CONFIDENTIAL. Limited circulation. For review only.

Robots that Suggest Safe Alternatives

Hyun Joe Jeong[1], Rosy Chen[2], and Andrea Bajcsy[2]

***Abstract*— Goal-conditioned policies, such as those learned via imitation learning, provide an easy way for humans to influence what tasks robots accomplish. However, these robot policies are not guaranteed to execute safely or to succeed when faced with out-of-distribution goal requests. In this work, we enable robots to know when they can confidently execute a user's desired goal, and automatically suggest safe alternatives when they cannot. Our approach is inspired by control-theoretic safety filtering, wherein a safety filter minimally adjusts a robot's candidate action to be safe. Our key idea is to pose alternative suggestion as a safe control problem in *goal* space, rather than in action space. Offline, we use reachability analysis to compute a goal-parameterized reach-avoid value network which quantifies the safety and liveness of the robot's pre-trained policy. Online, our robot uses the reach-avoid value network as a safety filter, monitoring the human's given goal and actively suggesting alternatives that are similar but meet the safety specification. We demonstrate our Safe ALTernatives (SALT) framework in simulation experiments with Franka Panda tabletop manipulation. We find that SALT is able to learn to predict successful and failed closed-loop executions, is a less pessimistic monitor than open-loop uncertainty quantification, and proposes alternatives that consistently align with those that people find acceptable.**

I. Introduction

Imagine that your friend asks you to grab a mug from the top kitchen shelf. Intuitively, you know that trying to reach it will be dangerous because you will drop the mug. Instead of attempting the unsafe task or asking your friend to get it for you, you may naturally suggest an alternative that you can safely accomplish: "I can't reach your mug, but are you ok with this cup on the lower shelf instead?" How can we get our robots to operate in same manner?

In this paper, we want to endow robots with the ability to know when they can confidently execute a user's desired goal and propose safe alternatives when they cannot. Specifically, we study goal-conditioned robot policies [1] such as those obtained via imitation learning [2]. While this paradigm has enabled robots to learn complex behaviors and adapt to specified goals online [3], these learned policies can degrade when faced with out-of-distribution goal requests or states [4]. In other words, given a pre-trained goal-conditioned policy, it is hard to ensure that the closed-loop robot behavior will always be safe (e.g., doesn't collide with the environment) and performant (e.g., will successfully pick up the cup) for any new user goal and initial state. Prior works have quantified policy uncertainty [5] or developed out-of-distribution input detectors [6], but these approaches are only a monitoring mechanism; they don't provide a way for the robot to actively propose alternatives that it can accomplish safely and effectively.

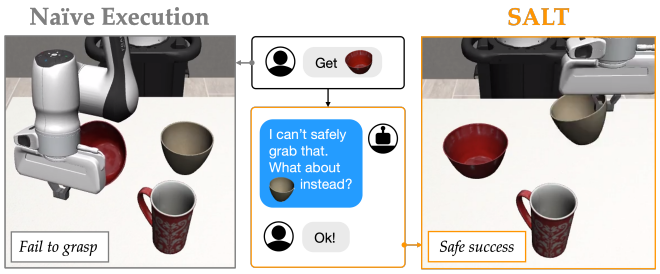

Figure 1: **Safe ALTernatives (SALT).** If a robot naively executes a user's request, it can fail for a variety of reasons. For example, a request to pick up the red bowl leads the robot to fail to grasp. Our SALT framework enables a robot to detect if it can successfully accomplish a user's original goal; if it cannot, the robot automatically proposes an alternative it can safely succeed at (e.g., get brown bowl). Videos on the project website: https://cmu-intentlab.github.io/salt/.

To close this gap, we propose that robots suggest safe alternatives. Our approach is inspired by safety filtering techniques from control theory [7]. Traditional safety filters take a candidate robot action (e.g., generated by a pre-trained policy) and minimally adjust (i.e., "filter") it to be safe. The filtering mechanism projects the candidate action onto the safe and live (i.e. goal-reaching) control set, where the control set is computed via methods like Control Barrier [8] or Control Lyapunov Functions [9], Hamilton-Jacobi (HJ) reachability analysis [10], or model predictive filters [11]. However, this action-space filtering does *not* enable the robot to naturally suggest alternatives in a human-verifable way. Our key idea is that an

*alternative suggestion can be modeled as safe control in **goal space**, rather than action space.*

We leverage HJ reachability analysis to synthesize a goal-conditioned reach-avoid value function that is used within our safe control framework. This computation is performed once offline and automatically quantifies how capable the robot's pre-trained policy is at accomplishing the task while staying safe, for a suite of possible goal inputs. Due to the dimensionality of the problem induced by goal parameterization, we leverage principled but approximate reachability solvers rooted in reinforcement learning [12], and empirically quantify the error of our learned value function. Online, the reach-avoid value function takes in the current state and a specified goal and determines whether the goal meets the reach-avoid criteria (safe and live). If not, we perform safety filtering over the goals to suggest alternatives.

[1]Department of Mechanical and Aerospace Engineering, UC San Diego. Email: hjjeong@ucsd.edu [2]Robotics Institute, Carnegie Mellon University. Email: abajcsy@cmu.edu

Manuscript 3120 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 3, 2025.

We call our overall framework for robots that suggest **S**afe **ALT**ernatives: **SALT**. We demonstrate SALT in simulation experiments grounded in tabletop manipulation with a Franka Panda arm. We find that compared to baselines that only consider open-loop uncertainty, SALT's understanding of closed-loop consequences of the robot's policy detect failures 25% more accurately, and the alternative goals that SALT proposes consistently align with those that people find acceptable.

II. Related Work

Safety Filtering. Safety filters—which detect unsafe actions and minimally modify them—are increasingly popular ways to ensure closed-loop safety [13]. The most popular methods are control barrier functions (CBFs) [8], Hamilton-Jacobi (HJ) reachability [14], [15], and model predictive shielding [16]. In this work we build off of HJ reachability due to its ability to handle non-convex target and constraint sets, control constraints and uncertainty in the system dynamics, and its association with a suite of numerical tools including recent neural approximations that scaled safe set synthesis to 15-200 dimensions [17], [18]. Our key idea is that by treating the human's goal as a virtual state, we can do safety value function synthesis and safety filtering on the goal (instead of on the actions). This enables the robot to minimally modify the human's desired goal and propose safe alternatives.

