Lowering barriers to human-robot communication

Submitted in partial fulfillment of the requirements of
for the degree of Doctor of Philosophy (PhD) in Robotics

By

Vidhi Jain

CMU-RI-TR-25-83

The Robotics Institute
CARNEGIE MELLON UNIVERSITY
Pittsburgh, Pennsylvania, 15213
August 2025

Abstract

For robots to be intuitive partners in learning and collaboration, they must interpret multimodal cues - visual gestures, natural language instructions, environmental context - and translate them into effective actions. However, existing robot policies typically rely on structured language goals and static visual observations, limiting both the richness of sensory understanding and the flexibility of human task specification. This thesis addresses three core challenges in developing embodied intent-awareness: (i) training policies to capture explicit human intent, (ii) understanding implicit intents, and (iii) enabling scalable benchmarks for these capabilities.

First, I develop policies that infer our explicit intents, which we broadly express via visual demonstrations and verbally. For visual cues, I extract task semantics from a single video prompt to guide simulated dish-loading tasks [92] and general manipulation task [93], using cross-attention transformers. For verbal instructions, I focus on the hierarchical language structure for goal specification; decomposing verbal commands into sequence of interaction points and relative waypoints for sample-efficiency [155].

Second, I incorporate implicit social and environmental constraints, such as predicting and minimizing robot noise at the listener's location, to augment planning costs and ensure comfortable human–robot coexistence [87].

Lastly, I focus beyond model development and towards scalable evaluation benchmarks [224] that expand the notions of generalization in robot manipulation and open challenges in robotics foundation models [78]. I present ActVQ-Arena [88], an embodied visual querying environment where agents must locate fine-grained information (e.g., "find the expiry date" or "check ingredients") under partial observability, varying poses, and diverse embodiments.

Together, these contributions advance robot autonomy by bridging human expressive modalities and practical control, paving the way for future work on multi-objective pragmatic instruction following, rapid adaptation from multimodal cues, and personalization to individual preferences.

Acknowledgements

First and foremost, I owe my deepest gratitude to my advisor, Yonatan Bisk. His unwavering support, insightful feedback, and boundless patience have shaped every aspect of this work. Yonatan's trust empowered me to pursue ambitious research questions, and our weekly meetings—always engaging, thought-provoking, and energizing—taught me how to formulate clear visions, choose appropriate technical approaches, and plan (and re-plan) realistic timelines. Through his mentorship, I have learned to ground complex ideas in simple examples, maintain sight of the big picture, and cultivate the strategic and creative instincts that drive good research. Beyond technical guidance, his openness and encouragement fostered a safe space for free-form discussion and reflection, for which I am profoundly grateful.

I am also deeply thankful to my committee members, Henny Admoni, Oliver Kroemer, and Dieter Fox, for their thoughtful questions and generous advice. Henny's pioneering work in multimodal human–robot interaction inspired many of my own ideas about gesture, gaze, and nonverbal communication. Oliver and his group welcomed my endless curiosity about versatile manipulation, and Dieter's research on multimodal training and large-scale simulation offered invaluable guidance for envisioning my experiments. Special thanks to Dorsa Sadigh and Jeanette Bohg for their mentorship and for encouraging me to realize the embodied visual queries project in Chapter 6.

I gratefully acknowledge my industry mentors, Akshara Rai and Debidatta Dwibedi. During my AI Residency at Meta FAIR (2021–22), Akshara's expert guidance laid the groundwork for the work in Chapter 2. In the summer of 2023 at Google DeepMind Robotics, Debidatta challenged me to extend those ideas in new directions, shaping the experiments in Chapter 3. Their practical insights and hands-on support were instrumental in translating theory into real-world robot applications.

My research has also benefited enormously from collaborations with Priyam Parasher and Chris Paxton (Meta), whose leadership on language-guided spatial attention underpins Chapter 4, and with Rishi Veerapaneni (CMU RI PhD), whose initial discussions and implementation efforts informed the audio-aware planning work in Chapter 5. I extend heartfelt thanks to Yixin Lin, Eric Undersander, Maria Attarian, Nikhil Joshi, Montserrat Gonzalez Arenas, Jacky Liang, Andy Zeng, Quan Vuong, and Pannag Sanketi for their contributions, encouragement, and the vibrant research communities they foster.

A special thank you to Yafei and Quanting for co-leading the survey of robotics foundation models; Sriram for HomeRobot OVMM; and Hao for immersive teleoperation with Stretch.

This journey was enriched by my broader network of peers, namely Tiffany, Abitha, Swami, Sanket, Sriram, Simran, Ceci, Tejus, Pranav, Homanga, Tarasha, Sarvesh, Minyoung, Harshita, Priya, Suvir, Satvik, Jenn, and all the others whose technical insights, manuscript feedback, video-production help, and camaraderie made this work so much richer. I am grateful to my mentees, Su Li and Soumya Teotia, for their enthusiasm and dedication to our shared research goals.

Finally, this work would not have been possible without my family's unwavering love and support. Though spread across many time zones, my parents and my partner have always been there for me. Thank you to my dadi (grandmother) for listening to my research ideas just as I once listened to her stories. Thank you to my sister, who excels where I don't and has held our family together. Thanks to my in-laws for always encouraging me over these years. And to my bhabhi and bhaiya, I'm deeply grateful for being my home away from home. I know it is likely I forgot someone, please forgive me and know how much I appreciate you all.

With a heartfelt thanks for all the love, guidance, and fun!

Vidhi

Contents

Abstract **i**

Acknowledgements **ii**

Contents **iv**

List of Figures **viii**

List of Tables **xvii**

1 Introduction **1**

1.1 Why is Human–Robot communication Hard? 1

1.2 Unified view of *Prompt* 2

1.3 Thesis objectives and contributions 3

1.4 Outline of the thesis structure 4

I Learning to act on explicit intent **9**

2 From demonstrations to preferences for dishwasher loading **10**

2.1 Introduction 10

2.2 Related Work 11

2.3 Transformer Task Planner (TTP) 13

2.3.1 Prompt-Situation Transformer 14

2.3.2 Multi-Preference Task Learning 15

2.4 Experiments 17

2.4.1 Simulated Dishwasher Loading 18

2.4.2 Real-world dish-rearrangement 21

2.5 Limitation 22

2.6 Conclusions 22

3 From pixels of videos to low-level actions for manipulation **23**

3.1 Introduction 24

3.2 Related Work 26

3.3 Approach . . . 27
3.3.1 Datasets . . . 28
3.3.2 Model Architecture . . . 29
3.3.3 Training . . . 31
3.4 Experiments . . . 34
3.4.1 Task-based success . . . 36
3.4.2 Cross-object motion transfer . . . 38
3.4.3 Ablations . . . 39
3.5 Limitations and Future Directions . . . 41
3.6 Conclusion . . . 41

4 From high-level instruction to sub-tasks for mobile manipulation 44
4.1 Introduction . . . 45
4.2 Related Work . . . 46
4.3 Spatial Language Attention Policies . . . 48
4.3.1 Architecture . . . 49
4.3.2 Training . . . 51
4.3.3 Task Planning . . . 52
4.4 Experiments . . . 52
4.4.1 Longitudinal Task Execution on Stretch . . . 53
4.4.2 Task planning with in-context learning and fine-tuning LLaMa . . . 54
4.5 Limitations . . . 56
4.6 Conclusion . . . 57

II Understanding implicit intents and constraints 58

5 From visuals and listener's distance to acoustic noise of robot's motion 59
5.1 Introduction . . . 60
5.2 Related Work . . . 61
5.3 Acoustic Noise Prediction (ANP) . . . 62
5.4 Simulation Experiments . . . 64
5.4.1 Results . . . 65
5.4.2 Ablations . . . 66
5.4.3 Analysis . . . 67
5.5 Real world experiments . . . 67
5.6 Limitations . . . 69
5.7 Conclusion . . . 70

III Enabling evaluation and scaling for intent-awareness 71

6 Active visual querying 72
6.1 Introduction . . . 72
6.2 Related Work . . . 73
6.3 Problem Setting . . . 74
6.4 ActVQ-Arena . . . 75

6.5 Baselines . . . 76
6.6 Results and Discussion . . . 77
6.7 Discussion . . . 79
6.8 Conclusion and Future Work . . . 79

7 Conclusion 81
7.1 Summary of Contributions and Next Steps . . . 81
7.2 Key Takeaways . . . 83
7.3 Broad Open Challenges . . . 84
7.4 Future Directions . . . 86

IV Appendix 88

A Appendix: From demonstrations to preferences for dishwasher loading 89
A.1 Hardware Experiments . . . 89
A.2 Simulation Setup . . . 94
A.3 Simulation Experiment Details: Baselines, Training and Metrics . . . 96
A.4 Failure Analysis . . . 100
A.5 Additional Ablation Experiments . . . 101
A.6 Limitations and Future scope . . . 107

B Appendix: From pixels of videos to low-level actions for manipulation 109
B.1 Training Details . . . 109
B.2 Dataset . . . 109
B.3 Hardware setup . . . 111
B.4 Vid2Robotor Long Horizon Tasks . . . 111
B.5 Qualitative Results on Rare Tasks . . . 112

C Appendix: From high-level instruction to sub-tasks for mobile manipulation 114
C.1 Training . . . 114
C.2 Relative Action Module . . . 117
C.3 High-level Task Planning and Execution . . . 118
C.4 Experimental Setup for Skills . . . 121
C.5 Additional Analysis . . . 128
C.6 Additional Related Work . . . 130

D Appendix: From visuals and listener's distance to acoustic noise of robot's motion 132
D.1 Real-world Experiments . . . 132
D.1.1 More details on ANAVI Framework . . . 132
D.1.2 Collecting Real-world Audio . . . 134
D.1.3 Qualitative Evaluations on Real-World Panoramas . . . 136
D.2 Simulation . . . 139

D.2.1 Training Data Generation 139
D.2.2 More details on Room Impulse Response (RIR) 139
D.2.3 Visualization of Fixed Robot and Fixed Listener maps 140
D.3 Discussion and Broader Scope 140

E Appendix: Active visual querying **144**
E.1 POMDP details 144
E.2 Submission Workflow 145
E.3 Reward Calculation 145
E.4 Train test split 146

Bibliography **147**

List of Figures

2.1 **Overview** Transformer Task Planner (TTP) makes decisions about "when to open drawers" and "what objects to pick" for a robot arm, such that it follows the preferred order and location, as indicated in a prompt demonstration. **Top**: Prompt summarized as an illustration. **Bottom**: Robot loads 4 dishes - (1-4) opens top drawer and places blue bowl and cup, then (5-8) opens bottom drawer to place plate and pink bowl. Note that the prompt can have different number of objects than in the situation where the robot acts. TTP learns to infer and apply preferences in simulation and transfers zero-shot to the real world. 12

2.2 **Architecture overview**. (Left) shows of how a scene is converted to a set of instances. Each instance is comprised of attributes, i.e. pose, category, timestep, and whether it is an object or place instance. (Right) Instance attributes (gray) are passed to the encoder, which returns instance embeddings for pickable objects (red), placeable locations (green), and <ACT> embeddings (blue). The transformer outputs a chosen pick (red) and place (green) instance embedding. 14

2.3 **Prompt-Situation Architecture**. (Left-half): prompt encoder ψ that takes as input a "prompt" (demonstration) and outputs a learned preference embedding γ. (Right-half): the situation decoder π, which is conditioned on preference embedding from prompt encoder and is acting on the state from given "situation" (environment). 15

2.4 **Dishwasher Loading demonstration** in AI Habitat Kitchen Arrange Simulator. Objects dynamically appear on the countertop (ii-iv), and need to be placed in the dishwasher. If the dishwasher racks are full, they land in sink (v). 16

