

Introduction:

Building on my work with preference learning over the summer, I worked on a research paper focusing on the active routing preference learning using counterfactual reasoning and environment design. The paper was intended for submission in October. My contributions included:

- Curating literature for the related works section.
- Designing system diagrams.
- Conducting experiments with the OpenStreetMap API and OSMnx, which involved enhancing road features such as type and elevation, and simulating traffic and elevation.

This effort deepened my understanding of the area for further research.

During this time, we proposed an intriguing idea: integrating active preference learning with the language-to-rewards framework by leveraging Linear Temporal Logic (LTL).

Why LTL?

LTL provides a formal, mathematical language for representing temporal and logical constraints, enabling nuanced preferences such as:

- **“Always avoid obstacles”** (G for Globally).
- **“Eventually reach the goal while avoiding dangerous areas”** (F for Future).

This structured representation ensures that complex temporal requirements are both verifiable and interpretable. LTL enables robots to:

- Formalize preferences into logical, environment-agnostic rules.
- Iteratively converge to accurate specifications with efficiency in user queries.
- Verify planned paths against learned constraints before execution.

For example, in a disaster-relief scenario, a drone tasked with package delivery may have ambiguous initial instructions, such as avoiding smoke-filled areas. By querying the user with pairs of candidate trajectories, the system identifies preferences like “Globally avoid smoke” (G (\neg SmokeArea)) or “Always maintain a safe distance from obstacles” (G (SafeDistance)), which solidify into verifiable LTL formulas.

Why Active Preference Learning?

When users cannot provide complete specifications upfront due to the complexity of articulating every constraint, traditional approaches often fail. I implemented an active preference learning system that strategically queries users to iteratively uncover their preferences, achieving:

- Reduced reliance on full demonstrations by leveraging targeted queries.
- Interactive learning of nuanced behaviors tailored to user intent.

- Efficient query strategies that minimize interactions while ensuring accurate specification learning.

How Does LTL Enhance Active Preference Learning?

After learning preferences, I encoded them as Linear Temporal Logic (LTL) formulas, which:

- Established a robust framework to verify whether planned actions satisfy learned preferences before execution.
- Provided formal guarantees on task execution through model checking.
- Enabled scalable integration into dynamic environments with rigorous validation.
- Infer behavioral rules that generalize preferences and states, since they are environment-agnostic yet behavior-specific.

We developed a framework that combines active preference learning with LTL, ensuring correctness, adaptability, and transparency. This integration empowers robots to generalize behaviors across diverse, dynamic environments. By embedding LTL specifications into preference learning, my work addresses key challenges, including:

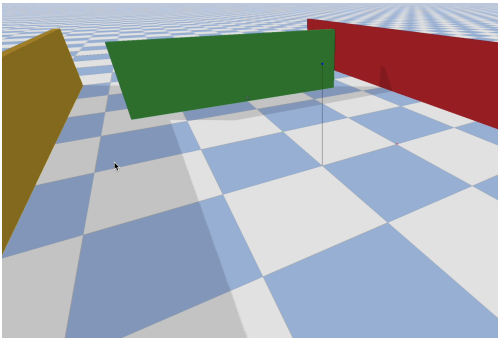
- Adaptability to dynamic and ambiguous conditions.
 - Generalization of learned behaviors to new scenarios.
 - Formal validation and verification of task execution.
-

Methodology:

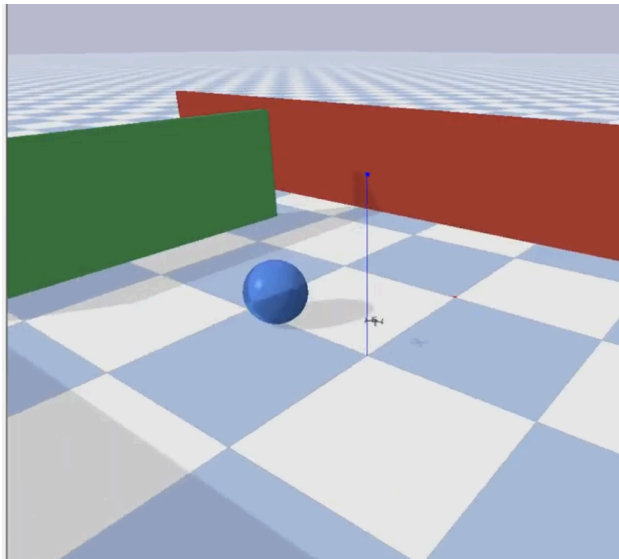
The methodology is structured into five phases, each addressing a critical component of the system:

Phase 1: Environment Setup

- **Environment A:** A basic obstacle avoidance scenario with static walls created in a [PyBullet simulation environment](#).



- **Environment B:** A complex setup featuring dynamic objects and multi-task constraints, designed to test the robot's ability to generalize learned behaviors without explicit instructions.



Phase 2: Active Preference Learning

- **LTL Rule Learning:** User specifications are translated into LTL formulas using LLMs (ChatGPT). Initial probabilities are assigned to inferred rules for dynamic refinement.

Phase 3: Planning and Verification

- **Conversion to Büchi Automaton:** LTL formulas are converted into Büchi Automata for integration with the robot's planning system.
- **Product MDP Construction:** Combines Büchi Automaton with the environment's MDP, enabling the computation of optimal trajectories satisfying LTL specifications.
- **Solve Product MDP:** Solved the product MDP using value iteration to give an optimal trajectory.

Phase 4: Feedback Loop

- **Violation Handling:** Action sequences violating LTL constraints are modified based on user feedback and alternative paths in the Product MDP.
- **Iterative Learning:** LTL rules are refined continuously using feedback from model checking and user preferences. Based on how likely each rule fits the observed actions, probabilities can be assigned to LTL formulas. If a robot avoids walls and reaches goals 90% of the time, ML might infer that this pattern (the LTL rule) is highly probable.

Phase 5: Evaluation and Novelty Demonstration

- The evaluation focuses on testing the robot's generalization across diverse environments with varying ambiguities, measuring success by adherence to learned preferences (LTL satisfaction) and behavior generalization. Uncertainty modeling will be assessed by comparing probabilistic inference with ground-truth preferences. Combining LTL with active preference learning enhances task generalization by enabling environment-agnostic yet behavior-specific rules, provides formal guarantees through pre-execution verification, and improves performance in ambiguous environments.
-

Literature Review:

This section highlights key works that informed our research approach, focusing on reward modeling, preference learning, and task specification frameworks.

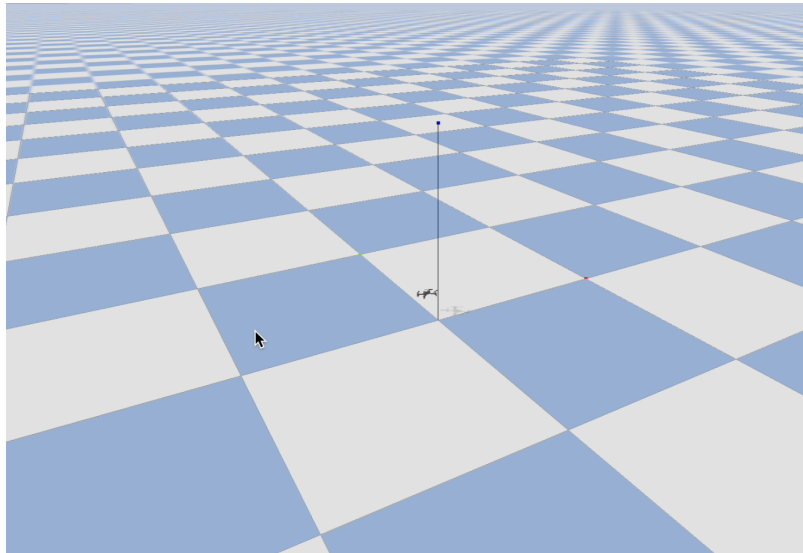
1. [Active Preference-Based Learning of Reward Functions:](#)
 - This work explores synthesizing pairwise comparison queries to extract maximum information about human preferences efficiently. The algorithm optimizes query selection by maximizing mutual information between human input and reward function parameters.
 - **Learning from this paper:** Coming up with (good) ‘k’ trajectories is a difficult and resource intensive process, but then ranking based on preferences over pairs is easier.
 - **Limitations:** The method relies on linear reward models and struggles to scale for complex tasks.
2. [Data-Efficient Learning of Natural Language to LTL Translators:](#)
 - Proposes mapping natural language instructions to LTL specifications using templates and language models. This bridges the gap between informal user input and formal task representations.
3. [Lang2LTL: Translating Natural Language Commands to Temporal Specifications:](#)
 - Demonstrates the use of large language models to convert natural language into temporal logic, which serves as the basis for task specification in robotic systems. This aligns well with our goals of formalizing human preferences in robot tasks.

