

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

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

Bradley Hayes

Christoffer Heckman

Nisar Ahmed

Stefanos Nikolaidis

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

Optimizing Interaction Through Environment Design: Reward Alignment and Intent Prediction in
Human-Robot Systems

Thesis directed by Prof. Alessandro Roncone

For humans and robots to collaborate safely and fluently, the robot must anticipate its partner at multiple 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 should instead be treated as a first-class design variable that a robot can actively configure to elicit more informative human behavior, reduce ambiguity, and improve coordination.

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 establish environment design as a complementary paradigm to learning and inference in human-robot collaboration. Across all three settings, the robot improves interaction not only by interpreting human behavior, but also by shaping the conditions under

which that behavior is expressed. This perspective suggests that future robot-learning systems should jointly consider how they infer human intent, how they elicit and collect human behavior, and how the surrounding environment influences that behavior. More broadly, environment design may support not only the robot’s understanding of the human, but also mutual understanding between collaborators, enabling more adaptive, predictable, and trustworthy human–robot teams.

Dedication

To my family whose boundless love and unconditional support made it possible for me to pursue this dream.

Acknowledgements

I am deeply grateful to my advisors, Dr. Alessandro Roncone and Dr. Bradley Hayes, for their guidance and support throughout my doctoral studies. They have profoundly shaped how I think, approach problems, and conduct research. They gave me the independence to pursue my intellectual interests while providing thoughtful and detailed feedback at every stage of the research process, from developing ideas and implementing systems to writing papers, preparing presentations, and navigating professional development. They continually encouraged me to consider the broader significance and potential impact of my work, and they have been among my strongest advocates throughout my career. Thank you for the extraordinary time, effort, and care you invested in training me as a scientist.

I am also grateful to the many collaborators whose contributions helped shape the research presented in this dissertation: Anuj Pasricha, Kayleigh Bishop, Matthew Luebbers, Himanshu Gupta, Gyanig Kumar, Wei Jiang, Yuni Wu, and Heyang Huang. I would also like to thank Shivendra Agrawal and Aaquib Tabrez for the many research discussions, valuable feedback, and thoughtful suggestions they provided throughout this work.

Thank you to the members of the Human Interaction and Robotics Group and the Collaborative AI and Robotics Laboratory at the University of Colorado Boulder for your ideas, feedback, technical support, and countless hours of discussion. I am equally grateful for the time we spent together outside the lab—for the social gatherings, shared experiences, and moments of fun that helped us build friendships and support one another through the challenges of the Ph.D.

I would also like to thank my M.Eng. advisor, Dr. Julie Shah. The work conducted in her

lab introduced me to human–robot interaction and sparked my interest to the possibilities of the field. That experience inspired me to pursue the path that ultimately led to this dissertation. I am also grateful to the members of the Interactive Robotics Group for welcoming me so warmly and helping me feel that I belonged at a time when I was still finding my confidence as a researcher.

Thank you to my friends in Boulder, who made my years here more joyful, adventurous, and meaningful. I am especially grateful to Allen, Amie, Everley, Jason, Luis, Doopy, Nathan, Angus, Henry, Yuni, Chenhao, and many others for the snowboarding trips, Taiwanese meals, shared celebrations, and countless adventures. Thank you for welcoming me into the Taiwanese community and for giving me a sense of home through our gatherings and holiday celebrations. I am also grateful to the friends I met through running, who helped me discover a lasting interest in the sport, and to the many people I had the opportunity to play tennis with. It has been a privilege to spend these years in such a beautiful state, surrounded by mountains, hiking trails, ski resorts, and seemingly endless opportunities for adventure.

To my partner, Natalie Yu, thank you for your patience, understanding, and support. Being in a long-distance relationship has not always been easy, but we have created unforgettable memories through our travels, visits, and the many places we have explored together. I am deeply grateful for the time we have shared and look forward to creating many more memories with you.

Finally, I want to thank my family, whose unwavering love and support made it possible for me to pursue this dream. Earning a Ph.D. is an extraordinary privilege, and I am profoundly grateful for everything they have done to help me reach this point. I remember my dad, who left us far too soon. From the time I was young, he nurtured my curiosity and helped shape the person I am today. I thank my mom, who taught me the value of hard work and perseverance, while also showing me how to find joy in everything I do. And I thank my brother, Jerry, for his companionship throughout our childhood, for introducing me to coding, and for helping inspire the decision to change my major as an undergraduate—a decision that ultimately set me on the path to this dissertation.

Contents

Chapter

1	Introduction	1
1.1	Motivation: Robot Chef Collaborator	1
1.2	The Environment as a Design Variable	1
1.3	Research Questions	2
1.4	Levels of Human Intent	2
1.5	Contributions	3
2	Just-in-Time Robotic Kitting via Bilevel Optimization	5
2.1	Introduction	5
2.2	Background and Related Work	7
2.3	Problem Formulation	8
2.4	Proposed Approach	10
2.4.1	Task Segmentation and Scheduling	10
2.4.2	Kit Arrangement	13
2.5	Experimental Validation	15
2.5.1	User Study	15
2.5.2	Shop Floor Simulation	18
2.6	Results	19
2.6.1	User Study	19

2.6.2	Simulation	21
2.7	Discussion	21
2.8	Chapter Summary and Future Work	22
3	Environment Design for Active Preference Learning	23
3.1	Introduction	23
3.2	Related Work	25
3.3	Preliminaries	27
3.4	Technical Approach	30
3.4.1	Counterfactual Reasoning	30
3.4.2	Environment Design	31
3.5	Experiments	32
3.5.1	Simulation Experiments	34
3.5.2	User Study	37
3.6	Chapter Summary and Future Work	39
4	Workspace Optimization for Human Motion Prediction	41
4.1	Introduction	41
4.2	Related Work	44
4.3	Legible Workspace Generation	45
4.3.1	Legibility Score	46
4.3.2	Optimization for Task Legibility	46
4.3.3	Search using Quality Diversity	48
4.4	Evaluation	49
4.4.1	Hypotheses	50
4.4.2	Overcooked Experiment	50
4.4.3	Tabletop Experiment	54
4.5	Results	56

4.5.1	Overcooked	57
4.5.2	Tabletop Experiment	57
4.6	Chapter Summary and Future Work	58
5	Discussion and Broader Impact	63
5.1	Synthesis Across Levels of Intent and Downstream Uses	63
5.2	Future Research Directions	64
5.2.1	Active Environment Generation for Foundation Models	65
5.2.2	Purposeful Diversity for Training and Evaluation	66
5.2.3	Structuring Interaction to Reduce Ambiguity	68
5.2.4	Integrating Environment Design with Task Execution	70
5.2.5	From Human Legibility to Mutual Legibility	72
5.3	Conclusion	73
	Bibliography	74
	Appendix	
A	VAE for Tabletop Object Positions	82
B	Task Graph for the Overcooked Experiment	83
C	Learned Reward Functions for Overcooked	84
D	Goal Prediction Results	86
E	Comparisons with other Optimization Methods	89

Tables

Table

4.1	The average determinant (det) and trace of the covariance matrices of the multivariate Gaussian models.	58
D.1	Overcooked Goal Prediction	86
D.2	Tabletop Goal Prediction	87
E.1	Comparisons of optimization methods on the best solution found (i.e., legibility scores) for different Overcooked environments.	89

Figures

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. 6
- 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 $T^H = \{a_1^H, a_2^H, a_3^H\}$ 9
- 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 $t + 1$ 11

- 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. 15
- 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 $p < .0001$ 17
- 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 $p < .001$ 20

3.1	The delivery robot above (goal: yellow pin) optimizes its path by considering factors like travel time and terrain types. Through active preference learning, it infers human rewards from trajectory rankings. (a) However, current state-of-the-art methods often struggle to efficiently generate informative trajectory pairs for these queries, leading to suboptimal results. To overcome this, CRED incorporates two key contributions: (b) counterfactual reasoning, which explores varied hypothetical preferences to produce more diverse trajectories, and (c) environment design, which “imagines” different scenarios—e.g. altering terrain from grass to gravel—to enhance the system’s generalization capabilities. (d) As a result, the robot aligns to human preferences when deployed.	25
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 $\{\xi_1, \dots, \xi_M\}$. The most informative trajectory pair is returned to the outer optimization.	27
3.3	Examples of preference queries with details and visualization across the three simulation domains.	33
3.4	Accuracy of the estimated rewards across three domains (columns) for each experiment (rows).	36
3.5	Results from the user study: higher is better for (a)-(c), lower is better for (d). Within each plot the central line denotes the median, the upper and lower edges correspond to the first (Q1) and third (Q3) quartiles, and the whiskers extend to 1.5 times the interquartile range from Q1 and Q3. Statistically significant differences indicated as: * $p \leq 0.05$, ** $p \leq 0.01$	38

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.	42
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.	42
4.3	Overcooked layouts used in our experiments. The environment generated by our approach, (d) <i>Legible</i> , optimizes object and virtual obstacle placements to elicit legible human motion.	52
4.4	Initial cube configurations used in our tabletop experiments. The environment generated by our approach, (d) <i>Both Optimized</i> , optimizes object and virtual obstacle placements to elicit legible human motion. (e) 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.	53

4.5	Behavior performance map generated in the Overcooked domain. MAP-Elites is able to explore legible workspace configurations across combinations of the features: number of obstacles and the ordering of ingredient placements, which improves the best solution found in complex search spaces.	54
4.6	Overcooked experiment results. (a) Mean accuracy of the Bayesian predictor using a cost function learned from MaxEnt IRL. Compared to the shortest distance heuristic shown in (b), MaxEnt IRL improved accuracy the most when less than 60% of the trajectory has been observed. The width of the shaded area is Tukey’s Q critical value, so overlaps indicate no statistical significance. The <i>Legible</i> condition elicits significantly higher accuracy for most cases.	60
4.7	Tabletop experiment results. (a) Mean accuracy of the time series multivariate Gaussian model. Our approach, <i>Both Optimized</i> , elicits significantly higher or comparable accuracy compared to the baselines. (b) Mean accuracy using a shortest distance heuristic where the closest cube to the current hand position is the predicted goal. (c) Mean accuracy of the time series multivariate Gaussian model with varying amounts of training data. <i>Both Optimized</i> elicits significantly higher accuracy compared to baselines starting at 33% training data, achieving around 90% accuracy with just 50% of the available training data.	61
4.8	Top-down view of the workspace plotting the mean and covariance of the time series multivariate Gaussian for each condition. The Gaussian models are fit to the hand trajectory data and predict the human’s goal (Section 4.4.3). The model in the <i>Both Optimized</i> condition has less covariance compared to the models trained in the other environment configurations.	62

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. . . .	83
C.1	Learned reward functions from MaxEnt IRL.	84
D.1	Confusion matrix for the Bayesian predictor in the different generated Overcooked layouts.	87
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.	88

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 multiple levels of intent: (1) **low-level motion**, (2) **high-level task structure**, and (3) **high-level preferences** and together highlight a common requirement: effective collaboration depends on a shared mental model between human and robot [72]. 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 [64], preference learning [12], and human behavior modeling. These approaches assume that the **environment** in which 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 [17, 86], systematically evaluate policy robustness [23], and reduce sim-to-real discrepancies in manipulation [1]. 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 inference, coordination, and interaction quality 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 Levels of Human Intent

We organize the thesis around the high- and low-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 table-top manipulation study (12 participants), showing improved goal-prediction accuracy and data efficiency across multiple classes of prediction models.

Chapter 5 summarizes the contributions of this thesis across task structure, reward alignment, and low-level motion, highlighting their shared formulation of environment design as an outer decision problem that improves a downstream task-execution or human-intent-inference process. It identifies two complementary roles for environment design: generating diverse and informative scenarios across interactions, and structuring individual interactions to reduce ambiguity and support coordination. The chapter then discusses future directions for active environment generation in foundation models, purposeful diversity in training and evaluation, adaptive interaction constraints, the integration of environment modification with task execution, and the extension of environment design from improving robot inference toward supporting mutual understanding between humans and robots.

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 [77].

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 [97].

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 [7, 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

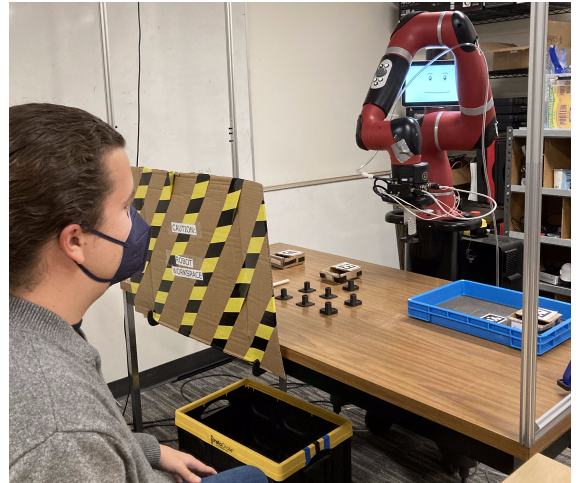


Figure 2.1: This work presents an optimization approach to automate just-in-time robot part delivery for collaborative assembly. A robot gathers parts required for an assembly task and places them on a kitting tray. The human then uses this “kit” to complete the next set of tasks.

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 [32] and assembly workers report higher cognitive loads [33, 67].

