment Design. In traditional preference learning, the environment is fixed. However, the transition dynamics determine which trajectories are feasible and which features are observable. Consequently, the informativeness of a preference query depends not just on the trajectories, but also on the environment in which they are embedded.

We formalize the problem of environment design for preference learning as the selection of a sequence of environment parameters $\theta \in \Theta$, where each θ defines a specific transition function $T_\theta \in \mathbb{T}$. These parameters capture aspects of the environment such as terrain types, obstacle placements, or object configurations, and they determine which trajectories are feasible and what features can be observed.

For a given environment θ, a trajectory ξ is generated by executing a policy under the transition function T_θ. Its features are represented by a function $\Phi(\xi, \theta)$, which depends on both the trajectory's state-action sequence and the environment. For example, the same path may traverse different terrain types depending on θ, resulting in different features. This makes the feature map $\Phi(\cdot, \theta)$ environment-dependent and highlights the role of environment design in shaping the information content of preference queries.

At each round i, the learner (robot):

(1) Selects an environment $\theta^{(i)} \in \Theta$,

(2) Generates a pair of trajectories $(\xi_A^{(i)}, \xi_B^{(i)})$ in $T_{\theta^{(i)}}$,

(3) Queries the human: "Which trajectory do you prefer?",

(4) Updates its belief over the reward weights $\mathbf{w}$.

This process allows the learner to actively shape the learning problem by selecting environments that yield the most informative trajectory comparisons for inferring the human's reward function.

3.4 Technical Approach

Active preference learning faces challenges in generating trajectories that optimize for information gain (Eqn. 3.2), as the objective function involves a pair of trajectories as variables. This task is further complicated by the fact that the optimization is typically constrained to a single environment, which may not adequately represent the full feature space, resulting in learned rewards that may fail to generalize effectively and suffer from sample inefficiency. Our approach using counterfactual reasoning and environment design to address these issues is summarized in Fig. 3.2.

3.4.1 Counterfactual Reasoning

Counterfactual reasoning explores different trajectories that could result if a hypothesized set of reward weights were the true weights. We maintain a belief over the weights (via an adaptive Metropolis algorithm [32]) while estimating the human's reward function, where each sample of weights could lead to a different policy and consequently different trajectories when the policy is executed. This allows us to pose counterfactual questions, such as "what if reward i is the true reward as opposed to reward j?" Let w_i be an instance of reward weights sampled from our belief. We can use reinforcement learning (RL) [75] to train a policy π_i that maximizes the reward function induced by w_i. Rolling out π_i in a given environment yields a trajectory ξ_i. Alternatively, we can directly optimize the trajectory with respect to $w_i^T \Phi(\xi)$ using trajectory optimization methods such as CHOMP [100] (Algorithm 2 line 4).

By sampling reward weights and generating trajectories, we construct a set of counterfactual trajectories that represent different human preferences. We then evaluate the information gain objective f (Eq. (3.2)) for each pair of trajectories to identify the most informative preference query (Algorithm 2 lines 6–7). To minimize the number of evaluations of the objective function, we

Algorithm 2 Counterfactual Reasoning

Require: Belief $P(w)$, N samples, M subset size
1: Sample $\{w_1, \ldots, w_N\} \sim P(w)$
2: Select M diverse weights (e.g., max cosine distance)
3: **for** each selected w_k **do**
4: Generate trajectory ξ_k by maximizing $w_k^\top \Phi(\xi)$ ▷ e.g., via RL or CHOMP
5: **end for**
6: Compute information gain f (Eq. 3.2) for all pairs ξ_i, ξ_j
7: Return most informative pair (ξ_i, ξ_j)

start by sampling N reward weights. We then select the most diverse M weights, where $M < N$, from this set by sequentially computing diversity based on cosine similarity, forming our final set of reward weights for evaluation (Algorithm 2 lines 1–2).

Algorithm 3 Environment Design

Require: Environment parameters Θ, Bayesian optimization iterations N
1: **for** $t = 1$ to N **do**
2: Propose θ^t using Bayesian optimization
3: Generate $(\xi_A^{(t)}, \xi_B^{(t)})$ via CR (Algorithm 2) in env θ^t
4: Compute information gain $F(\xi_A, \xi_B; \theta^t)$
5: Update GP model with $(\theta^t, F(\xi_A, \xi_B; \theta^t))$
6: **end for**
7: Return optimal θ^* found and corresponding (ξ_A, ξ_B)

3.4.2 Environment Design

While counterfactual reasoning generates trajectory pairs optimizing for different reward weights, the fixed environment can limit their ability to reveal crucial preference distinctions. We posit that if we have the ability to "imagine" new environments, we can better generate trajectories that show the differences between the different reward weights.

To formalize this idea, we make explicit the role of environment parameters in trajectory generation. Recall that Θ denotes the set of configurable environment parameters where $\theta \in \Theta$ induces a transition function T_θ. The feature function Φ is environment-dependent and written as $\Phi(\xi, \theta)$. Let $F(\xi_A, \xi_B; \theta)$ denote the information gain from presenting the trajectory pair (ξ_A, ξ_B) in environment θ, defined analogously to Eqn. 3.2, but accounting for the dependence of trajectories

and features on θ. We formulate environment design as a bilevel optimization problem[1] :

$$\max_{\theta \in \Theta} \max_{(\xi_A, \xi_B) \in \Xi(\theta)} F(\xi_A, \xi_B; \theta) \tag{3.3}$$

where $\Xi(\theta)$ denotes the set of feasible trajectory pairs under the MDP with transition T_θ. The outer optimization selects the environment parameters θ that maximize the informativeness of the resulting preference query, while the inner optimization identifies the most informative trajectory pair (ξ_A, ξ_B) that can be generated in that environment. This bilevel structure mirrors the kitting formulation of Chapter 2 — compare Eqns. 2.1–2.2 — with θ playing the role of the task segmentation index i and (ξ_A, ξ_B) playing the role of the kit layout P.

Since $F(\xi_A, \xi_B; \theta)$ is generally not differentiable with respect to θ, we use Bayesian optimization [73], which models F with a Gaussian process (GP) defined by a mean function $m : \Theta \to \mathbb{R}$ and a positive definite covariance function $K : \Theta \times \Theta \to \mathbb{R}$. We use the upper confidence bound (UCB) acquisition function, selecting θ that maximizes $UCB(\theta) = \mu(\theta) + \kappa\sigma(\theta)$, where κ balances exploitation against exploration. We use the `BayesianOptimization` Python library [58] which employs a Matérn kernel with hyperparameters fit via maximum likelihood estimation. Unless otherwise specified, we use the library's default settings. Algorithm 3 shows the pseudocode for environment design.

3.5 Experiments

We evaluate CRED across a suite of simulation experiments and a user study. Through these experiments we demonstrate that CRED enables more accurate and efficient inference of human reward functions than existing methods and characterize the contributions of its core components.

We begin with simulation experiments in three domains (Lunar Lander, Tabletop Manipulation, and Navigation) where we measure the accuracy of learned rewards and compare CRED against two established baselines [4, 11]. We conduct ablation studies to isolate the impact of coun-

[1] In principle, environment parameters θ and trajectory pairs (ξ_A, ξ_B) could be jointly optimized in a single-level formulation. However, the feasible set of trajectories $\Xi(\theta)$ depends on θ through the induced transition function T_θ, making joint optimization computationally expensive. The bilevel formulation leverages this structure by first selecting θ and then optimizing over $\Xi(\theta)$, which is more tractable in practice.

terfactual reasoning and environment design, followed by a comparison of our environment design strategy with domain randomization, an alternative that samples environments uniformly at random. Finally, we present user study results evaluating if real users can effectively teach preferences using CRED in manipulation and navigation tasks.

Our experiments address the following hypotheses:

- **H1:** CRED achieves a higher reward accuracy with fewer preference queries compared to baselines [4, 11].
- **H2:** Both counterfactual reasoning and environment design contribute significantly to CRED's performance.
- **H3:** CRED outperforms domain randomization by generating more informative environments that accelerate reward learning.
- **H4:** Real users prefer interacting with CRED over baseline methods in practical manipulation and navigation tasks and report lower mental workload as measured by NASA-TLX.

3.5.1 Simulation Experiments

For all simulation experiments, we simulate 10 users, each initialized with a distinct ground-truth reward weight vector. To encourage diversity, we sample 1000 random weight vectors and select the cluster centers obtained via K-means as the ground-truth weights. This setup allows us to test whether CRED can generalize across heterogeneous user preferences.

Environment Setup. **Lunar Lander.** We use the Gymnasium implementation of Lunar Lander [78], where a lander must safely reach a designated pad in a 2D environment. Beyond the environment's default objective of safe landing, we introduce user-dependent preferences such as approaching from the left versus right side and controlling descent speed. The feature vector Φ encodes the lander's vertical velocity and horizontal position, while the environment parameter θ specifies the wind power applied during descent. To generate trajectories, we train PPO [70]

agents using the Stable Baselines implementation. We use the default hyperparameters, with two exceptions: the target KL divergence is set to 0.01, and both the policy and value networks are two-layer MLPs with 256 hidden units per layer.

Tabletop Manipulation. We adapt a tabletop manipulation domain [63] in MuJoCo [77], where a robot delivers a cup of coffee across a cluttered table. To avoid spills on critical items such as electronics, the robot must learn user preferences for safe trajectories. Objects on the table include fruits, laptop, a phone, glasses, headphones, a camera, and a piggy bank. The features encode how often the robot's trajectory hovers above each object, while the environment parameters θ specify object placements. To make the search over possible object configurations tractable, we use a variational autoencoder (VAE) [12] to compress environments into a latent space Z (see Appendix A), where the environment parameters θ correspond to latent vectors $z \in Z$. We use CHOMP [100] to generate trajectories and augment the cost function with $-R$ to incorporate the rewards.

Navigation. We consider a delivery task where a robot must transport food to a customer by navigating between predefined start and goal locations over a street network derived from OpenStreetMaps [60]. The network is simulated in Webots [88] for experiments and user studies. The features encode the path length and the proportion of each terrain type (e.g., asphalt, paved, grass, concrete, and brick) along the route. The environment parameters θ control the surface types assigned to the edges on the shortest path, thereby altering the trade-offs between efficiency and terrain preferences. Trajectories are generated using value iteration [75], where graph nodes correspond to states and outgoing edges define the available actions at each node.

Metrics. We measure accuracy using the sample correlation coefficient r between ground truth and estimated rewards:

$$r = \frac{\sum_{i=1}^{n} \left(R_{gt}^{i} - \bar{R}_{gt}\right)\left(R_{est}^{i} - \bar{R}_{est}\right)}{\sqrt{\sum_{i=1}^{n} \left(R_{gt}^{i} - \bar{R}_{gt}\right)^{2}} \sqrt{\sum_{i=1}^{n} \left(R_{est}^{i} - \bar{R}_{est}\right)^{2}}}, \tag{3.4}$$

where R_{gt}^{i} and R_{est}^{i} are the ground truth and estimated rewards for sample i, and $\bar{R}$ denotes the sample mean. We compute R_{gt} and R_{est} by evaluating the true and learned weight vectors over a

grid of feature vectors, with feature ranges determined from a set of trajectories sampled in each environment.

Baselines. We compare CRED against two state-of-the-art preference learning baselines. The first optimizes the mutual information objective (Eqn. 3.2) over pre-generated trajectories from random rollouts [4], which we denote as **RR**. The second, inspired by [11] and used in later work such as PEBBLE [43], learns a reward function by training a neural network on preference data. In our Bayesian belief framework, we replicate this baseline by using the mean belief as the reward function. During rollouts, the policy trained on the learned reward function follows its action with probability $1 - \epsilon$ and explores with probability ϵ ($\epsilon = 0.25$ in our experiments). For CHOMP, exploration is simulated by adding Gaussian noise. We refer to this baseline as the **M**ean **B**elief **P**olicy (**MBP**).

Simulation Results.

Baseline Comparison. CRED consistently achieves the highest final accuracy, reaching 0.998, 0.979, and 0.906 in Lunar Lander, Tabletop, and Navigation, respectively. CRED also converges rapidly, surpassing 95% accuracy within 4 iterations for Lunar Lander and 10 for Tabletop. Navigation requires around 16 iterations, reflecting the greater difficulty of distinguishing preferences across multiple terrain types. By contrast, MBP plateaus around 60% accuracy and fails to produce informative queries after roughly 8 iterations. Because MBP cannot get feedback in novel environments, it is restricted to the features available in the current environment, limiting its effectiveness. Overall, CRED achieves both higher reward accuracy and faster convergence, supporting **H1**.

Ablations. To assess the contribution of each component in CRED, we evaluate two ablations. First, **CR** removes environment design, relying only on counterfactual reasoning. Second, **MBP-ED** removes counterfactual reasoning, applying environment design to the mean-belief policy baseline. Both ablations yield either lower final accuracy or slower convergence across all domains, indicating that environment design and counterfactual reasoning are complementary. These results support **H2**.

We further test the environment design strategy against **domain randomization (DR)**, which selects environment parameters $\theta \in \Theta$ uniformly at random. For fairness, the number of environments sampled under DR matches the number evaluated during Bayesian optimization. We construct three DR baselines: **RR-DR**, which generates random rollouts in randomized environments; **MBP-DR**, which executes trajectories from the mean-belief policy under random environments; and **CR-DR**, which applies counterfactual reasoning without guided environment selection. All DR variants underperform CRED (with the exception of RR-DR in Navigation), highlighting the benefit of using Bayesian optimization to actively search for informative environments rather than sampling them uniformly at random (**H3**). We note that RR-DR does not enforce a goal-reaching constraint on trajectories, enabling greater diversity in the feature space. However, our user study (Sec. 3.5.2) shows that RR-DR trajectories are harder for users to interpret, limiting their practicality despite their competitive accuracy in simulation.

3.5.2 User Study

Setup. We conducted an IRB-approved user study to evaluate CRED in both tabletop and navigation tasks. A total of 25 participants were recruited from the university campus (ages 20–39, $M = 26.8$, $SD = 4.5$; 18 male, 6 female, 1 other). Participants were randomly assigned to conditions in a within-subjects design, with each participant completing 6 preference queries per task. We compared CRED against two baselines: MBP [11] and RR-DR, which uses pre-generated trajectories [4] augmented with domain randomization. We did not include MBP with domain randomization (MBP-DR), as simulation results showed that it performed comparably to MBP alone. To avoid participant fatigue, we restricted the study to these three conditions. The order of conditions was randomized for each task to mitigate ordering effects.

In each trial, participants were presented with a pair of robot trajectories and asked to select the one they preferred. Trajectories were generated in simulation and visualized as arrows overlaid on the task environment, similar to Fig. 3.1. The reward functions inferred from these preferences were subsequently deployed on the physical robot platforms. For the tabletop task, the

preferred behavior was to avoid passing over electronics, while for the navigation task, the preferred behavior was to avoid crossing asphalt roads. Participants were informed that although the robot could traverse grass, such routes may be less energy efficient. These task instructions were chosen to define a ground-truth reward function[2] , allowing us to evaluate the accuracy of the learned rewards while also capturing realistic trade-offs in user preferences.

After each condition, participants completed a short questionnaire. Subjective workload was measured using the NASA-TLX [33] survey, and perceived ease of comparison was measured with a 7-point Likert item ("It was easy to choose between the trajectories that the robot showed me," 1=strongly disagree, 7=strongly agree). To assess learning performance, we compared the inferred reward functions to the ground truth rewards by computing the reward correlation (Eq. 3.4).

Results.

For the tabletop task, CRED achieved higher reward correlation with the ground truth rewards (median $r = 0.97$, IQR $= 0.01$), outperforming RR-DR (median $r = 0.75$, IQR $= 0.08$) and MBP (median $r = 0.51$, IQR $= 0.88$). Similarly, in the navigation task, CRED again showed better performance (median $r = 0.78$, IQR $= 0.25$), compared to RR-DR (median $r = 0.57$, IQR $= 0.21$) and MBP (median $r = 0.38$, IQR $= 0.59$). In terms of subjective results, participants reported lower mental workload under CRED (NASA-TLX: median = 1.83, IQR = 1.33) compared to DR-RR (2.67, 1.92) and MBP (2.33, 1.00). They also rated CRED higher on ease of comparison (6.0, 1.75) than DR-RR (4.5, 3.75) and MBP (5.0, 2.00), supporting **H4**.