The above works collectively informed our approach, emphasizing the need for combining preference-based learning with formal task representations like LTL. While existing methods offer foundational insights, our project extends these principles by focusing on ambiguous and dynamic environments, using LTL for generalization and verification.

Experiments:

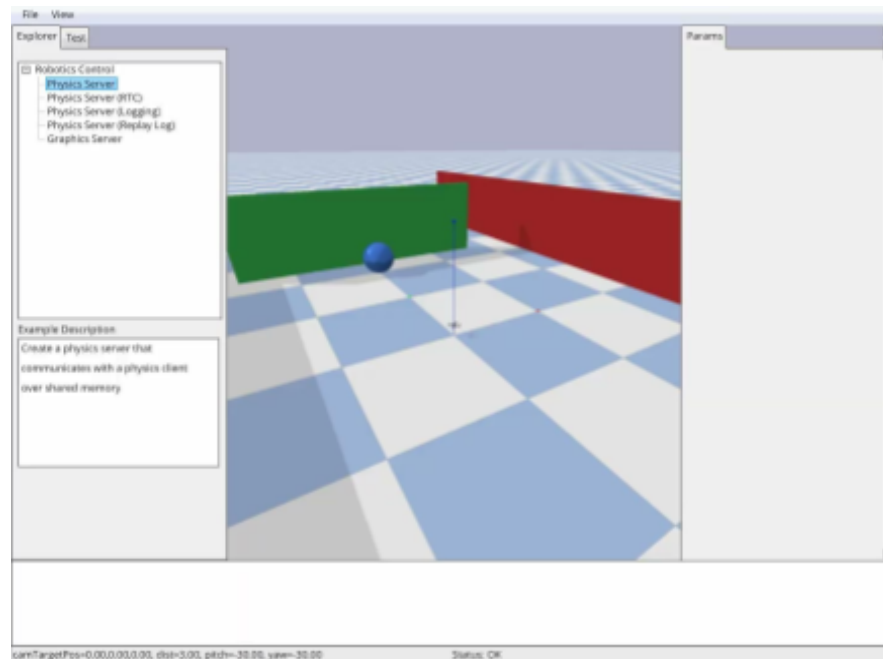
Simulation of PyBullet Drones:

- [Learning to Fly -- a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control](#): I used PyBullet Drones to test different algorithms for trajectory generation and learning.
- This is an example of the simulation environment:

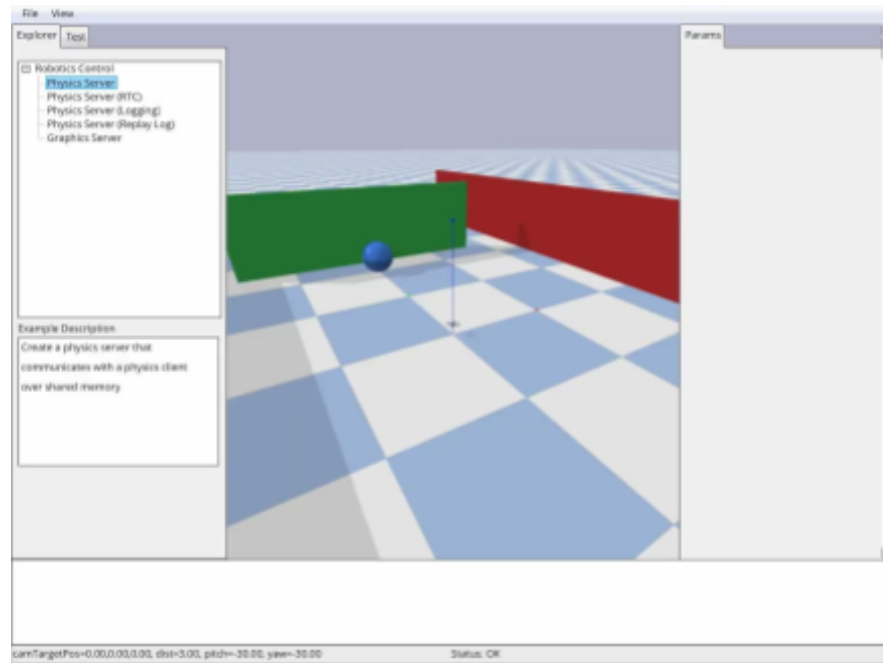


Examples and Simulations: (click to see the videos)

- Shows the drone in the environment avoiding static walls:



- Shows the drone in the environment avoiding dynamic objects:



Results:

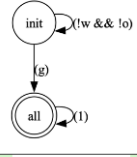
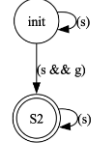
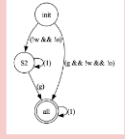
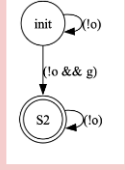
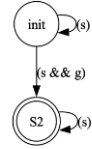
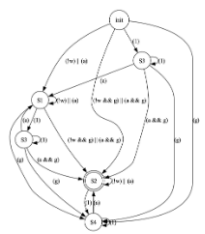
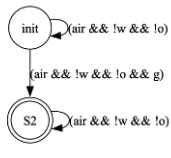
Natural Language	LTL Formula	Büchi Automata Diagram	Categorical Analysis
Eventually reach the goal without ever colliding with walls or obstacles.	$(\neg W \wedge \neg O) U G$		Correct
Always remain in a safe corridor and eventually reach the goal.	$G(S) \wedge F(G)$		Correct
Eventually reach the goal and never collide with walls or obstacles.	$F(G) \wedge (\neg W \wedge \neg O)$		Incorrect: This formula implies that the goal will eventually be reached, but it does not ensure that there are no collisions during the process.
The drone must avoid obstacles while navigating to the goal.	$G(\neg O) \wedge F(G)$		Incorrect: The drone never collides with obstacles ($G(\neg O)$) and eventually reaches the goal, but it fails to enforce drone to make progress toward the goal while avoiding obstacles.
Never come too close to any obstacle and eventually arrive at the goal.	$G(S) \wedge F(G)$		Correct
If the drone detects a wall ahead, it must perform an avoidance maneuver and still eventually reach the goal.	$G(W_{\text{ahead}} \rightarrow F(A)) \wedge F(G)$		Correct
The drone must never collide with obstacles or walls, remain within the allowed airspace at all times, and eventually land at the goal.	$G(Air \wedge \neg W \wedge \neg O) \wedge F(G)$		Correct

Table 1: Mapping of Natural Language Prompts to LTL Formulas

Future Direction:

- I plan to complete the phases 4 and 5, incorporating a verification and feedback loop that can refine LTL specifications.
- I plan to benchmark the algorithm against state-of-the-art methods, evaluating its performance on key metrics such as:
 - Generalization across diverse and complex environments.
 - Efficiency in learning user preferences with minimal queries.
 - Robustness in managing ambiguity and uncertainty effectively.
- I also want to work on multiple simulation environments with different tasks like home tasks using [ai2Thor](#) environment.

Through this independent study, I deepened my understanding of robotics, path planning, reinforcement learning and advanced topics like LTL, Omega Automata, and Büchi Automata, exploring their applications in trajectory validation and preference learning to enhance efficiency and robustness. This work involved simulations with existing models, analysis of recent advancements, and identifying opportunities for impactful contributions.