2.5 **Results** Comparisons of TTP, GNN-IL/RL and a Random policy in simulation across two metrics: SPA and TPA. TTP shows good performance at the task of dishwasher loading on seen and unseen preferences, and outperforms the GNN and random baselines. However TTP's generalization to unseen # objects is worse, but still close to GNN-IL (which was trained on *all* preferences), and better than GNN-RL and random. 19

2.6 **Ablations for #objects, training trajectories and preferences**. (a) Out-of-distribution generalization to #objects leads to degradation in performance, and the effect is more severe when increasing more objects than reducing objects. (b) Performance improves significantly with increase in the number of trajectories per preference. (c) Performance roughly plateaus with more unique variants of preferences. 19

3.1 **Overview of Vid2Robot**, a video-conditioned robot policy. Given a human demonstration (top), Vid2Robot recognizes the task semantics and performs the same task based on the robot's current visual observation (bottom left). A successful trajectory is presented on the bottom right. . . 23
3.2 **Dataset creation.** (top row) Here we show a Robot-Robot video pair for *placing the rxbar into top drawer*. We similarly pair existing robot-robot videos performing the same task. (middle row) Here, we show Hindsight Human-Robot paired videos for *picking a Coke can from the bottom drawer and placing it on the counter* task. We use the task instructions from robot trajectories, ask human participants to perform the task and record a demonstration video from the robot's perspective/view. (bottom row) Here, we show a Co-located Human-Robot pair of videos for *placing the pipe wrench in the toolkit*. We record a human demonstration and a robot teleoperation in the same workspace. We use different workspaces to perform the same task instruction, thus collecting paired videos with visually diverse prompts and robot state observations. More details in Section 3.3.1. 27
3.3 **Architecture.** Our model takes the prompt video and the robot's current observations as the input, encodes those into token embeddings for the prompt video and the robot's state, cross-attends to produce state-prompt encoding, and translates it into the expected robot action at the current timestep. More details in Section 3.3.2). 29
3.4 **Training Setup.** We show all the losses used for training Vid2Robot, particularly how each loss connects to its different modules. Along with (1) the main action prediction loss, we apply three auxiliary losses: (2) temporal video alignment loss, (3) a contrastive loss between the prompt and robot video performing the same task, and (4) a contrastive loss between a prompt/robot video with the language embedding. More details are in Section 3.3.3. 32
3.5 **Policy Rollouts.** Each row shows a prompt video of a human doing a task on the left, and on the right, we show the corresponding successful robot rollouts using Vid2Robot. Note how visually different the prompts are, while the policy rollouts have different lighting and backgrounds, as well as the number and placement of the distractor objects. 35
3.6 **Partial Success Rate for BC-Z and Vid2Robot.** Our policy Vid2Robot outperforms BC-Z in terms of *reaching* the correct object, *grasping* it, *releasing* it at the correct location and then *terminating* the episode correctly. Note that BC-Z does not have terminate control. 38
3.7 **Qualitative results for *cross-object motion transfer*.** (Top-blue) Prompt video of *placing coke can upright*; (Green) Policy rollouts with a *green can*, *chips bag*, *stapler* and a *soft toy* in front of the robot. Vid2Robot infers the motion of *place upright* in the prompt video and applies it to other objects. The policy adheres to the prompt video by picking the green can instead of the chips bag or banana. 42

3.8 **Ablation for auxilliary losses used in Vid2Robot.** We compare our proposed approach that has all auxiliary losses (green, left) with a variant without language contrastive loss that was originally proposed in BC-Z (orange, middle) and a version with no auxilliary losses (blue, right). More details in (§3.4.3) . . . 43
3.9 **Failure analysis with policy rollouts.** (Top) Policy predicts gripper pose and depends on the IK solver to move the arm. Sometimes, the IK solution can block the robot's camera view. (Middle) Grasping failures happen, especially with transparent and deformable objects. (Bottom) Distractor objects and differences in lighting and background may cause recognition errors, where policy might perform the correct motion but with an incorrect object(s). . . . 43

4.1 **Illustration** of SLAP, a framework that allows mobile manipulators to accomplish multi-step tasks given natural language goals in a sample efficient way. The demos are recorded on 'Hello Robot Stretch RE2'. (top) *Bring me a bottle*: Robot navigates to the countertop to pick the blue bottle in its view and then goes to the human to handover. (bottom) *Take the lemon out of the drawer*: Robot opens to drawer and picks up the lemon. 46
4.2 **Overview.** Spatial Language Attention Policies (SLAP) learn language-conditioned skills from few demonstrations in a wide variety of cluttered scenes. SLAP has two components: an *interaction prediction module* (IPM) to localize relevant features in a scene, and an *action prediction module* (APM) that uses the local context to predict an executable action. 48
4.3 **Architecture** of the interaction prediction module. The point cloud is downsampled to remove duplicates and encoded using two modified set-abstraction layers. The SA layers generate a local spatial embedding which is concatenated with proprioceptive features - in our case, the current gripper state. Both spatial and language features are concatenated and input into a PerceiverIO transformer backbone. We then predict an interaction score per spatial feature and the *argmax* is chosen as the interaction site for command l. . . . 50
4.4 **Training** SLAP. Demonstrations are collected and used to train the Interaction Prediction module and the Action Prediction Module separately. . . . 51
4.5 **Qualitative Results**: Examples of tasks executed on a Franka arm through our trained model in a clean setting. We trained numerous tasks (left) and tested on both seen and unseen objects (right). . . . 53
4.6 Output for interaction action predicted by PerAct (left) and interaction point predicted by SLAP (right) for the same input. We see a trend where π_I predictions for finer details are consistently more optimal . . . 56

4.7 A comparison of attention masks with (right) and without (left) the locality loss. Attention is more focused on relevant structures and less spread to irrelevant structures in the right image. The top row shows the attention mask for opening the top drawer: the previous model (left) spreads attention to the bottom handle due to its association with the "open" skill, while the new model (right) focuses only on the top drawer. Attention is more concentrated on handle for the "open bottom drawer" skill in the bottom row. 56

5.1 **Overview**. We generate data for acoustic impulse response in simulated 3D scenes from real-world environment scans. x-axis shows the relative distance of the listener from the source agent, y-axis shows the max decibels of a simulated Impulse Response (IR) at listener. The box area on the panaroma shows the listener's direction relative to the agent. Note the complex response pattern as materials, objects, and geometry have non-linear interactions with the sound. 61

5.2 Histogram of max decibel values from simulated data 62

5.3 **Architecture.** Our Acoustic Noise Prediction (ANP) model consists of *Image encoder*, *Direction-Distance encoder* and *Predictor* modules. The inputs are the 360°RGB panorama view at the robot's location, and the relative polar coordinates of the listener. The square box on the panorama highlights robot's current facing direction, and is drawn for illustration purposes only. The output is the max decibel (dB) of the Room Impulse Response (RIR) at the listener's location. 63

5.4 **Main Results.** Predicted Data Distribution plots for (left to right) Heuristic, DisLinReg, DirDisMLP, **Ours**(Pano)-VisDirDis. 65

5.5 $\epsilon-$threshold accuracy for Heuristic, DisLinReg, DirDisMLP, EgoVisDirDis, and PanoVisDirDis 66

5.6 **Ablations**. Data distribution plots for classification models trained with the cross-entropy loss with (a) 16, (b) 64, (c) 128 binned dB values, (d) regression model with an ego-centric view and (e) our proposed model on the right. We observe that the cross-entropy models are easier to train and capture outliers but provide a coarser prediction granularity. The learned distributions with ego-view miss outliers with very high values for the longer distances. 67

5.7 We present our ANP model's predictions for maximum decibels of room impulse response (max dB RIR) with real-world panorama scans. Note that max dB RIR is normalized between 0 to 1. **Top**: a large space in an office building. Here the north faces an open space area, with a corridor from west-to-east direction. There is a staircase close enough in the northeast direction that blocks most of the sound. In the south direction is a wall, and office rooms have closed wooden doors and bulletin boards. **Bottom**: a bedroom of an apartment. The west direction leads to a corridor and then the living area till 9.5m from the robot's location. The bed and a side table are placed in the north, with a wall at about 3.5m. Towards the north are windows and some furniture at about 3 m relative to the robot. In the south direction, a wall and an opened wooden door at 0.5m. The sound measurements are reported in Table 5.1. Our model's predictions align with that sound decays slowly in the open space with distance and peaks at shorter distances in corridors due to reverberations. The decay is much faster in a smaller room than in an open office space. . 68
5.8 Real-world audio recorded at the listener location (in blue) as the robot starts (from the triangle) and reaches the target (cross). The red path is approximately the shortest distance (32 seconds), but here, the robot goes quite close to the listener. A longer yet quieter path (46 seconds) is shown in green as it navigates behind the separator, thereby diffusing the robot's movement sound that reaches the listener. Best viewed in color. . 69

6.1 Environment tiers. (left) robot-free for direct control on camera and object pose, (middle) Single-arm Franka, (right) Bimanual ALOHA 74
6.2 Annotation pipeline for ActVQ-Arena, showing the four stages: (1) Object Import into MuJoCo, (2) Navigation to desired view by controlling the camera distance, look-at, azimuth, elevation and object's translation and rotation, (3) Region Annotation with top-left, top-right, bottom-right and bottom-left points marked by the user, and (4) Transform these to 3D points using depth and camera intrinsics, finally to Object Frame for later use in reward at runtime computation. 76
6.3 Success Rate based on Environment Tiers 78

A.1 Human demonstration of real-world rearrangement of household dishes. . 90
A.2 Pipeline for Real Hardware Experiments 90
A.3 Point cloud and grasps for different objects during policy rollout. 91
A.4 Coordinate Frame of reference in simulation (left) and real world setting (right). Red is x-axis, green is y-axis and blue is z-axis. 92
A.5 Human demonstration with point and click in simulation 94
A.6 Model workflow for GNN-IL and GNN-VAE-IL inspired from [120, 101]. The one-hot encoding per preference is passed as input to GNN-IL or GNN-GT-IL (there is no unseen preference for GNN-GT-IL). GNN-VAE-IL infers the preference from prompt through reconstruction and KL loss on preference embedding space. 97
A.7 Category level accuracy grouped by batch size for prompt-situation training. 99

A.8 Comparisons of TTP, GNN-IL, and GNN-VAE-IL simulation across two metrics: SPA and TPA. TTP shows good performance at the task of dishwasher loading on seen and unseen preferences and outperforms the GNN-IL and GNN-VAE-IL baselines. Note that GNN-IL has ground-truth training data on unseen preferences. 100

A.9 TE metric for held-out test settings. 100

A.10 Failure analysis for TTP. X-axis indicates enlists different error types concerning physical constraints (**Task-Obj Place**: object place pose is infeasible, **Task-Rack**: not placing when in top when its open, not placing in bottom when top is closed and bottom is open, placing object when both racks are closed), preference constraints (**Pref-Loc**: location violation in terms of incorrect rack, **Pref-Order**: order violation of preference) and No-ops (**Repeat Obj**: picking an already placed object, **Reposition**: Adjust the placement of an object within the same rack, **Repeat Rack**: picking a dishwasher part repeatedly; leads to termination, **Time Exceed**: leads to termination). Y-axis is the error count per rollout. Most errors are due to failing in the location-aspect and order-aspect of the preference. The **repeat-*** pattern triggers the termination in most rollouts, indicating that failure scenario usually obeys physical constraints but does not make progress towards the task, thereby resulting in lower SPA and TPA. . . . 101

A.11 Histogram per error type to estimate the frequency (Fig A.10 shows only mean and std). X-axis shows the value of the error count. Y-axis shows the number of times that value occurs in all of the rollouts. Most rollouts have 0 or very few preference based errors. Error count show peak near 0 and gradually reduces of towards higher numbers. This is an extended analysis of the error counts in figure A.10 showing the failure profiles for the different error types. 102

A.12 [Left-to-Right] Comparing different design choices of attribute encoders in terms of category token accuracy on held-out test prompt-situation session pairs. 103