Statistical comparisons were conducted using the Wilcoxon signed-rank test, a nonparametric test appropriate for ordinal measures (e.g., Likert ratings, NASA-TLX scores) and bounded non-normally distributed measures such as reward correlations. To account for multiple pairwise comparisons, we applied Holm–Bonferroni correction to control the family-wise error rate. CRED significantly outperformed both RR-DR and MBP in the tabletop and navigation tasks, with the

[2] Ground-truth rewards are defined as linear functions of trajectory features, normalized to unit-length weight vectors. For the tabletop task, features are the number of waypoints over `fruit`, `accessories`, and `electronics`, and the trajectory length, with weights $[-0.1, -0.1, -2.0, -1.0]$. For the navigation task, features are path length and the distances traversed on `paved`, `grass`, `asphalt`, and `concrete`, with weights $[-1.0, -0.1, -2.0, -5.0, -0.1]$.

exception of the CRED vs. RR-DR comparison in tabletop reward accuracy. We observed a few outliers in the tabletop task for both CRED and RR-DR, which may be attributable to participants not following task instructions. These results show that CRED more consistently recovers the true rewards across both navigation and manipulation domains. Furthermore, participants found CRED less mentally demanding and perceived the queries generated by CRED as easier to evaluate.

3.6 Chapter Summary and Future Work

We introduced CRED, a query generation framework that jointly optimizes environments and trajectories for active preference learning. Experiments show that CRED achieves higher sample efficiency and reward accuracy than prior methods, and ablations highlight the importance of counterfactual reasoning, environment design, and Bayesian optimization for selecting informative environments. A user study in tabletop and navigation tasks demonstrates that CRED produces queries that are easier to answer and impose lower mental workload. CRED is applicable across policy classes (e.g., PPO, value iteration, trajectory optimization) and domains with different preferences.

One limitation is the computational complexity: as a bilevel formulation, CRED requires an outer search over environment parameters and an inner search over trajectories. Future work can mitigate this cost through parallelization of environment evaluations and more efficient approximations of query informativeness.

Transition to Chapter 4. Chapters 2 and 3 both inferred high-level aspects of human intent — the task schedule and the reward function — by designing the environment. In both cases, the robot reasoned about the human's actions at the granularity of whole trajectories or whole subtasks. We now zoom in to the finest granularity of intent — the human's immediate motion — and show that environment design is equally effective there.

Chapter 4

Workspace Optimization for Human Motion Prediction

Chapter overview. So far, we have used environment design to improve the robot's ability to infer high-level intent: what the human will do (Chapter 2) and what they prefer (Chapter 3). Both operate at the level of task structure and reward. In close-proximity collaboration, however, such as a cook reaching for a knife across the robot's workspace, the robot must also predict the human's immediate motion on the scale of a single reach.

In this chapter, we present a workspace design approach that rearranges physical objects and projects virtual constraints via augmented reality (AR) to shape human motion. By structuring the workspace, the robot reduces variability in human trajectories, making low-level goal prediction more accurate and data-efficient.

Unlike Chapters 2 and 3, which rely on bilevel optimization over structured environment parameters, this chapter addresses a combinatorial, high-dimensional design space using MAP-Elites, a quality-diversity algorithm that enables efficient exploration of diverse workspace configurations [82].

4.1 Introduction

In human-robot collaborative tasks, shared mental models between agents enable the awareness and joint understanding required for effective teamwork [76]. With no shared notion of the task to be completed, the inherent stochasticity and opacity of human decision-making makes robot planning difficult [72]. To this end, prior research efforts have focused on developing robots that

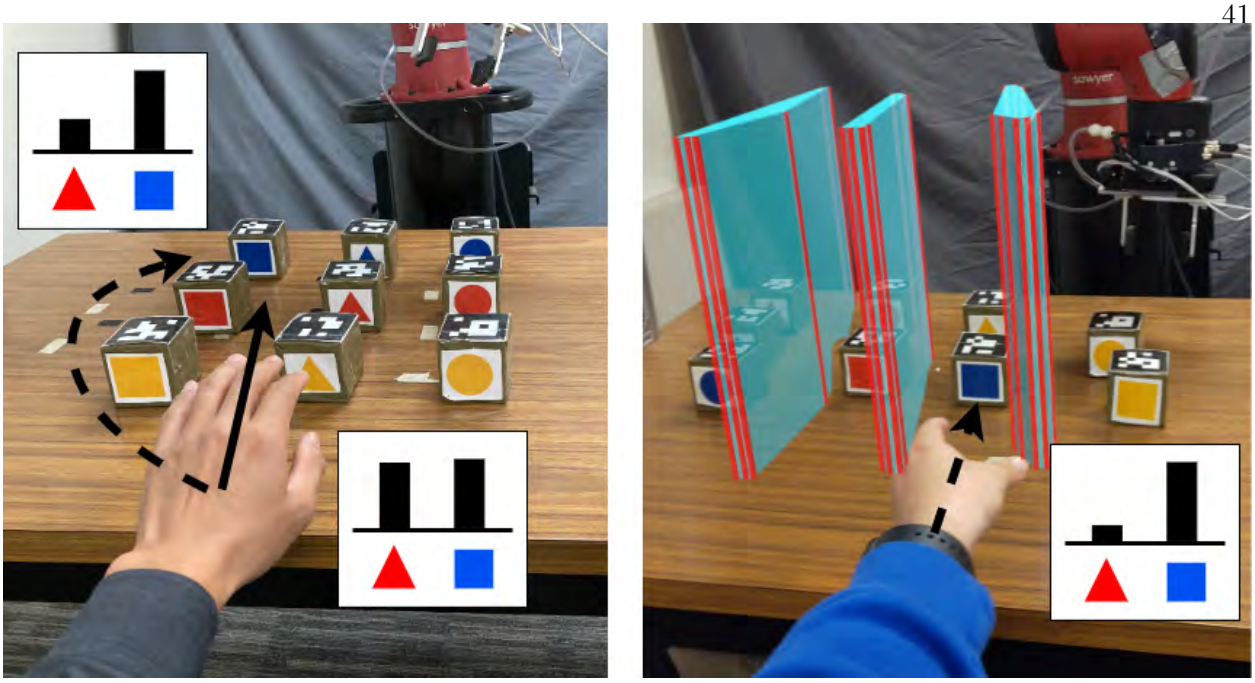


Figure 4.1: Workspace configuration affects the robot's ability to correctly predict the human's goal — the blue square cube. **Left**: The legible path (dotted) requires the human to take a circuitous route while the natural path (solid) is not legible. **Right**: Our approach generates a workspace configuration by arranging physical objects and projecting "virtual obstacles" in AR (cyan and red barriers), in order to induce naturally legible paths from the human.

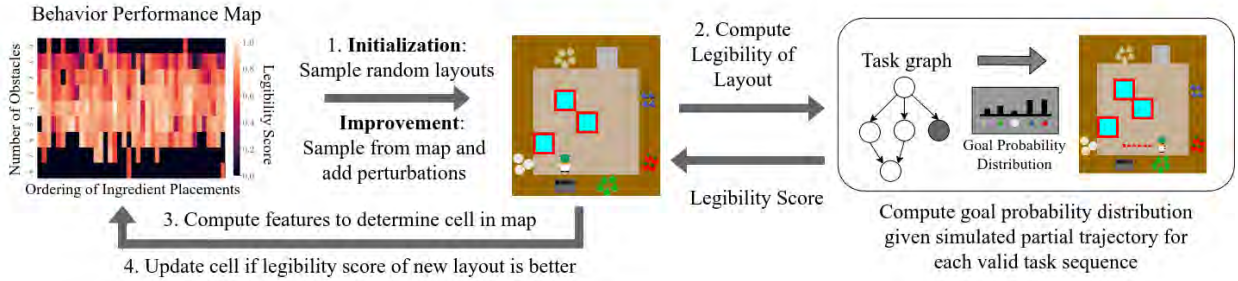


Figure 4.2: Our approach for generating workspace configurations that enable accurate human goal predictions. **(1)** In the initialization phase, we sample random environment layouts to populate the behavior performance map, which stores diverse and high performing solutions. This is followed by the improvement phase where we sample directly from the map and add perturbations to test whether the legibility is improved. **(2)** In both phases, we compute the legibility of the sampled layout by computing the probability of predicting the correct goal at each stage of the task execution. **(3)** We compute the features of the sampled layout to determine its location in the map. **(4)** The map is updated if the legibility score of the sampled layout is better than the existing one.

can predict human behavior [68, 42, 49], generating motion plans to safely interact in a shared

environment [62, 31]. However, these methods are limited by the quality of robot predictions of a human collaborator's intention and resultant behavior. With inaccurate human models or unexpected human behavior diverging from past experiences, the robot may produce unsafe interactions [52], especially in safety-critical or close-proximity settings [86].

To address the inherent challenges of accurately predicting human motion early in a demonstrated trajectory, our key insight is that robots can take an active role in structuring the environment to reduce the variance of human motion caused by dense and overlapping task spaces, thereby improving the performance of human behavior models.

In this work (see Fig. 4.1), we introduce an algorithmic approach for a robot to configure a shared human-robot workspace prior to interaction in order to improve a robot's ability to predict the human collaborator's goals during task execution. As detailed in Fig. 4.2, we present an objective function that scores potential workspace configurations in terms of how legible the actions of a human teammate are likely to be when performing a task in that environment. We use the mathematical formulation of legibility from [17], which computes the probability of successfully predicting an agent's goal given an observation of a snippet of its trajectory. Our approach finds workspace configurations that maximize legibility over the valid goals at each stage of task execution.

Each candidate workspace configuration combines a potential arrangement of physical objects and projection of "virtual obstacles" in augmented reality (AR) in the environment. While the arrangement of physical objects can be achieved prior to the shared task via a simple composition of robotic pick and place actions, the addition of augmented reality-based virtual obstacles is particularly effective in imposing explicit constraints on the possible motions of the human, without requiring additional physical changes to the environment. Note that our approach does not restrict the human to a particular goal and retains their ability to decide on the goal they're reaching towards. Humans also have the flexibility to alter their decisions midway through a task, since goal prediction is performed at every time step.

We efficiently explore the space of workspace configurations using a quality diversity (QD) algorithm called Multi-dimensional Archive of Phenotypic Elites, or MAP-Elites [54]. Instead

of finding a single optimal solution, MAP-Elites produces a map of performant solutions along dimensions of a feature space chosen by the designer. MAP-Elites enables efficient and extensive exploration of complex search spaces, leading to higher quality solutions as compared to other search algorithms [14, 57].

We empirically demonstrate that our workspace optimization approach improves the accuracy of human goal prediction models, showing results for both a Bayesian predictor with learned cost functions using Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) and a time series multivariate Gaussian model. We collect human motion data in an online 2D navigation experiment and a real world tabletop experiment involving a robotic manipulator, comparing our approach to other workspace arrangement strategies. In summary, we present two primary contributions: 1) an algorithm for optimizing the placement of physical and virtual objects in a shared human-robot workspace for maximizing human legibility, and 2) an evaluation of that algorithm via two human subjects experiments, showing that it influences human behavior in ways that improve a robot collaborator's prediction model.

4.2 Related Work

Planning using Human Motion Models. In human-robot collaboration, the robot needs to predict human motion in order to coordinate its actions with those of the human. Prior work has developed motion planning algorithms that model human motion and generate robot plans to safely interact with humans in tabletop [41] and navigation [26, 64] settings. To account for the uncertainty in human motion prediction, prior work has used partially observable Markov decision processes (POMDPs) to determine optimal actions for a robot given a probabilistic belief over the human's intended goals [62, 31]. These methods rely on the human motion model to achieve safe interactions, but human motion is inherently highly variable as humans can always move unexpectedly. The difficulty of the problem can be seen by the variety of approaches taken toward accurate human motion prediction: Gaussian models [45, 66], dynamic movement primitives [47, 89], latent representation learning [8, 53], and imitation learning [49, 20], among various other

methods [68]. Our work takes a different approach and addresses a fundamental challenge faced by all human motion prediction models; we reduce the uncertainty inherent in modeling the intentions of human collaborators by pushing them towards legible behavior via environment design. Our work improves human motion model predictions by increasing environmental structure to reduce uncertainties, facilitating more fluent human-robot interactions.

Environment Design in Robotics. Prior work has also explored designing or modifying environments in order to better achieve agent goals [80]. Zhang et al. [95] proposed a framework for designing environments that optimize an agent's reward, and Keren et al. [38] extended it for stochastic transitions. Kulkarni et al. [40] generate interpretable robot behaviors by modifying the environment. These works show the potential advantages of a robot using environment design to improve its task performance and interpretability. There has also been work on modifying the environment for collaborative teaming. Zhang et al. [96] optimizes warehouse layouts for multi-robot coordination, and Bansal et al. [2] explores the idea of robots moving objects to reduce the likelihood of future collisions in a tabletop task. Our technique differs from prior work in that the robot modifies its environment with the explicit goal of improving its ability to predict human behavior for fluent collaboration with human teammates. As discussed in Chapters 2 and 3, the idea of environment design as a lever for HRI threads through this entire thesis; this chapter's contribution is applying it to low-level motion.

Finding the optimal environment by simply iterating through all possible workspace configurations quickly becomes intractable as the number of objects or possible states increases. To address this, others have used quality diversity (QD) algorithms to generate diverse environments for evaluating the performance of shared autonomy algorithms [23] and explore diverse coordination behaviors as a result of environment design [25]. Inspired by the success of QD algorithms in finding diverse solutions in large search spaces [14, 57], we use MAP-Elites to search for environments that best elicit legible human behavior. Our work extends QD approaches to generate interaction scenarios that influence human behavior and address the robot's limitations when collaborating with humans.

4.3 Legible Workspace Generation

In this section, we describe our approach for modifying the shared human-robot workspace to maximize legibility and enable more accurate human goal predictions (summarized in Fig. 4.2).

4.3.1 Legibility Score

To evaluate the legibility of a workspace configuration, we consider the probability distribution of predicting that the human is approaching goal G given an observed trajectory from start state S to intermediate point Q. We use the formulation developed by [17] shown in Equation 4.1.

$$\Pr(G|\xi_{S\to Q}) \propto \frac{\exp(-C(\xi_{S\to Q}) - C(\xi^*_{Q\to G}))}{\exp(-C(\xi^*_{S\to G}))} \tag{4.1}$$

The optimal human trajectory from point X to point Y with respect to cost function C is denoted by $\xi^*_{X\to Y}$. Equation 4.1 evaluates how cost efficient (with respect to C) going to goal G is from start state S given the observed partial trajectory $\xi_{S\to Q}$ relative to the most efficient trajectory $\xi^*_{S\to G}$.

Let $\mathcal{G}$ be the set of valid goals at the current time step. We develop a legibility score (Eqn. 4.2) for use in our optimization objective that, for every valid goal at a given time step in the task execution, maximizes the margin of prediction between the human's chosen goal $G_{true} \in \mathcal{G}$ and all other valid goals. If the predicted goal is not G_{true}, the score is penalized by a fixed cost c multiplied by the length of the sampled human trajectory $|\xi_{S\to Q}|$. Otherwise, the score is the difference of the two highest probabilities (computed by the *margin* function). The notation $G_{(i)}$ denotes the i-th index of a sorted list of length n that represents the goal probabilities ordered from least to most likely given the observed trajectory $\xi_{S\to Q}$.

$$\text{EnvLegibility}(G_{true}) = \begin{cases} -c|\xi_{S\to Q}|, & \text{if } \arg\max_{G\in\mathcal{G}} \Pr(G|\xi_{S\to Q}) \neq G_{true} \\ G_{(n)} - G_{(n-1)}, & \text{otherwise} \end{cases} \tag{4.2}$$

4.3.2 Optimization for Task Legibility

To generate a workspace configuration with improved legibility of the agent's goals for a task, we maximize the legibility score from Equation 4.2 for all valid subtask sequences (Eqn. 4.3). We use precedence constraints introduced by the structure of the task (i.e., which subtasks are prerequisites for other subtasks) to identify the set of valid subtasks (and thus valid goals $\mathcal{G}$) at any given time step.

$$\max \sum_{T' \in \text{permutations}(T)} \mathbb{1}\{\text{valid}(T')\} \times \sum_{t \in T'} \sum_{G \in \mathcal{G}} \text{EnvLegibility}(G) \tag{4.3}$$

Let T' represent a valid subtask sequence, consisting of all k subtasks of task T: $t_1, \ldots, t_k$, ordered such that for each index i, subtask t_i has all precedence constraints satisfied. Each subtask has one or more goals that an agent can reach to complete it. For example, the task "Set Table" may have a subtask "Get/Place plate on place mat" with multiple satisfying goals (multiple place mats). Additionally, since there are often multiple valid subtask sequences given an observed set of subtasks $t_1, \ldots, t_i$, $\mathcal{G}$ represents the set of goals corresponding to all uncompleted subtasks with satisfied precedence constraints. The objective function considers all possible goals that the human might be reaching for at a given stage of task execution and maximizes the probability of correctly predicting the human's chosen goal.