Due to the physical and cognitive demands of manual kit preparation, robotic kitting has been explored to increase efficiency and flexibility. Caputo et al. [10] developed an economic model to show that automated systems are more cost effective and efficient, except in low-production settings. Boudella et al. [6] 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 [44], 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 [75]. Other production systems use material handling techniques like kitting trolleys [52] or automated delivery vehicles [96] 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 [55], and a non-linear program for flexible scheduling of robot tasks is presented in [89]. In addition to optimizing for efficiency, temporal considerations have been shown to enable fluent human-robot teaming in [46]. 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:

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

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.

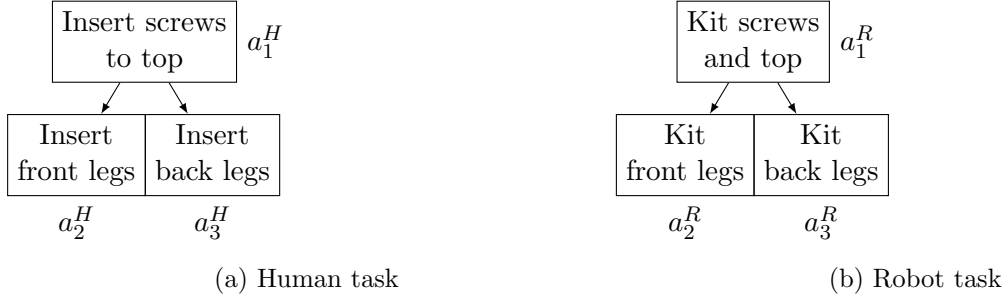


Figure 2.2: A task graph for assembly of a stool. The robot retrieves the parts required for the human to complete the assembly. The arrows indicate precedence constraints. The robot tasks are represented by $T^R = \{a_1^R, a_2^R, a_3^R\}$ and the human tasks are represented by $T^H = \{a_1^H, a_2^H,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

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 $\text{duration}^H(T_t)$ and $\text{duration}^R(T_t)$ 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 $S = \{0s, 35s\}$.

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:

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

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

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 [14]. 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: $K = \{a_1, a_2, \dots\}$ 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

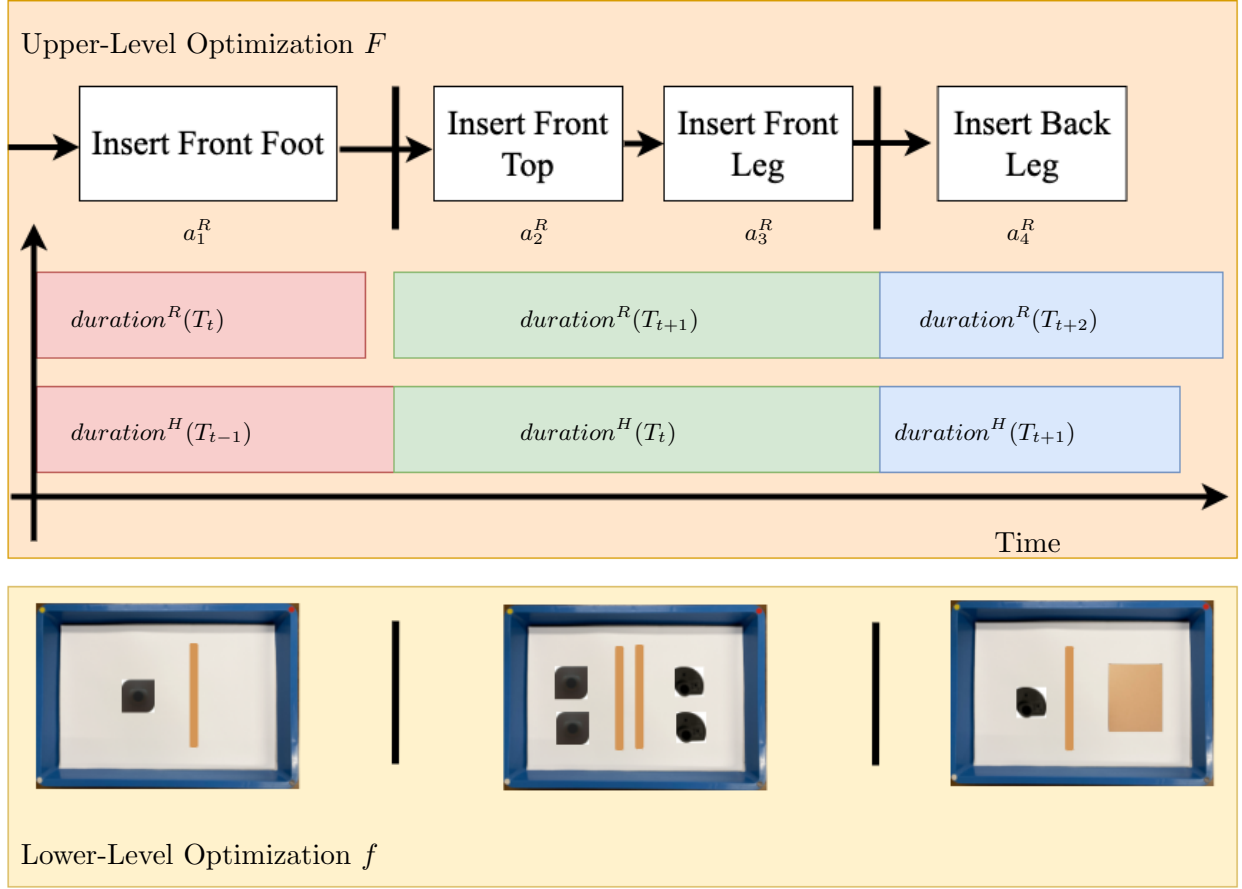


Figure 2.3: Overview of the bilevel optimization algorithm for just in time kitting. The upper level optimization segments and schedules tasks, and the lower level optimization arranges parts on the tray. By minimizing the differences between $\text{duration}^R(T_{t+1})$ and $\text{duration}^H(T_t)$ (shown in green) and the differences between $\text{duration}^R(T_{t+2})$ and $\text{duration}^H(T_{t+1})$ (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 k th 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:

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

$$- W_2 * i \quad (2.4)$$

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

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

$$- W_5 * \text{kit_fitness}(P) \quad (2.7)$$

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 $T_t = K_{0:i[0]}$, $T_{t+1} = K_{i[0]:i[1]}$, $\dots$, $T_{t+|i|} = K_{i[|i|-1]:|K|}$. 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, 26]. 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 j th 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:

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

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

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:

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

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:

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

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

$$\bigcup_{j=1}^i \text{parts_in}(K_j) \in P$$

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 [16]. Other stochastic methods such as simulated annealing [84] and genetic algorithms [28] 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**
 $\text{samples} \sim \mathcal{N}(\mu, \Sigma)$
 $\text{scores} \leftarrow \text{sorted}(\text{score}(\text{samples}))$ ▷ score samples and sort from largest to smallest
 $\mu \leftarrow \text{mean}(\text{samples}[:c])$
 $\Sigma \leftarrow \text{covariance}(\text{samples}[:c])$ ▷ 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.

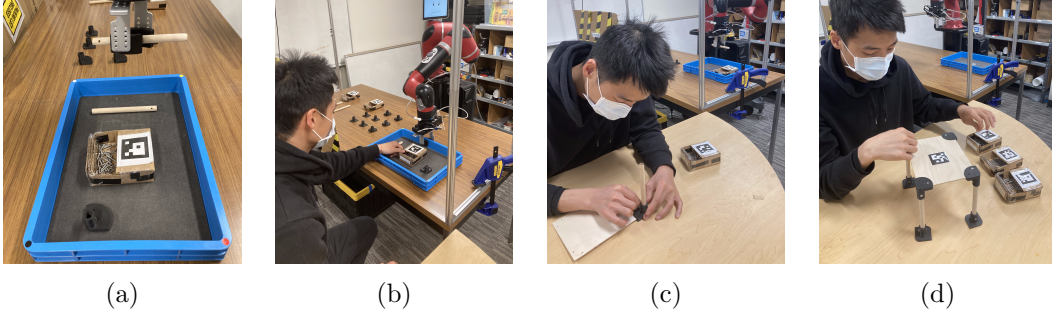


Figure 2.4: (a) The robot arranges parts on the tray according to the lower-level optimization. (b) The robot delivers a finished kit to the user. (c) The user begins the assembly task while the robot prepares the kit for the next tasks. (d) The user finishes assembling a flat-pack furniture table.

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 [92] 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.

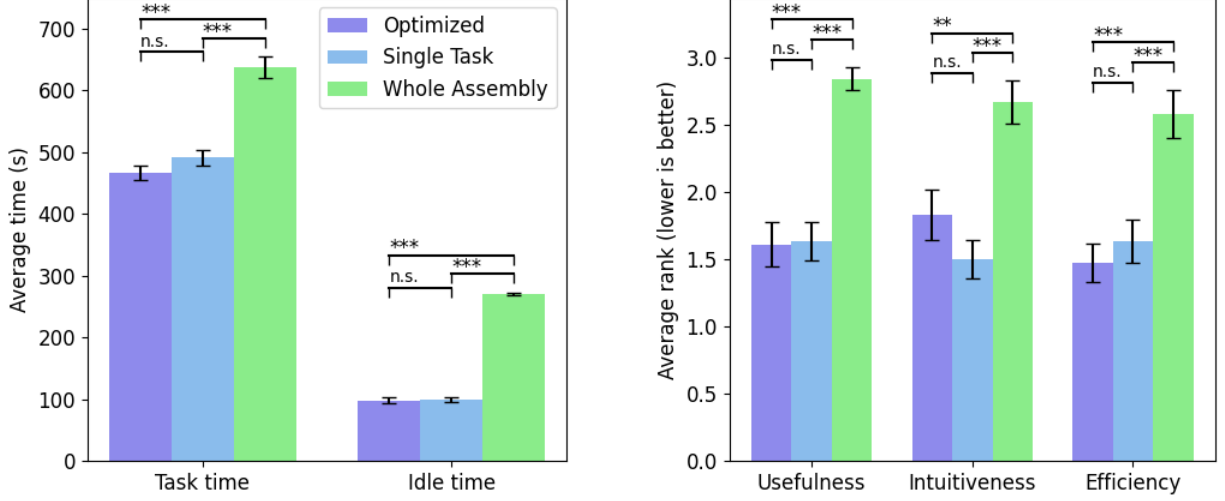


Figure 2.5: **Left:** Total task time and human idle time across all three conditions. Both the *Optimized* and *Single Task* approaches had significantly lower total times in both metrics than in the *Whole Assembly* condition. **Right:** Total user ranking scores for each condition across three subjective 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) [36]. We installed an Intel RealSense D435 Depth Camera above the robot workspace and used the Facebook Detectron2 algorithm [91] for image segmentation of the assembly parts. We used the ArUco library [27] 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 [61].

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:

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

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

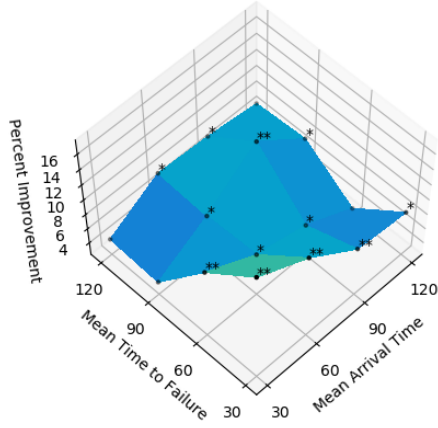
2.6.1 User Study

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.

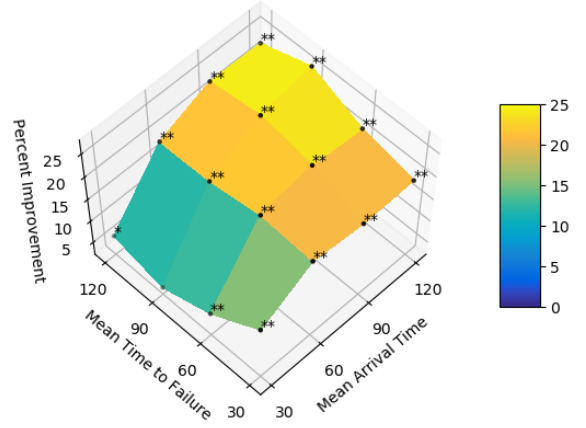
Total task time. The effect of condition significantly influenced total task time, $F(2, 34) = 57.60, p < .0001$, with a significant reduction in task time between the *Whole Assembly* condition and the *Single Task* condition ($p < .0001$, 95% C.I. = $[-188.211, -103.851]$), as well as between the *Whole 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. = $[-186.020, -160.141]$).

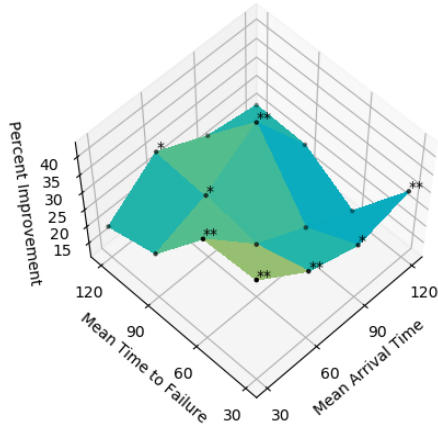
Subjective measures. We analyzed rank scale data on our post-experiment survey using a nonparametric Kruskal-Wallis Test with experimental condition as a fixed effect. Post-hoc comparisons used the Wilcoxon method for analyzing rank significance between condition pairs. The results are summarized in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



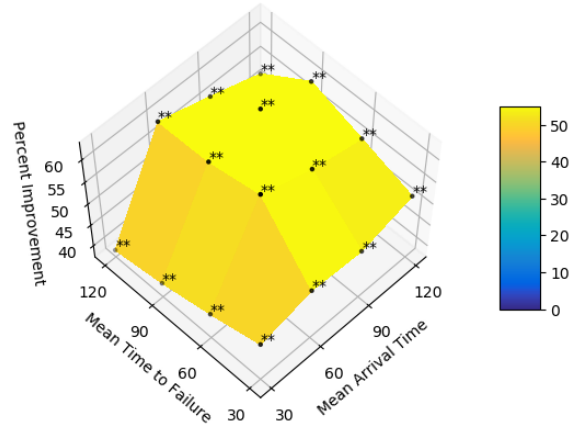
(a) Percent improvement in total task time of *Optimized* over *Single Task*.