A.13 Incorporating Context History instead just current observation. **Left**: Modified TTP for processing with previous Context History k. **Right**: Plot showing category level accuracy for the held-out test sessions for single preference training with context windows. While larger context window size learns faster, the asymptotic performance for all context windows converges in our setting. 104

A.14 Comparisons of TTP, Continuous Place Pose Prediction (TTP-ContP), and Discrete Category Token Input (TTP-DCat), in simulation across two metrics: SPA and TPA (on Y-axis), on seen preferences and objects during training. Continuous Pose prediction achieves much lower SPA due to infeasible object place pose prediction and thereby violates the *location-aspect* of the preference. As the TPA is based on the pick order sequence, TPA is high indicating that the policy learns the *order-aspect* of the preference. Discrete category prediction achieves low SPA and TPA due to as few as 7 categories mapped to the embedding of size 64; similar to original TTP with continuous category representation in terms of the bounding box. 105

A.15 Failure case analysis for continuous place pose prediction. The X-axis indicates the error count per rollout. The Y-axis enlists different error types concerning physical constraints (Task-Obj Place: object place pose is infeasible, Task-Rack: not placing when in the top when its open, not placing in the bottom when the top is closed and the bottom is open, placing object when both racks are closed), preference constraints (Pref-Loc, Pref-Obj) and No-ops (Repeat Obj: picking an already placed object, Reposition: Adjust the placement of an object within the same rack, Repeat Rack: picking a dishwasher part repeatedly; leads to termination, Time Exceed: leads to termination). 106

B.1 Qualitative results showing how Vid2Robot can perform long horizon tasks with chain prompt videos. 112
B.2 Qualitative results on rare tasks. 113

C.1 Regression model architecture with separate heads for each output. The point-cloud is cropped around the interaction point with some perturbation and passed to a cascade of set abstraction layers. Encoded spatial features are then concatenated with language and proprioception embeddings to predict position offset of action from interaction point, absolute orientation and gripper action as a boolean. 117
C.2 LSTM-based regression model architecture based on the regression head and PointNet++ embeddings introduced in Fig. C.1. LSTM-based architecture shows higher stability in learning action distribution with wider distribution due to the conditioning effect. 118
C.3 Diagram showing where immovable objects were placed in the environment (note that chair can move anywhere within the blue region). The three colored regions signify the placement assignment for different artifacts involved in the tasks. The circular symbol signifies the robot's pre-determined orientation where the beak represents where the robot will be facing. Position for robot's placement is determined based on predicted p_i 121
C.4 Within distribution objects used at training time and out-of-distribution objects introduced during testing in our experiments. 122
C.5 Seen objects and unseen distractors used in longitudinal experiments with Stretch. 122
C.6 Snapshot of test scenes from Franka evaluations to show the range of variation at test-time 122
C.7 Pour into bowl task: Showing variability and out-of-domain distribution covered by test against training samples. Top row: Training scenes. Note that the bowl was always placed somewhere on this particular region (right of sink) on the counter. Bottom row: Test scenes. Note bowl is surrounded by unseen clutter, placed in novel unseen environment and at relatively different positioning with respect to the camera and robot. Green boundary signifies successful episode, red a failed episode. 123

C.8 Close drawer task: Showing variability and out-of-domain distribution covered by test against training samples. Top row: Training scenes. Note the absence of any clutter and narrow range of relative positioning of drawer with respect to the camera and robot. Bottom row: Test scenes. Note presence of objects used in other tests in the same frame. Green boundary signifies successful episode, red a failed episode. 124
C.9 Examples of out of distribution predictions made by π_I. We show that it is able to handle heavy clutter around the implicated object to predict interaction points. Note that the prediction for bottle picking is sub-optimal in this example. 128
C.10 A generalization example of success for our model. The new bottle has same shape as the within distribution bottle but is much taller, different in color and wider in girth. The model is able to predict the interaction site and a feasible trajectory around it. We note though the execution of this trajectory was a failure; due to wider girth of the bottle the predicted grasp was not accurate enough to enclose the object. 128
C.11 Examples of failure cases for our baseline, PerAct, for the "place in drawer" and "place in bowl" tasks. In the top example, the gripper is moved from drawer's side towards inside, instead of from the top as demonstrated by expert. The gripper ends up pushing off the drawer to the side as our motion-planner is not collision-aware. Note that SLAP does not exhibit such behaviors as π_R implicitly learns the collision constraints present in demonstrated data. In the bottom example, each action prediction is disjointed from previous and semantically wrong. 129
C.12 An example out-of-distribution SLAP failure where an extreme sideways configuration of the drawer is paired with unseen distractors for the "open top drawer" skill. Note that the attention mask ranks other distractors in its top 5% and fails to choose an optimal interaction point. 129

D.1 Overview of ANAVI Framework: (a) Scan Environment, (b) Run ANP for listeners, (c) Weigh the audio noise vs. time costs, multiple listeners, and (d) plan with the overall cost. 133
D.2 We visualize how accurately estimating the noise cost (using the simulator as opposed to a distance heuristic) can enable better audio-aware paths. . 133
D.3 Panorama taken from the sound source location. Note the directions west, north are in the room, facing east is a wall and the south direction leads out of the room into the hallway. 135
D.4 **Setup for ANP Predictions on Real World Images**. This image visualizes ANP model predictions similar to Figure 7 from the main paper. The top row depicts a real-world panaromic image. The second row plots the maximum decibels of room impulse response $\hat{y}_{\text{ANP}}$ (max dB RIR). Note again that max dB RIR is normalized between 0 to 1. The third row plots the difference between the model's prediction and the linear regression prediction, i.e. $\hat{y}_{\text{ANP}} - \hat{y}_{linreg}$. 137
D.5 We show cherry-picked examples of model's prediction in (a) an open office area, and (b) a bedroom in a single family house. 138

D.6 Sim-to-real gap when applying the ANP model to real world panoramas. For each plot, the top row shows a real world panorama, second row the ANP predictions depending on the distance and angle to the sensor location, and the third row the difference compared a basic linear regression estimate (green = ANP predicts louder). 142

D.7 ANP Predictions over the Birds Eye View (BEV) map for (Left) Fixed listener (indicated with blue plus sign) and (right) Fixed robot (indicated with red triangle). Note that ANP uses panorama from robot's point of view, implying that the fixed listener maps use different visual features whereas fixed robot uses the same visual for each cell. The relative distance and direction from robot to the listener changes for each cell. We show all locations with 96 dB (more than 85%) in yellow, rest in blue and non-navigable regions at height 0 in purple. **Top**: The fixed listener predictions are spread more along the corridor (up-down) than (left-right). The fixed robot plot is more smoothly and uniformly spread across the map. **Middle**: The fixed listener predictions align with intuition that the sound should be louder in the open space compared to the area at same distance but separated by a wall. The fixed robot predictions plot follows the distance heuristic but not the common intuition. **Bottom**: Both fixed listener and fixed robot maps show the expected spread of sound at the cross-intersection. It is interesting to see that the fixed listener plots are quite noisy while the fixed robot one is very circular (and doesn't seem to care too much about walls) 143

E.1 ActVQ-Arena Policy Submission Workflow: Upon receiving a new agent policy, ActVQ-Arena loads the model weights and configures the policy for its designated environment tier. The system then instantiates simulation with randomized object layouts and camera poses to ensure robust evaluation across diverse scenarios. For each trial, ActVQ-Arena selects a specific task objective and retrieves the corresponding precomputed 3D annotation points that define where relevant information is. The policy predicts action for the agent, and ActVQ-Arena computes a reward at each timestep and number of steps taken. After completing all rollouts, ActVQ-Arena reports the performance metrics - such as task success rate, average visible area, and cumulative reward to the leaderboard, thereby ranking the submission among competing policies. 145

List of Tables

2.1 Three example preferences for dishwasher loading. Rack order and their respective contents (ordered by preference). 17
2.2 Attribute Ablations 20

3.1 Task Success Rate for Robot and Human prompts. 34
3.2 Real Robot Evaluation of Vid2Robot and BC-Z with more trials to ascertain the statistical significance of the results. 38
3.3 Cross-object motion transfer success. 39

4.1 **End-to-end performance.** Learned skills outperform heuristics except when Detic fails. 53
4.2 SLAP and PerAct [190] performance on real world Franka manipulation tasks. We evaluate both seen scenes (seen object positions and distractors), but in different arrangements, and unseen scenes with previously-unseen object positions and distractors. SLAP is notably better overall in both conditions. 54
4.3 SLAP on a mobile manipulator using a multi-task model across 4 skills, over 5 tries. With semantic predictions added to our feature space, we see the model perform better on unseen scenes with new distractors and unseen relative position of the robot with respect to the scene 55
4.4 Fine-tuning (FT) outperforms in-context learning (IC) for same latency. . . . 55

5.1 Acoustic measurements of Stretch and Unitree Go2 robot actions sounds, over the distance. The audio recordings were taken in a real Bedroom in an Apartment. 69

6.1 Quantitative results for all methods in the No-Robot tier. **SR**: Success Rate, **T2S**: Time-to-Success, **Δ Unseen-Objects** : Change in Success Rate with Unseen Objects, **Δ Unseen-Instructions (%)**: Change in Success Rate with Unseen Instructions (%) 77

B.1 Detailed Architecture for Vid2Robot 110
B.2 Hyperparameters for Vid2Robot 110
B.3 Data augmentation for training Vid2Robot 111

C.1 Data Distribution of procedurally generated samples across task families. 119
C.2 Examples of language used to train the model. 127
C.3 Examples of out-of-distribution language annotations used for evaluation . 127

D.1 Real Audio for a Bedroom in a Single Family House. 135

Chapter 1

Introduction

1.1 Why is Human–Robot communication Hard?

When we say *"Can you hand me that?"* while glancing at a fruit on the table, we expect implicit understanding; not just of the words, but also of our gaze, tone, and the shared context. For humans, this comprehension is effortless. Our ability to understand the world and communicate with others spans multiple modalities: language, gestures, gaze, intonation, and even silence, each layered with implicit intent.

Human communication is inherently multimodal, and each channel offers unique advantages. Language provides abstract and efficient goal specification, while visual demonstrations can convey nuanced motions or preferences that are difficult to describe verbally. This effortless fluency is not magic; it is the product of two coupled learning processes operating on very different time scales.

On the slow time scale, evolution equips us and other animals with robust perceptual and motor priors. On the fast time scale, culture and language are much faster, passed across generations and within communities, allowing shared nuance: pointing to an object, using emphasis in speech, or simply pausing with expectation can all be interpreted correctly by another human. Both of these two factors contribute to our multimodal fluency, enabling us to perform, coordinate, teach, and delegate tasks seamlessly.

Robots do not inherit either time scale: they lack both evolved perceptual priors and culturally shared conventions. Building robots that can interpret intent through multimodal signals is the key to their collaboration, trust, and usability in everyday settings. However, this human-like interaction remains hard for robots. The paradox that *"what seems easy for humans is difficult for machines, and vice versa"* was noted by Hans Moravec in 1988 [145]. He observed that *"it is comparatively easy to make computers exhibit adult-level performance on intelligence tests or playing checkers, and difficult to* *give them the skills of a one-year-old when it comes to perception and mobility*". This hints why we still lack general-purpose robots in our homes today.

Recent years have seen an accelerating progress in general-purpose AI models. Large language models (LLMs) [151, 68] can generate fluent prose and reason over abstract concepts; Visual language models (VLMs) [84] video generation models [20, 219] simulate coherent dynamics over time. However, we remain far from realizing the same fluency in real-world robotics, especially in domestic settings, where robots must perceive complex, cluttered environments and interpret diverse human cues to act appropriately.

Robots lack both the evolved perceptual priors and the culturally shared conventions that make human communication feel effortless. As a result, the bottleneck is not only perception or control but *communication*: how a human specifies intent, constraints, and preferences in forms a robot can ground. We therefore introduce a single interface for intent specification, the *prompt*, which unifies demonstrations, videos, language, information requests, and social budgets into one conditioning signal for planning and control. Section 1.2 formalizes this prompt space and explains how it connects to the systems developed in this thesis.