Uncertainty Quantification of Learned Robot Policies. For modular robot policies that utilize an "upstream" goal or intent estimator, prior works have quantified goal uncertainty [19], calibrated task plans inferred from language commands [20] and quantified their execution risk [21]. For end-to-end behavior cloned policies, prior works have quantified their generalizability via statistical bounds [22], and predicted policy success rate via value estimation [23]. In contrast, our work uses control-theoretic verification tools to analyze the closed-loop success of a robot's policy.

Robot Communication of Uncertainty & Capability. Prior works in human-robot interaction have enabled robots to communicate their task uncertainty via dialogue [24], their objectives via motion or haptics [25], [26], express physical capabilities [27], or explain their failures [28] to people. Instead of having robots only explain what they are uncertain about (or ask for help), we enable robots to actively suggest alternatives they can safely accomplish.

III. Method: Suggesting Safe Alternatives

We want robots to know when they can safely execute a user's given goal and propose safe alternatives when they cannot. Our key idea is to formalize alternative suggestion as a safe control problem in goal space. This goal-space filtering enables the robot to automatically suggest alternatives in a human-verifable way. By solving a goal-parameterized reachability problem *offline*, we obtain a reach-avoid value function that the robot uses *online* for filtering. We call our overall framework for suggesting **S**afe **ALT**ernatives, **SALT** (summarized in Fig. 2).

Setup. Let the robot's physical state be $s \in \mathcal{S}$ (e.g., positions, velocities, joint configuration) and the robot's action be $a \in \mathcal{A}$. We model the robot's state as evolving via the deterministic discrete-time dynamics $s_{t+1} = f(s_t, a_t)$. The human influences the robot's behavior by specifying a goal $g \in \mathcal{G}$ (e.g., an object a manipulator should pick up). We assume that the robot interprets this goal and executes its behavior based on a pre-trained goal-conditioned policy, $\pi(s; g)$. For example, this could be a behavior cloned policy[1] trained on a demonstration dataset $(s, a, g) \sim \mathcal{D}$ consisting of state-action-goal tuples.

***Offline*: Reach-Avoid Analysis of Goal-Conditioned Robot Policies.** We use reachability analysis to automatically verify for which initial states and desired goals can the robot's policy satisfy both safety and liveness constraints. Our core idea is to treat the space of possible goals that the human could ask about at deployment time ($g \in \mathcal{G}$) as a virtual state that has zero dynamics (i.e., $\dot{g} = 0$) during the reachability analysis. This approach, inspired by parameter-conditioned reachability [29], *enables us to quantify the safety and liveness sensitivity of the robot's policy as a function of all possible goal inputs.*

For offline analysis, we leverage Hamilton-Jacobi (HJ) reachability. This verification technique is compatible with nonlinear dynamical systems, arbitrary non-convex failure and target set representations, and has a suite of associated numerical tools, from exact grid-based solvers [30] to approximate but scalable techniques such as reinforcement learning [12], [17] and self-supervised learning [18].

We encode our safety specification via a failure set $\mathcal{F} \subset \mathcal{S}$ (e.g., the object slipped from the gripper) and liveness via a target set $\mathcal{T} \subset \mathcal{S}$ (e.g., the object height must be above the table). For computation, the target and failure sets are encoded via Lipschitz continuous margin functions $l(\cdot)$ and $h(\cdot)$ respectively: $\mathcal{T} := \{s \mid l(s) \leq 0\}$ and $\mathcal{F} := \{s \mid h(s) > 0\}$. One such function is the signed distance to the set boundary. Intuitively, these margin functions will measure the "closest" the robot's policy $\pi(s; g)$ ever got to violating safety and accomplishing the task given the specified goal g. We introduce goal-conditioned target and failure sets, $\mathcal{T}_g$ and $\mathcal{F}_g$, as well as corresponding goal-conditioned margin function, $l_g(s)$ and $h_g(s)$. With these in hand, we can define the *goal-* and *policy-conditioned* safety value function as:

$$V^{\pi}_{*}(s; g) = \min_{\tau \in \{0,1,\ldots\}} \max \Big\{ l_g(\xi_s^{\pi(\cdot;g)}(\tau)), \max_{\kappa \in \{0,\ldots,\tau\}} h_g(\xi_s^{\pi(\cdot;g)}(\kappa)) \Big\}, \quad (1)$$

where the robot's trajectory starting from state s and applying policy $\pi(\cdot; g)$ is denoted by $\xi_s^{\pi(\cdot;g)}$. Intuitively, the outer maximum acts as a mechanism to remember if the robot has ever entered the failure set $\mathcal{F}$ up to this time (right-hand side) and has satisfied the liveness property (left-hand side). If $\xi_s^{\pi(\cdot;g)}$ enters $\mathcal{F}$ at any time, then the inner maximum will be positive, and thus the overall value will also be positive.

[1]We assume that the policy is deterministic in this work.

Manuscript 3120 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 3, 2025.

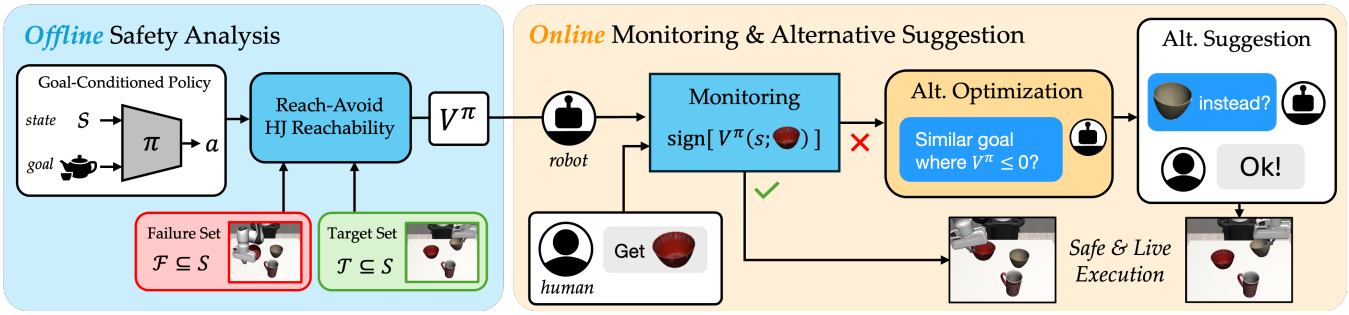

Figure 2: **Robots that Suggest Safe Alternatives (SALT) Framework.** (Left) Offline, a reach-avoid value network is learned to estimate the safety and liveness properties of a pre-trained goal-conditioned robot policy. (Right) Online, a human inputs a desired goal, which is first monitored by our reach-avoid value function. If the input goal satisfies both safety and liveness, then the policy is executed. Otherwise, the robot solves a safe control problem over alternative goals (e.g., objects in the scene) to propose an alternative. If the human accepts, then the robot confidently executes on the new goal.