(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. 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 how a robot can use task structure to construct a shared task model and optimize the environment for team performance. In that setting, the objective of the task is known, and the robot uses the structure of the task to improve coordination. The next chapter considers a more ambiguous setting: when there are multiple valid ways to complete a task, the robot may not know which outcomes, strategies, or tradeoffs the human prefers. We therefore study how environment design can support preference learning, enabling the human to teach the robot a reward function more efficiently.

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 [82]. 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 [47]. 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 [12, 93]. 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 [65, 4]. Prior work makes this optimization tractable by restricting the query set to pre-generated trajectories [5] or replay buffers [40], 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 [5]. 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 [63] 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 [54] 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,

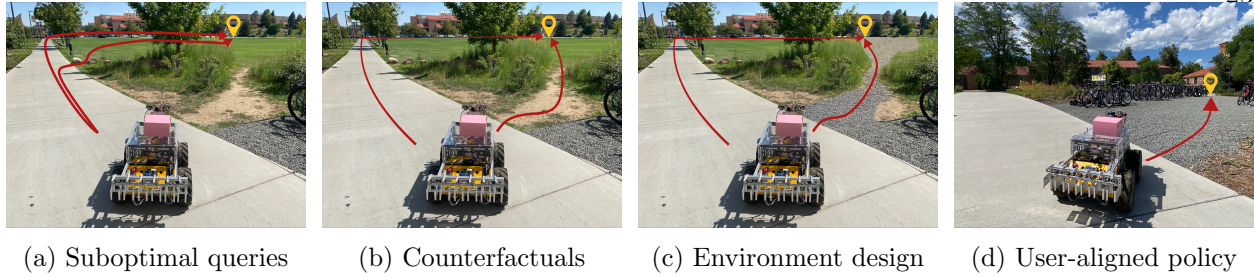


Figure 3.1: The delivery robot above (goal: yellow pin) optimizes its path by considering factors like travel time and terrain types. Through active preference learning, it infers human rewards from trajectory rankings. (a) However, current state-of-the-art methods often struggle to efficiently generate informative trajectory pairs for these queries, leading to suboptimal results. To overcome this, CRED incorporates two key contributions: (b) counterfactual reasoning, which explores varied hypothetical preferences to produce more diverse trajectories, and (c) environment design, which “imagines” different scenarios—e.g. altering terrain from grass to gravel—to enhance the system’s generalization capabilities. (d) As a result, the robot aligns to human preferences when deployed.

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.

3.2 Related Work

Learning from Preferences. Preference learning infers a human’s reward function by iteratively asking them to choose between robot trajectories [5]. 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 [65]. Subsequent work explores different query selection objectives: maximize mutual information of the query and the estimated weights [5], maximize regret [90], and maximize uncertainty in reward predictions [40]. For example, Lee et al. [40] 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 [5] and [40] 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. [34] learn a visual representation space to extrapolate preferences to out-of-distribution terrains, Mao et al. [47] train a bird’s-eye view costmap from human feedback, and Zhang et al. [93] 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 [86] or by modifying parameters to maximize an agent’s learning potential [17]. In human-robot interaction, environment modification has been used to generate interpretable robot behaviors [37], legible human motion [80, 78] (discussed in detail in Chapter 4), and to support collaborative teaming in settings like warehouse design [95] or tabletop reorganization [2]. Most relevant to our

setting is the work of [8], 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.

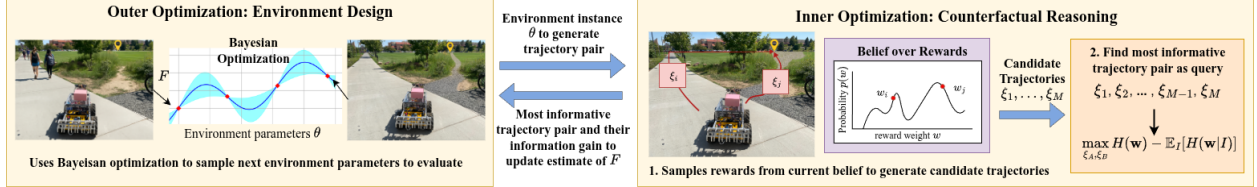


Figure 3.2: System overview of CRED as a bilevel optimization problem. **Outer optimization (Environment Design):** Bayesian optimization selects environment parameters θ to evaluate, seeking those that maximize estimates of query informativeness F . **Inner optimization (Counterfactual Reasoning):** Given θ , the system samples reward weights from the current belief to generate candidate trajectories $\{\xi_1, \dots, \xi_M\}$. 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, $T : S \times A \times S \rightarrow [0, 1]$ is the transition function, $R : S \times A \rightarrow \mathbb{R}$ 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: $\xi = ((s_0, a_0), (s_1, a_1), \dots, (s_H, a_H))$ 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 [71] or trajectory optimization methods such as CHOMP [99].

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 ?” [4]. 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.

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

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 [30] 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 [5], 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 [5]. Our objective function f is

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

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 [30]) 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) [71] 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 [99] (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 $\{w_1, \dots, w_N\} \sim P(w)$
 - 2: Select M diverse weights (e.g., max cosine distance)
 - 3: **for** each selected w_k **do**
 - 4: Generate trajectory ξ_k by maximizing $w_k^\top \Phi(\xi)$ $\triangleright$ 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 problem¹ :

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

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 [69], which models F with a Gaussian process (GP) defined by a mean function $m : \Theta \rightarrow \mathbb{R}$ and a positive definite covariance function $K : \Theta \times \Theta \rightarrow \mathbb{R}$. 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 [54] 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 [5, 12]. 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.

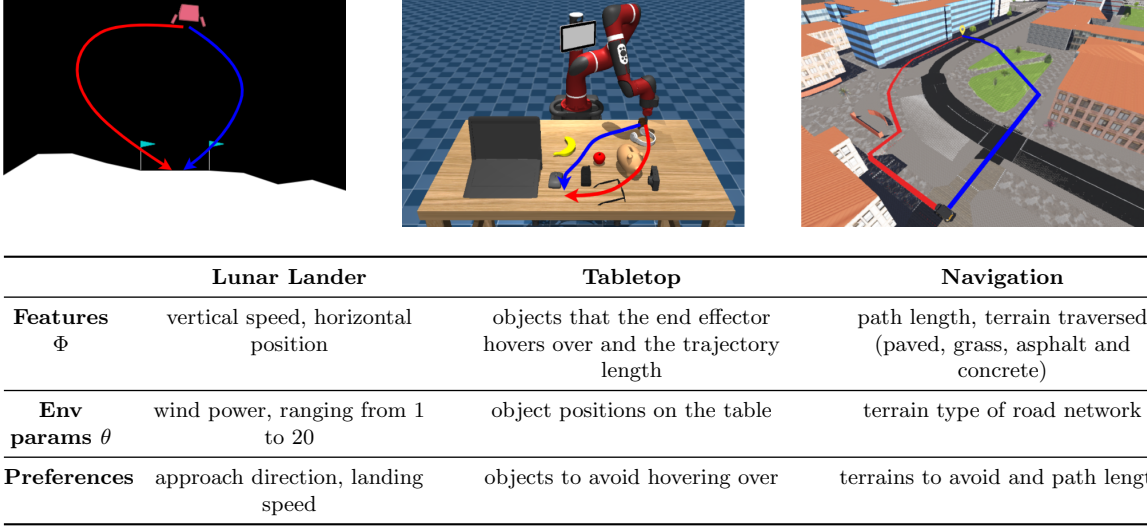


Figure 3.3: Examples of preference queries with details and visualization across the three simulation domains.

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 [5, 12].
- **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 [74], 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 [66] 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 [59] in MuJoCo [73], 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) [13] 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 [99] 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 [56]. The network is simulated in Webots [87] 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 [71], 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:

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

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 [5], which we denote as **RR**. The second, inspired by [12] and used in later work such as PEBBLE [40], 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 **Mean Belief Policy (MBP)**.

Simulation Results.

Baseline Comparison. Figure 3.4 shows that 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

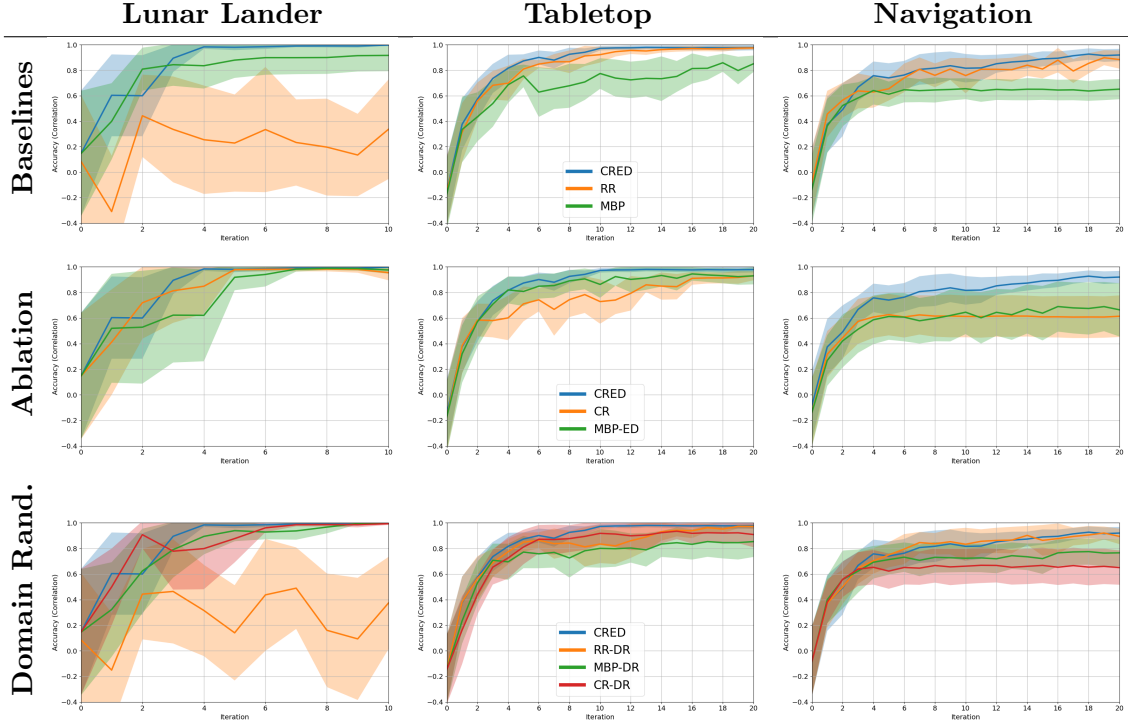


Figure 3.4: Accuracy of the estimated rewards across three domains (columns) for each experiment (rows).

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 [12] and RR-DR, which uses pre-generated trajectories [5] 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. 3.3. The reward functions inferred from these preferences were subsequently deployed on the physical robot platforms. For the tabletop task, the

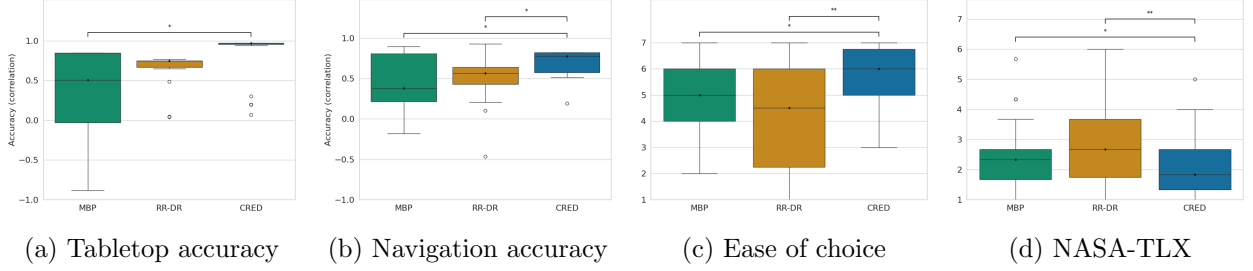


Figure 3.5: Results from the user study: higher is better for (a)-(c), lower is better for (d). Within each plot the central line denotes the median, the upper and lower edges correspond to the first (Q1) and third (Q3) quartiles, and the whiskers extend to 1.5 times the interquartile range from Q1 and Q3. Statistically significant differences indicated as: * $p \leq 0.05$, ** $p \leq 0.01$.

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 function², 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 [31] 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.

Figure 3.5 shows that, 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,

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 $[-1.0, -0.1, -2.0, -5.0, -0.1]$.

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 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 models human behavior at the granularity of complete subtasks or trajectories. We now turn to the finer-grained problem of interpreting the human’s ongoing motion. Chapter 4 shows that modifying the physical and virtual workspace can elicit more legible motion, reduce behavioral ambiguity, and improve the robot’s ability to predict the human’s intended goal.

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.