4.3.3 Search using Quality Diversity

Iterating through all possible workspace configurations to find the optimal solution is intractable since the number of possible configurations is exponential in the number of goals, virtual obstacles, and size of the workspace. We use MAP-Elites [54] to approximate the optimal solution.

In Algorithm 5, MAP-Elites maintains a behavior performance map, or solution map S, that stores high performing solutions across features or behaviors of interest. To find the most legible workspace (objective function F), the designer chooses a set of features or behaviors (computed by measure function M) such as the distance between the objects or the number of virtual obstacles.

Algorithm 4 `improve_workspace`

Require: Workspace configuration w, objective score of w s_w, objective function F, measure function M, Solution map S, Solution values V

Best configuration $w^* \leftarrow w$; Best score $s^* = s_w$
$A \leftarrow$ sample_perturbations(w)
for $a \in A$ **do**
    Generate new workspace $w' =$ apply_perturbation(w, a)
    Compute legibility score $s_{w'} = F(w')$
    if $s_{w'} > s^*$ **then**
        Update best workspace $w^* = w'$
        Update best score $s^* = s_{w'}$
    end if
    Determine features $\boldsymbol{m} = M(w')$
    if $S[\boldsymbol{m}] = \emptyset$ or $s_{w'} > V[\boldsymbol{m}]$ **then**
        Update solution map $S[\boldsymbol{m}] = w'$
        Update solution values $V[\boldsymbol{m}] = s_{w'}$
    end if
end for
if $w^* \neq w$ **then**
    return improve_workspace(w^*)
end if
return w^*

Algorithm 5 Workspace Generation with MAP-Elites

Require: Human Trajectory Generator G_H, Objective function F, measure function M

Solution map $S \leftarrow \emptyset$; Solution values $V \leftarrow \emptyset$
for $i = 1, \ldots, N$ **do**
    if $i < N_{init}$ **then**
        Generate workspace $w =$ random_workspace()
    else
        Sample workspace from map $w =$ random(S)
        Run $w =$ improve_workspace(w)
    end if
    Determine features $\boldsymbol{m} = M(w)$
    Determine objective score $s = F(w)$
    if $S[\boldsymbol{m}] = \emptyset$ or $s > V[\boldsymbol{m}]$ **then**
        Update solution map $S[\boldsymbol{m}] = w$
        Update solution values $V[\boldsymbol{m}] = s$
    end if
end for
return S, V

The algorithm would then find the most legible workspace for each possible combination of features found. As input, the algorithm also requires a model G_H that outputs human trajectory given a goal. G_H can be learned from data via inverse optimal control [49] or approximated via shortest path to goal [17].

MAP-Elites (Alg. 5) consists of two phases: initialization and improvement. In the initialization phase, we randomly sample workspaces for N_{init} iterations and store them in the corresponding cell in the solution map by computing the features. In the solution map S, the cell associated with the vector of feature values m is denoted $S[m]$. For $N - N_{init}$ iterations, we perform the improvement phase: we first randomly sample from the solution map and then empirically approximate the gradient to improve the solution. Following differentiable QD [22], we use gradient information to speed up search. Unlike [22], our implementation only estimates gradients of the objective function and not the measure function. Since we empirically approximate the gradient using stochastic sampling (Alg. 4), the environment updates may not always align with the steepest ascent direction. This can lead to "suboptimal" solutions that can, however, contribute to increased diversity.

As detailed in Algorithm 4 (the *improve_workspace* function), line 2 samples perturbations to the workspace (i.e., changing an item's position, adding or removing a virtual obstacle). For each perturbation, a new workspace configuration w' is generated by applying the perturbation. We keep track of the current best workspace w^* in terms of the objective score and update the solution map if a better environment was found for the features $\boldsymbol{m}$. If there was an improvement to the workspace, we run Algorithm 4 again. Otherwise, a local minimum has been found, and we return the best workspace found.

4.4 Evaluation

We evaluate our approach in two environments: a) single-player Overcooked [10] and b) tabletop collaborative pick-and-place with a Sawyer robot. These experiments validate our approach across navigation and tabletop domains while analyzing two distinct forms of human-borne motion data. To predict the human's goal given a partial trajectory in Overcooked, we use the Bayesian

predictor developed by Dragan et al. [17] (Eqn. 4.1). For the tabletop experiment, we implement the time series multivariate Gaussian model proposed in [66] to predict which cube the human is reaching for. We chose these models as representative examples of two distinct approaches to human motion prediction. Collectively, these experiments demonstrate the broad applicability of our approach across varied environments and tasks, as well as varied human goal classification techniques.

4.4.1 Hypotheses

H1: Environments generated by our approach will enable more accurate predictions of the human goal throughout the task execution compared to baseline environments.

H2: The prediction models will have better accuracy than predictions based on heuristics such as predicting the goal that is closest to the current trajectory.

H3: Prediction models trained in environments generated by our approach will be more data-efficient, requiring fewer examples to reach peak performance levels as compared to those trained in baseline environments.

4.4.2 Overcooked Experiment

Experimental Setup. The Overcooked game (Fig. 4.3a–d) requires participants to fetch ingredients, place them in a pot, plate the cooked dish, and deliver the dish to a serving station in a grid world environment. We chose to run the experiment with a single human agent in order to isolate the effects of environment design on the goal predictability of human agents. An investigation on combining robot policy and environment design to influence goal prediction is an interesting direction for future work. We conducted an IRB-approved Amazon Mechanical Turk experiment with 20 participants aged 18 or older and with at least a 95% approval rating. Each participant played in five rounds — one training round followed by four conditions in randomized order. Each round had three soup deliveries with the following ingredients: 2 tomatoes + 1 onion, 2 onions + 1 cabbage, and 3 fish. The soups and the ingredients could be delivered or placed

in the pot in any order. At each time step, we recorded the game state and action taken by the participant.

Goal Prediction Model. We use the Bayesian formulation (Equation 4.1) to predict the human goal in the Overcooked experiment. The predicted goal is given by $\arg\max_G P(G|\xi)$ where ξ is the observed partial trajectory. We use Maximum Entropy Inverse Reinforcement Learning (MaxEntIRL) [99] to estimate the cost function C_θ that is modeled as a linear function of the state features f_s, $C_\theta(s) = \theta^T f_s$. We used the position of the player as the features in our experiments. The cost of a trajectory is the sum of the cost of states in the trajectory. MaxEntIRL models the trajectories ξ from the training data as a Boltzmann distribution $p_\theta(\xi) = \frac{1}{Z}\exp(-C_\theta(\xi))$, where Z is the partition function computed via dynamic programming. We optimize the cost function parameters θ by maximizing the likelihood given training data $\max_\theta \sum_{\xi \in D} p(\xi|\theta)$ [99].

Conditions. We compare four conditions (Fig. 4.3a–d):

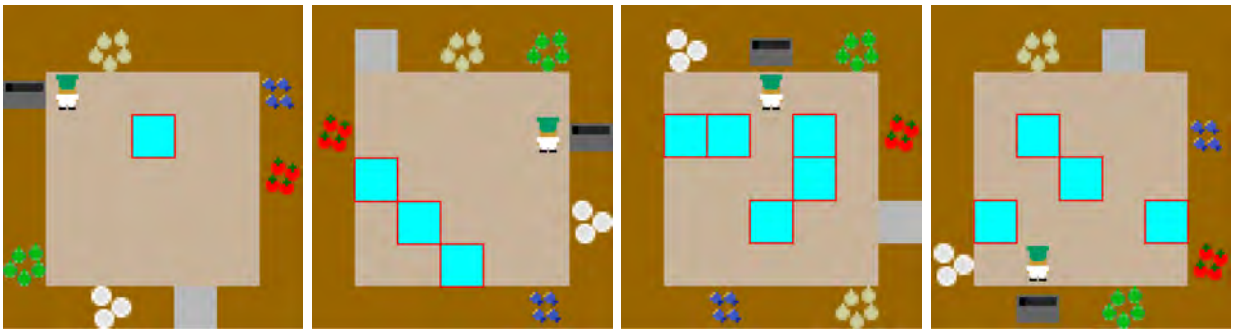
Random: We sample 1000 environments where each workstation is randomly placed in the perimeter, and each inner cell has a 20% chance of being assigned a virtual obstacle. This is the same as the initialization step of the MAP-Elites implementation. We then sort the layouts by the legibility score and select the layout with the median score (Fig. 4.3a).

Efficient: We use MAP-Elites to minimize the distance traveled to complete a task (Fig. 4.3b). This condition represents a common objective when designing workspace configurations — optimizing for task efficiency.

Legible-Efficient: The MAP-Elites objective function includes terms for both maximizing legibility and minimizing the distance traveled (Fig. 4.3c).

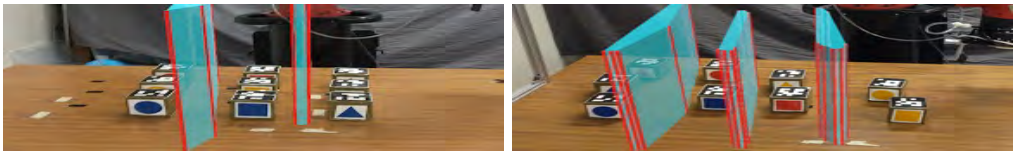
Legible: Our approach — using MAP-Elites to maximize legibility (Fig. 4.3d).

Workspace Generation. The workstations of the Overcooked game consist of the cooking pot, dish, serving, and four ingredient stations: onions, tomatoes, cabbages, and fish. For multiple agents to efficiently collaborate, an agent has to predict the ingredients the other agents are picking up so that efforts aren't duplicated. Therefore, for the objective function and evaluation, we only consider trajectories starting from the pot or serving stations and ending at the dish or ingredient

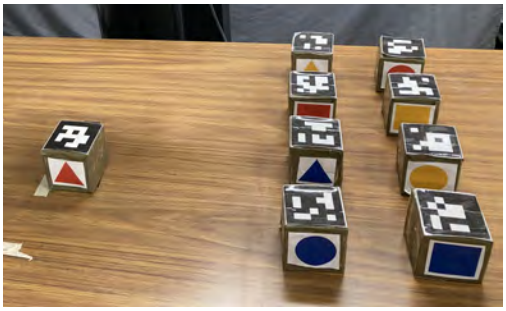


(a) Random (b) Efficient (c) Legible + Efficient (d) Legible

(e) Baseline (f) Placement Optimized



(g) Virtual Obstacle Optimized (h) Both Optimized



(i) Desired configuration

Figure 4.3: (a–d) Overcooked layouts and (e–h) initial cube configurations used in our experiments. The environments generated by our approach, (d) *Legible* and (h) *Both Optimized*, optimize object and virtual obstacle (shown in cyan with red edges) placements to elicit legible human motion. (i) The tabletop experiment involves the human and the robot collaboratively placing cubes into the desired configuration — two columns on the right with a given ordering.

stations. We allow the algorithm to move the workstations in the perimeter which is representative of reorganizing objects. The algorithm can also add or remove virtual obstacles onto any space within the perimeter. We add a constraint that the workstations have to be at least two cells away so that trajectories will be longer than a single step.

All conditions, except the Random condition, are generated using MAP-Elites with different objective functions. The features are the number of obstacles and the ordering of workstation placements from top to bottom and left to right. After 1000 iterations of the improvement phase, we use the solution with the highest legibility score in the map.

1) Initialization. The workspace is randomly sampled by finding non-overlapping placements for each workstation in the perimeter. For each cell not in the perimeter, there is a 20% chance the cell is a virtual obstacle. We verify that the sampled workspace has valid paths between the workstations before placing it in the behavior performance map. This phase is run for 1000 iterations.

2) Improvement. The available perturbations are moving a station to an empty counter, swapping positions with another station, and adding or removing virtual obstacles.

4.4.3 Tabletop Experiment

Experimental Setup. We conducted an IRB-approved human subjects study with 12 participants (8 male, 4 female), with an age range of 19 to 31 years old ($M = 24.42$, $SD = 3.37$), recruited from a university campus to collect human hand trajectory data in different tabletop workspace setups. We collected 8 trajectories per participant for each of the four conditions, presented in a randomized, counterbalanced order. The study involves the human and an autonomously operating Sawyer 7 DoF manipulator robot working collaboratively to place cubes into a desired configuration (Fig. 4.3i). To prevent the robot from picking up the same cube, the participant is asked to first reach for a cube while the robot maintains a probability distribution over the possible cubes the human is reaching for in real time. Once the robot is sufficiently confident of the human's goal, the robot will select its own cube to pick up and move to grasp it. The human and robot team

continues to pick a cube each until the task is completed. The precedence constraints are set such that the first column of the desired configuration must be completed before the second column can start.

Goal Prediction Model. We use a time series multivariate Gaussian model [66] to predict human goals given the human hand trajectory collected from the tabletop experiment. For a training data set D, we use dynamic time warping (DTW) [55] to align the trajectories so they all have length K. For each time step $k \in K$ and goal G, we train a multivariate Gaussian with mean $\mu_G[k] = \frac{1}{|D|}\sum_{i=1}^{|D|} f_i[k]$ and covariance $\Sigma_G[k] = \frac{1}{|D|-1}\sum_{i=1}^{|D|}(f_i[k]-\mu[k])(f_i[k]-\mu[k])^T$ where $f_i[k]$ are the feature values of the ith trajectory at time step k. At prediction time, the probability of a goal G given a partial trajectory ξ is $P(G|\xi) \propto \prod_{k=1}^{K}[\mathcal{N}(\mu_G[k], \Sigma_G[k])]^{\frac{1}{K}}$. The goal with the highest probability is the predicted goal. We use two features, the x and y positions of the hand, captured via an Intel RealSense RGB camera and a real-time hand tracking algorithm [84].

Conditions. We compare workspace setups using our optimization against **Baseline**, where the cubes are initially sorted by their color and shape (Fig. 4.3e). We also perform an ablation study, removing the algorithm's ability to organize the cubes or project virtual obstacles. As such, the remaining conditions are:

Placement Optimized: We optimize for legibility by only rearranging the cubes, with no virtual obstacles (Fig. 4.3f).

Virtual Obstacle Optimized: The cubes are sorted by their color and shape, and we optimize for legibility by projecting virtual obstacles (Fig. 4.3g).

Both Optimized: Our approach that optimizes for legibility by rearranging the cubes and projecting virtual obstacles (Fig. 4.3h).

Workspace Generation. The feature dimensions for MAP-Elites are the minimum distance between the cubes and the ordering of cubes by the x-axis. In contrast to the layout generation for the Overcooked game, we do not add virtual obstacles in the initialization step. Due to the continuous state space and the difficulty of randomly sampling a useful virtual obstacle, we choose fixed size virtual obstacles and insert them between two randomly selected cubes in the improve-

ment phase. By employing this heuristic, we can effectively explore configurations that result in altered human reaching motions, which are approximated using shortest paths in a visibility graph. We experimented with virtual obstacles of different sizes and found that obstacles measuring 30 cm $\times$ 1 cm were sufficient in inducing distinct reaching motions.