1.2 Unified view of *Prompt*

To address the communication bottleneck outlined in Section 1.1, we formalize a *prompt* as the conditioning variable that binds human-specified intent to the robot's perception and control. In single-task regimes, a policy $\pi : s_t \rightarrow a_t$ needs no intent; in multi-task settings, a human must convey *what* to do (goal), *where and how* to do it (referents and preferences), and *under what limits* (constraints). We capture these a *prompt* conditioning c so that $\pi : s_t, c \rightarrow a_t$.

Throughout this thesis, understanding the *prompt* is not merely mapping text to actions; it requires grounding semantic terms (e.g., "on top rack in front", "upright," "front label," "quietly") in the robot's sensory context and in human preferences. Large language models (LLMs) provide strong priors over everyday semantics, but they do not, by themselves, bind meanings to the robot's geometry, dynamics, and sensors. The goal of this thesis is to explore how each modality, including demonstration, video, language, and sound, can serve as a single conditioning variable to the robot policy, so that semantics are resolved on-the-fly for the current implicit state of the user.

1.3 Thesis objectives and contributions

This thesis tackles the multifaceted challenge of human intent by developing a suite of foundational capabilities. Learning representations from multimodal signals, including video, language, and audio, enables robots to interpret human intent with less burden on the user. Rather than pursuing a single monolithic model, I approach several distinct systems as a deliberate research strategy: to isolate and solve key subproblems.

To investigate intent-understanding problem, I train robot policies for learning explicit preferences from vision (Chapter 2, 3), grounding high-level language in motion (Chapter 4), understanding implicit social norms (Chapter 5), and establishing robust evaluation paradigms (Chapter 6). My belief is that progress on these distinct fronts is a necessary prerequisite for building a truly integrated and general-purpose home robot.

These works span task planning [92], navigation [87], low-level manipulation [93, 11], and mobile manipulation [155, 224]. Alternately, we categorize these works by modalities of human expression: language [155, 89, 11, 224, 150], visual [91, 93], and audio [87], as it provides a human-centric understanding of the robot's capabilities.

Thesis statement Intent-aware embodied agents require: (i) methods that can follow explicit human intents (preferences, demonstrations, instructions), (ii) mechanisms that respect implicit social constraints, and (iii) evaluation settings that force agents to actively seek information rather than passively receive it. Concretely, the thesis contributes: § 2 TTP (Transformer Task Planner) for preference-aware sequencing; § 3 Vid2Robot for end-to-end video-conditioned manipulation; § 4 SLAP for spatial-language policies over point clouds; § 5 ANP for acoustic-awareness; and § 6 ActVQ-Arena, a benchmark for active visual querying across embodiments.

Intended audience and scope. This work targets researchers and practitioners building household and service robots that must personalize to users, operate safely around people, and explain their behavior. The contributions prioritize (a) grounded semantics over pure language benchmarks, (b) interaction efficiency, and (c) real-robot feasibility under latency and compute constraints. We do not address in-hand dexterity or multi-robot coordination, focusing instead on the perception–planning–control loop for single-robot, user-facing tasks.

1.4 Outline of the thesis structure

Part I forms the majority of the core contributions, where I focus on robot policies that understand explicit intent, specifically expressed via visual demos and verbal instructions.

Part I: Learning to Act on Explicit Intent

Video-based intent inference (TTP and Vid2Robot). We perceive our surroundings and act mainly on the basis of our eyes. When we are first learning a new skill, our parents often show us how to do it. We recognize patterns and preferences to imitate, which we can generalize to new scenarios. An ideal household robot should be able to notice where we put away certain dishes or how we organize our home, simply through observation, without being told. To give robots this skill, I introduce the Transformer Task Planner (TTP) [92] in Chapter 2. TTP learns high-level actions from demonstrations using object-centric representations. The policy predicts actions that adhere to the underlying task semantics shown in the demonstration prompt. TTP can be transferred to the real world, with a Franka arm performing the dish rearrangement. This work has been published and presented at CoRL 2022 [92].

As we grow, we learn a variety of skills, like a new cooking recipe, by watching videos on the Internet. We execute based on our understanding of 2D pixels despite occlusions and lack of 3D structure. While we want our robots to learn from videos directly, prior work [9] suggests it is a bigger challenge than language conditioning. Prior works [25, 148] show the success of video-conditioned policies but largely in simulation. In Chapter 3, I present end-to-end video-conditioned policies that can adapt to new manipulation skills in real-time. This work has been published and presented at RSS 2024 [93].

Language-based subtask understanding (SLAP). Observation-based learning is valuable, but most human communication relies on natural language. Language has been a successful interface for reasoning about questions with ChatGPT [209] or imagining a picture with Diffusion models [50]. For robot assistants, language can provide a succinct and abstract representation of our goals (e.g. *Move the book to the couch*). Language also unlocks common knowledge and assumptions about the world, like *Where should I look for a book? Where is the couch most likely to be?* For pre-thesis research, I explore the role of (i) Capturing commonsense co-occurrence knowledge, (ii) Leveraging Large Language Models (LLM) and Visual Language Models (VLM) for grounding semantics and motion control and (iii) Improving visual grounding for general-purpose VLMs. In this thesis, I present Chapter 4 on decomposing high-level instruction into reusable substeps for sample efficient multi-task policy. Below I provide a brief summary of pre-thesis research and Chapter 4.

Find the keys! Capturing commonsense co-occurrence knowledge. Humans often search by first scanning the area coarsely and finding objects where it is likely to find the *keys*, for example, on the *table*. On getting closer, if *keys* are not found, one would navigate to the next most likely location. We search for smaller, visually imperceivable objects like *keys* based on associated big objects that are easily detectable, like a *table* or a *dresser*. I study whether frequently co-occurring objects can guide the search for small objects in simulated homes [191]. In this work, graph-based embeddings based on physical distance and commonsense priors outperform baselines for longer horizon tasks. This work was in NeurIPS Object Representations for Learning and Reasoning workshop 2020 [89].

Zero-shot prompting strategies for Large Language Models (LLM) and Visual Language Models (VLM) for grounding semantics and motion control. LLMs for robotics [116, 192] have mostly been used for high-level planning, and assume access to a skill library for low-level motion control. But LLMs also contain vast knowledge and can be prompted iteratively to generate low-level code for tasks like *opening drawer*, without any task-specific training. For robotic tasks, I contributed to development and evaluation of prompting paradigms to generate code for executing several low-level skills in PromptBook [11], which has been applied to diverse tasks, from *interacting with cabinets*, to *bimanual tasks* to *whisking*. In my prompts, I used examples, robot API documentation, state estimation tools, and chain-of-thought to write robot code. We improved the prompt iteratively using LLMs and human feedback. This work has been presented in conference proceedings at ICRA 2024 [11] and at CoRL LangRob workshop 2023 [10].

Improving visual grounding for general-purpose VLMs. LLMs/VLMs have demonstrated a wide range of general-purpose capabilities for robotics, as used in PromptBook [11]. However, early VLMs often struggled with hallucinations and incorrect grounding, in part due to their reliance on textual data over visual information during training. To address this, I contributed to FlexCap [57], where we trained VLM to autoregressively generate length-conditioned captions for the local regions in an image. Here, *length* serves as a proxy for *information density*, controlling how detailed the captions should be. Our extensive experiments show that FlexCap effectively provides a large set of dense captions for any selected subregion in an image, which enables: (a) diverse descriptions of selected object's attributes, (b) LLMs for reasoning over detailed image and video descriptions, and (c) outperforming state-of-the-art on image-QA and video-QA tasks. For more details on textually grounding the visual world, visit our FlexCap page.

Decomposing high-level instruction into reusable substeps for multi-task mobile manipulation. In this thesis Chapter 4, I investigate whether robots can be trained, just as humans, to perform diverse skills with very few demonstrations or instructions per task. I look into converting language requests into task and motion plans in which the language influences every part of the plan. When we have to *pick up a mug* or *open a kitchen drawer*, we reuse our ability to *approach* and *grasp*. This modularity enabled me to challenge strong assumptions about language-based goal conditioning in robot policies. In the work led by Parasher [156], we present spatial-language attention policies to efficiently sample and learn everyday tasks using the hierarchical decomposition of language into a sequence of interaction points and their relative waypoints. I define 'atomic skill' like `pick('mug')` as a task specified by an interaction point `IP='mug'`, and a sequence of relative waypoints `W=[`*`approach, grasp, lift`*`]`. This skill decomposition captures the invariance in semantically related actions and avoids compounding errors in behavior cloning. We show that our method outperforms PerAct [162] for tabletop (Franka arm) and mobile tasks (Hello Robot Stretch). This work is published in CoRL 2023 [155].

Overall, *Part I* outlines training methods and systems for robot policies that enable low-friction intent specification using vision (Chapter 2, 3) and language (Chapter 4). The following *Part II* discusses how to make robots understand implicit social norms, specifically about sound awareness in homes.

Part II: Understanding Implicit Constraints

Our world is governed by several implicit social norms, some of them as basic as how to move around quietly to not disturb others in homes and offices. Humans inherently understand how their actions impact the acoustic environment around them: we know to whisper when a baby is asleep or that it is okay to make noise during a lively party. As robots become increasingly integrated into our homes, we need this ability in robots to visually understand how acoustics differ by geometry and material composition.

In Chapter 5, I discuss how robots should perform implicit task specifications by estimating audio from visual observations of indoor environments through ANAVI (*Audio Noise Awareness using Visuals of Indoor environments for NAVIgation* framework. Since sound depends upon the geometry and material composition of rooms, I utilized an audio-visual 3D home simulator to train Acoustic Noise Predictor (ANP). ANP is trained to use visual observations for passively perceive loudness, without audio sensing. I collect acoustic profiles corresponding to different robot actions for navigation. Unifying ANP with action acoustics, I demonstrate how any robot, be it wheeled (Hello Robot Stretch) or legged (Unitree Go2), can estimate how loud it will be due to any of its actions and thereby plan to minimize the noise for the listener in the environment. This approach unified multimodal learning principles, demonstrating how auditory awareness complements visual understanding in making robots more adaptive and human-friendly. This work has been published in CoRL 2024 [87]. This is a classic case where the robot must read between the lines to be effective companions.

So far, I have discussed how to design policies that capture human intentions across different modalities. These modalities have far more nuanced interactions in everyday activities. In *Part III*, I investigate how best to evaluate these policies and study the axes of generalizations at scale.

Part III: Enabling evaluation and scaling for intent-awareness

Benchmarks shape their research field as they provide not only common ground for comparing methods but also crystallize what a community values as progress. In robotics, evaluation is uniquely hard due to the interplay of perception, control, and real-world constraints. To enable reproducible and generalizable robotics, we advocate for benchmarks that are grounded in realistic assumptions, span simulation-to-real transfer, and test the true generality of agents across tasks and embodiments.

My pre-thesis research collaboration with the OVMM benchmark [224] emphasized open vocabulary manipulation in household settings. OVMM demonstrated how integrating pre-trained vision-language models with goal-conditioned policies could enable multi-object and multi-task reasoning. This work was presented at CoRL 2023 [224], and was also hosted as a challenge at NeurIPS 2023 [225]. However, it also highlighted a gap - evaluation protocols often failed to test the agents' understanding of abstract instructions, temporal ordering, or non-trivial information-seeking behaviors.