4.1 Introduction

In human-robot collaborative tasks, shared mental models between agents enable the awareness and joint understanding required for effective teamwork [72]. With no shared notion of the task to be completed, the inherent stochasticity and opacity of human decision-making makes robot planning difficult [68]. To this end, prior research efforts have focused on developing robots that can predict human behavior [64, 39, 45], generating motion plans to safely interact in a shared environment [58, 29]. 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

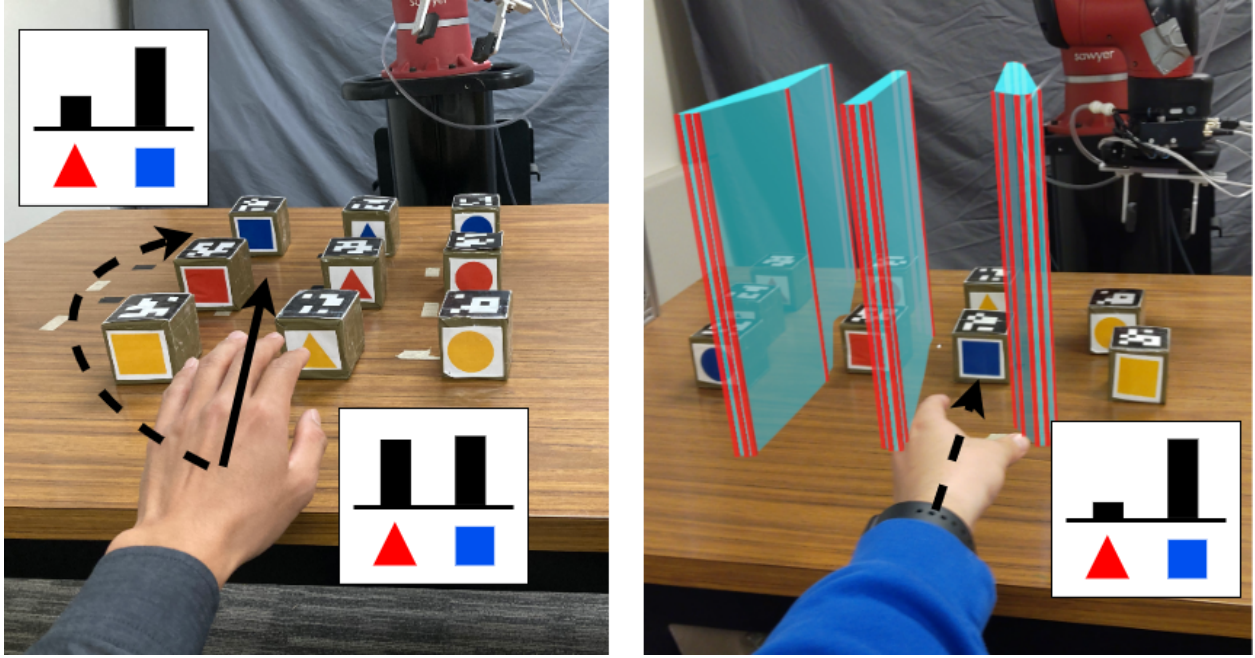


Figure 4.1: Workspace configuration affects the robot’s ability to correctly predict the human’s goal — the blue square cube. **Left:** The legible path (dotted) requires the human to take a circuitous route while the natural path (solid) is not legible. **Right:** Our approach generates a workspace 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.

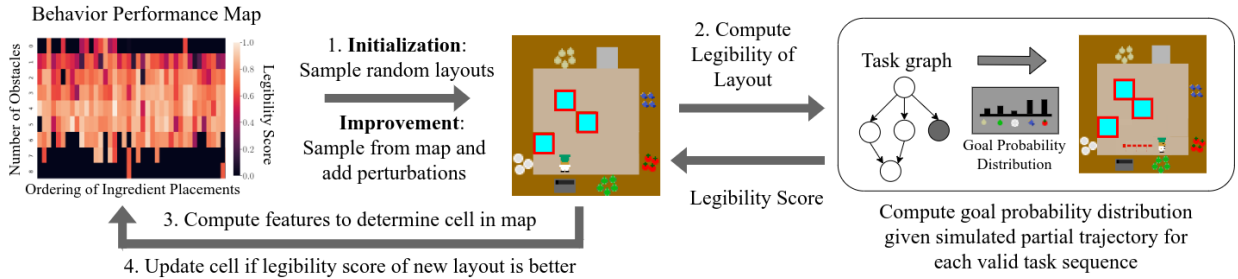


Figure 4.2: Our approach for generating workspace configurations that enable accurate human goal predictions. **(1)** In the initialization phase, we sample random environment layouts to populate the behavior performance map, which stores diverse and high performing solutions. This is followed by the 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.

[48], especially in safety-critical or close-proximity settings [85].

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 [18], 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 [50]. 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 [15, 53].

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 [38] and navigation [25, 60] 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 [58, 29]. 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 [42, 62], dynamic movement primitives [43, 88], latent representation learning [9, 49], and imitation learning [45, 20], among various other methods [64]. 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 [77]. Zhang et al. [94] proposed a framework for designing environments that optimize an agent’s reward, and Keren et al. [35] extended it for stochastic transitions. Kulkarni et al. [37] 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. [95] 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 [24]. Inspired by the success of QD algorithms in finding diverse solutions in large search spaces [15, 53], 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 [18] shown in Equation 4.1.

$$\Pr(G|\xi_{S \rightarrow Q}) \propto \frac{\exp(-C(\xi_{S \rightarrow Q}) - C(\xi_{Q \rightarrow G}^*))}{\exp(-C(\xi_{S \rightarrow G}^*))} \quad (4.1)$$

The optimal human trajectory from point X to point Y with respect to cost function C is denoted by $\xi_{X \rightarrow Y}^*$. 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 $\xi_{S \rightarrow Q}$ relative to the most efficient trajectory $\xi_{S \rightarrow G}^*$.

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 $|\xi_{S \rightarrow Q}|$. 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 $\xi_{S \rightarrow Q}$.

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

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.

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

Let T' represent a valid subtask sequence, consisting of all k subtasks of task T : $t_1, \dots, t_k$, 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 $t_1, \dots, t_i$, $\mathcal{G}$ 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.

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

- 1: Best configuration $w^* \leftarrow w$; Best score $s^* = s_w$
 - 2: $A \leftarrow \text{sample_perturbations}(w)$
 - 3: **for** $a \in A$ **do**
 - 4: Generate new workspace $w' = \text{apply_perturbation}(w, a)$
 - 5: Compute legibility score $s_{w'} = F(w')$
 - 6: **if** $s_{w'} > s^*$ **then**
 - 7: Update best workspace $w^* = w'$
 - 8: Update best score $s^* = s_{w'}$
 - 9: **end if**
 - 10: Determine features $\mathbf{m} = M(w')$
 - 11: **if** $S[\mathbf{m}] = \emptyset$ or $s_{w'} > V[\mathbf{m}]$ **then**
 - 12: Update solution map $S[\mathbf{m}] = w'$
 - 13: Update solution values $V[\mathbf{m}] = s_{w'}$
 - 14: **end if**
 - 15: **end for**
 - 16: **if** $w^* \neq w$ **then**
 - 17: **return** `improve_workspace`(w^*)
 - 18: **end if**
 - 19: **return** w^*
-

Algorithm 5 Workspace Generation with MAP-Elites

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

```

1: Solution map  $S \leftarrow \emptyset$ ; Solution values  $V \leftarrow \emptyset$ 
2: for  $i = 1, \dots, N$  do
3:   if  $i < N_{init}$  then
4:     Generate workspace  $w = \text{random\_workspace}()$ 
5:   else
6:     Sample workspace from map  $w = \text{random}(S)$ 
7:     Run  $w = \text{improve\_workspace}(w)$ 
8:   end if
9:   Determine features  $\mathbf{m} = M(w)$ 
10:  Determine objective score  $s = F(w)$ 
11:  if  $S[\mathbf{m}] = \emptyset$  or  $s > V[\mathbf{m}]$  then
12:    Update solution map  $S[\mathbf{m}] = w$ 
13:    Update solution values  $V[\mathbf{m}] = s$ 
14:  end if
15: end for
16: return  $S, V$ 

```

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 [50] 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. 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 [45] or approximated via shortest path to goal [18].

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 $\mathbf{m}$. If there was an improvement to the workspace, we run Algorithm 4 again. Otherwise, a local maximum has been found, and we return the best workspace found.

4.4 Evaluation

We evaluate our approach in two environments: a) single-player Overcooked [11] 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. [18] (Eqn. 4.1). For the tabletop experiment, we implement the time series multivariate Gaussian model proposed in [62] 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) [98] 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 $\max_\theta \sum_{\xi \in D} p(\xi|\theta)$ [98].

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).

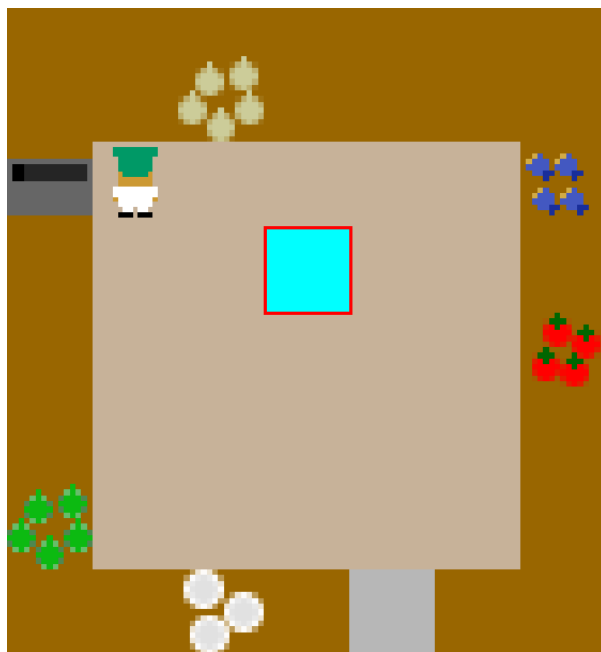
Efficient: We use MAP-Elites to minimize the distance traveled to complete a task (Fig. 4.3b). This condition represents a common objective when designing workspace configurations — optimizing for task efficiency.

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

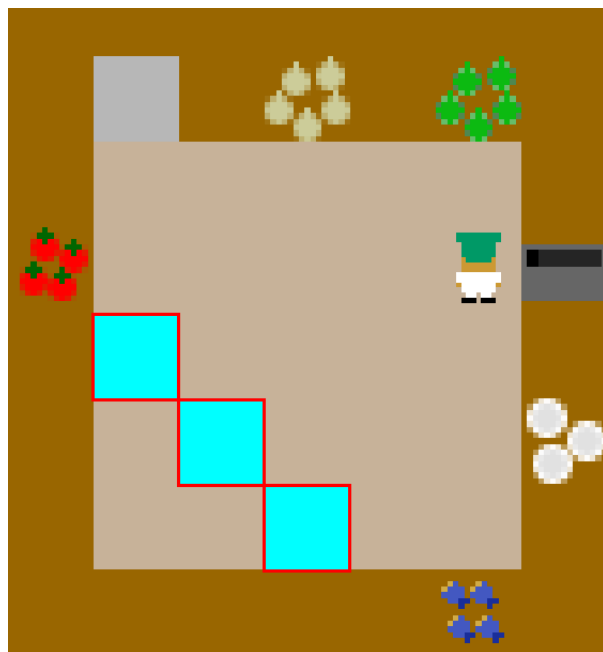
Legible: Our approach — using MAP-Elites to maximize legibility (Fig. 4.3d).

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 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.