1) Initialization. We randomly sample (x, y) positions for the cubes and ensure that they don't overlap. The positions are continuous values uniformly sampled within the permissible boundaries.

2) Improvement. The improvement step consists of first changing cube positions and then adding virtual obstacles. We sample new cube positions from a Gaussian centered at the cube with variance = 7 cm. The virtual obstacles are placed between two randomly sampled cubes.

The boundaries of the algorithmically generated virtual obstacles are passed to an AR interface, implemented using a Microsoft HoloLens 2 head-mounted display. The AR interface renders those obstacles directly in the environmental context of the shared workspace as holograms of cyan barriers with red outlines (Fig. 4.3h). These barriers appear to the human as if they are physically located in the environment, and indicate regions of the environment the human should not enter.

4.5 Results

In both experiments, we run a stratified 4-fold cross validation with 3 repeats and different randomization in each repetition. The stratified cross validation preserves the percentage of samples for each class (i.e., the valid goals at each time step) in each fold. A prediction is defined as correct if the single highest probability value in the probability distribution is the same as the target goal. A trajectory is the human motion when reaching towards a goal: x, y positions in the Overcooked grid and hand positions in the x-y plane for the tabletop experiment. We use the one-way analysis of variances (ANOVA) and perform post-hoc analysis using Tukey's HSD test for multiple comparisons to test for effects between condition pairs. We plot Tukey's Q critical value as the width of the shaded area in the accuracy figures such that overlaps indicate an absence of a statistically significant difference.

4.5.1 Overcooked

The Bayesian predictor results show that the *Legible* condition elicits significantly higher or comparable accuracy than the best performer among other conditions except when 30% of the trajectory has been observed, supporting hypothesis *H1*. The counterintuitive bump at 30% is because the optimization penalizes incorrect predictions more for longer trajectories (Eqn. 4.2). Thus, it made a trade-off: the accuracy is lower when predicting goals about 30% into a subtask but higher for longer trajectories. Using the shortest Manhattan distance as a heuristic for the cost function produces lower mean accuracy when only a small percentage of the trajectory has been observed, suggesting that IRL enables faster accurate goal predictions which supports hypothesis *H2*.

4.5.2 Tabletop Experiment

The environment generated by our approach, *Both Optimized*, elicits significantly higher prediction accuracy than baseline environments when less than 50% of the trajectory has been observed. When using the shortest distance heuristic — where the closest cube to the current hand position is the predicted goal — all conditions elicit lower accuracy compared to the predictions when using the Gaussian models; the shortest distance heuristic exhibits poor performance even when the entire trajectory is observed. Both hypotheses *H1* and *H2* are supported by these results.

To evaluate the learning curve of the Gaussian model, we perform stratified splits cross validation with 10 repeats. *Both Optimized* elicits significantly higher accuracy than the baselines with 33% training data and achieves around 90% accuracy with 50% training data, supporting our hypothesis *H3* that our approach generates environments where the training of prediction models is more data-efficient.

The Gaussian models in the *Both Optimized* condition have less covariance, a measure of uncertainty, compared to the Gaussian models trained in baseline environment configurations. Table 4.1 shows the average determinant and trace of the covariance matrices of the Gaussian

Table 4.1: The average determinant (det) and trace of the covariance matrices of the multivariate Gaussian models.

	Baseline	Placement Optimized	Virtual Obstacle Optimized	Both Optimized
Det	9.36e-06	8.50e-06	6.14e-06	**1.09e-06**
Trace	7.12e-03	5.90e-03	3.90e-03	**2.77e-03**

models. The determinant is a measure of the magnitude of variation among the variables, and the trace is the sum of the variances of the individual variables but does not consider the correlations between the variables. The Gaussian models trained in the *Both Optimized* layout have lower values for both the determinant and trace compared to the models trained in baseline environments.

Overall, our results show that our workspace configuration approach reduces the uncertainty in human motion data and improves the accuracy of goal prediction models in planar navigation and tabletop manipulation tasks. The models are also more data-efficient in our generated environments, requiring less data to reach their best performance levels as compared to those trained in baseline environments.

4.6 Chapter Summary and Future Work

In this chapter, we introduced an algorithmic approach for autonomous workspace optimization to improve robot predictions of a human collaborator's goals. By rearranging physical objects in the workspace and projecting AR-based virtual obstacles into the environment prior to interaction, the robot influences the human into more legible behavior during task execution, thereby reducing the uncertainty of their motion. Through dual experiments in 2D navigation and tabletop manipulation, we show that our approach results in more accurate model predictions across two distinct goal inference methods, requiring less data to achieve these correct predictions. Importantly, we demonstrate that environmental adaptations can be discovered and leveraged to compensate for shortfalls of prediction models in otherwise unstructured settings.

Our approach is applicable for domains where the following conditions hold: 1) multiple agents share the same physical space and the agents do not have access to other agents' controllers

or decision making processes, 2) the environment allows physical or virtual configurations, and 3) environment configuration can be performed prior to the interaction. By improving goal prediction, we envision that our framework enhances human-robot teaming across domains such as shared autonomy for assistive manipulation, warehouse stocking, and cooking assistance, among others. Through our results, we demonstrate the generalizability of our overall approach across varied tasks, environments, and human motion prediction models, showing its potential to realize fluency improvements in a number of human-robot collaborative domains.

One limitation is that the improvement of our approach on prediction models is dependent on the task and the environment. If the configurability of the environment is limited, the effect size of our algorithm will likely be small. Future work should explore beyond objective measures of human predictability, investigating the subjective implications of our approach on the human's experience (i.e., do more complex configurations increase cognitive load for the human?). Lastly, it would be valuable to evaluate our approach over extended interactions to determine whether virtual obstacles remain necessary after multiple interactions or if humans naturally adapt to more legible trajectories after "training."

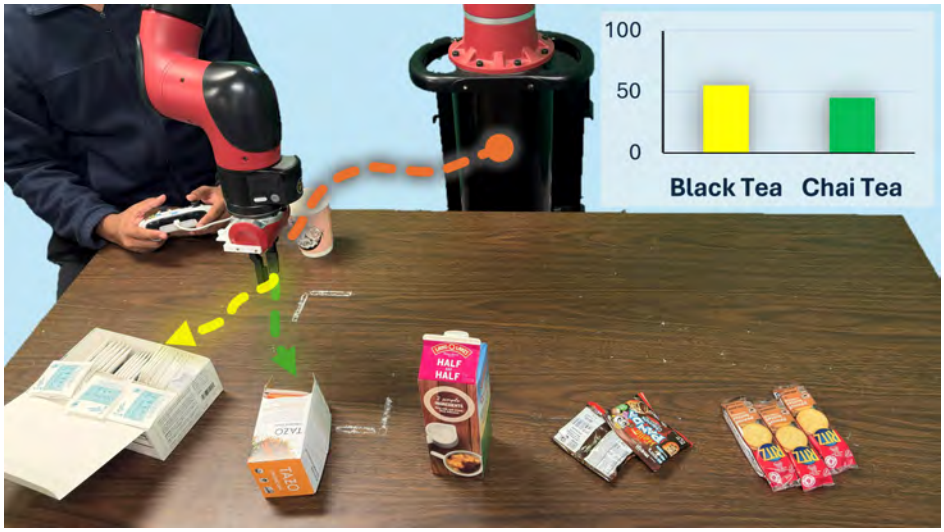
Chapter 5

Environment Design for Reliable Shared Autonomy

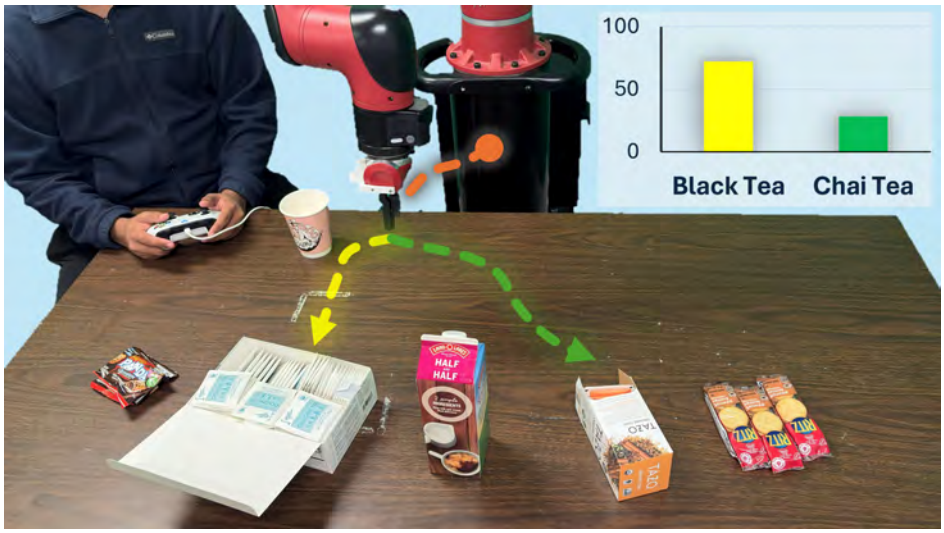
Chapter overview. In Chapter 4, we optimized workspace configurations by maximizing the **legibility** of candidate goals along a nominal trajectory, using cross-entropy between the inferred goal distribution and the true goal as the objective. That formulation assumes a deterministic nominal trajectory and does not model uncertainty in the user's inputs, so the optimizer does not account for how execution noise perturbs the trajectory or how that noise affects goal distinguishability. In this chapter, we propose a new workspace optimization objective that explicitly accounts for inference uncertainty under a stochastic user model. We present conditions under which the optimized workspace provides a probabilistic guarantee of correct goal inference, and we empirically evaluate the approach in simulation and on a physical robot across a range of shared-autonomy scenarios.

We consider a shared-autonomy setting in which a user controls a robotic manipulator through a low-dimensional interface (e.g., a joystick), while the robot continuously infers the intended goal from the observed trajectory and commits to autonomous assistance once sufficiently confident. We treat the physical arrangement of objects as a design variable that shapes the difficulty of this inference problem under stochastic user input. Under a noisy control model, object placement determines how distinguishable candidate goals remain once execution noise is accounted for, and thus how reliably the robot can infer and commit to the correct goal. We therefore design workspaces that are robust to input uncertainty, improving both inference accuracy and decision timing. As a result, we obtain a probabilistic correctness guarantee: under bounded Gaussian noise, the true

goal is identified with probability at least $1 - \alpha$, where α is a user-specified failure probability.



(a) Baseline setup. Without workspace optimization, the robot struggles to accurately infer which tea bag the user intends to pick up, leading to ambiguous goal predictions.



(b) Optimized workspace. The robot quickly infers that the user is picking up a black tea bag to place in the cup.

Figure 5.1: Given an initial trajectory (orange), the robot infers a probability distribution over possible goals (yellow and green). The workspace optimization introduced in this chapter rearranges objects to enhance the accuracy and speed of intent prediction under noisy user inputs.

5.1 Introduction

Shared autonomy enables a human and a robot to collaboratively control a robotic system by blending human input with autonomous assistance. Human control remains crucial in unstructured or safety-critical environments, where full autonomy is unreliable due to factors such as perceptual uncertainty, task ambiguity, or the need to leverage human expertise, as commonly seen in assistive robotics [18, 27] and surgical robotics [1]. Pure teleoperation, however, imposes a high cognitive and physical burden on the operator, since a low-dimensional interface (e.g., a joystick or keyboard) must be used to control a high-dimensional robotic system, such as a 7-DoF manipulator. Shared-autonomy systems alleviate this burden by inferring the user's intent from observed inputs and assisting execution once the inferred goal is sufficiently confident [46]. Prior work has focused on improving intent inference through algorithmic advances, including planning under uncertainty [34] and leveraging vision-language models to identify user goals [46], while treating the environment as fixed.

In Chapter 4, we showed that object arrangement shapes the difficulty of goal inference for collaborative reaching, and that a MAP-Elites search over workspace configurations produces layouts that elicit more legible human motion. That chapter's objective minimizes the cross-entropy between the Boltzmann posterior over goals and the true goal along a nominal trajectory. The formulation is an **average-case** objective: a layout may achieve low cross-entropy on average while still leaving one pair of goals nearly indistinguishable under realistic execution noise. For shared autonomy, where the robot must commit to a single goal and cede control, the worst-case is what matters.

In this chapter, we formulate workspace design for shared autonomy as an optimization problem over object placements that maximizes the separability of candidate goals under a probabilistic user model. Building on a bounded Gaussian model of joystick noise, we derive an objective that provides a probabilistic correctness guarantee: the system identifies the correct goal with probability at least $1 - \alpha$. This yields a principled connection between environment design, inference

uncertainty, and performance guarantees. We evaluate our approach in six tabletop manipulation scenarios of increasing difficulty in simulation and demonstrate that optimized workspaces improve goal inference reliability compared to random baselines. We also deploy the framework on a physical Sawyer manipulator in three realistic shared-autonomy scenarios: tea preparation with snack selection, LEGO block sorting, and assistive feeding.

This chapter (i) formulates workspace design for shared autonomy as an optimization problem that explicitly accounts for intent inference under user input noise; (ii) derives a probabilistic correctness guarantee linking workspace geometry to goal inference reliability; and (iii) demonstrates empirically that optimized workspaces improve inference performance in simulation and support deployment on a physical manipulator.

5.2 Workspace Optimization for Intent Inference

We formalize the problem of arranging tabletop objects to enable a shared-autonomy system to infer the human operator's intended goal both rapidly and reliably. The key observation is that object placement is a controllable design variable: by selecting object configurations, the robot directly shapes the geometry of the underlying inference problem.

5.2.1 Preliminaries

Consider a set of M candidate goal objects at positions $\theta = \{\mathbf{g}_1, \ldots, \mathbf{g}_M\} \subset \mathbb{R}^d$ on a shared workspace, and a robot end-effector (EE) initialized at $\mathbf{x}_0 \in \mathbb{R}^d$. At each discrete timestep t, the human provides a noisy control input

$$\mathbf{u}_t = \boldsymbol{\pi}_h(\mathbf{x}_t, \mathbf{g}^*) + \boldsymbol{\epsilon}_t, \qquad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \tag{5.1}$$

where $\mathbf{g}^* \in \theta$ denotes the user's intended goal, $\boldsymbol{\pi}_h(\mathbf{x}, \mathbf{g}) = (\mathbf{g} - \mathbf{x})/\|\mathbf{g} - \mathbf{x}\|$ is a unit vector pointing toward $\mathbf{g}$, and $\boldsymbol{\Sigma}$ is the joystick noise covariance. Under first-order integrator dynamics, the EE evolves as $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{u}_t \, \Delta t$, producing a trajectory $\xi = \mathbf{x}_{0:T}$.

The robot maintains a belief over goals using the Boltzmann-rational model [17] introduced

in Chapter 4, which assigns higher probability to goals that render the observed trajectory efficient according to a cost function C:

$$P(\mathbf{g} \mid \xi_{0:t}) \propto P(\mathbf{g}) \, \exp\big(-\beta \, C(\xi_{0:t}, \, \mathbf{g})\big), \tag{5.2}$$

where $\beta > 0$ is a rationality coefficient. The system commits to a goal once the posterior confidence exceeds a threshold, $\max_{\mathbf{g}} P(\mathbf{g} \mid \xi_{0:t}) \geq p_{\text{thresh}}$. Following Dragan et al. [17], we take the cost function $C(\xi_{0:t}, \mathbf{g})$ to be the sum of the trajectory length (under the ℓ_2 norm) and the remaining distance-to-go to the goal.

5.2.2 Workspace Optimization with Probabilistic Guarantees

To design workspaces that are robust to execution uncertainty, we introduce a formulation that directly enforces **probabilistic correctness** of goal inference under input noise. The approach is based on a margin criterion that measures separation between competing goal hypotheses.