In contrast, if the robot's trajectory never enters $\mathcal{F}$, then the overall value will be negative if and only if the robot reaches the target $\mathcal{T}$ (which is encoded via the subzero level set of l). The outer minimum ensures that the value function "remembers" these events over the entire time horizon. In other words, the value can only be negative if the target $\mathcal{T}$ is reached without ever violating the safety constraint $\mathcal{F}$ along the way.

Following prior work [12], it can be shown that this value function must satisfy the fixed-point *reach-avoid Bellman equation*:

$$V_*^\pi(s;g) = \max\Big\{h_g(s),\ \min\big\{l_g(s), V_*^\pi(s_+^\pi;g)\big\}\Big\}. \quad (2)$$

where $s_+^\pi := f(s,\pi(s;g))$ is the next state the robot reaches after applying the control from the goal-conditioned policy $\pi(\cdot;g)$ and the $*$ indicates optimality of the reach-avoid value function, V_*^π, computed with a perfect solver.

To scale reach-avoid analysis to higher dimensional state and goal spaces, we leverage the principled time-discounted formulation of Eq. 2 introduced in [12], rendering the reach-avoid problem compatible with reinforcement learning approximations (e.g., Q-learning, REINFORCE [17], [31]):

$$\begin{aligned} V^\pi(s;g) = \gamma \max\Big\{h_g(s), \min\big\{l_g(s), V^\pi(s_+^\pi;g)\big\}\Big\} \\ +(1-\gamma)\max\big\{l_g(s), h_g(s)\big\}. \end{aligned} \quad (3)$$

Here, $\gamma \in [0,1)$ represents the time discount factor, where $V^\pi \to V_*^\pi$ as $\gamma \to 1$. Note that V^π is an over approximation of V_*^π, therefore it will always be more conservative than V_*^π [12]. In the **SALT** framework, the reach-avoid value function $V^\pi(s;g)$ is trained offline *before* deploying the robot policy, but will be queried *online* given any new human goal g to monitor the robot policy performance and to automatically propose an alternative to the human.

***Online*: Alternative Suggestion as Safe Control in Goal Space.** Once V^π is trained offline, we instantiate the problem of suggesting alternatives *online* as a safe control problem. Our key idea is to treat the goal as a virtual action that the robot can minimally modify to ensure the policy will be accomplished safely. Specifically, we formalize a "smooth blending" safety filter inspired by control barrier functions [8], but instead of filtering *actions* as is done typically, we filter *goals*. Let the human's original goal input be $g_\mathcal{H}$. The robot seeks an alternative goal $g_\mathcal{R}$ that satisfies:

$$\begin{aligned} g_\mathcal{R} = \arg\min_{g\in\mathcal{G}}\ & d(\mathcal{E}(g), \mathcal{E}(g_\mathcal{H});\theta) \\ \text{s.t. } & V^\pi(s;g) \leq 0, \end{aligned} \quad (4)$$

where $d(\cdot)$ is a similarity measure, and $\mathcal{E}(\cdot)$ is an encoder that maps goal representations to the goal space. The similarity measure is parameterized by human intent θ, since the notion of similarity can differ based on what the user intends to do with a goal object or after a robot reaches a desired state. If the human's original goal $g_\mathcal{H}$ and robot's initial state s does *not* satisfy safety and liveness (i.e., $V^\pi(s;g_\mathcal{H}) > 0$), then the optimization above will be solved to find an alternative goal $g_\mathcal{R}$ that's similar to $g_\mathcal{H}$. We stress that the representation of the goals g, and the corresponding similarity measure $d(\cdot)$ are a key design decision, and one which we study in Sec. V-C. In our experiments, we first start with pose-based representations of g and an Euclidean distance function as a similarity measure, and then explore semantic similarity and goal representations (conditioned on intent).

IV. Experimental Setup

Environment: Manipulation (20D). We study our SALT framework in a high-dimensional manipulation task. A tabletop manipulator has to lift a person's desired object: a red mug, a brown bowl, or a red bowl (Fig 1). These comprise our discrete goal parameters: $g \in \mathcal{G} := \{\text{RedMug}, \text{BrownBowl}, \text{RedBowl}\}$. We use the robosuite simulation environment [32] and the Franka Panda manipulator. For both the base policy and for reachability analysis, we model the 20-dimensional robot state $s \in \mathbb{R}^{20}$ consisting of robot end-effector (EE) pose ($p_{\text{EE}} \in \mathbb{R}^7$, xyz position and quaternion), gripper state (left and right gripper opened or closed $\delta_\text{L}, \delta_\text{R} \in \{0,1\}$), the pose of the person's desired object ($p_g \in \mathbb{R}^7$), the relative position of the EE to the object ($p_{\text{EE},g}^{rel} \in \mathbb{R}^3$). We augment the state space with the 1-dimensional desired object id ($g \in \mathcal{G}$). The robot's action

space $a \in \mathbb{R}^7$ controls the EE linear (x,y,z) and angular (roll, pitch, yaw) velocity and gripper open and close. All control inputs are bounded to a magnitude of $[-1, 1]$. The target margin function is $l_g(s) = ||p_g^z - 0.8||_2^2 - \epsilon$, encoding the z-distance of the object from the target height above the table (0.8 m), and the failure margin function is $h_g(s) = \min\{\min(\delta_{\text{L}}, \delta_{\text{R}}) < 0.001, ||p_{\text{EE},g}^{rel}|| - 0.1\}$, which measures if the robot is gripping without the object in hand.