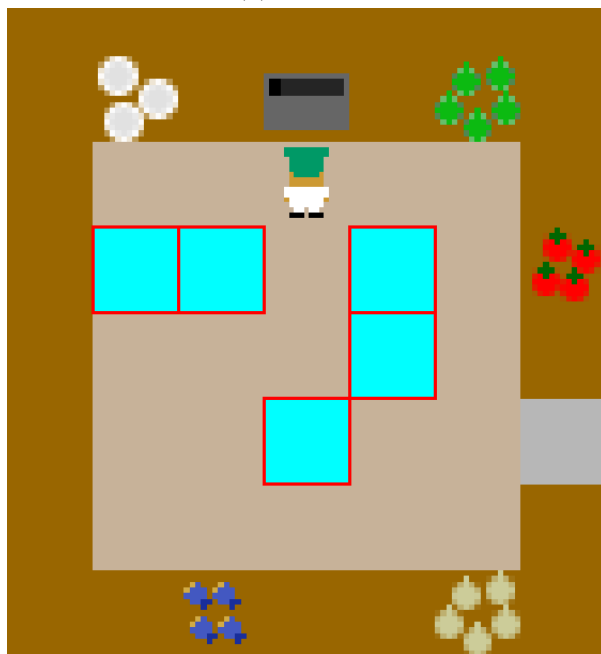
All conditions, except the Random condition, are generated using MAP-Elites with different objective functions. The features are the number of obstacles and the ordering of workstation



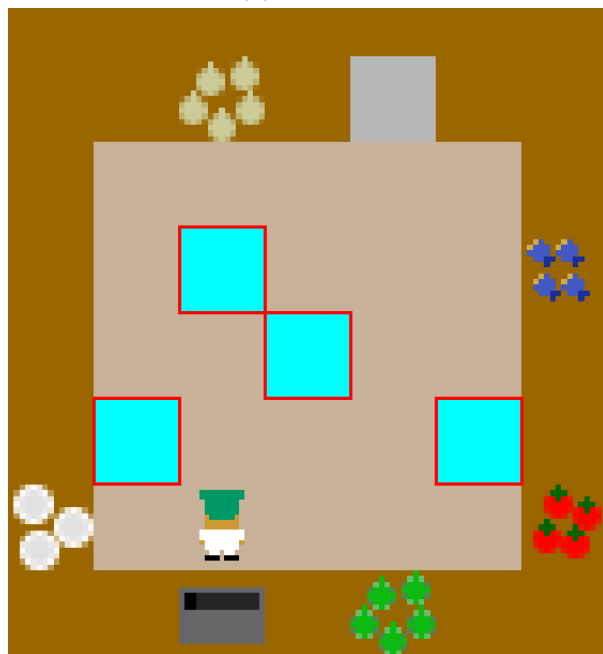
(a) Random



(b) Efficient

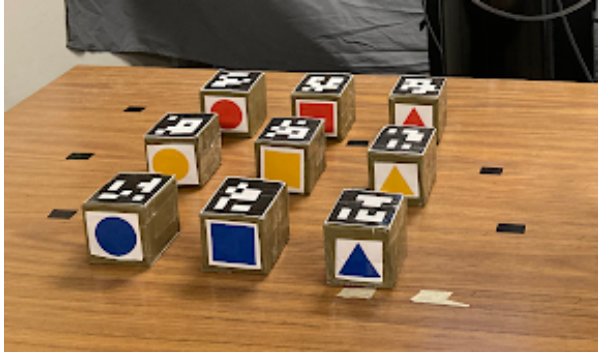


(c) Legible + Efficient



(d) Legible

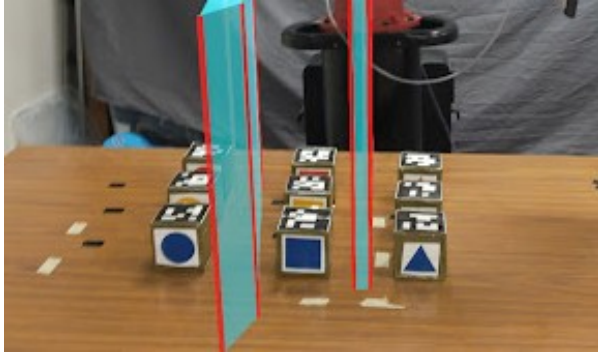
Figure 4.3: Overcooked layouts used in our experiments. The environment generated by our approach, (d) *Legible*, optimizes object and virtual obstacle placements to elicit legible human motion.



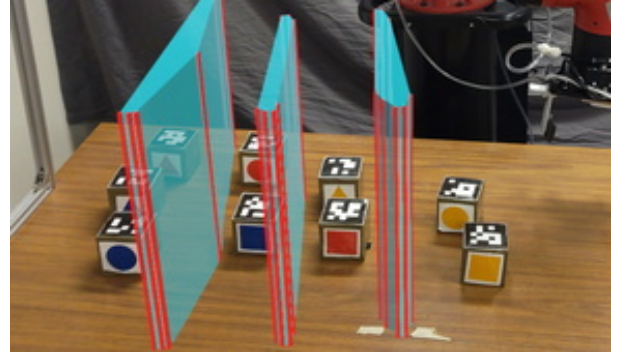
(a) Baseline



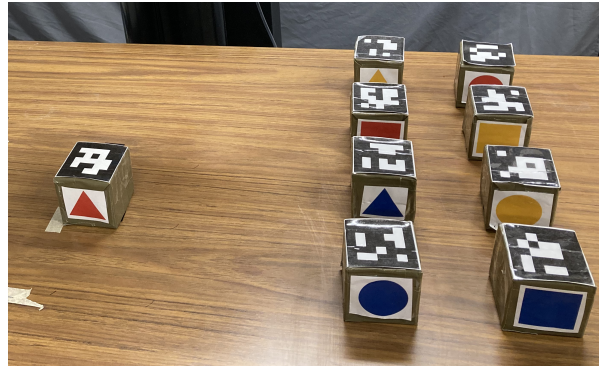
(b) Placement Optimized



(c) Virtual Obstacle Optimized



(d) Both Optimized



(e) Desired configuration

Figure 4.4: Initial cube configurations used in our tabletop experiments. The environment generated by our approach, (d) *Both Optimized*, optimizes object and virtual obstacle placements to elicit legible human motion. (e) 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.

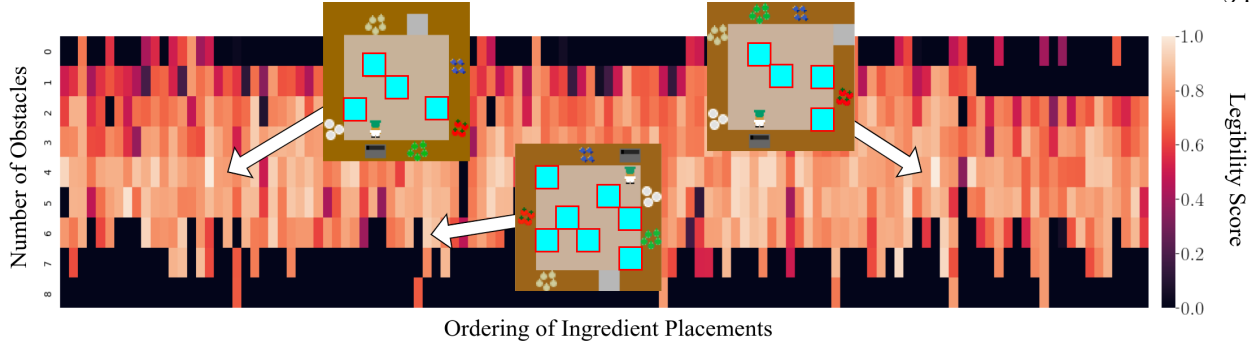


Figure 4.5: Behavior performance map generated in the Overcooked domain. MAP-Elites is able to explore legible workspace configurations across combinations of the features: number of obstacles and the ordering of ingredient placements, which improves the best solution found in complex search spaces.

placements from top to bottom and left to right. Figure 4.5 shows the behavior performance map generated by MAP-Elites. 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. 4.4e). 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 [62] 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) [51] 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 i th 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 [83].

Conditions. We compare workspace setups using our optimization against **Baseline**, where the cubes are initially sorted by their color and shape (Fig. 4.4a). 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. 4.4b).

Virtual Obstacle Optimized: The cubes are sorted by their color and shape, and we optimize for legibility by projecting virtual obstacles (Fig. 4.4c).

Both Optimized: Our approach that optimizes for legibility by rearranging the cubes and projecting virtual obstacles (Fig. 4.4d).

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 improvement 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 $30\text{ cm} \times 1\text{ cm}$ 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 $= 7\text{ cm}^2$. 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. 4.4d). 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

Figure 4.6 shows that, using the Bayesian predictor, 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

Figure 4.7 shows that 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

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

	Baseline	Placement Optimized	Virtual Obsta- cle 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

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. Figure 4.8 visualizes this effect, and Table 4.1 shows the average determinant and trace of the covariance matrices of the Gaussian 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.

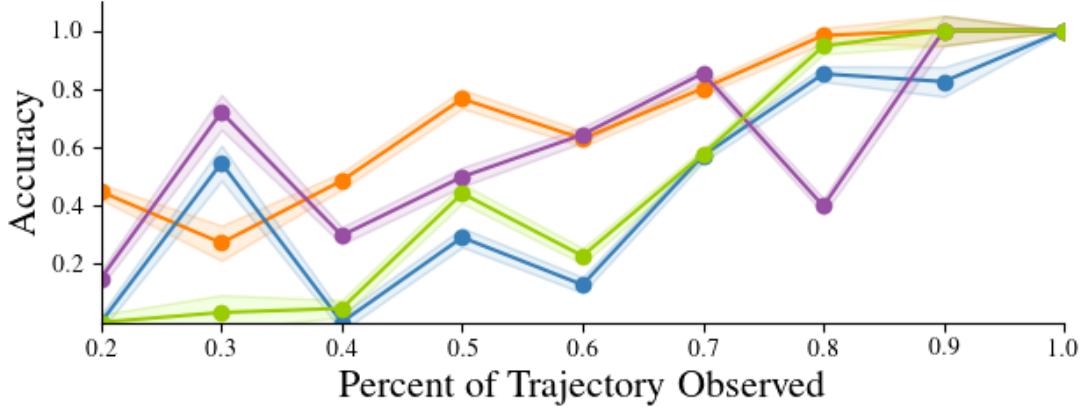
4.6 Chapter Summary and Future Work

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

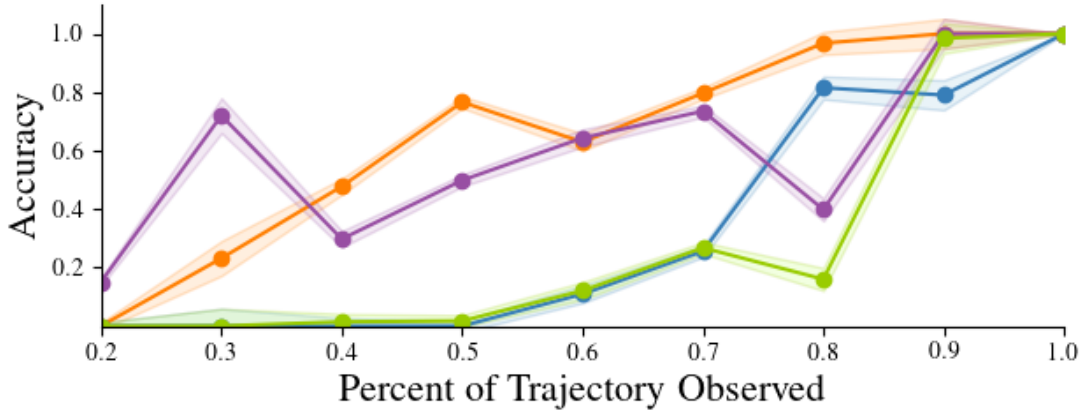
reducing the uncertainty of their motion. Through dual experiments in 2D navigation and tabletop manipulation, we show that our approach results in more accurate model predictions across two distinct goal inference methods, requiring less data to achieve these correct predictions. Importantly, we demonstrate 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 [81], 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.”



(a) MaxEnt IRL cost function



(b) Shortest distance heuristic

Figure 4.6: Overcooked experiment results. (a) Mean accuracy of the Bayesian predictor using a cost function learned from MaxEnt IRL. Compared to the shortest distance heuristic shown in (b), MaxEnt IRL improved accuracy the most when less than 60% of the trajectory has been observed. The width of the shaded area is Tukey's Q critical value, so overlaps indicate no statistical significance. The *Legible* condition elicits significantly higher accuracy for most cases.

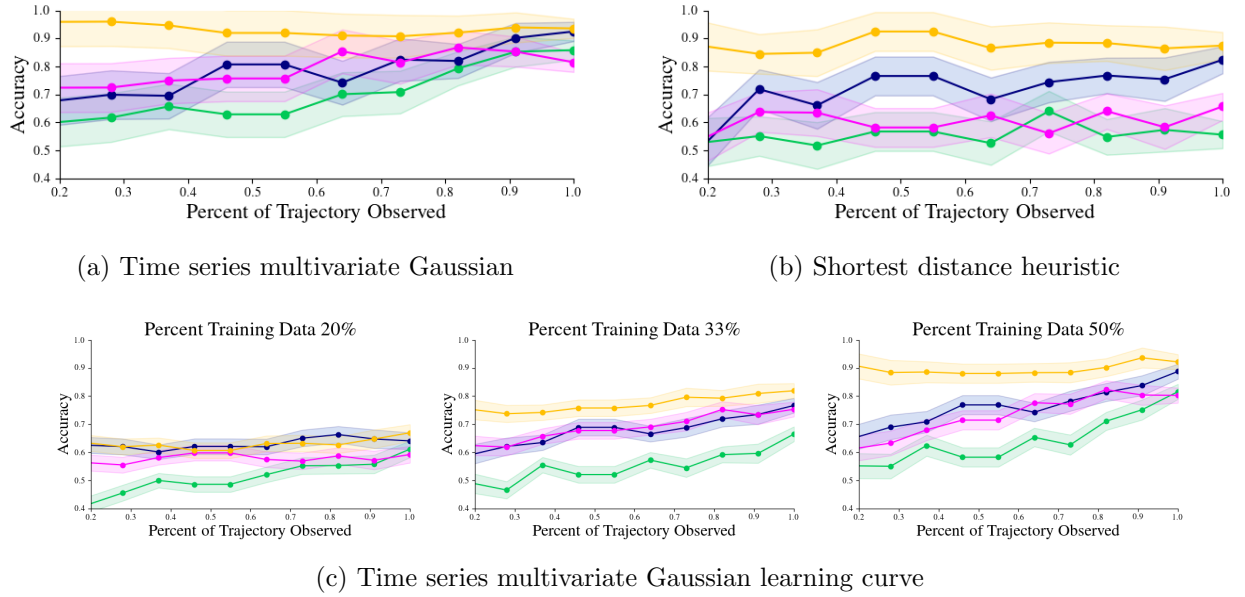


Figure 4.7: Tabletop experiment results. (a) Mean accuracy of the time series multivariate Gaussian model. Our approach, *Both Optimized*, elicits significantly higher or comparable accuracy compared to the baselines. (b) Mean accuracy using a shortest distance heuristic where the closest cube to the current hand position is the predicted goal. (c) Mean accuracy of the time series multivariate Gaussian model with varying amounts of training data. *Both Optimized* elicits significantly higher accuracy compared to baselines starting at 33% training data, achieving around 90% accuracy with just 50% of the available training data.