Cost difference and margin. Consider the nominal trajectory $\xi_{0:T}$ toward the true goal $\mathbf{g}^*$. We define the pairwise cost difference between the true goal and an alternative $\mathbf{g}$ as

$$\delta c(\mathbf{g}) = C(\xi_{0:T}, \mathbf{g}) - C(\xi_{0:T}, \mathbf{g}^*), \tag{5.3}$$

which measures how much less efficient the trajectory is under $\mathbf{g}$ compared to $\mathbf{g}^*$. When $\delta c(\mathbf{g}) > 0$, the trajectory is more consistent with the true goal. Under the Boltzmann model, the log-posterior margin $M_g(\xi)$ is defined as the log ratio of posterior probabilities between the true goal and an alternative:

$$M_g(\xi) = \log \frac{P(\mathbf{g}^* \mid \xi)}{P(\mathbf{g} \mid \xi)} = \beta \, \delta c(\mathbf{g}) + \log \frac{P(\mathbf{g}^*)}{P(\mathbf{g})}. \tag{5.4}$$

We assume uniform priors, which reduces the mean margin to $m_g = \beta \, \delta c(\mathbf{g})$.

From margin to correctness. Correct goal inference requires that the true goal have higher posterior probability than all alternatives:

$$M_g(\xi) > 0, \qquad \forall \mathbf{g} \neq \mathbf{g}^*. \tag{5.5}$$

In practice, execution noise makes $M_g(\xi)$ a random variable, and we enforce this condition with high probability:

$$\mathbb{P}(M_g(\xi) > 0,\ \forall \mathbf{g} \neq \mathbf{g}^*) \geq 1 - \alpha. \tag{5.6}$$

Uncertainty from trajectory noise. Joystick noise perturbs the executed trajectory, introducing uncertainty into the cost difference for goal inference. Under integrator dynamics with isotropic Gaussian control noise, the terminal state $\mathbf{x}_T$ satisfies

$$\mathbf{x}_T \sim \mathcal{N}(\bar{\mathbf{x}}_T, \Sigma_{x_T}), \qquad \Sigma_{x_T} = T\,\sigma_u^2\,\Delta t^2\,\mathbf{I}, \tag{5.7}$$

where $\bar{\mathbf{x}}_T$ is the nominal terminal state obtained in the absence of noise, σ_u^2 is the per-timestep joystick noise variance, T is the horizon, Δt is the timestep, and $\mathbf{I}$ is the identity matrix. The covariance Σ_{x_T} captures how uncertainty accumulates over time. Because $C(\xi, \mathbf{g})$ is generally nonlinear in the trajectory, δc is not Gaussian even if $\mathbf{x}_T$ is. To obtain a tractable approximation, we linearize $\delta c(\mathbf{g})$ with respect to the terminal state around the nominal endpoint:

$$\delta c(\mathbf{g}; \mathbf{x}_T) \approx \delta c(\mathbf{g}; \bar{\mathbf{x}}_T) + \nabla_{\mathbf{x}_T} \delta c(\mathbf{g})\big|_{\bar{\mathbf{x}}_T}^{\top} (\mathbf{x}_T - \bar{\mathbf{x}}_T). \tag{5.8}$$

The approximation is locally valid when execution noise remains within a neighborhood of the nominal trajectory where higher-order terms are bounded; we formalize this condition and bound the linearization error in Appendix F. Let $\mathbf{a}_g := \nabla_{\mathbf{x}_T} \delta c(\mathbf{g})\big|_{\bar{\mathbf{x}}_T}$ denote the sensitivity of the cost difference with respect to the terminal state. Under this first-order approximation, the pairwise log-posterior margin is Gaussian with variance $v_g = \beta^2\,\mathbf{a}_g^{\top} \Sigma_{x_T} \mathbf{a}_g$. This variance captures how sensitive the goal separation is to execution noise: small perturbations in the terminal state that induce large changes in $\delta c(\mathbf{g})$ make the corresponding goals more easily confused.

Probabilistic guarantee. For each competing goal $\mathbf{g} \neq \mathbf{g}^*$, the pairwise log-posterior margin is modeled as a Gaussian random variable under the linearized approximation, $M_g \sim \mathcal{N}(m_g, v_g)$. Correct identification of the true goal requires $M_g \geq 0\ \forall \mathbf{g} \neq \mathbf{g}^*$. For a fixed competing goal $\mathbf{g}$, the misclassification probability is bounded by a standard Gaussian tail:

$$\mathbb{P}(M_g < 0) = \Phi\left(-\frac{m_g}{\sqrt{v_g}}\right),$$

where Φ is the standard normal CDF. To enforce $\mathbb{P}(M_g < 0) \leq \alpha_g$, it suffices that $m_g \geq \Phi^{-1}(1 - \alpha_g)\sqrt{v_g}$. Applying a union bound over all $M-1$ alternative goals, we allocate $\alpha_g = \alpha/(M-1)$ to each pairwise comparison and define the **trajectory margin slack**:

$$s_g(\theta) := m_g(\theta) - \Phi^{-1}(1 - \alpha_g)\sqrt{v_g(\theta)}. \tag{5.9}$$

A positive slack ensures robustness of the margin against execution noise. This yields the following sufficient condition for correct inference.

Theorem 1 (Sufficient condition for correct goal inference) Fix a workspace parameter θ and true goal $\mathbf{g}^*$. Suppose:

(1) The trajectory induces a Gaussian terminal state $\mathbf{x}_T \sim \mathcal{N}(\bar{\mathbf{x}}_T, \Sigma_{x_T})$.

(2) For each $\mathbf{g} \neq \mathbf{g}^*$, the pairwise log-posterior margin M_g admits a first-order linear approximation and is Gaussian: $M_g \sim \mathcal{N}(m_g, v_g)$.

(3) The local linearization error is negligible within a $(1-\alpha)$-probability neighborhood of $\bar{\mathbf{x}}_T$.

If $s_g(\theta) > 0 \ \forall \mathbf{g} \neq \mathbf{g}^*$, then the probability of correct goal inference satisfies

$$\mathbb{P}\big(\hat{\mathbf{g}} = \mathbf{g}^*\big) \geq 1 - \alpha. \tag{5.10}$$

The slack $s_g(\theta)$ measures how robustly the true goal is separated from an alternative $\mathbf{g}$ under execution noise. The mean term m_g captures nominal distinguishability between goals, while the variance term v_g penalizes sensitivity to noise. Ensuring positive slack for all competing goals guarantees that the true goal remains the most likely hypothesis with probability at least $1 - \alpha$.

Workspace design objective. We optimize the workspace to maximize the worst-case margin slack across all possible true goals and competing alternatives:

$$\max_{\theta} \min_{\mathbf{g}^*} \min_{\mathbf{g} \neq \mathbf{g}^*} s_g(\theta). \tag{5.11}$$

When this objective is positive, the system guarantees correct identification of the goal with probability at least $1 - \alpha$ for **every** goal, providing a design-time certificate of inference reliability.

5.2.3 Quality-Diversity Optimization

The workspace design objective (Eq. 5.11) is non-convex and multi-modal, with multiple disjoint regions of the layout space yielding comparable objective values. In such settings, standard optimization methods are prone to mode collapse or local maxima, returning a single suboptimal solution that does not reflect the full structure of the solution space. As in Chapter 4, we use MAP-Elites [54], which maintains an archive of solutions indexed by low-dimensional behavioral descriptors. Each cell in the archive stores the highest-quality solution observed for a given region of the behavior space. Here, the quality of a layout is measured by its worst-case trajectory margin slack, and the behavior descriptors capture geometric properties of the layout.

Behavior space. To encourage diversity in spatial configurations, we define the behavior space by (i) the mean pairwise distance between objects, which captures overall spatial spread, and (ii) the offset of the object centroid from the workspace center, which captures global asymmetry.

Optimization. We instantiate the quality-diversity framework using CMA-ME [24], which leverages covariance matrix adaptation to efficiently explore the continuous layout space. The algorithm iteratively proposes candidate layouts, evaluates their margin slack, and inserts them into the archive based on their behavioral descriptors.

Constraints. All candidate layouts must satisfy geometric feasibility constraints, including workspace bounds, inter-object clearance, and minimum distance from the robot's initial pose. These constraints ensure that archived solutions are physically realizable and safe for execution. Implementation details are provided in Appendix G.

5.3 Evaluation

We evaluate workspace optimization across six tabletop manipulation scenarios of increasing difficulty in simulation. The experiments address two research questions. **RQ1:** Does workspace optimization improve goal inference compared to unoptimized (random) layouts? **RQ2:** How does the approach scale with the number of candidate goals? We consider scenarios with $M = 2$ to

Table 5.1: Evaluation environments. M is the number of pick objects. "Total" includes fixed objects that are not picked but contribute clutter.

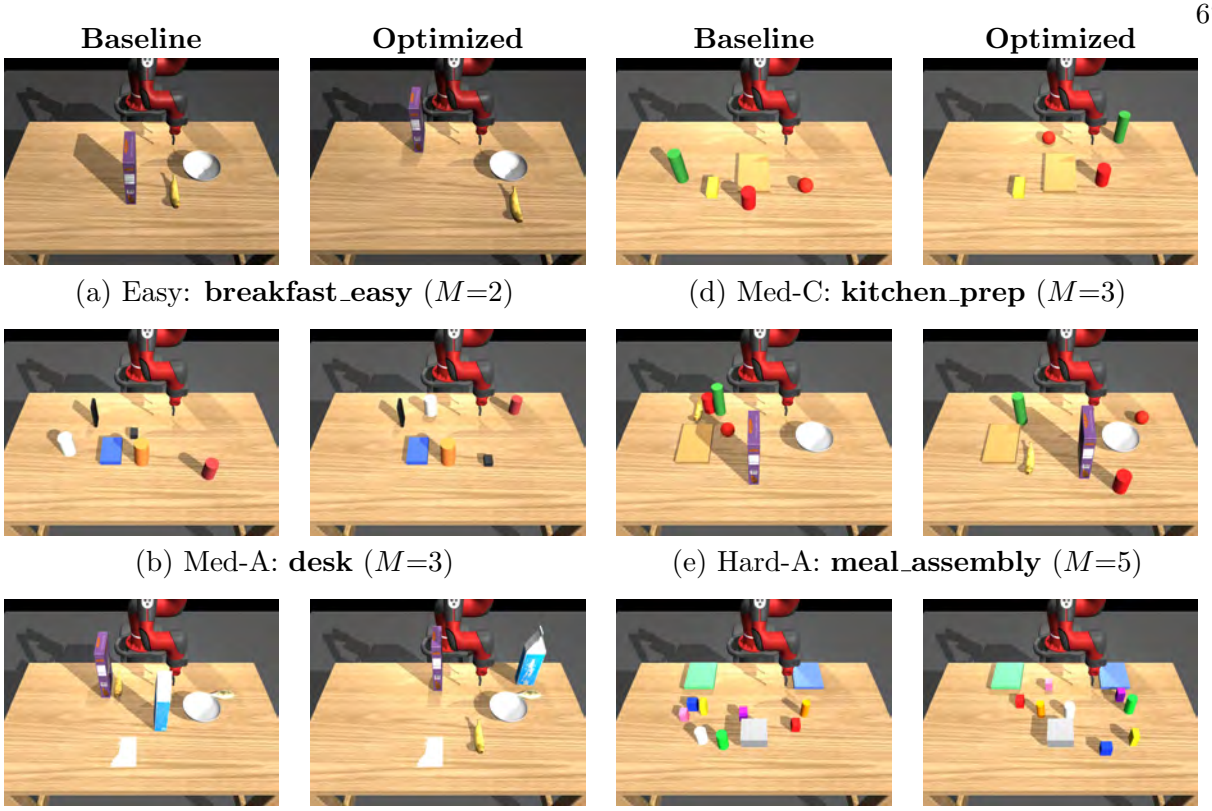
Tier	**Scene**	M	**Total**	**Pick objects**
Easy	breakfast_easy	2	3	cereal box, yellow banana
Med-A	desk	3	5	white mug, black stapler, maroon pen cup
Med-B	breakfast	3	6	cereal box, yellow banana, milk carton
Med-C	kitchen_prep	3	5	red apple, red can, green bottle
Hard-A	meal_assembly	5	7	cereal box, yellow banana, red apple, red can, green bottle
Hard-B	cluttered	8	11	8 colored blocks and cylinders

$M = 8$ competing objects and report both inference performance and optimization cost.

5.3.1 Experimental Setup

Scenarios. We construct six tabletop manipulation tasks using a shared Sawyer workspace (Fig. 5.2), grouped into Easy, Medium, and Hard settings based on the number of candidate goals M. In each task, the robot must infer the human operator's intended object from early end-effector motion. All objects remain valid candidates at every decision point, requiring the system to repeatedly disambiguate among M alternatives. After each pick, the selected object is returned to its initial position and the inference process is reset, allowing multiple independent trials within the same layout. To evaluate robustness across task sequences, we consider multiple permutations of object pick orderings; for tasks with $M \geq 5$, we sample up to 120 permutations uniformly at random.

Grasp library. Each object is assigned a fixed grasp pose in SE(3) from a hand-authored library that includes both top-down and side grasps. The choice of grasp type reflects the intended use of the object in downstream tasks (e.g., side grasps for objects that are typically lifted or poured,



(a) Easy: **breakfast_easy** (M=2)

(d) Med-C: **kitchen_prep** (M=3)

(b) Med-A: **desk** (M=3)

(e) Hard-A: **meal_assembly** (M=5)

(c) Med-B: **breakfast** (M=3)

(f) Hard-B: **cluttered** (M=8)

Figure 5.2: Random vs. SE(3)-optimized workspace layouts. Each pair shows a random baseline (left) and an SE(3)-optimized layout (right) produced by MAP-Elites over (x, y, yaw) using the trajectory-margin objective. All layouts satisfy identical geometric constraints. By optimizing in SE(3), the method jointly places and orients objects to maintain separability of reachable grasp trajectories across M candidate goals. Tasks share the same workspace and robot home pose.

and top grasps for objects that are picked and placed from above). For each object, the grasp pose is parameterized relative to its geometry, with offsets computed from the object's bounding box so that the gripper fingertips make contact with the object surface. This ensures consistent and physically feasible grasps across different object instances. The grasp library is fixed across all experimental conditions; only object positions and orientations vary. Fixing grasps isolates the effect of workspace layout on goal inference and task execution.

Conditions. We compare two layout conditions:

(1) **Random** ($n = 30$ layouts per scene): Object positions (x, y) are sampled uniformly within the workspace bounds, subject to minimum inter-object clearance (12 cm), minimum dis-

tance from the robot end-effector (15 cm radius), and table-edge margins. Object orientations are sampled uniformly from $[-\pi, \pi)$.

(2) **ME-optimized**: Layouts are generated using CMA-ME, producing a diverse archive of feasible solutions. We report results for the highest-performing elite in the archive. The behavior space is defined by the mean pairwise distance and the centroid offset from the center of the workspace.

Simulator. All experiments are conducted using the MuJoCo simulator with the Sawyer robot controlled via 3D Jacobian pseudoinverse velocity control at 20 Hz. User input is simulated using the stochastic policy $\mathbf{u}_t = \boldsymbol{\pi}_h(\mathbf{x}_t, \mathbf{g}^*) + \boldsymbol{\epsilon}_t$ with Gaussian noise $\sigma_v = 0.03\,\mathrm{m/s}$ and a maximum end-effector speed of 0.05 m/s. The end-effector is driven toward the target grasp pose, and the inference loop terminates when either the posterior probability exceeds a commit threshold, the EE arrives at the goal pose, or a 30 s timeout is reached.

Observer. Goal inference is performed using the Boltzmann-rational observer of Eq. 5.2, where C is the path-efficiency cost ratio using an SE(3) distance metric with rotation weight $\lambda_R = 0.04$. We use $\beta = 5$ and a default commit threshold $p_{\text{thresh}} = 0.9$.

Feasibility check. Before evaluation, each layout is validated for grasp feasibility. For each object, we test inverse-kinematics reachability and collision-free execution at the corresponding grasp pose. Objects without a valid IK solution or with collisions between the robot and non-target objects are marked infeasible. During inference, infeasible grasps are excluded from candidate goals; however, they are counted as failures in the task success metric to reflect the practical impact of poor layout design.

Metrics.

- **Argmax accuracy**: fraction of picks where the final inferred goal matches the true goal.
- **Time to inference** T_{infer}: average time (in seconds) from end-effector motion onset to the first commitment, averaged over all pick steps.