Base Robot Policy (π). The robot's base policy is trained via behavior cloning (BC). We collected demonstrations in the robomimic [33] environment via expert human teleoperation and obtained 100 successful demonstrations per object. The BC policy is a 2-layer MLP with 1024 neurons per layer and ReLU nonlinearity. It is trained with the AdamW [34] optimizer for 100 epochs. Every 5 epochs, we rollout the policy 50 times and use the policy with highest success rate.

Reach-Avoid Value Learning (V^π). We use the off-the-shelf reach-avoid reinforcement learning (RARL) solver from [12] to approximate the value function in Eq. (3). We train on a single NVIDIA 4090ti GPU for 400k epochs total, checkpointing at every 50k epochs. We anneal the discount factor γ from 0.9 to 0.9999 throughout training, use a 3-layer MLP with 512 neurons per layer and tanh nonlinearity, and the AdamW optimizer. We warm-up the value network for 50k iterations by sampling random points in the state space to properly learn $l_g(s)$ and $h_g(s)$. 500 max episode steps are set for the task. Note that the each episode step is 0.1 seconds, so the time horizons is 50 seconds. We use the MuJoCo [35] simulation environment.

V. Experimental Results

To understand each component of **SALT**, we study four questions in simulation experiments: (1) how accurate is the reach-avoid value function approximation?, (2) how does **SALT** compare to alternative runtime monitoring schemes (e.g., ensembles)?, (3) how should **SALT** measure "similar" alternative goals?, and (4) how aligned are the alternatives that **SALT** proposes with human-acceptable ones?

A. How Accurate is SALT's Reach-Avoid Value Function?

Metrics. To evaluate the reliability of the learned value network, we measure the true success rate (**TSR**: network predicts the policy can safely accomplish the task and in reality it can), true failure rate (**TFR**: network predicts the policy *cannot* safely accomplish the task and in reality it cannot), false success rate (**FSR**: network predicts the policy *can* safely accomplish the task but in reality it cannot), and false failure rate (**FFR**: network predicts the policy *cannot* safely accomplish the task but in reality in can). F_1-Score represents the predictive performance of a binary classifier (1.0 indicates perfect precision).

Evaluation Approach. Since exhaustive gridding of the state space is not feasible for our high-dimensional manipulation example, we randomly sample initial physical states s and goals g, perform a rollout in our simulator, and check if the value of the network accurately reflects policy execution outcome (safe success, or not). We do this on 1,000 initial (s, g) pairs sampled near the zero level set of the approximate value network ($V^\pi(s, g) \approx 0$), since an accurate boundary matters for monitoring, and across 10 random seeds.

Environment	TSR (%)	TFR (%)	FSR (%)	FFR (%)	F_1-Score
Manipulation	59.98 (±2.10)	14.55 (±1.43)	12.64 (±1.28)	12.83 (±1.13)	0.82

TABLE I: **Quality of the SALT's value function approximation.** Results from 1,000 initial conditions sampled near the zero level boundary of the value network across 10 random seeds.

Results. Table I shows accuracy metrics for manipulation. We find that our value function has a 0.82 F_1-score, indicating that we have learned a non-trivial discrimination of safe and unsafe initial states and goals. We hypothesize that errors in the approximated value function come from the high dimensionality of the system and task complexity of the grasping and pickup task. Future work should investigate post-hoc adjustment techniques to further minimize the FFR and FSR (e.g., [36]).

B. What is the Benefit of SALT as a Runtime Monitor?

Method	TNR % (↑)	TPR % (↑)	FPR % (↓)	FNR % (↓)	F_1-Score (↑)
Ensemble	34.61 (±1.68)	27.45 (±1.96)	32.65 (±1.27)	5.26 (±0.89)	0.65
RewardSum	64.47 (±1.61)	4.67 (±0.68)	8.85 (±1.01)	22.01 (±1.21)	0.81
SALT (ours)	61.21 (±1.91)	13.23 (±1.56)	13.23 (±1.23)	12.33 (±0.85)	**0.83**

TABLE II: **Evaluating Runtime Monitors.** Confusion matrix for 1,000 random initial states across 10 random seeds.

Baselines. We compare our reachability-based monitor to two baselines: **Ensemble** and **RewardSum** safety monitors. Following [37], we use an **Ensemble** of behavior cloned policies as an *open loop* monitor: high ensemble disagreement measures uncertainty in the robot's action prediction. If the disagreement exceeds a threshold, then the robot stops and asks for help. We use $M = 5$ policies as ensemble members, and take the variance σ^2 of the action prediction as the uncertainty measure; the robot stops when $\sigma^2 > \epsilon$. We use $\epsilon = 0.0175$, which are heuristically tuned for lowest FSR and FFR. Similar to our approach, **RewardSum** is a *closed loop* safety monitor whose value function captures long-term outcomes of executing the base policy. However, the two methods differ in their optimization objective: **RewardSum** computes the value via the typical *expected sum* of discounted rewards used in reinforcement learning, while **SALT** uses the reach-avoid objective which remembers the *closest* the robot ever gets to safety and liveness violations (as in Eq. (3)).

Metrics. We measure the accuracy of each monitor stopping to alert the human. We once again randomly sample (s, g) pairs, roll our the policy to obtain the ground-truth success or failure label, and then compare each monitor to this label. We define a 2x2 confusion matrix of monitor predictions (flag raised or not) and actual robot outcomes (success or fail). True negative rate (TNR) is when the monitor does not

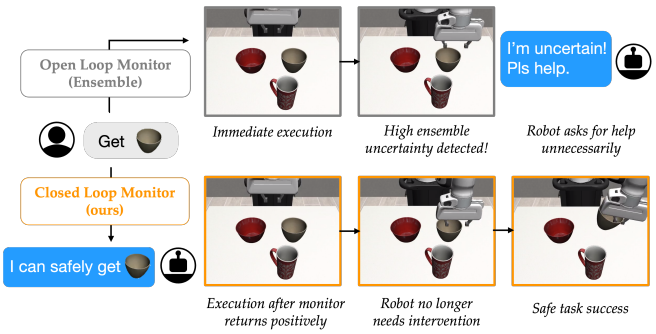

Figure 3: **Open Loop vs. Closed Loop Monitoring.** Ensemble detects uncertainty before grasping and asks for help unnecessarily. SALT's closed loop monitoring first checks the safety and liveness via the V^π, then executes confidently.

raise a flag and the robot executes successfully, true positive rate (TPR) is when the monitor raises a flag and the robot would have actually failed, false positive rate (FPR) is when the monitor raises a flag unnecessarily, and false negative rate (FNR) is when when the monitor did not raise a flag when it should have (robot failed). We also measure the F_1 score as in Section V-A.