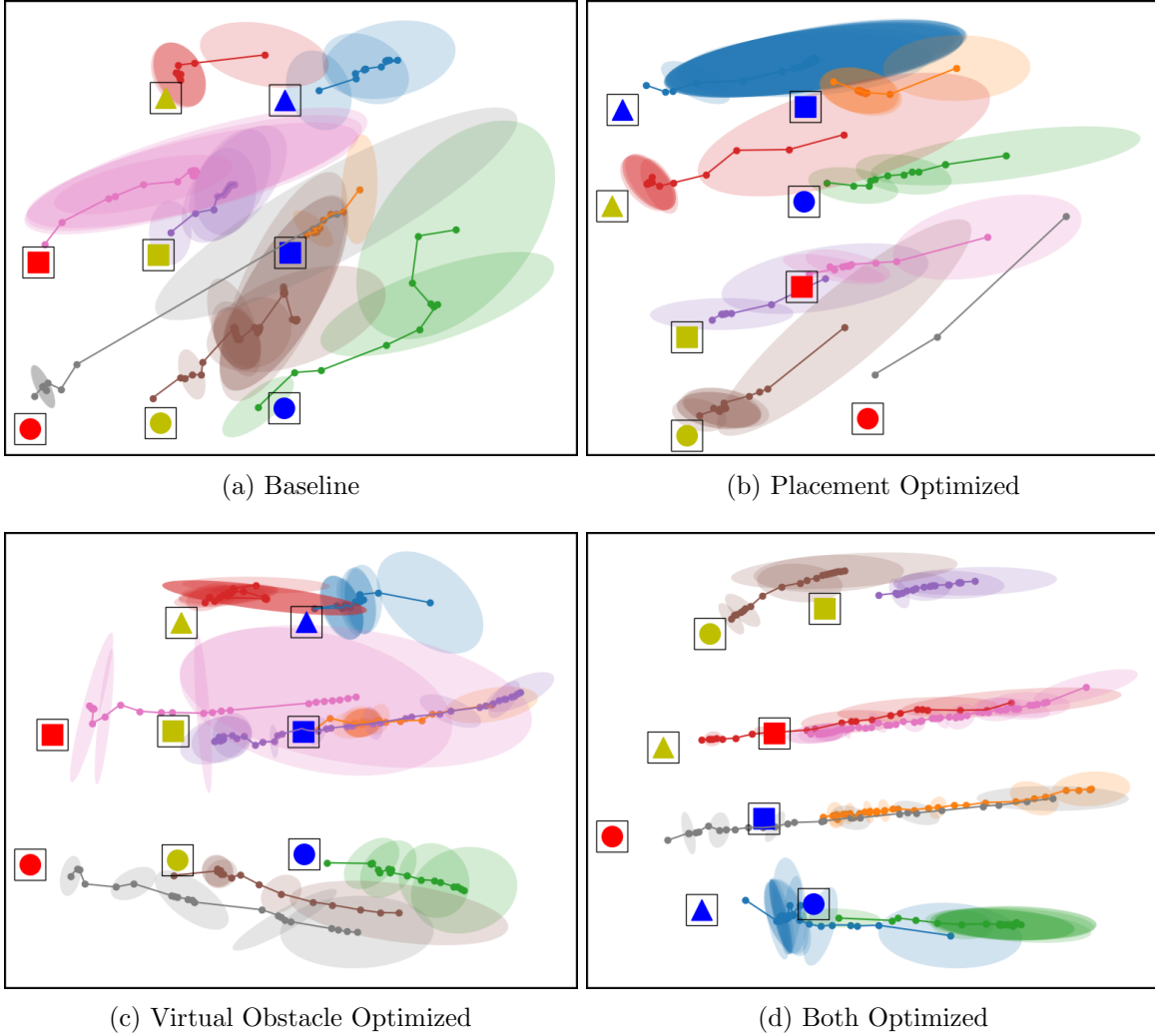


Figure 4.8: Top-down view of the workspace plotting the mean and covariance of the time series multivariate Gaussian for each condition. The Gaussian models are fit to the hand trajectory data and predict the human’s goal (Section 4.4.3). The model in the *Both Optimized* condition has less covariance compared to the models trained in the other environment configurations.

Chapter 5

Discussion and Broader Impact

5.1 Synthesis Across Levels of Intent and Downstream Uses

This thesis opened with the idea of having a cooking collaborator: a robot that must reason about its human partner’s motion, task structure, and preferences in order to be a safe and fluent teammate. We showed in three separate studies that environment design is a productive lever for each of these layers of inference.

Task structure (Chapter 2). In a flat-pack furniture assembly task, a robot that adaptively segments and schedules its kitting actions, while arranging parts on a tray, reduces total task time and improves user-reported subjective metrics relative to fixed kitting strategies. The environmental design variable — which parts to include in the next kit, and where on the tray — directly shapes the human’s upcoming workflow and gives the robot a lever to implement the just-in-time principle. This addresses **RQ1** by identifying **part selection and physical kit layout** as modifiable aspects of the environment, and **RQ2** by solving the resulting bilevel optimization via a cross-entropy method inside a limited-horizon greedy search.

Reward alignment (Chapter 3). When the human’s preferences are themselves the unknown, the robot can use environment design to imagine new scenarios to query the human, thus improving its inference. CRED’s bilevel formulation — an outer Bayesian optimization over environment parameters, an inner counterfactual-reasoning loop over trajectory pairs — achieves higher reward accuracy with fewer queries than state-of-the-art baselines, and users report lower mental workload when answering CRED’s queries. This addresses **RQ1** by identifying **terrain**

types and object placements as modifiable aspects of the environment, and **RQ2** by framing environment selection as Bayesian optimization with a UCB acquisition function.

Low-level motion (Chapter 4). At the finest granularity, the robot can rearrange objects and project virtual AR obstacles to elicit legible human motion, reducing the variance of learned goal-prediction models and improving their data efficiency. This method follows the same bilevel structure as the preceding works: an outer environment-design problem searches for effective workspace configurations, while an inner learning problem evaluates how accurately the robot can infer human goals from the motion induced by each configuration. Because the joint space of physical object placements and virtual obstacles is high-dimensional and combinatorial, the outer problem is solved using quality-diversity search. Specifically, MAP-Elites maintains a diverse population of high-performing environments, reducing sensitivity to local optima while revealing multiple distinct workspace designs that support reliable goal inference. This addresses **RQ1** by identifying **physical object placement plus virtual AR obstacles** as jointly modifiable aspects of the environment, and **RQ2** by using MAP-Elites to explore the high-dimensional configuration space.

Across all three contributions, environment design forms the outer decision problem, while the inner problem models the task execution, preference-inference, or motion-inference process that the environment is intended to improve.

5.2 Future Research Directions

There are two complementary ways in which environment design can improve human-robot collaboration. Across interactions, a robot can generate diverse scenarios that elicit informative human behavior, improve coverage of possible situations, and expose weaknesses in its learned models. Within a particular interaction, the robot can structure the environment to reduce ambiguity, coordinate future actions, and establish a shared workflow. Environment design can therefore both broaden the distribution of experiences from which a robot learns and constrain individual interactions so that they are easier to interpret.

These insights suggest several directions for future research. First, foundation models could treat the environment as an active component of the data-generation process rather than as a fixed source of passive observations. Second, methods for environment generation could move beyond diversity alone and explicitly seek scenarios that are informative, realistic, and relevant to deployment. Third, robots could learn when and how to structure an interaction without unnecessarily restricting human flexibility or agency. Fourth, environment design could be integrated with task execution, enabling robots to decide whether and when the expected future benefits of reorganizing the environment justify its immediate costs. Finally, environment design could be extended beyond improving the robot’s understanding of the human to support mutual understanding between both collaborators.

5.2.1 Active Environment Generation for Foundation Models

Recent advances in robot learning have been driven in part by larger datasets, larger models, and increasingly diverse sources of demonstration data. However, human behavior is inherently multimodal: people may use different motions, strategies, or task decompositions while pursuing the same objective [76]. Consequently, increasing the number of observations may improve coverage without resolving the ambiguity present in each observation. A large dataset can contain many examples of behavior while still providing limited evidence about the intentions, preferences, or constraints that produced that behavior.

The results of this dissertation suggest that robots should reason not only about how much data they collect, but also about the process through which that data is generated. Chapter 3 selects environments that produce preference queries capable of distinguishing among competing reward hypotheses. Chapter 4 modifies physical and virtual workspace features to elicit motions that make human goals easier to infer. In Chapter 2, the robot structures the human’s upcoming workflow through adaptive task segmentation and kit layout. Across these settings, environment design changes the distribution of behavior that the robot observes, allowing the robot to actively create conditions under which learning and inference become easier.

Future foundation models could incorporate this principle by jointly reasoning about the model, the data-collection policy, and the environment in which interaction occurs. Rather than collecting additional demonstrations from a fixed task distribution, a robot could identify uncertainties in its current model and select or construct environments that are expected to resolve them. For example, a robot learning human preferences could generate scenarios that separate otherwise similar reward hypotheses. A manipulation system could reconfigure a workspace to determine whether a failure results from perceptual uncertainty, insufficient task knowledge, or ambiguity in the user’s intended goal. A general-purpose robot could similarly construct curricula that progressively expose weaknesses in its representations or control policies.

This approach introduces several open problems. A robot must estimate the expected information gained from an environmental intervention and balance that benefit against the physical, cognitive, and operational cost of modifying the environment. It must also determine when intervention is appropriate: reconfiguring the workspace may sometimes be more effective than collecting another demonstration, whereas in other cases it may be preferable to ask the human a direct question or act conservatively. The environment-design policy must generalize across users, tasks, and robot embodiments, while adapting to individual differences when needed. Finally, active environment generation must preserve safety and ecological validity so that the behaviors it elicits remain representative of those that will occur during deployment.

These challenges motivate a shift from passive dataset construction toward active robot learning through environment design. Under this view, foundation models would not only learn from the interactions available to them. They would also reason about which interactions are missing and how the surrounding environment could be modified to produce them.

5.2.2 Purposeful Diversity for Training and Evaluation

The ability to generate diverse environments is a key enabler of active data collection. In Chapter 3, diverse environments produce preference queries that reveal different aspects of the human’s reward function. In Chapter 4, MAP-Elites identifies a repertoire of distinct workspace

configurations that support reliable goal inference. These examples show that diversity can help a robot explore multiple ways of eliciting informative human behavior instead of committing to a single locally optimal environment.

Diversity also plays an important role throughout robot learning. Domain randomization varies properties of simulated environments to improve transfer to the physical world [1]. Data augmentation increases variation in the observations used to train perceptual and control models [3]. Curriculum-generation methods construct sequences of tasks with increasing difficulty [17], while automated testing methods search for scenarios that expose failures or unsafe behavior [23]. These approaches can all be viewed as forms of environment design, but they pursue different objectives. Domain randomization primarily seeks coverage over nuisance variation, active learning seeks information about an unknown model or reward, and adversarial evaluation seeks situations in which a system is likely to fail.

Future work should therefore distinguish between environments that are merely different and environments that are useful for a particular learning or evaluation objective. Random variation may produce broad visual or geometric diversity while providing little new information about the model’s capabilities. Conversely, a small number of carefully selected environments may be highly diagnostic because they separate competing hypotheses, elicit rare behaviors, or reveal specific failure modes. A central research question is thus how to generate environments that are simultaneously diverse, informative, realistic, safe, and relevant to downstream deployment.

Quality-diversity methods provide one promising framework for addressing this problem. Rather than searching for a single maximally difficult or informative scenario, these methods can maintain a repertoire of high-performing environments that differ along meaningful behavioral or task dimensions. Such a repertoire could be used as an active curriculum for a foundation model, with environments selected according to the model’s current uncertainty, competence, or deployment risk. The behavioral descriptors used by quality-diversity algorithms could represent factors such as task complexity, degree of human–robot coupling, type of perceptual ambiguity, or category of model failure.

For human–robot interaction, however, environment diversity raises additional considerations. Changing the environment does not merely perturb the robot’s observations; it can alter human behavior, strategy, workload, and sense of agency. An environment that is highly informative to the robot may feel unnatural, confusing, or burdensome to the human. Generated scenarios may also elicit behaviors that are technically valid but unrepresentative of ordinary interaction. Future methods should therefore optimize diversity and informativeness jointly with human-centered criteria such as naturalness, comfort, cognitive load, and willingness to participate.

Environment generation could also support validation before real-world deployment. Instead of evaluating a robot on a fixed collection of benchmark tasks, a testing system could search for environments that expose uncertainty, distribution shifts, or unsafe interactions. This would allow evaluation to move beyond average-case performance toward a more systematic understanding of where and why a model fails. For systems that interact closely with people, such testing could identify combinations of environmental conditions and human behaviors that are rare but consequential, providing stronger evidence for safe and reliable deployment.

5.2.3 Structuring Interaction to Reduce Ambiguity

Environment design does not always need to increase variation. Within a particular interaction, the robot may instead improve collaboration by introducing structure that reduces the number of plausible actions or interpretations. Chapter 2 illustrates this role through adaptive task segmentation and kit layout. By deciding which parts to present and where to place them, the robot shapes the sequence of actions available to the human and establishes a shared task structure. Similarly, Chapter 4 uses physical and virtual constraints to make human motion more legible and therefore easier for the robot to interpret.