Table 5.2: Trajectory-margin slack, argmax accuracy, and time to inference across tiers. Random denotes 30 random feasible layouts (mean ± std). ME-optimized is the best layout found with MAP-Elites. Slack is the worst-case pairwise trajectory-margin slack computed at Bonferroni-corrected significance $\alpha = 0.05$; positive slack implies a nominal argmax-accuracy guarantee of $\geq 1 - \alpha = 95\%$ under the straight-line, linear-Gaussian approximation (Eq. 5.9).

Tier	**Layout**	M	Slack	T_{infer} (s)	Argmax acc.
Easy	Random	2	—	5.5 ± 2.7	88%
	ME-optimized		8.66	6.1	100%
Med-A	Random	3	—	7.2 ± 3.1	94%
	ME-optimized		7.68	4.1	100%
Med-B	Random	3	—	8.8 ± 2.7	89%
	ME-optimized		6.52	7.7	100%
Med-C	Random	3	—	7.3 ± 4.0	97%
	ME-optimized		7.91	3.5	100%
Hard-A	Random	5	—	10.5 ± 2.8	83%
	ME-optimized		2.28	10.3	100%
Hard-B	Random	8	—	12.3 ± 1.7	67%
	ME-optimized		0.03	12.0	97%

5.4 Results

5.4.1 Inference Accuracy and Speed

We evaluate whether workspace optimization improves intent inference (RQ1) and how performance scales with the number of candidate goals (RQ2). Table 5.2 summarizes results across all six scenarios.

Accuracy. ME-optimized layouts consistently achieve higher argmax accuracy than random baselines across all scenarios, directly supporting **RQ1**. While random layouts degrade from 88% (Easy, $M = 2$) to 67% (Hard-B, $M = 8$), optimized layouts maintain near-perfect performance (100% in most cases and 97% in Hard-B). This gap widens as the number of candidate goals increases, providing strong evidence for **RQ2**. As the workspace becomes more cluttered and the number of competing hypotheses grows, random placements increasingly produce ambiguous trajectories, whereas optimized layouts preserve distinguishability between goals. Workspace optimization is therefore particularly beneficial in high-ambiguity regimes.

Time to inference. Workspace optimization reduces the time required for intent inference in moderate-difficulty scenarios, further supporting **RQ1**. In Med-A and Med-C, optimized layouts enable earlier disambiguation (e.g., 4.1 s vs. 7.2 s and 3.5 s vs. 7.3 s), indicating that increased goal separation allows the robot to reach confident predictions with less trajectory evidence. This trend is less consistent in the Easy tier, where improvements in accuracy do not always translate to faster inference. As the number of candidate goals increases (RQ2), the time-to-inference gap narrows: in Hard-A and Hard-B, both random and optimized layouts require longer trajectories to disambiguate among many competing goals. Optimized layouts nonetheless maintain substantially higher accuracy, indicating that in high-M regimes the primary benefit shifts from speed to robustness.

5.4.2 Optimizer Compute Cost

Table 5.3 reports wall-clock time for MAP-Elites across all tiers. Optimization time ranges from 340–920 s per scene and does not increase monotonically with M. Instead, runtime is influenced by the structure of the search space, while archive coverage remains consistently high (93–100%). The approach scales favorably in practice, even as the number of candidate goals increases.

A key advantage of MAP-Elites is the diversity of solutions produced. With hundreds of feasible layouts spanning the behavior space, practitioners can select among top-performing candidates at deployment time. This provides robustness to potential mismatch between the optimization objective and the runtime inference model.

5.4.3 Real-World System Demonstration

We deploy the framework in three shared-autonomy scenarios on a physical Sawyer manipulator: (a) making tea with snacks, (b) sorting LEGO blocks, and (c) assistive feeding with brownies and fruits (Fig. 5.3). In the tea-making task, the user prepares tea and then selects one of several snacks. In the LEGO sorting task, the robot assists the user in sorting colored blocks, without

Table 5.3: Optimizer compute cost (wall-clock seconds, single seed). We also report the number of solutions in the MAP-Elites archive and the archive coverage.

Tier	M	ME (s)	Elites	Coverage
Easy	2	597	374	93.5%
Med-A	3	920	399	99.8%
Med-B	3	653	400	100%
Med-C	3	727	397	99.3%
Hard-A	5	461	400	100%
Hard-B	8	340	400	100%

prior knowledge of which container corresponds to each color. In the assistive feeding task, the user chooses between fruits (e.g., strawberry or grape) and brownies, with the option to dip them into chocolate or sprinkles before feeding.

In the baseline condition, objects are placed intuitively — similar or related items are grouped together (e.g., teas placed side-by-side, or fruits clustered together). While this arrangement appears natural, it can make intent inference ambiguous for the robot, leading to lower confidence during teleoperation and delayed assistance.

The optimized condition applies the workspace optimization of Eq. 5.11 to generate object configurations that improve goal separability. Unlike the simulation experiments, which assume a fixed set of candidate goals, real-world tasks are sequential and context-dependent. We exploit this structure by optimizing only the subset of relevant goals at each stage of the task, as defined by the task graph (in the style of Appendix B). This produces layouts that adapt to the current decision context, enabling more reliable disambiguation as the task progresses. By strategically placing and orienting objects to maximize predictability, the robot infers the user's intent earlier and transitions more effectively to autonomous assistance. These demonstrations show how environment design can be leveraged to improve shared autonomy in realistic, multi-stage tasks. We note that this section provides a qualitative demonstration of system integration; a controlled user study is deferred to future work (see Section 5.5).

Figure 5.3: Three shared-autonomy scenarios: (a) grabbing snacks for tea, (b) sorting LEGO blocks into containers, and (c) reaching dips for a strawberry. The first row (BASELINE) presents intuitive arrangements, where similar or related objects are grouped together. The second row (ME-OPTIMIZED) presents the configurations produced by the MAP-Elites algorithm, which optimizes the workspace for improved intent prediction.

5.5 Chapter Summary and Future Work

In this chapter, we presented a framework for improving intent inference in shared autonomy through workspace design. Rather than treating the environment as fixed, we showed that object placement fundamentally shapes the difficulty of the inference problem. By formulating workspace design as an optimization problem over goal separability, we derived a margin-based probabilistic correctness condition that guarantees the true goal is identified with probability at least $1 - \alpha$ under bounded noise. Optimized workspaces consistently improved inference accuracy across a range of scenarios and maintained high performance as the number of candidate goals increased. While gains in time to inference were most pronounced in moderate-difficulty settings, the primary benefit in more complex scenarios was improved robustness, with optimized layouts preserving reliable goal disambiguation despite increased ambiguity. We further deployed the framework in real-world tabletop tasks, showing that task-aware workspace optimization can adapt to multi-stage interactions by focusing on relevant subsets of goals.

This chapter sharpens the formulation of Chapter 4 in two ways that matter for shared-autonomy deployments. First, by modeling joystick noise explicitly, the optimizer targets the worst-case goal pair rather than an average-case cross-entropy, which is the regime where a committing controller actually fails. Second, a positive margin slack produces a design-time certificate: before the user ever touches the joystick, the robot can verify that any layout it proposes satisfies a $(1-\alpha)$ correctness guarantee.

One limitation is that the guarantee rests on the Gaussian joystick model and the local linearization of the pairwise cost difference around the nominal trajectory. In practice, human joystick inputs exhibit heavier tails, systematic biases, and state-dependent noise that this model does not capture. Future work should extend the formulation to richer user models (e.g., direction-dependent covariance or learned input distributions) and characterize how the guarantee degrades when the model is misspecified. A complementary direction is to deploy the framework in a user study that measures how the margin-slack objective affects subjective fluency and trust, building on the null subjective result reported in Chapter 4. Finally, the current approach optimizes the workspace offline; a natural extension is to update the layout online as the task progresses and the set of relevant candidate goals changes.

Bibliography

[1] Garth H Ballantyne and Fred Moll. The da Vinci telerobotic surgical system: the virtual operative field and telepresence surgery. Surgical Clinics, 83(6):1293–1304, 2003.

[2] Shray Bansal, Rhys Newbury, Wesley Chan, Akansel Cosgun, Aimee Allen, Dana Kulić, Tom Drummond, and Charles Isbell. Supportive actions for manipulation in human-robot coworker teams. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 11261–11267. IEEE, 2020.

[3] Erdem Biyik and Dorsa Sadigh. Batch active preference-based learning of reward functions. In Conference on robot learning. PMLR, 2018.

[4] Erdem Biyik, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, Dorsa Sadigh, et al. Asking easy questions: A user-friendly approach to active reward learning. In Conference on Robot Learning. PMLR, 2020.

[5] Mohamed El Amine Boudella, Evren Sahin, and Yves Dallery. Kitting optimisation in just-in-time mixed-model assembly lines: assigning parts to pickers in a hybrid robot–operator kitting system. International Journal of Production Research, 56(16):5475–5494, 2018.

[6] Yavuz A Bozer and Leon F McGinnis. Kitting versus line stocking: A conceptual framework and a descriptive model. International Journal of Production Economics, 28(1):1–19, 1992.

[7] Thomas Kleine Buening, Victor Villin, and Christos Dimitrakakis. Environment design for inverse reinforcement learning. In Proceedings of the 41st International Conference on Machine Learning, pages 24808–24828, 2024.

[8] Judith Bütepage, Hedvig Kjellström, and Danica Kragic. Anticipating many futures: Online human motion prediction and generation for human-robot interaction. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 4563–4570, 2018. doi: 10.1109/ICRA.2018.8460651.

[9] Antonio Casimiro Caputo, Pacifico Marcello Pelagagge, and Paolo Salini. A model for planning and economic comparison of manual and automated kitting systems. International Journal of Production Research, 59(3):885–908, 2021.

[10] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. Advances in neural information processing systems, 32, 2019.

[11] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. Advances in neural information processing systems, 30, 2017.

[12] Lucas Pinheiro Cinelli, Matheus Araújo Marins, Eduardo Antonio Barros Da Silva, and Sérgio Lima Netto. Variational methods for machine learning with applications to deep networks, volume 15. Springer, 2021.

[13] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. Annals of Operations Research, 153(1):235–256, Sep 2007. ISSN 1572-9338. doi: 10.1007/s10479-007-0176-2. URL https://doi.org/10.1007/s10479-007-0176-2.

[14] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. Nature, 521(7553):503–507, 2015.

[15] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. Annals of operations research, 134(1):19–67, 2005.

[16] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. Advances in neural information processing systems, 2020.

[17] Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. Legibility and predictability of robot motion. In 2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 301–308, 2013. doi: 10.1109/HRI.2013.6483603.

[18] Andrea Eirale, Mauro Martini, and Marcello Chiaberge. Human following and guidance by autonomous mobile robots: A comprehensive review. IEEE Access, 2025.

[19] Kevin Ewacha, Ivan Rival, and George Steiner. Permutation schedules for flow shops with precedence constraints. Operations research, 38(6):1135–1139, 1990.

[20] Muhammad Fahad, Zhuo Chen, and Yi Guo. Learning how pedestrians navigate: A deep inverse reinforcement learning approach. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 819–826, 2018. doi: 10.1109/IROS.2018.8593438.

[21] AFH Fansuri, ANM Rose, NMZ Nik Mohamed, and H Ahmad. The challenges of lean manufacturing implementation in kitting assembly. In IOP Conference Series: Materials Science and Engineering, volume 257, page 012069. IOP Publishing, 2017.

[22] Matthew Fontaine and Stefanos Nikolaidis. Differentiable quality diversity. Advances in Neural Information Processing Systems, 34:10040–10052, 2021.

[23] Matthew C Fontaine and Stefanos Nikolaidis. Evaluating human–robot interaction algorithms in shared autonomy via quality diversity scenario generation. ACM Transactions on Human-Robot Interaction (THRI), 11(3):1–30, 2022.

[24] Matthew C Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference, pages 94–102, 2020.

[25] Matthew C Fontaine, Ya-Chuan Hsu, Yulun Zhang, Bryon Tjanaka, and Stefanos Nikolaidis. On the importance of environments in human-robot coordination. arXiv preprint arXiv:2106.10853, 2021.

[26] David Fridovich-Keil, Andrea Bajcsy, Jaime F Fisac, Sylvia L Herbert, Steven Wang, Anca D Dragan, and Claire J Tomlin. Confidence-aware motion prediction for real-time collision avoidance. The International Journal of Robotics Research, 39(2-3):250–265, 2020. doi: 10.1177/0278364919859436. URL https://doi.org/10.1177/0278364919859436.

[27] Ze Fu, Pinhao Song, Yutong Hu, and Renaud Detry. TASC: Task-aware shared control for teleoperated manipulation. arXiv preprint arXiv:2509.10416, 2025.

[28] Michael R Garey, David S Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. Mathematics of operations research, 1(2):117–129, 1976.

[29] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco Madrid-Cuevas, and Rafael Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. Pattern Recognition, 51, 10 2015. doi: 10.1016/j.patcog.2015.09.023.

[30] David E Goldberg and John Henry Holland. Genetic algorithms and machine learning. Machine Learning, 1988.

[31] Himanshu Gupta, Bradley Hayes, and Zachary Sunberg. Intention-aware navigation in crowds with extended-space pomdp planning. In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, pages 562–570, 2022.

[32] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive Metropolis algorithm. Bernoulli, 7(2):223 – 242, 2001.

[33] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In Advances in psychology, volume 52, pages 139–183. Elsevier, 1988.

[34] Shervin Javdani, Henny Admoni, Stefania Pellegrinelli, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared autonomy via hindsight optimization for teleoperation and teaming. The International Journal of Robotics Research, 37(7):717–742, 2018.

[35] Lisa Jeanson, JM Christian Bastien, Alexandre Morais, and Javier Barcenilla. Tell me how you kit, i'll tell you how you think. In H-Workload 2018: 2nd International Symposium on Human Mental Workload: Models and Applications, page 232, 2018.

[36] Jamil Joundi, Peter Conradie, Jan Van Den Bergh, and Jelle Saldien. Understanding and exploring operator needs in mixed model assembly. In EISMS19, Workshop on Research and Practice Challenges for Engineering Interactive Systems while Integrating Multiple Stakeholders Viewpoints, 2019.

[37] Haresh Karnan, Elvin Yang, Garrett Warnell, Joydeep Biswas, and Peter Stone. Wait, that feels familiar: Learning to extrapolate human preferences for preference-aligned path planning. In 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024.

[38] Sarah Keren, Luis Pineda, Avigdor Gal, Erez Karpas, and Shlomo Zilberstein. Equi-reward utility maximizing design in stochastic environments. HSDIP, 2017:19, 2017.

[39] Anis Koubaa. Robot Operating System (ROS): The Complete Reference (Volume 2). Springer Publishing Company, Incorporated, 1st edition, 2017. ISBN 331954926X.

[40] Anagha Kulkarni, Sarath Sreedharan, Sarah Keren, Tathagata Chakraborti, David E Smith, and Subbarao Kambhampati. Designing environments conducive to interpretable robot behavior. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 10982–10989. IEEE, 2020.

[41] Przemyslaw A. Lasota and Julie A. Shah. Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration. Human Factors, 57(1):21–33, 2015. doi: 10.1177/0018720814565188. URL https://doi.org/10.1177/0018720814565188. PMID: 25790568.

[42] Przemyslaw A. Lasota and Julie A. Shah. A multiple-predictor approach to human motion prediction. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 2300–2307, 2017. doi: 10.1109/ICRA.2017.7989265.

[43] Kimin Lee, Laura M Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In International Conference on Machine Learning, pages 6152–6163. PMLR, 2021.

[44] Joel Lehman and Kenneth O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11, page 211–218, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450305570. doi: 10.1145/2001576.2001606. URL https://doi.org/10.1145/2001576.2001606.

[45] Qinghua Li, Zhao Zhang, Yue You, Yaqi Mu, and Chao Feng. Data driven models for human motion prediction in human-robot collaboration. IEEE Access, 8:227690–227702, 2020. doi: 10.1109/ACCESS.2020.3045994.

[46] Huihan Liu, Rutav Shah, Shuijing Liu, Jack Pittenger, Mingyo Seo, Yuchen Cui, Yonatan Bisk, Roberto Martín-Martín, and Yuke Zhu. Casper: Inferring diverse intents for assistive teleoperation with vision language models. arXiv preprint arXiv:2506.14727, 2025.