"Find the keys in the living room" differs fundamentally from "find the expiry date on this bottle." The first is a global search over a large space with sparse hits; the second is a local viewpoint selection problem where the object is known, but the informative detail is occluded, tiny, or low contrast. In Chapter 6, I introduce ActVQ-Arena [88], a platform designed for the latter form of visual queries in embodied environments. Here I evaluate whether agents can extract semantically rich, localized information from everyday objects (e.g., 'expiry date', 'ingredients') and perform short-horizon manipulation to expose occluded attributes. While VQAs and VLMs today answer these queries, they need a clear view of the information and have no understanding how to act in the world to retrieve that information; at best, they can guide a human about where to move the camera next because they have a grounding for where the back of an arbitrarily posed object might be. I provide easy no-robot environments and scale from single-camera, single-arm setups to more complex bimanual and cluttered scenes. ActVQ-Arena provides a unified framework for evaluating agents across various real object scans, with a single annotation per information that can be reused across different pose configurations, compositional tasks, and environment tiers. We observe Diffusion Policies demonstrate the strongest success rate in seen settings, but face generalization challenge in unseen objects and instructions. With open-source environments, annotation pipelines, and evaluation metrics, I hope to establish a shared foundation for robust, general-purpose home robots. Just like in LLM and vision communities, I hope that benchmarks and arenas help us define the critical user-focused capabilities we need in our robot policies today: from grounded perception to compositional planning, from controlled simulation to reproducible real-world deployment.

In Chapter 7, I summarize key contributions, limitations, and unified takeaways from this thesis. I broadly explain the open challenges for the community and finally pitch four topics as next frontiers for research, namely foundation models for integrating modalities in embodied and robot learning, multi-objective pragmatic instruction following, personalization to user behaviors, and utilizing reasoning for embodiment adaptation.

Part I

Learning to act on explicit intent

Chapter 2

From demonstrations to preferences for dishwasher loading

Publication Information
Vidhi Jain, Yixin Lin, Eric Undersander, Yonatan Bisk, and Akshara Rai. "Transformers are adaptable task planners." Presented at the 6th Annual Conference on Robot Learning at Auckland, New Zealand. December 2022. In proceedings, pp. 1011-1037. PMLR, 2023.

Every home is different, and every person likes things done in their own way. Therefore, home robots of the future need to both reason about the sequential nature of day-to-day tasks and generalize to user's preferences. To this end, we propose a Transformer Task Planner (TTP) that learns high-level reasoning from demonstrations by leveraging object attribute-based representations. TTP is pre-trained on multiple preferences in a simulated dishwasher loading task and shows generalization to *unseen* preferences using a single demonstration as a prompt. Further, we demonstrate real-world dish rearrangement using TTP with a Franka robotic arm. See videos and code at `https://sites.google.com/andrew.cmu.edu/ttp/home`.

2.1 Introduction

Consider a robot tasked with loading a dishwasher. Such a robot has to account for task constraints (e.g. only an open dishwasher rack can be loaded), and dynamic environments (e.g. more dishes may arrive once the robot starts loading the dishwasher). Dishwasher loading is such a canonical task with user-specific preferences, like a user may place mugs on the top rack and plates on the bottom or load dirtier bowls before easier-to-clean cups, encoding their preference. Additionally, they pull out a rack before loading it, inherently encoding a structural constraint. This also holds in other household tasks, like cooking. For example, a person may start cooking potatoes before frying onions, as potatoes often take longer to cook, while another person first caramelizes the onions before adding in potatoes for flavor. In a kitchen, a user might have a preferred cabinet for plates, and another for cups. Learning temporal and spatial constraints and preferences from demonstrations requires policies with temporal context that can consider the *sequence of actions* demonstrated. Classical task planning [66] deals with such constraints through symbolic task description, but such descriptions are difficult to design and modify for new preferences in complex tasks. Building easily adaptable long-horizon task plans, under constraints and uncertainty, is an open problem in robotics.

Transformers are well-suited to this problem, as they have been shown to learn long-range relationships [28], although not in temporal robotic tasks. Recent work [102] has shown Transformers [208] can learn temporally-consistent representations, and generalize to new scenarios [21, 179, 125]. Our central question is: *Can a Transformer learn task structure, adapt to user preferences, and achieve complex long-horizon tasks using no symbolic task representations?* We propose Transformer Task Planner (TTP) - an adaptation of a classic transformer architecture that includes temporal, pose, and category embeddings to learn object-oriented relationships over space and time. By pre-training TTP on multiple preferences, TTP learns to infer unseen preferences from a single demonstration and adhere to it with a variable number of objects and dynamic environments.

Our main contributions are to (i) introduce transformers as a promising architecture for learning task plans from demonstrations using object-centric embeddings (§ 2.3), (ii) show that preference-conditioned pre-training generalizes at test time to new, unseen preferences and outperforms competitive baselines (§ 2.4.1), and (iii) transfer TTP to a rearrangement problem in the real world, where a Franka arm places dishes in two drawers, using a single demonstration (Fig. 2.1 and § 2.4.2).

2.2 Related Work

Learning for Sequential manipulation aims to overcome the challenges in designing and scaling classical Task and Motion Planning approaches. Object-centric pick-place representations using off-the-shelf perception methods are common in learning policies [210, 47, 231, 240, 226, 89], specifically for sequential manipulation [66, 159]. Transporter-Nets [232] primarily deal with table-top Pick&Place generalized to new object locations and do not consider in 3D sequential reasoning problems. SORNet [229] assumes access to a task planner to reason about sequential reasoning instead of a learned policy. Such a task planner is often tedious to design for complex tasks, and difficult to modify per

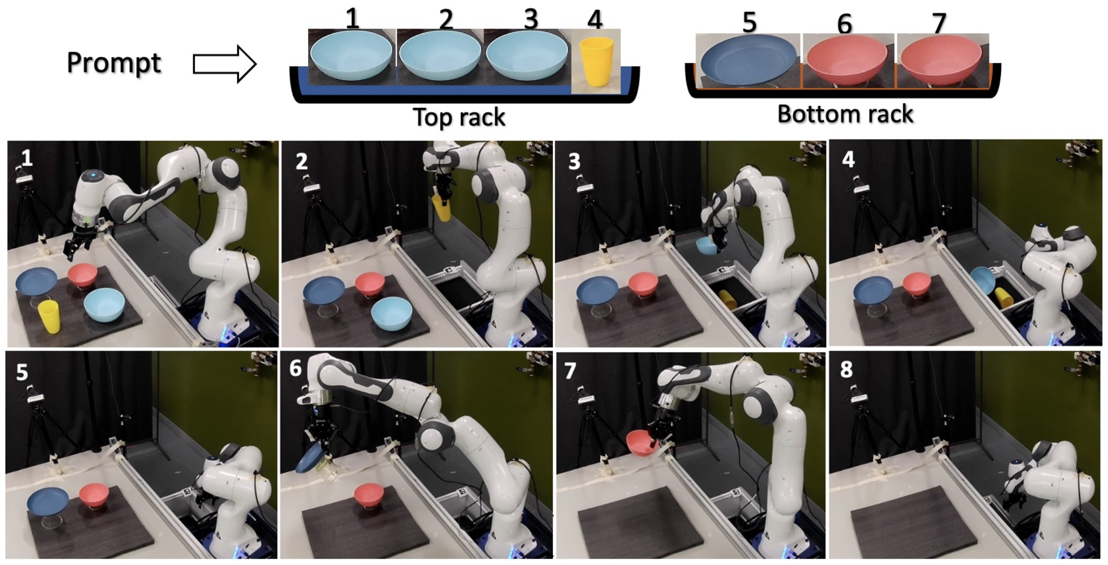


Figure 2.1: **Overview** Transformer Task Planner (TTP) makes decisions about "when to open drawers" and "what objects to pick" for a robot arm, such that it follows the preferred order and location, as indicated in a prompt demonstration. **Top**: Prompt summarized as an illustration. **Bottom**: Robot loads 4 dishes - (1-4) opens top drawer and places blue bowl and cup, then (5-8) opens bottom drawer to place plate and pink bowl. Note that the prompt can have different number of objects than in the situation where the robot acts. TTP learns to infer and apply preferences in simulation and transfers zero-shot to the real world.

preference. While [120] presents a learned sequential reasoning approach using GNNs, and [101] shows generalization of GNNs to preferences, neither is directly suitable for our task, so we combine [120, 101] to create a stronger baseline. Recent works have repurposed Transformers for other sequence modeling tasks [124, 199, 90, 35, 97, 169]. Prompt DT [222] considers a single model to encode the prompt and the successive sequence of state. This limits the size of the prompt and requires special tokens to demarcate between prompt and current state. We use the encoder-decoder setup and share the parameters for instance encoding. PlaTe [199] proposes planning from videos, while [35, 97, 169] model a sequential decision-making task. We consider long-horizon tasks with partially observable state features and user-specific preferences.

Preferences and prompt training There are several ways of encoding preferences in recent works. [101] propose VAE to learn user preferences for a spatial arrangement based on just the final state, while our approach models temporal preference from demonstrations. Preference-based RL learns rewards based on human preferences [211, 113, 117, 108], but do not generalize to unseen preferences. For complex long-horizon tasks, [40] shows that modeling human preferences enables faster learning than RL. We show generalization to an unseen preference by learning to infer them from the demonstration. Large language and vision models have shown generalization through prompting [37, 178]. Prompting has been used to guide a model to quickly switch between multiple task objectives [175, 115, 195]. Recent language models learn representations that can be easily transferred to new tasks in a few-shot setting [181, 122, 21, 37]. Our approach similarly utilizes prompts for preference generalization in sequential decision-making robotic tasks.

2.3 Transformer Task Planner (TTP)

We introduce TTP a transformer-based policy architecture for learning sequential manipulation tasks. We assume low-level generalized pick-place primitive actions that apply to both objects like plates and bowls, as well as the dishwasher door and racks. TTP learns a high-level policy for pick-up place according to the task structure and preference shown in the demonstrations. The following sections are described with dishwasher loading as an example, but our setup is applicable to most long-horizon manipulation tasks with generalized pick-place.

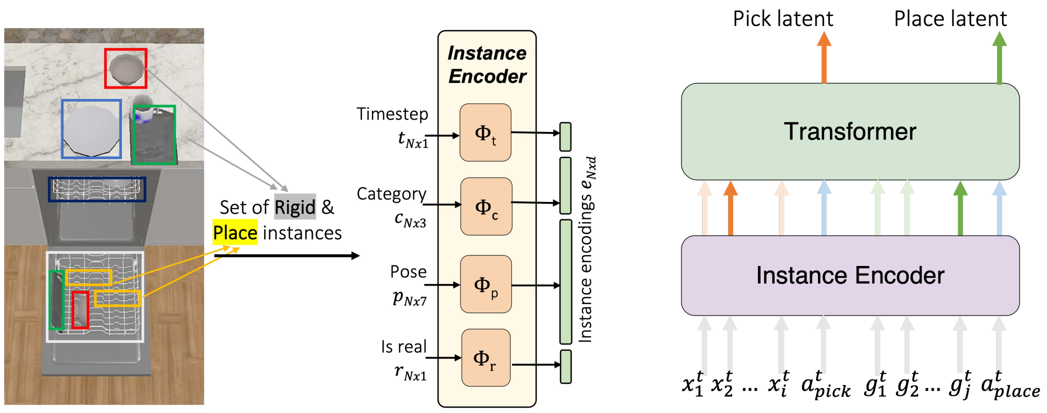
State-Action Representations We consider a high-level policy that interacts with the environment in discrete time steps. At every timestep t, we receive observation $\boldsymbol{o}_t$ from the environment which passes through a perception pipeline to produce a set of rigid-body 'pick'-able instances $\{x_i\}_{i=1}^{n}$, corresponding to the n objects currently visible. The pick *state* S_t^{pick} is described as the set of instances that are **visible** in $\boldsymbol{o}_t$: $S_t^{pick} = \{x_1, x_2, \cdots, x_n\}$.

Once a 'pick' object is chosen, we decide where to place it. We pre-compile a list of discrete placement poses corresponding to viable place locations for each object category, created by randomly placing objects in the dishwasher and measuring the final pose they land in.