Results. Results are in Table II: Ensemble had the highest TPR (27.45%) but also the highest FPR (32.65%), and the lowest F_1-score of 0.65. In total, Ensemble triggered human help around 60% of the time, when it should have been triggered less than half of the time. Figure 3 shows an example of this pessimism: Ensemble asks for help mid-execution even though the robot was capable of doing the task safely. Since actions are being evaluated at every timestep, Ensemble can only know that it is uncertain, rather than describe a high-level alternative that would make it confident. In contrast, ours checks pre-execution if the closed-loop behavior is predicted to be safe and live, and requires no human supervision during execution. Finally, comparing closed loop monitors, SALT is consistently a more reliable safety monitor compared to RewardSum, having a high TPR+TNR of 74.44% (vs. 69.14%), a lower FNR of 12.33% (vs. 22.01%), and higher F_1-score of 0.83 (vs. F_1-score of 0.81).

C. How Should SALT Reason About "Similar" Alternatives?

We hypothesize that suggesting similar alternatives may require a *semantic* representation of the goals—capturing visual or functional properties of the object—and a corresponding similarity measure. In this section, we ablate the encoder models $\mathcal{E}(\cdot)$ and similarity measure $d(\cdot)$ used within our SALT framework. We also study how similar alternatives change as a function of human intent θ in Equation 5.

Methods: Encoders, Intents, and Similarity Measures. We investigate three methods that leverage different goal representations and similarity measures: CosineSim, LLM, and SIRL. In our experiments, CosineSim uses a textual description of the goal (e.g., as returned by a semantic object detector) and measures similarity via the cosine similarity between the textual embedding of the human's original goal ($g_\mathcal{H}$) and any alternative goal (g). We use the pre-trained BERT [38] sentence-transformer model to obtain the textual embedding. Mathematically, given a language description $\mathcal{L}_g$ of a goal g and a textual description of the intent $\mathcal{L}_\theta$, our encoder produces an embedding vector $\mathcal{E}_{\text{BERT}}(\mathcal{L}_g; \mathcal{L}_\theta) = \vec{w}_g$ and our similarity measure is $d := \vec{w}_g \cdot \vec{w}_{g_\mathcal{H}} / ||\vec{w}_g|| \; ||\vec{w}_{g_\mathcal{H}}||$. LLM Fuzzy Matching [39] also uses a textual representation of the goal and intent, but uses a pre-trained large language model (LLM) to directly reason about the semantic similarity (without looking at the embedding similarity). In our experiments, we use GPT-4o [40] as our language model. Mathematically, $d := \text{LLM}(\mathcal{L}_g, \mathcal{L}_{g_\mathcal{H}}; \mathcal{L}_\theta, \mathcal{P})$, where the prompt to the LLM is $\mathcal{P}$ = "The user intends to $\mathcal{L}_\theta$. Given $\mathcal{L}_{g_\mathcal{H}}$, which item is the closest related to it?".

Unlike LLM and CosineSim (which use pre-trained language models), SIRL requires training a *personalized* representation which explicitly learns an end-user's notion of similarity from their preference data, enabling us to study how a personalized model of similarity influences our SALT framework compared to pre-trained models that are not fine-tuned on individual data. This model uses a privileged, hand-engineered feature space $\Phi_g \in \mathbb{R}^m$ as input based on any given goal object g; for example, in our experiments, given a g is a red cup, Φ_g would be a 20-dimensional vector consisting of the object's RGB color values and functional and material properties. We train an intent-parameterized encoder $\mathcal{E}_{\text{SIRL}}(\Phi(g); \theta) = \phi_g^\theta$ via contrastive learning, which returns an embedding vector $\phi_g^\theta \in \mathbb{R}^n$, $n < m$ that represents the most relevant features of a goal given the user's intent. Finally, SIRL measures similarity via L2 distance in embedding space: $d := ||\phi_g^\theta - \phi_{g_\mathcal{H}}^\theta||_2$.

Implementation Details. SIRL learns relevant similarity features by asking the end user to select the two most similar goals given a triplet of goals. Throughout this section, we use simulated human data for training SIRL and evaluation, enabling us to have access to a ground-truth representation of the human's notion of similarity given the intent. SIRL is trained on triplets $\mathcal{D}_\theta = \{(\Phi^i_{g_1}, \Phi^i_{g_2}, \Phi^i_{g_3})\}_{i=1}^K$, which are feature spaces corresponding to three distinct goals in the environment given a human intent θ. The simulated human ranks the two most similar ones using their ground-truth intent-relevant features. We train the encoder $\mathcal{E}_{\text{SIRL}}(\Phi(g); \theta) = \phi_g^\theta$ to minimize the loss from [41]:

$$\mathcal{L}(\phi_g^\theta) = \sum_{i=1}^{|\mathcal{D}_\theta|} \mathcal{L}_{trip}(\Phi^i_g, \Phi^i_{g_+}, \Phi^i_{g_-}) + \mathcal{L}_{trip}(\Phi^i_{g_+}, \Phi^i_g, \Phi^i_{g_-})$$

where Φ^i_g and $\Phi^i_{g_+}$ are ranked as most similar and $\Phi^i_{g_-}$ is most dissimilar. $\mathcal{L}_{trip}(\Phi^i_g, \Phi^i_{g_+}, \Phi^i_{g_-})$ is the triplet loss [42] that uses Φ^i_g as the anchor, $\Phi^i_{g_+}$ as the similar example and $\Phi^i_{g_-}$ as the dissimilar example. This loss function pushes together embeddings for similar objects with respect to the intent-relevant features while pushing apart embeddings for dissimilar objects.