[47] Ren.C Luo and Licong Mai. Human intention inference and on-line human hand motion prediction for human-robot collaboration. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5958–5964, 2019. doi: 10.1109/IROS40897.2019.8968192.

[48] Riccardo Maderna, Matteo Poggiali, Andrea Maria Zanchettin, and Paolo Rocco. An online scheduling algorithm for human-robot collaborative kitting. In 2020 IEEE international conference on robotics and automation (ICRA), pages 11430–11435. IEEE, 2020.

[49] Jim Mainprice, Rafi Hayne, and Dmitry Berenson. Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces. IEEE Transactions on Robotics, 32(4):897–908, 2016. doi: 10.1109/TRO.2016.2581216.

[50] Michail Maniadakis, Emmanouil Hourdakis, Markos Sigalas, Stylianos Piperakis, Maria Koskinopoulou, and Panos Trahanias. Time-aware multi-agent symbiosis. Frontiers in Robotics and AI, 7, 2020. ISSN 2296-9144. doi: 10.3389/frobt.2020.503452. URL https://www.frontiersin.org/article/10.3389/frobt.2020.503452.

[51] Luisa Mao, Garrett Warnell, Peter Stone, and Joydeep Biswas. Pacer: Preference-conditioned all-terrain costmap generation. IEEE Robotics and Automation Letters, 2025.

[52] Gustav Markkula and Mehmet Dogar. How accurate models of human behavior are needed for human-robot interaction? for automated driving? arXiv preprint arXiv:2202.06123, 2022.

[53] Julieta Martinez, Michael J. Black, and Javier Romero. On human motion prediction using recurrent neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[54] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. arXiv preprint arXiv:1504.04909, 2015. URL https://arxiv.org/pdf/1504.04909.pdf.

[55] Meinard Müller. Dynamic time warping. Information retrieval for music and motion, pages 69–84, 2007.

[56] Mr Swapnil Nikam, A Joel, and J Dutta. Design and implementation of kitting trolley for just in time production in textile industry. International Research Journal of Engineering and Technology (IRJET), 5(4):1301–1304, 2018.

[57] Olle Nilsson and Antoine Cully. Policy gradient assisted map-elites. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 866–875, 2021.

[58] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014. URL https://github.com/bayesian-optimization/BayesianOptimization.

[59] Ernesto Nunes and Maria Gini. Multi-robot auctions for allocation of tasks with temporal constraints. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, page 2110–2216. AAAI Press, 2015. ISBN 0262511290.

[60] OpenStreetMaps. Planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org, 2017.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[62] Stefania Pellegrinelli, Henny Admoni, Shervin Javdani, and Siddhartha Srinivasa. Human-robot shared workspace collaboration via hindsight optimization. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 831–838, 2016. doi: 10.1109/IROS.2016.7759147.

[63] Shaoting Peng, Haonan Chen, and Katherine Driggs-Campbell. Towards uncertainty unification: A case study for preference learning. arXiv preprint arXiv:2503.19317, 2025.

[64] Mark Pfeiffer, Ulrich Schwesinger, Hannes Sommer, Enric Galceran, and Roland Siegwart. Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2096–2101, 2016. doi: 10.1109/IROS.2016.7759329.

[65] Neha Prajapat and Ashutosh Tiwari. A review of assembly optimisation applications using discrete event simulation. International Journal of Computer Integrated Manufacturing, 30 (2-3):215–228, 2017.

[66] Claudia Pérez-D'Arpino and Julie A. Shah. Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 6175–6182, 2015. doi: 10.1109/ICRA.2015.7140066.

[67] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In IJCAI, volume 7, pages 2586–2591, 2007.

[68] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. The International Journal of Robotics Research, 39(8):895–935, 2020.

[69] Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In Robotics: Science and Systems, 2017.

[70] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[71] S Shaikh, SVG Cobb, D Golightly, JI Segal, and CM Haslegrave. Effects of takt time on physical and cognitive demands in a mixed model assembly line and a single model assembly line. In Contemporary Ergonomics and Human Factors 2012: Proceedings of the international conference on Ergonomics & Human Factors 2012, Blackpool, UK, 16-19 April 2012, page 137. CRC Press, 2012.

[72] Thomas B Sheridan. Human–robot interaction: status and challenges. Human factors, 58 (4):525–532, 2016.

[73] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25, 2012.

[74] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11(4):341–359, Dec 1997. ISSN 1573-2916. doi: 10.1023/A:1008202821328. URL `https://doi.org/10.1023/A:1008202821328`.

[75] Richard S Sutton. Reinforcement learning: An introduction. A Bradford Book, 2018.

[76] Aaquib Tabrez, Matthew B Luebbers, and Bradley Hayes. A survey of mental modeling techniques in human–robot teaming. Current Robotics Reports, 1:259–267, 2020.

[77] Emanuel Todorov. Mujoco: A physics engine for model-based control. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5026–5033, 2012.

[78] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. arXiv preprint arXiv:2407.17032, 2024.

[79] Beverly Townsend. Line Balancing And Jit Kitting. Boca Raton, Fl: Crc Press, 2012.

[80] Yi-Shiuan Tung, Kayleigh Bishop, Bradley Hayes, and Alessandro Roncone. "bilevel optimization for just-in-time robotic kitting and delivery via adaptive task segmentation and scheduling". IEEE International Conference on Robot & Human Interactive Communication, 2022.

[81] Yi-Shiuan Tung, Matthew B Luebbers, Alessandro Roncone, and Bradley Hayes. Improving human legibility in collaborative robot tasks through augmented reality and workspace preparation. In 6th International Workshop on Virtual, Augmented, and Mixed-Reality for Human-Robot Interactions (VAM-HRI). ACM, Mar 2023.

[82] Yi-Shiuan Tung, Matthew B Luebbers, Alessandro Roncone, and Bradley Hayes. Workspace optimization techniques to improve prediction of human motion during human-robot collaboration. In Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction, 2024.

[83] Yi-Shiuan Tung, Gyanig Kumar, Wei Jiang, Bradley Hayes, and Alessandro Roncone. CRED: Counterfactual reasoning and environment design for active preference learning. In 2026 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2026.

[84] Andrey Vakunov, Chuo-Ling Chang, Fan Zhang, George Sung, Matthias Grundmann, and Valentin Bazarevsky. Mediapipe hands: On-device real-time hand tracking. 2020. https://mixedreality.cs.cornell.edu/workshop.

[85] Peter J. M. van Laarhoven and Emile H. L. Aarts. Simulated annealing: Theory and applications. In Mathematics and Its Applications, 1987.

[86] Milos Vasic and Aude Billard. Safety issues in human-robot interactions. In 2013 IEEE International Conference on Robotics and Automation, pages 197–204, 2013. doi: 10.1109/ICRA.2013.6630576.

[87] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. arXiv preprint arXiv:1901.01753, 2019.

[88] Webots. http://www.cyberbotics.com. URL `http://www.cyberbotics.com`. Open-source Mobile Robot Simulation Software.

[89] Dominik Widmann and Yiannis Karayiannidis. Human motion prediction in human-robot handovers based on dynamic movement primitives. In 2018 European Control Conference (ECC), pages 2781–2787, 2018. doi: 10.23919/ECC.2018.8550170.

[90] Ronald Wilcox, Stefanos Nikolaidis, and Julie Shah. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. Robotics, 8(441):10–15607, 2013.

[91] Nils Wilde, Dana Kulić, and Stephen L Smith. Active preference learning using maximum regret. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.

[92] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[93] Sofya Zeylikman, Sarah Widder, Alessandro Roncone, Olivier Mangin, and Brian Scassellati. The hrc model set for human-robot collaboration research. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1845–1852. IEEE, 2018.

[94] Arthur Zhang, Harshit Sikchi, Amy Zhang, and Joydeep Biswas. Creste: Scalable mapless navigation with internet scale priors and counterfactual guidance. arXiv preprint arXiv:2503.03921, 2025.

[95] Haoqi Zhang, Yiling Chen, and David C Parkes. A general approach to environment design with one agent. In Twenty-First International Joint Conference on Artificial Intelligence. Citeseer, 2009.

[96] Yulun Zhang, Matthew C. Fontaine, Varun Bhatt, Stefanos Nikolaidis, and Jiaoyang Li. Multi-robot coordination and layout design for automated warehousing. In Edith Elkind, editor, Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23, pages 5503–5511. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/611. URL https://doi.org/10.24963/ijcai.2023/611. Main Track.

[97] Binghai Zhou and Zhaoxu He. A static semi-kitting strategy system of jit material distribution scheduling for mixed-flow assembly lines. Expert Systems with Applications, 184:115523, 2021.

[98] Xiaowei Zhu, S. Jack Hu, Yoram Koren, and Samuel P. Marin. Modeling of Manufacturing Complexity in Mixed-Model Assembly Lines. Journal of Manufacturing Science and Engineering, 130(5), 08 2008. ISSN 1087-1357. doi: 10.1115/1.2953076. URL https://doi.org/10.1115/1.2953076. 051013.

[99] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In AAAI, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[100] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. The International journal of robotics research, 32(9-10):1164–1193, 2013.

Appendix A

VAE for Tabletop Object Positions

This appendix provides additional details on the variational autoencoder (VAE) used to compress the tabletop environment space in Chapter 3.

We learn a low-dimensional latent space Z over tabletop object layouts with a convolutional VAE. Each environment is represented as a $C \times H \times W$ binary grid x (here $C = 7$ object channels; $H = W = 4$), where channel c encodes the occupancy of object c over the grid.

The encoder $q_\phi(z \mid x)$ consists of two stride-2 convolutions (32 and 64 filters, kernel 3×3, ReLU), followed by flattening and linear heads to the mean $\mu(x) \in \mathbb{R}^d$ and log-variance $\log \sigma^2(x) \in \mathbb{R}^d$ of a diagonal Gaussian in Z (latent dimension $d = 20$). We sample $z = \mu + \sigma \odot \epsilon$, $\epsilon \sim N(0, I)$.

The decoder $p_\theta(x \mid z)$ maps z through a linear layer back to the flattened conv feature size, then uses two stride-2 transposed convolutions ($64 \to 32$, $32 \to C$; kernel 4×4, ReLU) with a final sigmoid to produce per-cell, per-object probabilities $\hat{x} \in [0, 1]^{C \times H \times W}$.

Appendix B

Task Graph for the Overcooked Experiment

This appendix presents the task graph for the Overcooked game environment used in the user study of Chapter 4.

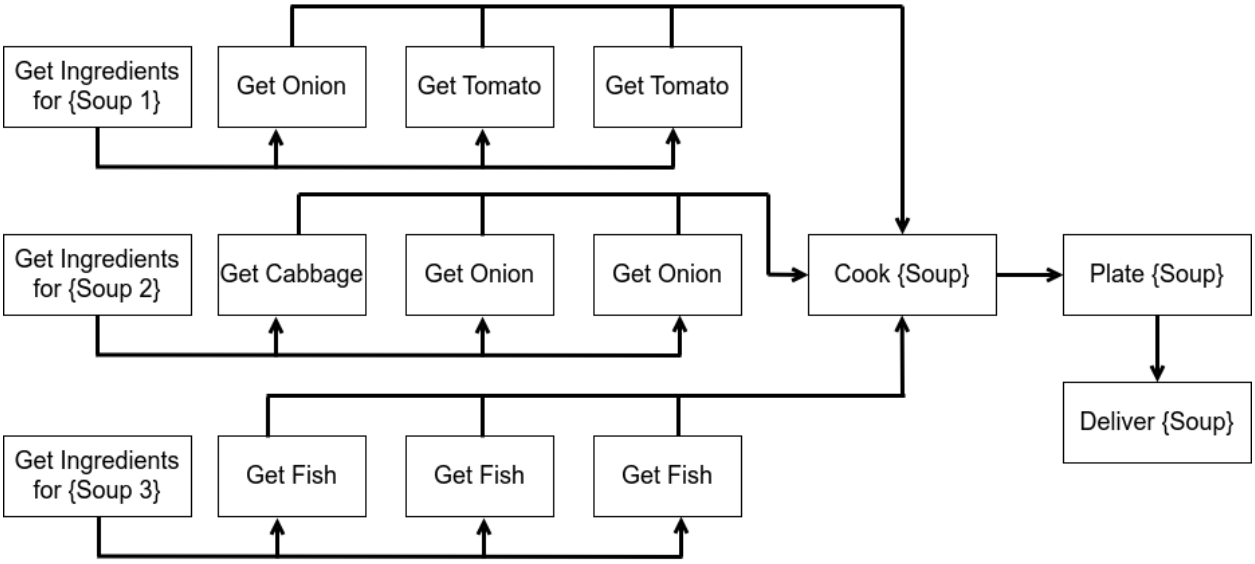
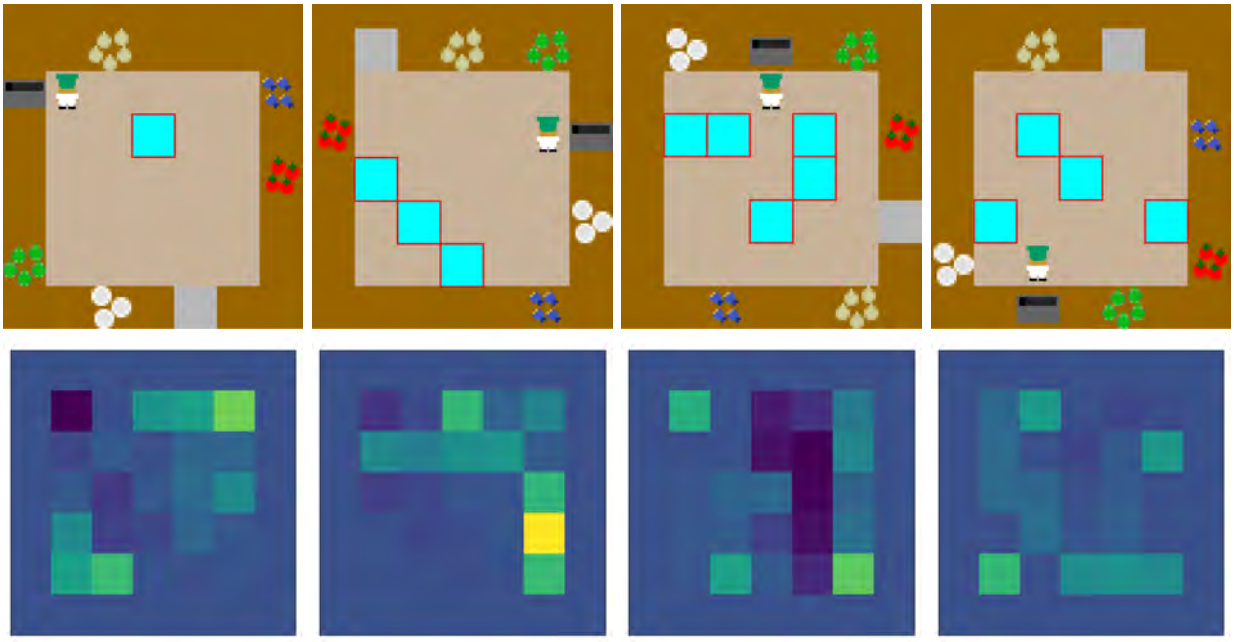


Figure B.1: Task graph for the Overcooked game environment. Arrows indicate precedence constraints such that if $t_i \rightarrow t_j$ then subtask t_i has to be completed before subtask t_j can begin. A total of 3 soups have to be delivered, and each soup consists of 3 ingredients. The soups and ingredients can be delivered or placed in any order.

Appendix C

Learned Reward Functions for Overcooked

This appendix shows the reward functions estimated via Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) for the four Overcooked conditions of Chapter 4.



(a) Random (b) Efficient (c) Legible + Efficient (d) Legible

Figure C.1: Learned reward functions from MaxEnt IRL.

We use MaxEnt IRL to estimate the reward functions for each layout. We use the negative of the reward function as the cost function in Equation 4.1 and predict the target goal of trajectories in the test data set. Figure C.1 shows the learned reward function normalized to values between 0

(dark color) and 1 (bright color) for one of the cross validation splits. As expected, the squares near workstations have higher rewards. We can also observe some preferred paths such as taking the right-most lane to go from the pot to the onion station in the *Legible + Efficient* condition.