This suggests viewing environment design as a form of interaction regularization. Just as regularization introduces assumptions that make a learning problem better posed, environmental structure can make a collaborative task easier to coordinate by reducing unnecessary degrees of freedom. Tools can be grouped according to an assembly sequence, handoff regions can be made

spatially distinct, and virtual cues can separate otherwise ambiguous motion paths. These interventions need not fully determine the human’s behavior. Instead, they can preserve several valid strategies while making their consequences more predictable to the robot.

A major direction for future work is to learn when environmental structure is helpful and when it becomes overly restrictive. Stronger constraints may improve predictability but reduce flexibility, efficiency, creativity, or human agency. A workspace optimized for one user or one task strategy may be inefficient for another. Environmental interventions may also cause people to adapt strategically, invalidate previously learned behavior models, or become less effective as users gain experience. The robot must therefore reason about the tradeoff between reducing ambiguity and preserving meaningful choice.

Future systems could adapt the amount of structure according to the robot’s confidence and the needs of the interaction. When intent is clear, the robot may leave the environment largely unconstrained. As uncertainty increases, it could introduce lightweight cues, change the ordering or placement of task materials, or restrict only those actions that would create substantial risk. The environment could then be relaxed again once the robot and human have established a shared understanding. Such methods would turn environment design into a closed-loop process rather than a one-time optimization performed before the task.

This direction also requires models of how people respond to environmental interventions over time. An arrangement that initially improves legibility may eventually become unnecessary as the collaborators develop shared conventions. Conversely, repeated changes to the workspace may make the robot less predictable and increase the human’s workload. Longitudinal studies could examine how environmental structure interacts with learning, trust, expertise, and team adaptation. These studies would help determine whether robots should optimize environments independently for each interaction or gradually develop persistent conventions with particular users.

More broadly, diversity and constraint should not be treated as contradictory objectives. They operate at different scales. Across interactions, generating diverse environments can improve coverage and expose the robot to a broad range of human behaviors. Within an interaction,

selectively constraining the environment can reduce ambiguity and support reliable coordination. A long-term goal is to develop systems that can manage uncertainty at both scales, generating broad experience during learning while creating sufficient structure for effective collaboration during execution.

5.2.4 Integrating Environment Design with Task Execution

The methods in this dissertation largely assume that the robot can modify the environment at a designated stage of the interaction. In Chapter 4, for example, the workspace is optimized before the human begins the task, and the resulting configuration remains fixed during execution. This separation between environment design and task execution makes the optimization problem more tractable, but it assumes that the robot has sufficient time, knowledge, and physical capability to reorganize the environment in advance. In many real-world settings, these assumptions may not hold. The robot may learn relevant information only after the task has begun, the environment may change unexpectedly, or some modifications may be impractical until particular objects become available.

A natural extension is to consider environment design as an online process. Rather than selecting a single configuration before execution, the robot could reorganize the workspace in response to its evolving beliefs about the human, the task, and its own capabilities. For example, if the robot becomes uncertain about the human’s intended goal, it could reposition future task materials to make subsequent actions easier to distinguish. If an object placement causes repeated grasp failures, the robot could move that object or clear nearby clutter before attempting later actions. Similarly, the robot could introduce or remove physical and virtual constraints as the interaction progresses, adapting the amount of structure to the current level of uncertainty.

Online reorganization is not always beneficial, however. Modifying the environment consumes time and robot effort, may interrupt the human’s workflow, and can make the interaction less predictable if changes occur too frequently. A robot must therefore compare the immediate cost of an environmental intervention with its expected future benefit. This benefit may include more

accurate intent inference, easier manipulation, reduced risk, or improved performance on later stages of the task. Future work could formalize this tradeoff by treating environment-modification actions as decisions within a sequential planning problem.

Under this formulation, the robot would plan not only actions that directly advance the task, but also actions that improve the conditions under which future task actions will be performed. Moving an object may provide no immediate task reward, yet it may simplify a later grasp, improve visibility, or make the human’s subsequent motion more legible. Environment design thus becomes analogous to information-gathering or setup actions in task and motion planning: the robot may incur a short-term cost to improve future execution. The planning problem must determine which modifications are worthwhile, when they should occur, and how they should be coordinated with the human’s ongoing activity.

Chapter 2 provides an initial example of this tighter coupling between environment design and task execution. The robot selects and arranges parts for the next stage of assembly while the larger task is still underway, using just-in-time kitting to shape the human’s future workflow. Future systems could extend this principle beyond discrete kit preparation to continuous workspace reorganization, jointly optimizing task progress, environmental configuration, and human–robot coordination over time.

This direction also raises questions about predictability and human control. Even when reorganization improves objective performance, people may find a continually changing workspace disruptive or difficult to understand. The robot may therefore need to communicate proposed changes, preserve stable regions of the workspace, or allow the human to approve, reject, and reverse environmental interventions. Effective online environment design will require not only planning algorithms that account for long-term benefits, but also interaction policies that make reorganization timely, legible, and compatible with the human’s workflow.

5.2.5 From Human Legibility to Mutual Legibility

The methods developed in this dissertation primarily use environment design to improve the robot’s understanding of the human. The robot rearranges objects, selects task components, or introduces virtual constraints so that human preferences and goals become easier to infer. However, effective collaboration also depends on the human’s ability to understand the robot. A natural extension is therefore to design environments that support mutual legibility, making the goals, plans, capabilities, and uncertainty of both collaborators easier to interpret.

Environmental cues could complement conventional communication modalities such as robot motion, speech, or graphical interfaces. For example, the arrangement of tools and parts could indicate the sequence of actions the robot expects to perform. AR cues could display intended motion corridors, handoff locations, or areas the robot considers unsafe. Changes in the workspace could communicate which task the robot believes is active or which object it expects the human to use next. Rather than requiring the human to infer the robot’s internal state solely from its motion, the environment itself could become a persistent communication medium.

This perspective is particularly relevant when a robot’s uncertainty affects how the human should behave [79]. A robot could use environmental cues to indicate that it is uncertain about a goal, that it requires additional information, or that it intends to defer a decision to the human. Such cues may help people identify when intervention is needed without requiring continuous attention to the robot. They may also make failures easier to anticipate by exposing mismatches between the human’s expectations and the robot’s current plan.

Future work could study environment design as a bidirectional negotiation process. Rather than having the robot unilaterally optimize the workspace, the human and robot could jointly modify it to establish conventions, allocate responsibilities, or communicate preferences. A person might move an object to indicate task priority, while the robot might reposition materials to propose a more efficient sequence. Over repeated interactions, these modifications could develop into shared environmental conventions that encode aspects of the team’s workflow.

Mutual environment design also raises questions of control and consent. Environmental interventions can influence human behavior even when that influence is subtle. A robot should therefore make its modifications understandable, reversible, and proportionate to the needs of the task. Users should be able to reject or override arrangements that conflict with their preferences. Evaluations should consider not only efficiency and inference accuracy, but also whether users understand why an intervention occurred, whether they feel able to act differently, and whether the resulting interaction remains collaborative rather than coercive.

Ultimately, environment design could become a shared language between humans and robots. Physical arrangements, virtual constraints, and task structure could communicate expectations in both directions while simultaneously making behavior easier to predict. Extending environment design from one-way inference to mutual legibility would support teams in which both the human and the robot can understand, influence, and adapt to the evolving structure of their collaboration.

5.3 Conclusion

Together, these directions suggest a broader view of environment design as an adaptive capability that operates throughout the lifecycle of a robotic system. Environments can be generated to improve learning, structured to reduce ambiguity during interaction, and reorganized online to support future task performance. Realizing this vision will require robots to reason jointly about information, task progress, intervention costs, safety, and human agency. Ultimately, environment design should not be treated as a one-time preprocessing step or a capability exercised solely by the robot, but as an ongoing and potentially collaborative process through which humans and robots create the conditions for effective coordination.

Bibliography

- [1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik’s cube with a robot hand. arXiv preprint arXiv:1910.07113, 2019.
- [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] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pages 10134–10141. IEEE, 2024.
- [4] Erdem Biyik and Dorsa Sadigh. Batch active preference-based learning of reward functions. In Conference on robot learning. PMLR, 2018.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] Judith Bütepage, Hedvig Kjellström, and Danica Kragic. Anticipating many futures: On-line 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.
- [10] 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.

- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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>.
- [15] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. Nature, 521(7553):503–507, 2015.
- [16] 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.
- [17] 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.
- [18] 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.
- [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, 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.

- [25] 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>.
- [26] 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.
- [27] 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.
- [28] David E Goldberg and John Henry Holland. Genetic algorithms and machine learning. Machine Learning, 1988.
- [29] 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.
- [30] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive Metropolis algorithm. Bernoulli, 7(2):223 – 242, 2001.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] 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.
- [35] Sarah Keren, Luis Pineda, Avigdor Gal, Erez Karpas, and Shlomo Zilberstein. Equi-reward utility maximizing design in stochastic environments. HSDIP, 2017:19, 2017.
- [36] Anis Koubaa. Robot Operating System (ROS): The Complete Reference (Volume 2). Springer Publishing Company, Incorporated, 1st edition, 2017. ISBN 331954926X.
- [37] 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.

- [38] 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.
- [39] 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.
- [40] 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.
- [41] 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>.
- [42] 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.
- [43] 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.
- [44] Riccardo Maderna, Matteo Poggiali, Andrea Maria Zanchettin, and Paolo Rocco. An on-line scheduling algorithm for human-robot collaborative kitting. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 11430–11435. IEEE, 2020.
- [45] 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.
- [46] Michail Maniadakis, Emmanouil Hourdakakis, 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>.
- [47] Luisa Mao, Garrett Warnell, Peter Stone, and Joydeep Biswas. Pacer: Preference-conditioned all-terrain costmap generation. *IEEE Robotics and Automation Letters*, 2025.
- [48] 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.
- [49] 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.

- [50] 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>.
- [51] Meinard Müller. Dynamic time warping. Information retrieval for music and motion, pages 69–84, 2007.
- [52] 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.
- [53] Olle Nilsson and Antoine Cully. Policy gradient assisted map-elites. In Proceedings of the Genetic and Evolutionary Computation Conference, pages 866–875, 2021.
- [54] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014. URL <https://github.com/bayesian-optimization/BayesianOptimization>.
- [55] 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.
- [56] OpenStreetMaps. Planet dump retrieved from <https://planet.osm.org>. <https://www.openstreetmap.org>, 2017.
- [57] 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.
- [58] 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.
- [59] Shaoting Peng, Haonan Chen, and Katherine Driggs-Campbell. Towards uncertainty unification: A case study for preference learning. arXiv preprint arXiv:2503.19317, 2025.
- [60] 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.
- [61] 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.
- [62] 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.

- [63] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In IJCAI, volume 7, pages 2586–2591, 2007.
- [64] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Darius M Gavrilu, and Kai O Arras. Human motion trajectory prediction: A survey. The International Journal of Robotics Research, 39(8):895–935, 2020.
- [65] 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.
- [66] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
- [67] 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.
- [68] Thomas B Sheridan. Human–robot interaction: status and challenges. Human factors, 58(4): 525–532, 2016.
- [69] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25, 2012.
- [70] 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>.
- [71] Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. Journal of Cognitive Neuroscience, 11(1):126–134, 1999.
- [72] 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.
- [73] 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.
- [74] 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.
- [75] Beverly Townsend. Line Balancing And Jit Kitting. Boca Raton, Fl: Crc Press, 2012.
- [76] Yi-Shiuan Tung, Matthew B Luebbers, Alessandro Roncone, and Bradley Hayes. Stereoscopic virtual reality teleoperation for human robot collaborative dataset collection. In 7th International Workshop on Virtual, Augmented, and Mixed-Reality for Human-Robot Interactions.

- [77] 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.
- [78] 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.
- [79] Yi-Shiuan Tung, Himanshu Gupta, Wei Jiang, Bradley Hayes, and Alessandro Roncone. Causal influence detection for human robot interaction. Workshop on Causal Learning for Human-Robot Interaction (Causal-HRI), 2024.
- [80] 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.
- [81] Yi-Shiuan Tung, Himanshu Gupta, Gyanig Kumar, Heyang Huang, Bradley Hayes, and Alessandro Roncone. Environment design for reliable shared autonomy with probabilistic guarantees. arXiv preprint arXiv:2607.15487, 2026.
- [82] 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.
- [83] 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>.
- [84] Peter J. M. van Laarhoven and Emile H. L. Aarts. Simulated annealing: Theory and applications. In Mathematics and Its Applications, 1987.
- [85] 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.
- [86] 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.
- [87] Webots. <http://www.cyberbotics.com>. URL <http://www.cyberbotics.com>. Open-source Mobile Robot Simulation Software.
- [88] 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.
- [89] 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.

- [90] 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.
- [91] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [92] 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.
- [93] 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.
- [94] 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.
- [95] 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.
- [96] 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.
- [97] 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.
- [98] 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.
- [99] 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 ($64 \rightarrow 32$, $32 \rightarrow C$; 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.