Evaluation Setup. We use 10 kitchen objects (e.g., cups,

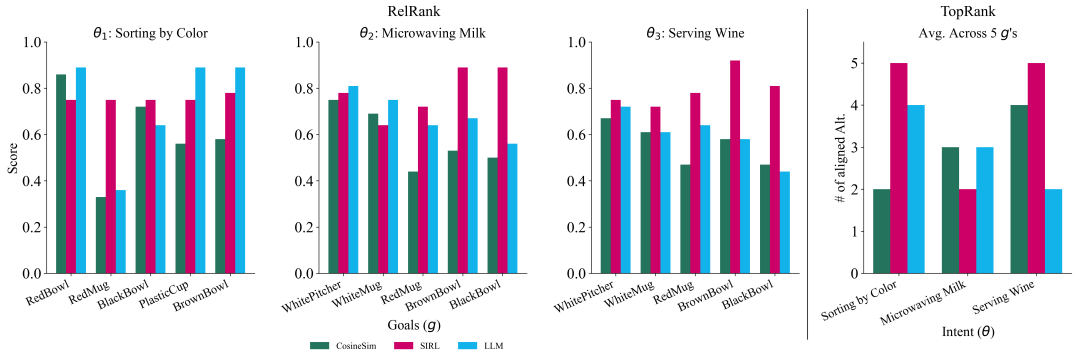

Figure 4: **Accuracy of Different Similarity Measures.** (Left Three Plots) `RelRank` scores for 5 aligned $g_{\mathcal{H}}$ are scored across baselines and intents. (Right Plot) `TopRank` scores for 5 aligned $g_{\mathcal{H}}$ are scored across baselines and intents.

bowls, mugs, pitchers, teapots, ramekins) from the Google Research dataset [43] and study three increasingly complicated user intents: $\theta_1 =$ *Sorting Kitchen by the Same Color*, $\theta_2 =$ *Microwaving Soup*, and $\theta_3 =$ *Serving Wine at Formal Dinner*. For each intent, we select 5 initial goal objects from the dataset that make sense for the intent. Given one of these five objects, we query each method for its similarity score compared to all other 9 objects. We determine the ground-truth similarity score using a simulated human with privileged knowledge about their intent-relevant features.

Metrics. We want to measure how well the distance functions in each method captures the relative distance between any pairs of objects. We use two metrics: `TopRank` returns one if the most similar item outputted by a similarity measure is the most similar item for the simulated human (and zero otherwise); `RelRank` returns a real-valued score (between 0 to 1) which measures how correctly a method ranks all other goals relative to a given goal[2] . Mathematically, let $g_{\mathcal{H}}$ be a given goal and let $\mathcal{G}^*_{g_{\mathcal{H}}} = \{g_1, g_2, ...g_n\}$ be a list of all other n goals ranked by how similar they are to $g_{\mathcal{H}}$ by the simulated human. We define $\mathcal{G}^{\omega}_{g_{\mathcal{H}}}$ to be a list of all the n goals ranked by their similarity to $g_{\mathcal{H}}$ by any method, $\omega \in \{$**SIRL**, **CosineSim**, **LLM**$\}$. Let $r_{\mathcal{G}^*_{g_{\mathcal{H}}}}(g_i)$ be the rank position of goal g_i in the ordered list $\mathcal{G}^*_{g_{\mathcal{H}}}$ and let $R_{\mathcal{G}^*_{g_{\mathcal{H}}}}(g_i, g_j) = \text{sign}[r_{\mathcal{G}^*_{g_{\mathcal{H}}}}(g_j) - r_{\mathcal{G}^*_{g_{\mathcal{H}}}}(g_i)]$. Here, $R_{\mathcal{G}^*_{g_{\mathcal{H}}}}(g_i, g_j)$ returns 1 if g_i is more similar to $g_{\mathcal{H}}$ than g_j and -1 for the converse (our lists assume no ties). Finally, we define an indicator function $1[R_{\mathcal{G}^*_{g_{\mathcal{H}}}}(g_i, g_j) = R_{\mathcal{G}^{\omega}_{g_{\mathcal{H}}}}(g_i, g_j)]$ that returns 1 if the two lists have the same relative rankings of the goals and 0 otherwise. The `RelRank` metric is then:

$$\texttt{RelRank}(\mathcal{G}^*_{g_{\mathcal{H}}}, \mathcal{G}^{\omega}_{g_{\mathcal{H}}}) = \frac{1}{\binom{n}{2}} \sum_{1 \le i < j \le n} 1[R_{\mathcal{G}^*_{g_{\mathcal{H}}}}(g_i, g_j) = R_{\mathcal{G}^{\omega}_{g_{\mathcal{H}}}}(g_i, g_j)]. \tag{5}$$

[2]Our metric is inspired by from Kendall's rank correlation coefficient [44]. However, while Kendall's penalizes (with -1) for every incorrectly ordered pair, our metric does not (assigns 0). This modification is appropriate for our context, where the top rankings are of more practical importance (since we want to return *maximally* similar alternatives). Thus, we should not punish the method for having noisy bottom rankings since the differences may not represent meaningful distinctions.

Results: Most Similar Goal Accuracy. In the right of Figure 4 we show the `TopRank` results averaged across all five goals per each intent. **SIRL** most consistently ranks the human's preferred goal as most similar (in total, 12) compared to **CosineSim** and **LLM** (9 for both). Furthermore, the explicit training of **SIRL** makes it more robust to intents while **LLM** and **CosineSim** (parameterized by the intent) sometimes ignore critical intent features. (e.g., for $\theta_1 =$*Sorting Kitchen by Color* and $g_{\mathcal{H}} =$ RedMug, both methods return White Mug, neglecting color). We hypothesize that this occurs because the models are only approximately optimal in their rankings and selecting the top-1 choice requires precision; for example, we observed that the ground-truth top choice typically appeared within the top-3 results of each method.