All possible placement locations for the picked object category, whether free or occupied, are used to create a set of 'place' instances $\{g_j\}_{j=1}^{l}$. We describe an *action* $\boldsymbol{a}_t$ in terms of what to pick and where to place it. Specifically, a policy π outputs the pick action, which is an object in $\boldsymbol{o}_t$, i.e. $\pi(S^{pick}) \rightarrow x_{target}$.

The place state is $S^{place} = S^{pick} \cup \{g_j\}_{j=1}^{l}$. The *same* policy π then chooses where to place the object, $\pi(S^{place}) \rightarrow g_{target}$ (Fig. 2.2). Note that the input for predicting place includes both objects and place instances since the objects determine whether a place instance is free to place or not. On the other hand, the 'pick' action only looks at the current object instances to determine the next best object to pick. A demonstration is a state-action sequence $\mathcal{C} = \{(S_0, a_0), (S_1, a_1), \cdots, (S_{T-1}, a_{T-1}), (S_T)\}$. See A.2.

Instance Encoder We describe both 'pick' and 'place' instances in terms of their attributes such as $\{p, c, t, r\}$, where p is the pose of the object, c is its category, t is the



(A) Scene to Instance embeddings (B) Instances to prediction

FIGURE 2.2: **Architecture overview**. (Left) shows of how a scene is converted to a set of instances. Each instance is comprised of attributes, i.e. pose, category, timestep, and whether it is an object or place instance. (Right) Instance attributes (gray) are passed to the encoder, which returns instance embeddings for pickable objects (red), placeable locations (green), and <ACT> embeddings (blue). The transformer outputs a chosen pick (red) and place (green) instance embedding.

timestep while recording $\boldsymbol{o}_t$ and r is a boolean indicator, for instance, type – 'pick' or 'place' (Fig. 2.2a).

The pose is embedded as Φ_p, using a positional encoding scheme similar to NeRF [139] to encode the 3D positional coordinates and 4D quaternion rotation. To encode the category, we use the dimensions of the 3D bounding box of the object to build a continuous space of object types and process this through an MLP Φ_c. For each discrete 1D timestep, we model Φ_t as a learnable embedding, similar to the positional encodings in BERT [49]. The concatenated d-dimensional embedding for an instance at timestep t is represented as $f_e(x_i) = \Phi_t||\Phi_c||\Phi_p||\Phi_r = e_i$. See A.5 for instance encoding design. The encoded state at time t is represented as: $S_t^{enc} = \{e_0, e_1, \cdots e_N\}$. We drop $()^{enc}$ for brevity.

2.3.1 Prompt-Situation Transformer

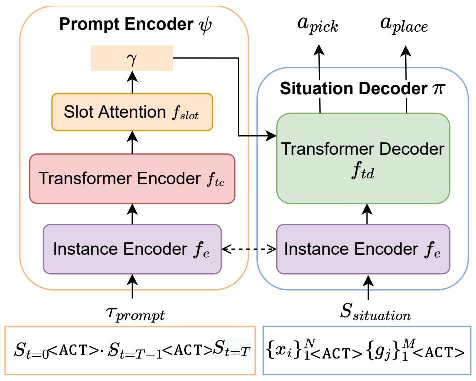
We use Transformers [208], a deep neural network that operates on sequential data, to learn a high-level pick-place policy π. The input to the encoder is a d-dimensional token[1] per instance $e_i \in [1, ..N]$ in the state S. In addition to instances, we introduce special <ACT> tokens[2], a zero vector for all attributes, to demarcate the end of one state and start of the next. These tokens help maintain the temporal structure; all instances between two <ACT> tokens in a sequence represent one observed state. A demonstration without actions is $\tau = \left[S_{t=0}, \text{<ACT>}, \cdots, S_{t=T-1}, \text{<ACT>}, S_{t=T}\right]$. To learn a common policy for multiple preferences, we propose a prompt-situation

[1]Terminology borrowed from natural language processing where tokens are words; here, they are instances.

[2]Similar to $< CLS >$ tokens used for sentence classification.

architecture, where "prompt" is a demonstration with some underlying preference, and "situation" is a different environment scenario where the policy acts. The policy cannot simply copy the prompt actions because the prompt object set is different from the situation. Instead, it needs to infer the underlying preference from the prompt and then apply it in the given situation.

In Fig. 2.3, the prompt encoder ψ is on the left and the situation decoder or policy π acting on the given state of the situation is on the right. The prompt encoder $\psi : f_{slot} \circ f_{te} \circ f_e$ consists of an instance encoder f_e, transformer encoder f_{te}, and a slot-attention layer f_{slot} [127]. Slot attention is an information bottleneck, which learns semantically meaningful representations of the prompt as a fixed and reduced dimensional embedding. ψ takes the demonstration τ_{prompt} as input and returns $\gamma = \psi(\tau_{\text{prompt}})$. The situation decoder $\pi : f_{td} \circ f_e$ receives as input N instance tokens from the current scene S, consisting of objects, as well as, placement instances (Fig. 2.2b). It consists a transformer decoder [208] with self-attention layers on the N input tokens, followed by a cross-attention layer with preference embedding γ from the prompt encoder. We select the output of the situation decoder at the <ACT> token and calculate dot-product similarity with the N input tokens e_i. The token with the maximum dot-product is chosen as the predicted instance: $x_{pred} = \max_{e_i, i \in \{1,N\}} \left(\hat{x}_{\text{<ACT>}} \cdot e_i\right)$. π is trained with cross-entropy to maximize the similarity of output latent with the expert's chosen instance embedding as target. To summarize, the prompt encoder receives a prompt demonstration as input and outputs a learned representation of the preference. These output prompt tokens are input to a situation decoder π, which also receives the current state as input. The decoder π predicts the action chosen by the expert in the given situation, adhering to the preference indicated in the prompt.



Prompt Encoder ψ
γ
Slot Attention f_{slot}
Transformer Encoder f_{te}
Instance Encoder f_e
τ_{prompt}
$S_{t=0}$<ACT>$\cdot$ $S_{t=T-1}$<ACT>$S_{t=T}$
a_{pick}
a_{place}
Situation Decoder π
Transformer Decoder f_{td}
Instance Encoder f_e
$S_{situation}$
$\{x_i\}_1^N$<ACT>$\{g_j\}_1^M$<ACT>

FIGURE 2.3: **Prompt-Situation Architecture**. (Left-half): prompt encoder ψ that takes as input a "prompt" (demonstration) and outputs a learned preference embedding γ. (Right-half): the situation decoder π, which is conditioned on preference embedding from prompt encoder and is acting on the state from given "situation" (environment).

2.3.2 Multi-Preference Task Learning

We adopt a prompt-situation architecture for multi-preference learning. This design (1) enables multi-preference training by disambiguating preferences, (2) learns task-level rules shared between preferences (e.g. dishwasher should be open before placing objects), (3) can generalize to unseen preferences at test time, without fine-tuning. Given a 'prompt'

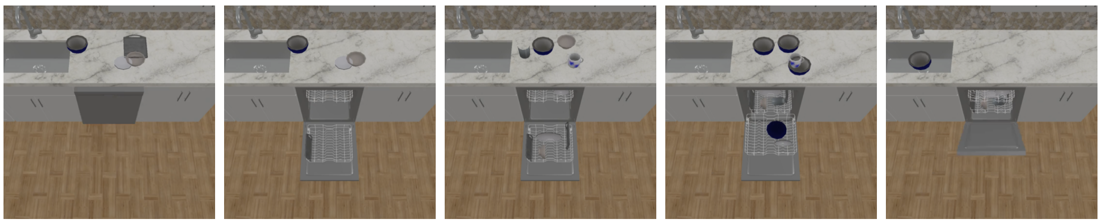


FIGURE 2.4: **Dishwasher Loading demonstration** in AI Habitat Kitchen Arrange Simulator. Objects dynamically appear on the countertop (ii-iv), and need to be placed in the dishwasher. If the dishwasher racks are full, they land in sink (v).

demonstration of preference m, our policy semantically imitates it in a different 'situation' (i.e. a different initialization of the scene). To this end, we learn a representation of the prompt γ^m which conditions π to imitate the expert.

$$\psi(\tau^m_{\text{prompt}}) \rightarrow \gamma^m \tag{2.1}$$

$$\pi(S_{\text{situation}}|\gamma^m) \rightarrow a^m_t = \{x^m_{\text{pred}}, g^m_{\text{pred}}\} \tag{2.2}$$

We train neural networks ψ and π together to minimize the total prediction loss on all preferences using the multi-preference training dataset $\mathcal{D}$. The overall objective is:

$$\min_{m \sim M, \tau \sim \mathcal{D}_m, (S,a) \sim \mathcal{D}_m} \mathcal{L}_{CE}(a, \pi(S|\psi(\tau))) \tag{2.3}$$

For every preference m in the dataset, we sample a demonstration from $\mathcal{D}_m$ and use it without actions as a prompt. We also sample a state-action pair (S, a) from any demonstration $\mathcal{C} \in \mathcal{D}_m$.

At test time, we record one prompt demo from a seen or unseen preference to output action: $a = \pi(S|\psi(\tau_{\text{prompt}}))$. All policy weights are kept fixed during testing, and generalization to new preferences is zero-shot using the learned preference representation γ. Unlike [101], γ captures not just the final state, but a temporal representation of the whole demonstration. Building a temporal representation is crucial to encode demonstration preferences like the order of loading racks and objects. Even though the final state is the same for two preferences that only differ in which rack is loaded first, our approach is able to distinguish between them using the temporal information in τ_{prompt}. To the best of our knowledge, our approach is the first to temporally encode preferences inferred from a demonstration in learned task planners.

Table 2.1: Three example preferences for dishwasher loading. Rack order and their respective contents (ordered by preference).

First?	*Top*	*Bottom*
Top	1. cups 2. glasses 3. small bowl	1. big plates 2. small plates 3. trays 4. big bowls
Bottom	1. cups 2. glasses 3. small bowl	1. big plates 2. small plates 3. trays 4. big bowl
Bottom	1. small plate 2. glasses 3. cups	1. big bowls 2. trays 3. big plates 4. small bowl

2.4 Experiments

We present the "Replica Synthetic Apartment 0 Kitchen"[3] (see figure 2.4, appendix and video), an artist-authored interactive recreation of the kitchen of the "Apartment 0" space from the Replica dataset [198]. We use selected objects from the ReplicaCAD [202] dataset, including seven types of dishes, and procedurally generate dishwasher loading demonstrations. See dataset details in A.2.

We define a preference in terms of expert demonstration 'properties', like which rack is loaded first with what objects? Table 2.1 describes the preferences of dishwasher loading in terms of three properties - first loaded tray, objects in the top and bottom tray. Preferences 1 & 2 vary in the order of which rack is loaded first, while 2 & 3 both load the bottom rack first with similar categories on top and bottom but with different orderings for these categories. Our dataset consists of 12 preferences (7 train, 5 held-out test) with 100 sessions per preference. In a session, $n \in \{3, ..., 10\}$ instances are loaded in each rack. The training data consists of sessions with 6 or 7 objects allowed per rack. The held-out test set contains 5 unseen preferences for $\{3, 4, 5, 8, 9, 10\}$ objects per rack. See preferences examples in A.2.

Additionally, to simulate a dynamic environment, we randomly initialize new objects mid-session on the kitchen counter. This simulates situations where the policy does not know every object to be loaded at the start of the session and has to learn to be reactive to new information. See A.2.