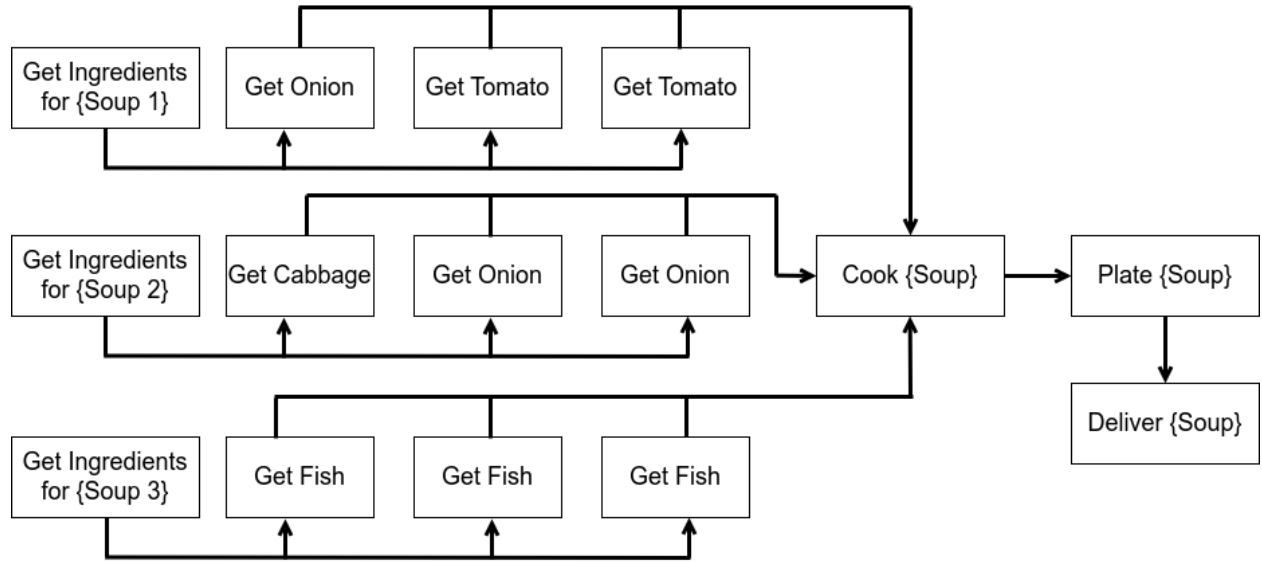


Figure B.1: Task graph for the Overcooked game environment. Arrows indicate precedence constraints such that if $t_i \rightarrow t_j$ then subtask t_i has to be completed before subtask t_j can begin. A total of 3 soups have to be delivered, and each soup consists of 3 ingredients. The soups and ingredients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

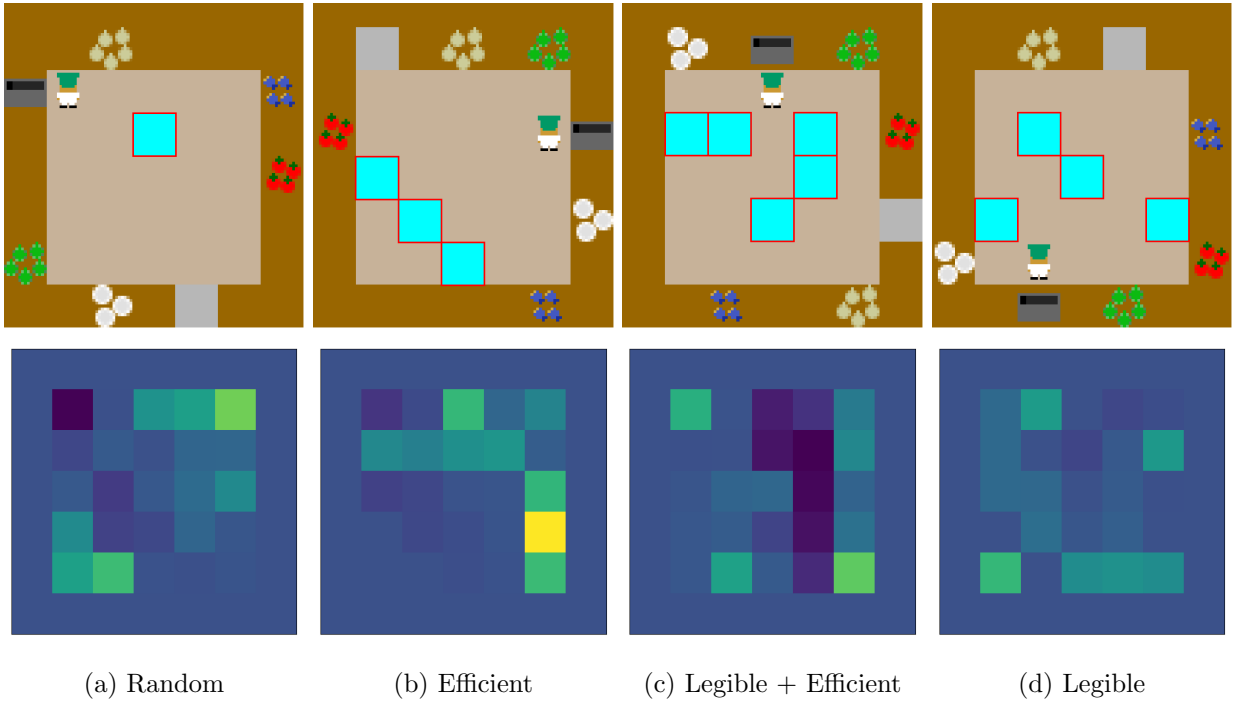


Figure C.1: Learned reward functions from MaxEnt IRL.

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, and 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 [57].

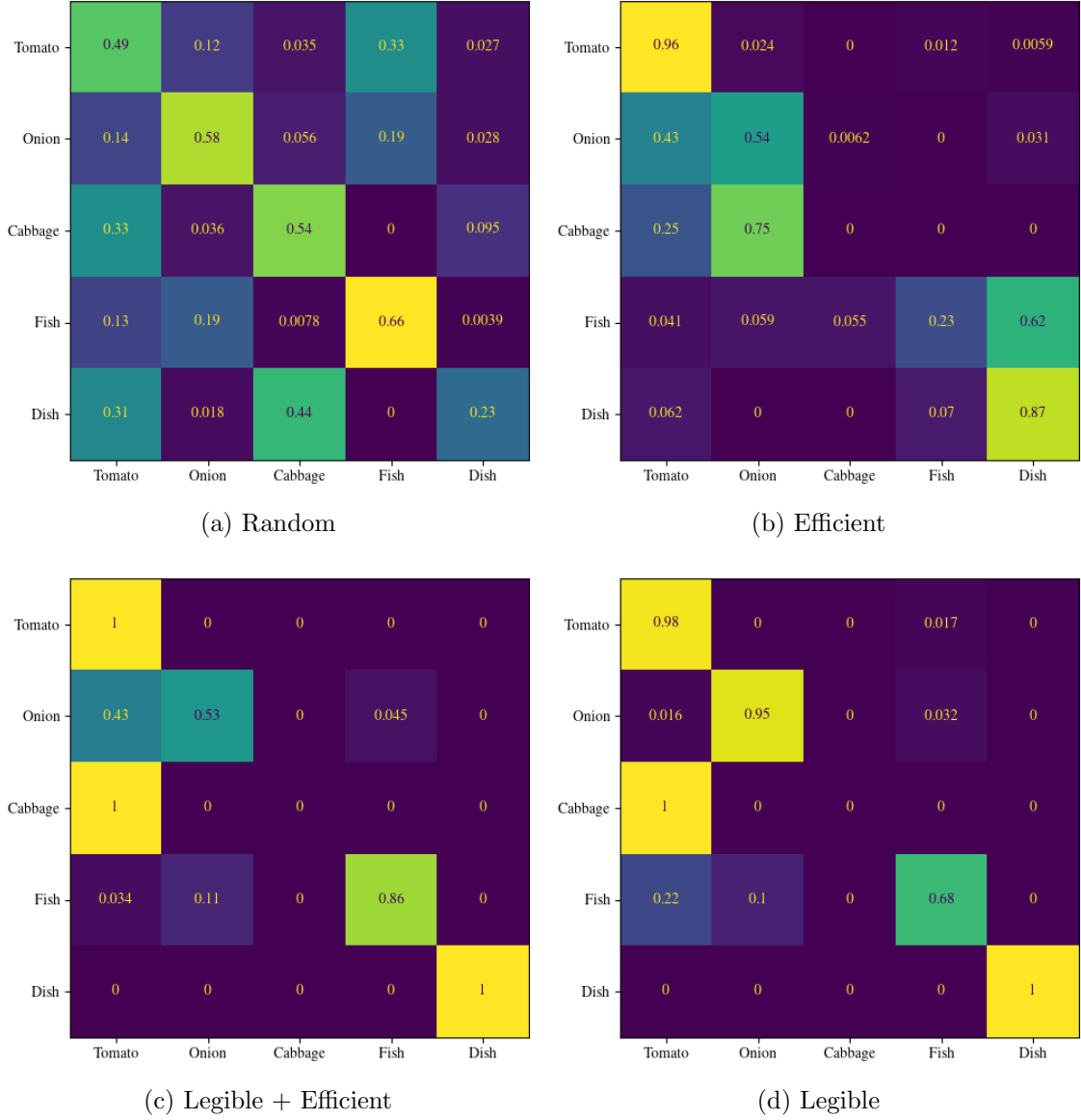


Figure D.1: Confusion matrix for the Bayesian predictor in the different generated Overcooked layouts.

Table D.2: Tabletop Goal Prediction

	Baseline	Placement Optimized	Virtual Obsta- cle 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

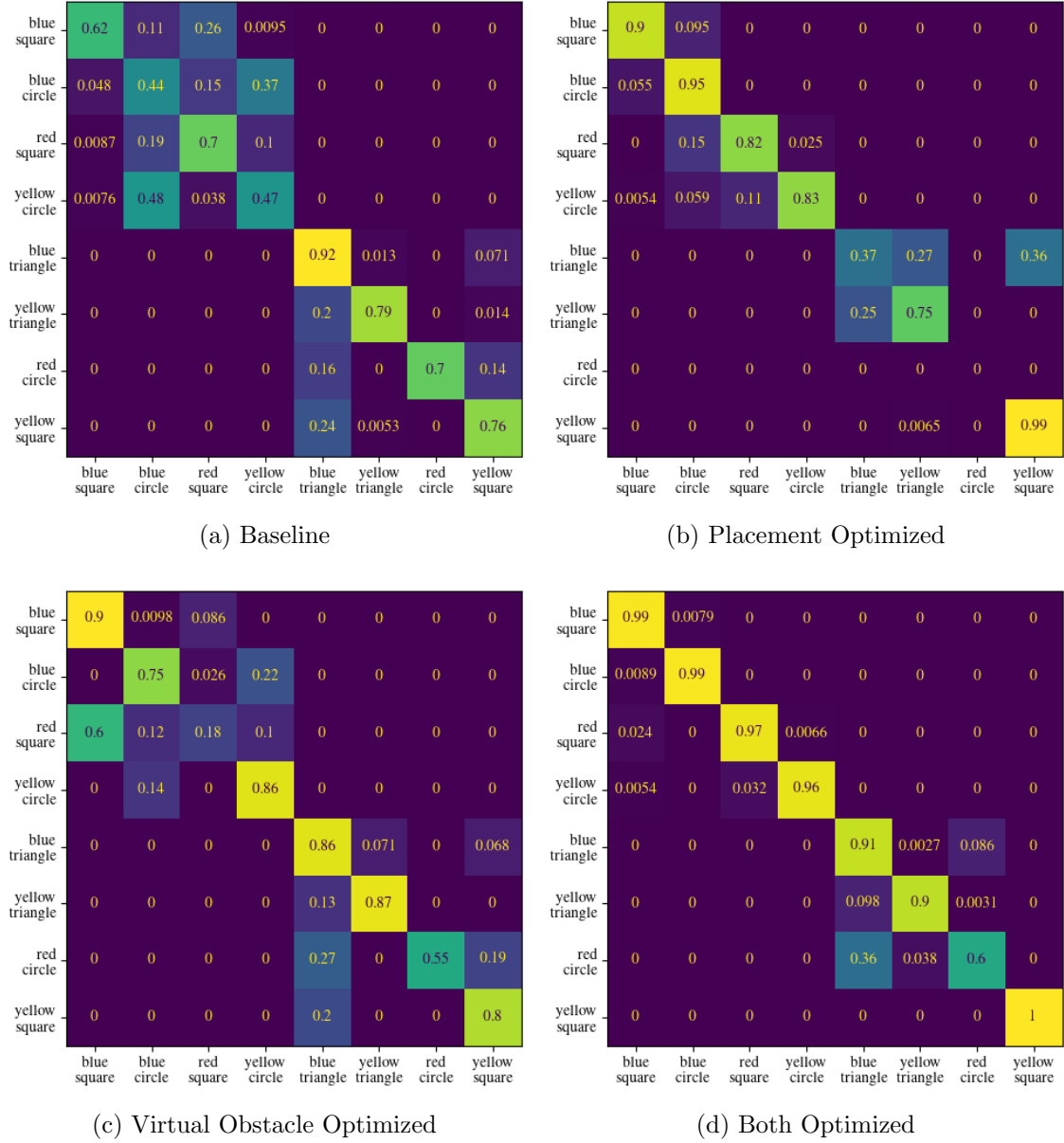


Figure D.2: Confusion matrix for the time series multivariate Gaussian in the different generated workspace configurations. The x -axis shows the true target, and the y -axis is the predicted target.

Appendix E

Comparisons with other Optimization Methods

This appendix provides supplementary comparisons of MAP-Elites against Novelty Search with Local Competition (NSLC) [41] and Differential Evolution (DE) [70], 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 (2.19)	-202.52 (3.96)	-229.22 (4.68)	-186.31 (3.27)
NSLC	-95.44 (10.73)	-175.43 (15.92)	-190.54 (11.89)	-169.63 (20.39)
MAP-Elites	-86.37 (7.55)	-167.88 (16.13)	-173.52 (8.20)	-162.97 (10.60)