Results: Overall Similarity Measure Accuracy. We report the `RelRank` metric (which quantifies the overall performance of a similarity measure) in the three plots left of Figure 4 for each intent $\theta_1, \theta_2, \theta_3$. Across all intents and goals, **SIRL**'s similarity measure is more consistently accurate compared to **CosineSim** and **LLM**, and is the best performing measure for intent θ_3. This is because **SIRL** is optimized to solve a personalized metric learning problem—identifying an embedding space that understands similarity according to the user's internal state—while the other two approaches that use pre-trained models have an implicit semantic understanding of similarity. As the intents become increasingly complicated (e.g., $\theta_3 =$*Serving Wine*, **CosineSim** and **LLM**'s pre-trained similarity struggle to capture how the human evaluates these alternatives, while **SIRL** maintains performance due to its privileged access to human's internal states. Between the two pre-trained approaches, **LLM** typically out-performs **CosineSim** in terms of accuracy.

Our main takeaway is that personalized preference data enables more accurate goal representations and better similarity measures, particularly for complex intents. However, off-the-shelf language models and LLM Fuzzy Matching can still be valuable semantic similarity measures that require no additional training data and provide a more naturalistic and intuitive interface for people to specify their goals (e.g., via language rather than explicit featurizations of the world).

D. How Acceptable are SALT's Proposed Alternatives?

Finally, we study SALT's overall performance at detecting when the robot can safely accomplish a desired goal and suggesting a similar safe alternatives when it cannot.

Alternatives Dataset. To measure how acceptable are SALT's alternatives, we obtained a validation dataset of initial goal ($g_{\mathcal{H}}$) and acceptable alternative pairs annotated by 20 expert users from labs at Carnegie Mellon and UC San Diego. In the manipulation setting, people were shown a tabletop with three objects (as in Fig. 1) with two distinctive intents: θ_1 = *Drinking Soup* and θ_2 = *Sorting Kitchen by the Same Color*. These users were asked which alternative objects would be acceptable given an initial object.

Approach. For each initial goal $g_{\mathcal{H}}$ in the validation dataset, we queried SALT to obtain our algorithm's suggested alternatives. We perform this across a suite of initial robot states s. We queried 1,000 random initial conditions for each goal and saved our algorithm's suggestion, $g_{\mathcal{R}}$. Note that SALT returns the initial goal (i.e., $g_{\mathcal{R}} = g_{\mathcal{H}}$) if it is safe and live.

Metrics. We measure alternative alignment (%): given an initial goal $g_{\mathcal{H}}$, if the alternative goal proposed by SALT matches or if the input goal is safe and live, then alignment is a success. We compute mean and standard deviation across all users and initial s, and report results per initial goal. We compare alignment across several distance measures: **Euclidean**, **CosineSim**, **SIRL**, and **LLM**.

Results: Quantitative. Alternative alignment results are shown in Figure 5. On aggregate, we find that SALT can detect if the original goal was safe and suggest safe alternatives that strongly align with human preferences (alignment scores between 70%-100% when using the **SIRL** or **LLM** objective functions. We note that when θ_1 =*Drinking Soup*and $g_{\mathcal{H}}$ = RedMug, 35% of the users wanted neither alternative, dropping the maximum possible alignment rate.

Furthermore, we break down our results into only those scenarios where the human's initial goal ($g_{\mathcal{H}}$) was *not* safe, and thus the robot *had* to suggest an alternative. The alignment scores for these scenarios are shown in Figure 5 in a cross-hatched pattern (called AltSuggest). Note that for $g_{\mathcal{H}}$ = BrownBowl, the robot is always capable of grabbing this initial object safely, and thus it is not part of the AltSuggest breakdown. First, we see that **Euclidean** has low alignment for most of the AltSuggest scenarios, highlighting the need for semantic similarity measures. For intent θ_1 =*Drinking Soup* and θ_2 =*Sorting by Color*, **CosineSim**'s alignment rate drops significantly (to 2.13% and 10% respectively) when $g_{\mathcal{H}}$ = RedBowl, indicating that this similarity measure is not capable of making the closed-loop system suggest safe *and* aligned alternatives. Consistent with Section V-C, we see that when SALT uses **SIRL** and **LLM**, the overall system is able to consistently return safe and similar alternatives that align with an end-user's notion of similarity.

Results: Qualitative. Figure 1 shows our algorithm in manipulation where a user first asks for the red bowl to be picked up with the intent θ_1 = *Drinking Soup*. Realizing the it would likely mis-grasp the red bowl and fail, SALT proposes to pick up the brown bowl with **SIRL** (right, Figure 1), and safely completes the task.

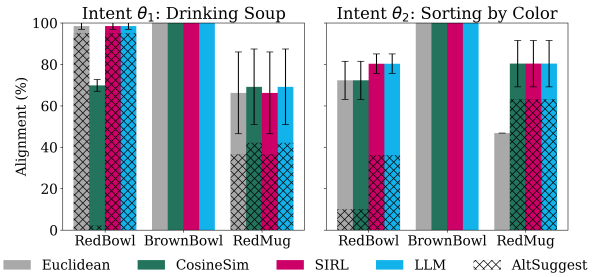

Figure 5: **User Alignment Success Across 1,000 Initial Conditions.** **Euclidean** is in grey, **CosineSim** in green, **SIRL** in pink, and **LLM** in blue. Scenarios where the initial goal $g_{\mathcal{H}}$ was *not* safe and thus the robot had to suggest an alternative are denoted as AltSuggest. **SIRL** and **LLM** consistently out-perform **Euclidean** and **CosineSim** when it comes to guiding SALT to generate safe and aligned alternatives.

VI. Conclusion

In this work, we propose SALT: a framework for robots to monitor if their goal-conditioned policies can safely accomplish a given goal, and automatically suggest safe alternatives when they cannot. By actively proposing a safe alternative pre-execution, we can not only minimize possible safety violations, but also human intervention efforts. For safety, we find that open loop monitors that interpret safety as uncertainty cannot differentiate clearly between a safe and unsafe state, while our closed-loop monitor reliably predicts success and failure. One limitation of our work is that we rely on privileged state information about the environment as well as goal representations to quantify safety and suggest alternatives. Another limitation is the use of a simulated human during our similarity measure evaluation, since it may not accurately model real user preferences. We are excited about further verifying this with a rigorous user study. In future work, we seek to investigate image-based goal-conditioned policies, as well as image representations of goals and similarity queries. One exciting future work is reasoning about task-level alternatives, rather than high-level alternatives like goals.