To measure success, we rollout the policy from an initial state and compare the resultant trajectory with an expert demonstration. Rollouts require repeated decisions in the

[3]"Replica Synthetic Apartment 0 Kitchen" was created with the consent of and compensation to artists and will be shared under a Creative Commons license for non-commercial use with attribution (CC-BY-NC).

environment, without any resets. A mistake made early on in a rollout can result in poor performance, even if the prediction accuracy is high. For example, if a policy mistakenly fails to open a dishwasher rack, the performance will be poor, despite good prediction accuracy in later steps. For in-distribution evaluation, 10 sessions are held-out per preference. For out-of-distribution evaluation, we create sessions with unseen preferences and unseen numbers of objects. Specifically, we measure:

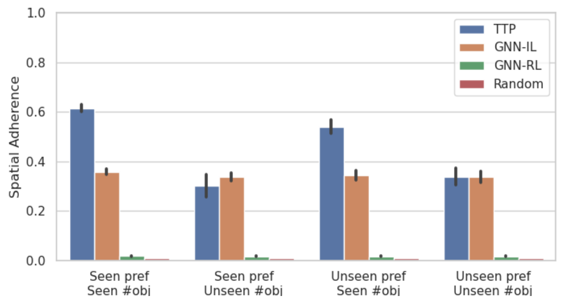
Spatial Preference Adherence (SPA) : SPA measures how well is the final state packed per user's spatial preference. Let $\hat{a}_i$ be the number of objects placed in the i^{th} receptacle adhering to the preference by the learned policy, and a_i be the number for an expert, then $SPA = \frac{1}{N}\sum_{i=1}^{N}\frac{\hat{a}_i}{max(a_i,\hat{a}_i)}$. SPA measures the packing efficiency of the final state of the dishwasher after policy rollout, while respecting the spatial preference of top or bottom rack per category. Note that if the policy follows the wrong spatial preference (places objects in the wrong rack), then SPA is low, even if the dishwasher is full.

Temporal Preference Adherence (TPA): TPA measures how much the policy deviates from the order of actions shown in an expert demonstration. Let $\hat{\tau}, \tau$ be the sequence of picked instances by the learned policy and expert respectively, then TPA $= 1 - \frac{EditDistance(\hat{\tau},\tau)}{max(|\tau|,|\hat{\tau}|)}$. TPA accounts for the order in which objects are picked, irrespective of where they are placed, and measures the deviation from the temporal actions of the user. TPA is high if the policy follows both sequential constraints of task (open dishwasher before loading), and the order-related preferences (bowls before cups).

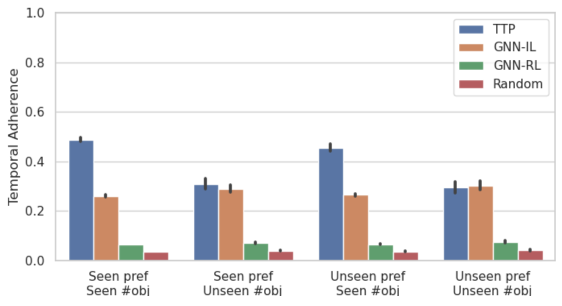
If the expert preference was to load the top rack first and the learned policy loads the bottom first, SPA would be perfect at 1.0, but TPA would be low. Both metrics are low if the policy repeatedly violates the task's physical constraints or does not make progress towards the task. See more details on metrics in A.3.

2.4.1 Simulated Dishwasher Loading

We compare our approach against Graph Neural Network (GNN) based preference learning from [120, 101] at this task of dishwasher loading. Neither of these works is directly suitable for our task, so we combine [120, 101] to create a stronger baseline. We use ground-truth preference labels, represented as a categorical distribution, and add them to the input features of the GNN, similar to [101]. We use the output of the GNN to make sequential predictions, as in [120]. Thus, by combining the two works, we create a GNN baseline that can act according to preference in a dishwasher-loading scenario. The GNN policy is trained on all preferences, including the 'unseen' preferences. **For unseen preferences, this is privileged information** that our approach does not have access to. We train the GNN policy using imitation learning (IL), and reinforcement learning (RL), following [120]. GNN-IL is trained from the same set of demonstrations

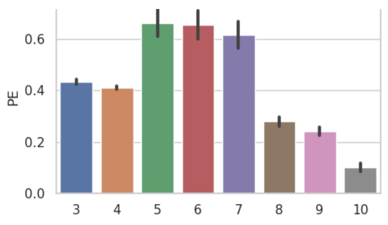


(A) Spatial Preference Adherence (SPA)

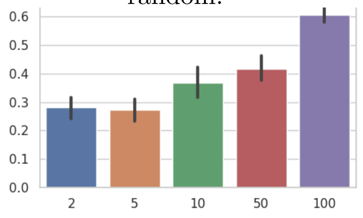


(B) Temporal Preference Adherence (TPA)

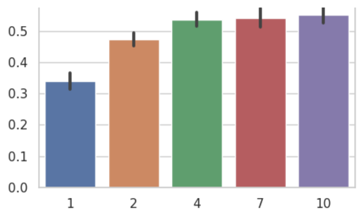
FIGURE 2.5: **Results** Comparisons of TTP, GNN-IL/RL and a Random policy in simulation across two metrics: SPA and TPA. TTP shows good performance at the task of dishwasher loading on seen and unseen preferences, and outperforms the GNN and random baselines. However TTP's generalization to unseen # objects is worse, but still close to GNN-IL (which was trained on *all* preferences), and better than GNN-RL and random.



(A) Performance decays for OOD number of objects, i.e 3-5 & 8-10



(B) Performance improves with the # of unique sessions in training



(C) Performance improves with the # of unique preferences in training

FIGURE 2.6: **Ablations for #objects, training trajectories and preferences**. (a) Out-of-distribution generalization to #objects leads to degradation in performance, and the effect is more severe when increasing more objects than reducing objects. (b) Performance improves significantly with increase in the number of trajectories per preference. (c) Performance roughly plateaus with more unique variants of preferences.

as TTP using behavior cloning (see Fig. A.6a). For GNN-RL, we use Proximal Policy Optimization [183] from [176]. GNN-RL learns from scratch by directly interacting with the dishwasher environment and obtaining a dense reward. For more details on baselines, including an alternate variant of GNN-IL, see Appendix A.3.

GNN-IL does not reach the same performance as TTP for in-distribution tasks (SPA of 0.34 for GNN-IL vs 0.62 for TTP). Note that unseen preferences are also in-distribution for GNN-IL since we provide ground-truth preference labels to the GNN. Hence, there is not a drop in performance for unseen preferences for GNN, unlike TTP. For detailed failure analysis, see Appendix A.4. Despite having no privileged information, TTP outperforms GNN-IL in unseen preferences and performs comparably on unseen #objects. Due to the significantly long time horizons per session (more than 30 steps), GNN-RL fails to learn a meaningful policy even after a large budget of 32,000 environment interactions and a dense reward (SPA 0.017 ± 0.002 using GNN-RL). Lastly, we find that the random policy RP is not able to solve the task at all due to the large state and action space. TTP is able to solve dishwasher loading using unseen preference prompts

TABLE 2.2: Attribute Ablations

Pose	Cat	Time	SPA
×	×	×	0.0
×	×	✓	0.0
×	✓	×	0.027
×	✓	✓	0.142
✓	×	×	0.411
✓	×	✓	0.419
✓	✓	×	0.517
✓	✓	✓	0.606

well (SPA 0.54). In contrast, classical task planners like [66] need to be adapted per new preference. This experiment shows that Transformers make adaptable task planners, using our proposed prompt-situation architecture. However, TTP's performance on unseen #objects deteriorates (SPA of 0.62 for seen versus 0.34 on unseen #objects) , and we look more closely at that next.

Generalization to unseen # objects: Fig.2.6a examines SPA on out-of-distribution sessions with lesser i.e. 3-5 or more i.e. 8-10 objects per rack. The training set consists of demonstrations with 6 or 7 objects per rack. The policy performs well on 5-7 objects, but poorer as we go further away from the training distribution. Poorer SPA is expected for larger number of objects, as the action space of the policy increases, and the policy is more likely to pick the wrong object type for a given preference. Poorer performance on 3-4 objects is caused by the policy closing the dishwasher early, as it has never seen this state during training. Training with richer datasets, and adding more randomization in the form of object masking might improve out-of-distribution performance of TTP.

Ablations

We study the sensitivity of our approach to training hyperparameters. First, we vary the number of training sessions per preference and study generalization to unseen scenarios of the same preference. Figure 2.6b shows that the performance of TTP improves as the number of demonstrations increases, indicating that our model might benefit from further training samples. Next, we vary the number of training preferences and evaluate generalization performance to unseen preferences. Figure 2.6c shows that the benefits of adding additional preferences beyond 4 are minor, and similar performance is observed when training from 4-7 preferences. This is an interesting result since one would assume that more preferences improve generalization to unseen preferences. But for the kinds of preferences considered in our problem, 4-7 preference types are enough for generalization.

Finally, we analyze which instance attributes are the most important for learning in an object-centric sequential decision-making task. We mask different instance attributes to remove sources of information from the instance tokens. In Table 2.2, all components of our instance tokens play a significant role (0.606 with all, versus the next highest of 0.517). The most important attribute is the pose of the objects (without the pose, top SPA is 0.142), followed by the category. The timestep is the least important, but the best SPA comes from combining all three. More ablations are in Appendix A.5.

2.4.2 Real-world dish-rearrangement

As a proof-of-concept, we demonstrate zero-shot transfer of our policy trained in simulation to robotic hardware, by assuming low-level controllers. We use a Franka Panda with a Robotiq 2F-85 gripper, controlled using the Polymetis control framework [121]. For perception, we use three Intel Realsense D435 RGBD cameras [106] whose outputs are combined using Open3D [236] and segmented [214] to generate 3D point clouds of visible objects, used to calculate object-centric attributes used by TTP. For low-level pick, we use a grasp candidate generator [60] applied to a chosen object's point cloud and use it to grasp the target object. 'Place' is approximated as a 'drop' action in a pre-defined location. More details are in Appendix A.1.

Our hardware setup mirrors our simulation, with different categories of dishware (bowls, cups, plates) on a table, and a "dishwasher" (cabinet with two drawers). The objective is to select an object to pick and place it into a drawer (rack) (see Fig. 2.1). We use a policy trained in simulation and apply it to a scene with four objects (2 bowls, 1 cup, 1 plate) using the hardware pipeline described above. We start by collecting a prompt human demonstration which is manually segmented into pick and place motions, and the environment state recorded at each timestep. The learned policy, conditioned on the prompt, is applied to two variations of the scene, and the predicted actions executed. The hardware success is defined in terms of pick only, shown in Fig. 2.1, since we do not reason about 'place' on hardware. The policy successful places all 4 objects in the correct order in the drawer once and 3 out of 4 objects in the second attempt. The hardware failure case was caused by a perception error; a bowl was classified as a cup. This demonstrates that TTP can be trained in simulation and applied directly to hardware, and the policy is robust to minor hardware errors, like noise in object location. It is also reactive to failure, such as if a bowl grasp fails, it just repeats the grasping action. However, it is sensitive to perception errors like failing to detect objects, or mis-classifying them. See Appendix A.6.

2.5 Limitation

While our task is complex by virtue of its strict sequential nature, it is incomplete as we assume doors and drawers can be easily opened and perception is perfect. In complex real settings, the policy needs to learn how to recover from its low-level execution mistakes. Moreover, preferences can have further nuanced differences based on visual or textural patterns on objects, for which the instance encoder needs to be modified to encode such attributes. Another important question to address is how more complex motion plans interact with or hinder the learning objective, due to different human and robot affordances. Finally, prompts are only presented via single demonstration, while language might be a more natural interface for users to instruct the robot.

2.6 Conclusions

We present Transformer Task Planner (TTP): a high-level, sequential, preference-based policy from a single demonstration using a prompt-situation architecture. We introduced a simulated dishwasher loading task with demonstrations that exhibit different preferences. TTP can solve this complex, long-horizon task in simulation and transfers to the real world. This is a first step towards learning to adapt to different users' preferences in terms of both ordering and spatial location.

Chapter 3

From pixels of videos to low-level actions for manipulation

Publication Information

Vidhi Jain, Maria Attarian, Nikhil J. Joshi, Ayzaan Wahid, Danny Driess, Quan Vuong, Pannag R. Sanketi, Pierre Sermanet, Stefan Welker, Christine Chan, Igor Gilitschenski, Yonatan Bisk, Debidatta Dwibedi. "Vid2robot: End-to-end video-conditioned policy learning with cross-attention transformers." In Proceedings of Robotics: Science and Systems (RSS) at Delft, Netherlands. July 2024.