Appendix D

Goal Prediction Results

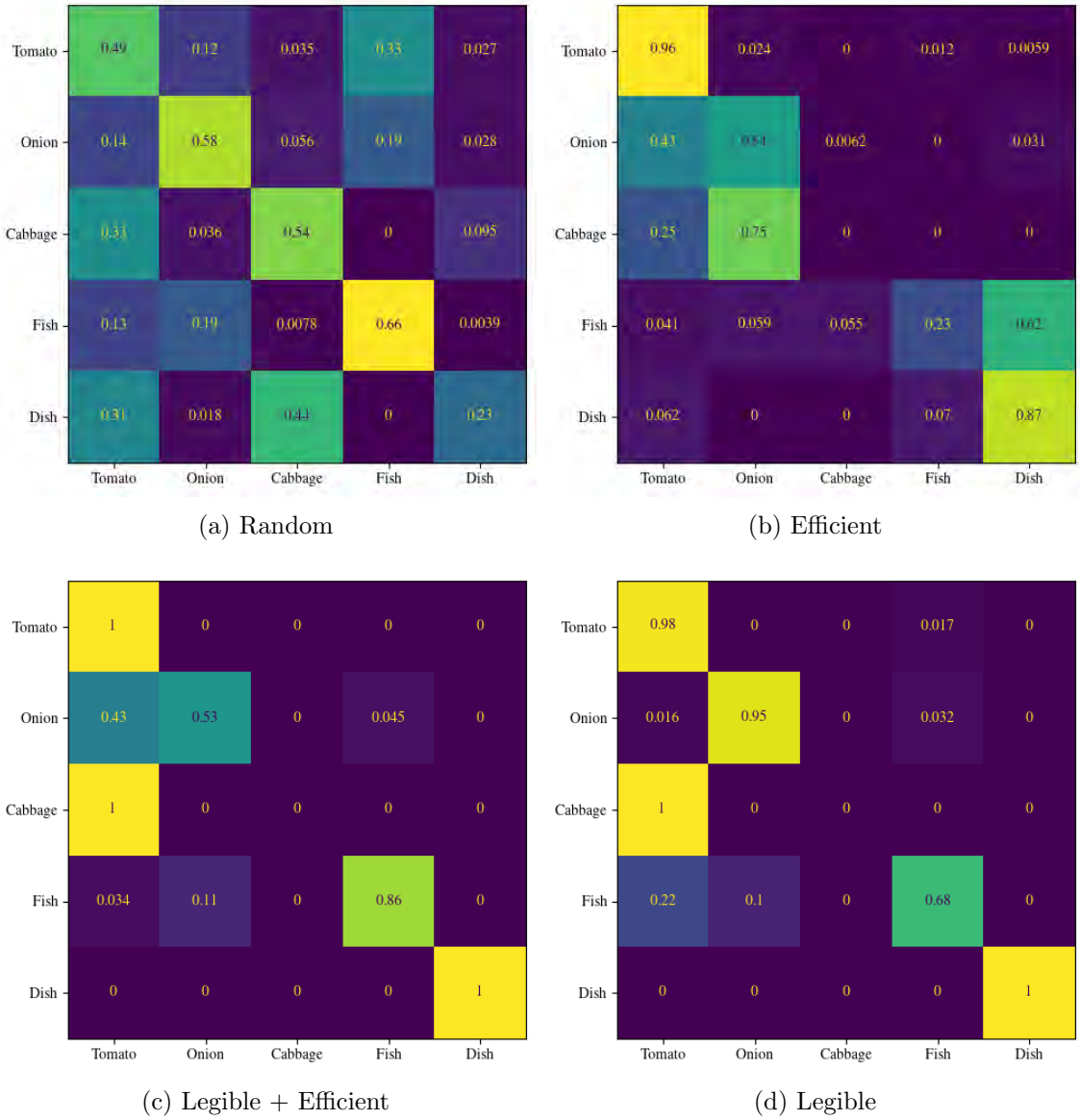
This appendix provides additional goal-prediction results (confusion matrices and F1/precision/recall metrics) for the Overcooked and tabletop experiments of Chapter 4.

Figure D.1 shows the confusion matrix for the goal prediction across the different conditions in Overcooked. The x-axis shows the true target, and the y-axis is the predicted target. The layout generated by our approach, *Legible*, has 100% correct predictions for the target goals tomato and dish. Due to the environment and task complexity, the optimization trades off the ability to predict the goal cabbage for accurate predictions of the other target goals. Cabbage is only used once in the recipes and thus incurs less cost for incorrect predictions. Note that we only include the results when a partial trajectory is observed since the goal prediction is always correct given the entire trajectory.

Table D.1: Overcooked Goal Prediction

	Random	Efficient	Legible + Efficient	Legible
F1	0.477	0.512	0.744	**0.799**
Precision	0.564	0.648	0.792	**0.824**
Recall	0.482	0.554	0.751	**0.812**

Tables D.1 and D.2 show the F1 score, precision, and recall of the goal predictions of the Bayesian predictor in Overcooked and the time series multivariate Gaussian in the tabletop experiments. We account for label imbalance by computing the metrics for each label and computing their average weighted by support. We use the implementation from scikit-learn [61].



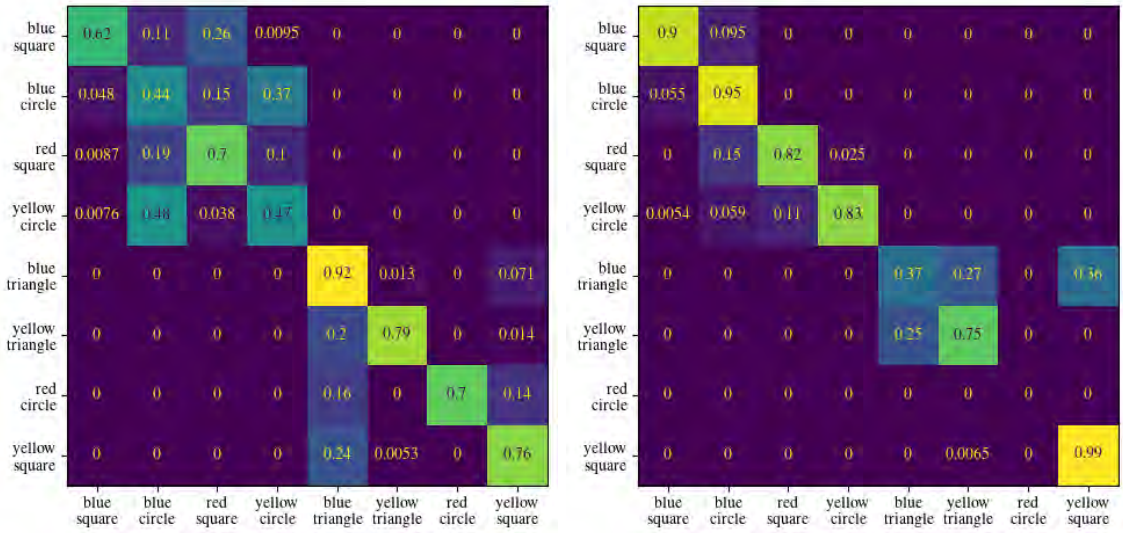
(a) Random (b) Efficient

(c) Legible + Efficient (d) Legible

Figure D.1: Confusion matrix for the Bayesian predictor in the different generated Overcooked layouts.

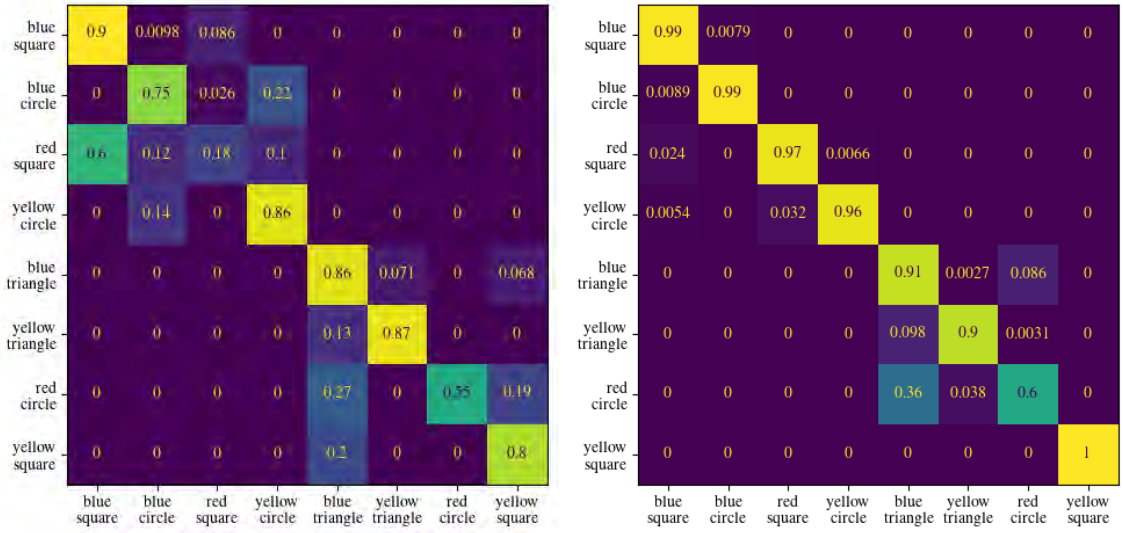
Table D.2: Tabletop Goal Prediction

	Baseline	Placement Optimized	Virtual Obstacle Optimized	Both Optimized
F1	0.704	0.753	0.796	**0.936**
Precision	0.751	0.770	0.803	**0.941**
Recall	0.689	0.761	0.800	**0.937**



(a) Baseline

(b) Placement Optimized



(c) Virtual Obstacle Optimized

(d) Both Optimized

Figure D.2: Confusion matrix for the time series multivariate Gaussian in the different generated workspace configurations. The x-axis shows the true target, and the y-axis is the predicted target.

Appendix E

Comparisons with other Optimization Methods

This appendix provides supplementary comparisons of MAP-Elites against Novelty Search with Local Competition (NSLC) [44] and Differential Evolution (DE) [74], for the Overcooked domain of Chapter 4.

We report the average legibility scores across 10 runs with different seeds (larger values are better) in Table E.1. This result shows that MAP-Elites outperforms NSLC and DE, but the best choice of optimization method is likely conditioned on the task and the environment. In the implementation, we set the hyperparameters such that they have the same number of iterations performing the update step for an individual from the population. MAP-Elites initializes the solution map with 100 random layouts and performs the improvement phase for 100 iterations. NSLC maintains a population size of 20 that evolves for 10 generations. DE performs a maximum of 200 generations over which the entire population is updated.

Table E.1: Comparisons of optimization methods on the best solution found (i.e., legibility scores) for different Overcooked environments.

	Square 5×5	Square 7×7	Square 9×9	Rectangle 9×5
DE	-129.28	-202.52	-229.22	-186.31
	(2.19)	(3.96)	(4.68)	(3.27)
NSLC	-95.44	-175.43	-190.54	-169.63
	(10.73)	(15.92)	(11.89)	(20.39)
MAP-Elites	$\mathbf{-86.37}$	$\mathbf{-167.88}$	$\mathbf{-173.52}$	$\mathbf{-162.97}$
	(7.55)	**(16.13)**	**(8.20)**	**(10.60)**

Appendix F

Local Validity of the Linearized Margin

This appendix supplements the derivation of the probabilistic correctness guarantee in Chapter 5 (Section 5.2.2). The pairwise cost difference $\delta c(\mathbf{g})$ is, in general, a nonlinear function of the trajectory and the terminal state. As a result, the log-posterior margin is not exactly Gaussian under Gaussian execution noise. To obtain a tractable probabilistic characterization, we approximated δc using a first-order Taylor expansion around the nominal terminal state $\bar{\mathbf{x}}_T$. We now show that this linearization is locally accurate with high probability, provided the execution noise remains within a neighborhood where the curvature of $\delta c(\mathbf{g})$ is bounded.

Proposition. Let $f_g(\mathbf{x}_T) := \delta c(\mathbf{g}; \mathbf{x}_T)$ denote the pairwise cost difference as a function of the terminal state. Assume f_g is twice continuously differentiable in a neighborhood $\mathcal{N}_g$ of $\bar{\mathbf{x}}_T$, and that its Hessian is bounded:

$$\|\nabla^2 f_g(\mathbf{x}_T)\|_2 \leq L_g, \qquad \forall \mathbf{x}_T \in \mathcal{N}_g. \tag{F.1}$$

Then for any $\mathbf{x}_T \in \mathcal{N}_g$, the second-order Taylor expansion gives

$$f_g(\mathbf{x}_T) = f_g(\bar{\mathbf{x}}_T) + \nabla f_g(\bar{\mathbf{x}}_T)^\top (\mathbf{x}_T - \bar{\mathbf{x}}_T) + R_g(\mathbf{x}_T), \tag{F.2}$$

where the remainder satisfies

$$|R_g(\mathbf{x}_T)| \leq \frac{L_g}{2} \|\mathbf{x}_T - \bar{\mathbf{x}}_T\|^2. \tag{F.3}$$

Suppose $\mathbf{x}_T \sim \mathcal{N}(\bar{\mathbf{x}}_T, \Sigma_{x_T})$, and let $r_\alpha > 0$ satisfy

$$\Pr(\|\mathbf{x}_T - \bar{\mathbf{x}}_T\| \leq r_\alpha) \geq 1 - \alpha. \tag{F.4}$$

If the ball $\{\mathbf{x}_T : \|\mathbf{x}_T - \bar{\mathbf{x}}_T\| \leq r_\alpha\}$ is contained in $\mathcal{N}_g$, then with probability at least $1 - \alpha$,

$$|R_g(\mathbf{x}_T)| \leq \frac{L_g}{2} r_\alpha^2. \tag{F.5}$$

Implication. Within this high-probability region, the nonlinear cost difference $f_g(\mathbf{x}_T)$ is well-approximated by its linearization. Therefore, the log-posterior margin can be approximated as a Gaussian random variable with bounded error. In particular, if

$$\frac{L_g}{2} r_\alpha^2 \ll m_g, \tag{F.6}$$

where m_g is the nominal separation margin, then the linearized model preserves the sign and magnitude of the margin with high probability. This justifies using the Gaussian approximation to derive the sufficient chance constraint for correct goal inference in Theorem 1.

Appendix G

Quality-Diversity Optimization Details for Shared Autonomy

This appendix provides implementation details for the quality-diversity optimization procedure described in Section 5.2.3 of Chapter 5.

Solution parameterization. Each candidate layout is represented by the (x, y, yaw) positions of the M movable objects, yielding a $3M$-dimensional decision vector.

Algorithm. We use CMA-ME [24], a variant of MAP-Elites that employs Covariance Matrix Adaptation Evolution Strategy (CMA-ES) emitters to explore the solution space. We use three emitters operating in parallel, each with a batch size of 30 and initial step size $\sigma_0 = 0.08$. Emitters are ranked using the two-stage improvement criterion described in [24].

Behavior space discretization. The archive is a 20×20 grid defined over two behavioral dimensions:

- mean pairwise distance between objects, and
- centroid offset from the workspace center.

The range of each dimension is determined by sampling 10,000 random layouts and computing the 5th and 95th percentiles of the resulting feature values.

Constraint handling. Candidate layouts are required to satisfy the following constraints:

- **Workspace bounds:** $x \in [0.30, 0.85]$ m, $y \in [-0.45, 0.45]$ m.
- **Minimum inter-object clearance:** $\|\mathbf{g}_i - \mathbf{g}_j\| \geq 0.12$ m $\;\forall i \neq j$.

- **Minimum distance from robot end-effector:** $\|\mathbf{g}_i - \mathbf{x}_0\| \geq 0.15$ m.

Constraint violations are handled via additive penalties in the objective function, discouraging infeasible layouts from entering the archive.

Evaluation. Each candidate layout is evaluated using the trajectory margin slack objective defined in Section 5.2.2. The archive stores the layout with the highest objective value in each cell.