Acknowledgment

This work is supported by NSF Award #2246447 and the Robotics Institute Summer Scholars (RISS) program at Carnegie Mellon University. The authors would like to thank Ken Nakamura on his insights on reach-avoid RL, Ravi Pandya on his help with robosuite and robomimic, and Lasse Peters for his discussions at various stages in the project.

References

[1] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. Devin, B. Eysenbach, and S. Levine, "Learning to reach goals via iterated supervised learning," *arXiv preprint arXiv:1912.06088*, 2019.

[2] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, "Goal-conditioned imitation learning," *Advances in neural information processing systems*, vol. 32, 2019.

[3] A. Peng, A. Bobu, B. Z. Li, T. R. Sumers, I. Sucholutsky, N. Kumar, T. L. Griffiths, and J. A. Shah, "Preference-conditioned language-guided abstraction," in *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, 2024, pp. 572–581.

[4] M. Liu, M. Zhu, and W. Zhang, "Goal-conditioned reinforcement learning: Problems and solutions," 2022. [Online]. Available: https://arxiv.org/abs/2201.08299

[5] O. Lockwood and M. Si, "A review of uncertainty for deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 18, no. 1, 2022, pp. 155–162.

[6] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *International Journal of Computer Vision*, pp. 1–28, 2024.

[7] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," *Annual Review of Control, Robotics, and Autonomous Systems*, 2023.

[8] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[9] J. A. Primbs, V. Nevistić, and J. C. Doyle, "Nonlinear optimal control: A control lyapunov function and receding horizon perspective," *Asian Journal of Control*, vol. 1, no. 1, pp. 14–24, 1999.

[10] K. Margellos and J. Lygeros, "Hamilton-Jacobi formulation for reach-avoid differential games," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1849–1861, 2011.

[11] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," 2021. [Online]. Available: https://arxiv.org/abs/1812.05506

[12] K.-C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac, "Safety and liveness guarantees through reach-avoid reinforcement learning," *Robotics: Science and Systems*, 2021.

[13] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," *Annual Reviewsof Control, Robotics, and Autonomous Systems*, 2024.

[14] A. Li, L. Sun, W. Zhan, M. Tomizuka, and M. Chen, "Prediction-based reachability for collision avoidance in autonomous driving," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7908–7914.

[15] K. Driggs-Campbell, R. Dong, and R. Bajcsy, "Robust, informative human-in-the-loop predictions via empirical reachable sets," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 300–309, 2018.

[16] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[17] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging hamilton-jacobi safety analysis and reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8550–8556.

[18] S. Bansal and C. J. Tomlin, "Deepreach: A deep learning approach to high-dimensional reachability," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1817–1824.

[19] J. F. Fisac, A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, S. Wang, C. J. Tomlin, and A. D. Dragan, "Probabilistically safe robot planning with confidence-based human predictions," *Robotics: Science and Systems*, 2018.

[20] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, Z. Xu, D. Sadigh, A. Zeng, and A. Majumdar, "Robots that ask for help: Uncertainty alignment for large language model planners," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[21] J. Lidard, H. Pham, A. Bachman, B. Boateng, and A. Majumdar, "Risk-calibrated human-robot interaction via set-valued intent prediction," *Robotics: Science and Systems*, 2024.

[22] J. A. Vincent, H. Nishimura, M. Itkina, P. Shah, M. Schwager, and T. Kollar, "How generalizable is my behavior cloning policy? a statistical approach to trustworthy performance evaluation," *arXiv preprint arXiv:2405.05439*, 2024.

[23] C. Gokmen, D. Ho, and M. Khansari, "Asking for help: Failure prediction in behavioral cloning through value approximation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5821–5828.

[24] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley *et al.*, "Robots that ask for help: Uncertainty alignment for large language model planners," *Conference on Robot Learning*, 2023.

[25] S. H. Huang, D. Held, P. Abbeel, and A. D. Dragan, "Enabling robots to communicate their objectives," *Autonomous Robots*, vol. 43, pp. 309–326, 2019.

[26] J. F. Mullen, J. Mosier, S. Chakrabarti, A. Chen, T. White, and D. P. Losey, "Communicating inferred goals with passive augmented reality and active haptic feedback," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8522–8529, 2021.

[27] M. Kwon, S. H. Huang, and A. D. Dragan, "Expressing robot incapability," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 87–95.

[28] C. Tagliamonte, D. Maccaline, G. LeMasurier, and H. A. Yanco, "A generalizable architecture for explaining robot failures using behavior trees and large language models," in *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, 2024, pp. 1038–1042.

[29] J. Borquez, K. Nakamura, and S. Bansal, "Parameter-conditioned reachable sets for updating safety assurances online," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 553–10 559.

[30] I. Mitchell, "A toolbox of level set methods," *http://www. cs. ubc. ca/mitchell/ToolboxLS/toolboxLS. pdf, Tech. Rep. TR-2004-09*, 2004.

[31] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[32] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu, "robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[33] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *arXiv preprint arXiv:2108.03298*, 2021.

[34] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

[35] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[36] A. Lin and S. Bansal, "Verification of neural reachable tubes via scenario optimization and conformal prediction," *arXiv preprint arXiv:2312.08604*, 2023.

[37] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[39] J. Duan, W. Pumacay, N. Kumar, Y. R. Wang, S. Tian, W. Yuan, R. Krishna, D. Fox, A. Mandlekar, and Y. Guo, "Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation," *arXiv preprint arXiv:2410.00371*, 2024.

[40] OpenAI, "Gpt-4o system card," 2024. [Online]. Available: https://arxiv.org/abs/2410.21276

[41] A. Bobu, Y. Liu, R. Shah, D. S. Brown, and A. D. Dragan, "Sirl: Similarity-based implicit representation learning," in *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 565–574. [Online]. Available: https://doi.org/10.1145/3568162.3576989

[42] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks." in *Bmvc*, vol. 1, no. 2, 2016, p. 3.

[43] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke, "Google scanned objects: A high-quality dataset of 3d scanned household items," 2022. [Online]. Available: https://arxiv.org/abs/2204.11918

[44] A. Stepanov, "On the kendall correlation coefficient," 2015. [Online]. Available: https://arxiv.org/abs/1507.01427