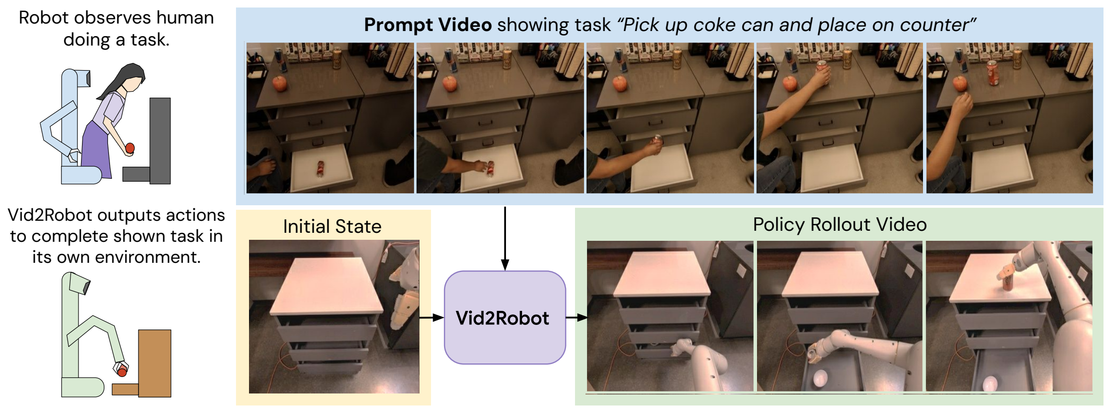

Robot observes human doing a task.
Prompt Video showing task "Pick up coke can and place on counter"
Vid2Robot outputs actions to complete shown task in its own environment.
Initial State
Vid2Robot
Policy Rollout Video

FIGURE 3.1: **Overview of Vid2Robot**, a video-conditioned robot policy. Given a human demonstration (top), Vid2Robot recognizes the task semantics and performs the same task based on the robot's current visual observation (bottom left). A successful trajectory is presented on the bottom right.

Large-scale multi-task robotic manipulation systems often rely on text to specify the task. In this work, we explore whether a robot can learn by observing humans. To do so, the robot must understand a person's intent and perform the inferred task despite differences in the embodiments and environments. We introduce Vid2Robot, an end-to-end video-conditioned policy that takes human videos demonstrating manipulation tasks as input and produces robot actions. Our model is trained with a large dataset of prompt video-robot trajectory pairs to learn unified representations of human and robot actions from videos. Vid2Robot uses cross-attention transformer layers between video features and the current robot state to produce the actions and perform the same task as shown in the video. We use auxiliary contrastive losses to align the prompt and robot video representations for better policies. We evaluate Vid2Robot on real-world robots and observe over 20% improvement over BC-Z when using human prompt videos. Further, we also show cross-object motion transfer ability that enables video-conditioned policies to transfer a motion observed on one object in the prompt video to another object in the robot's own environment. Videos available at vid2robot.github.io.

3.1 Introduction

The path to creating versatile robots that assist in people's daily routines requires them to learn new skills on-the-go. These skills can vary from simple preferences for arranging dishes in the dishwasher in a specific household to completely different approaches to household cleaning. Humans can communicate in natural language for tasks that are already known. However, we revert to demonstrations when we want to learn a novel skill with nuance. For example, we might show how a particular microwave works or how to organize our cabinets. Robots need the same ability for generalization from demonstration, which comes so naturally to humans.

Humans can infer the intentions of other humans based on third-person visual observations. Often, we use social reasoning and common sense to understand others' goals implicitly. This ability is crucial for learning everyday tasks, such as kneading dough or knitting, where the intricacies are challenging to convey through still images or text [14]. We often turn to How-To videos [138]) to learn how to perform such tasks. If robots could act based on videos, it would enable efficient task learning and improved interaction.

This work focuses on visual imitation learning, where robots learn to perform tasks by watching video demonstrations. This setup offers several advantages. First, it allows robots to learn from agents with a different embodiment, enabling new skill acquisition without teleoperation. Second, it allows robots to infer tasks from expert demonstrations, even if the expert is not showing how to perform tasks in the same environment as the robot. Finally, visual imitation learning is ideal for teaching tasks that are difficult or impossible to describe in words.

Existing multi-task robot manipulation models (e.g. RT-1 [19], RT-2 [241], and RT-X [150]) use language conditioning to output a robot trajectory. Relying on text alone for task specification makes it difficult for robots to handle polysemy and tasks whose executions vary dramatically based on context. For example, '*open* drawer', '*open* cabinet', '*open* container with lid' and '*open* jar with screw cap' might share the same verb but require very different motor control for each interaction. Here, the agent should not generalize its policy, whereas it should generalize from one *drawer* to others that vary in type, color, and shape. For this reason, there are a broad range of tasks for which it is hard to design primitives for high-level planning approaches [116, 11].

Another common approach has been to use a final goal image in goal-conditioned behavior cloning tasks [148, 110]. However, *how* to act is often ignored in such specifications. For example, 'hold the flag' and 'wave the flag' can have the same final goal image. We can resolve this ambiguity by using several sub-goal frames, that is quite close to conditioning robot policies with videos.

Current success of video conditioned policies in [186] assume that the provided video is from the same workspace with limited variability. Video-conditioned policies also lag in performance compared to language-conditioned policies work [96]. Based on these and related work, we identify three main challenges for video-conditioned policies: (1) Raw videos are high dimensional data that require more computational power and memory. (2) While unlabeled video data are abundant on the internet, finding robot-specific video and motion data is hard. (3) People can perform the same task differently due to variations in objects, lighting conditions, and other background distractions.

Despite these challenges, video-conditioned policy learning is a core challenge robots need to master. To this end, we study how end-to-end models with video-conditioning can be used to specify tasks to the robot. Vid2Robot is an end-to-end system that enables rapid adaptation to tasks specified as video demonstrations. Unlike prior work that either learned representations from videos for only object and verb recognition [96] or learned motor control in simulation [216], our work demonstrates the applicability of end-to-end learned video representations for real-world robotic control.

We present the key contributions of our work as follows: (i) We present a transformer-based policy to encode video task specification, demonstrated by either robot or human agent embodiments (§3.3). (ii) We encourage alignment between the prompt and robot video representations using three contrastive auxiliary losses during training (§3.3.3) (iii) Through actual robot experiments, we find our video-conditioned policy is better than baselines on human prompt videos (§3.4.1). Furthermore, our policy is better at cross-object motion transfer (§3.4.2).

3.2 Related Work

Task Specifications for Robots The development of general-purpose robots hinges on effectively grounding task specifications. Videos are a dense source of information that provides information about what to do and how to do it in the physical world. Recent works have used videos for task specification [13, 98, 186]. Another line of work uses videos to learn world models to predict future visual observations [137, 126, 24, 148, 54].

Recall our example of "open drawer", "open cabinet", and "open jar" in the §3.1. Video-conditioned policies like Vid2Robot are capable of doing these tasks because these policies identify the tasks of "open jar" and "open drawer" from visuals, unlike language-conditioned which have the same embedding of the verb 'open' for each task. Note that language is not an input to Vid2Robot. Therefore, the verb does not directly influence the action; this is a critical difference between Vid2Robot and existing language-conditioned robot policies. While language [224, 19, 150, 156], final goal images [110, 18], and others like hand-drawn inputs [197] have been proposed as means for task specification, learning from prompt videos is complementary to these approaches and inevitable for rapid adaptation of trained polices to perform new manipulation skills at deployment.

Learning from Human Demonstrations As videos of humans performing various tasks proliferate on the internet, several works aim to address how to leverage this information for robot learning. The difference in robot versus human embodiment poses a significant challenge, for which existing approaches range from translating the image of a human into the robot [194] to inpainting for agent-agnostic representations [12]. Many prior works propose to leverage off-the-shelf models for hand pose estimation and contact tracking [13, 48, 163], object-centric representations [165, 92], as well as reward functions for reinforcement learning [12, 129, 194].

XIRL [230], GraphIRL[112] and other RL approaches [182, 160] take a lot of time and manual effort for resetting scenes during the policy learning phase, limiting their applicability to real robots. Furthermore, RL often leads to unsafe situations with real-world robots. Other methods [149, 216, 13] cast this problem into visual representation learning to accelerate learning of downstream motor control tasks. While these modular learning solutions work well in limited datasets, they are prone to compounding errors in each component and are not efficiently scalable. End-to-end training approaches for goal-conditioned imitation learning [46, 187, 72, 51] are also largely limited in simulation and hindered by sim-to-real gap. In contrast, we tackle this as an end-to-end large multi-task learning from human videos with real robot evaluations.

Imitation via Paired Demonstrations Our setup of paired prompt videos and robot trajectory is most similar to the One-Shot Visual Imitation literature. Many prior works assume access to pairs, where the first video demonstrates the task, and the second

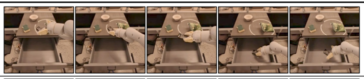
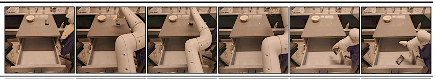
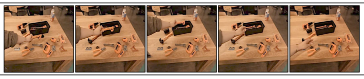
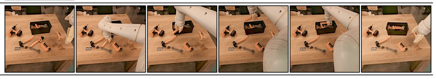
Dataset Name	Prompt Video Embodiment	Prompt v/s Robot Scene	Prompt Video	Robot Video
Robot-Robot	Robot	Different		
Hindsight Human-Robot	Human	Different		
Co-located Human- Robot	Human	Same		

FIGURE 3.2: **Dataset creation.** (top row) Here we show a Robot-Robot video pair for *placing the rxbar into top drawer*. We similarly pair existing robot-robot videos performing the same task. (middle row) Here, we show Hindsight Human-Robot paired videos for *picking a Coke can from the bottom drawer and placing it on the counter* task. We use the task instructions from robot trajectories, ask human participants to perform the task and record a demonstration video from the robot's perspective/view. (bottom row) Here, we show a Co-located Human-Robot pair of videos for *placing the pipe wrench in the toolkit*. We record a human demonstration and a robot teleoperation in the same workspace. We use different workspaces to perform the same task instruction, thus collecting paired videos with visually diverse prompts and robot state observations. More details in Section 3.3.1.

video shows the agent's visual observations. Some of the early works [55] proposed training a demonstration network via temporal convolution and neighborhood attention to condition a manipulation policy network. In more recent approaches like [46, 134, 92], paired demonstrations and observations are used to train a transformer policy, often with additional constraints like inverse dynamics prediction[46] or contrastive representation learning [134]. However, evaluating these approaches is usually limited to a specific set of simulated tasks and hardly to real robots.

BC-Z [96] is most similar to our work, which reports real robot evaluations. While our training setup has similarities with BC-Z, our model Vid2Robot couples large image encoders, cross-attention layers, and contrastive auxiliary losses to learn a manipulation policy that imitates a human showing a task. Recent approaches for self-supervised skill discovery like XSkill [221] learn skills from unpaired human and robot videos, while our approach uses text descriptions of the task to pair them explicitly. The paired human and robot videos contain different backgrounds, lighting, and object arrangements, thereby training the visual representations to be invariant to these settings, and focus on the task semantics instead.

3.3 Approach

Preliminaries Our objective is to design a robotic system that takes in a *prompt video* of a manipulation task and outputs actions that accomplish the task demonstrated in the video. This system must infer the underlying task from the prompt video (which might have a different setup or embodiment than the robot) and then manipulate the objects in its environment to achieve the inferred task. The input of our model is a prompt video V and the robot state $S_t = \{x_i\}_{i=t-k-1}^{t}$ where x_i is the frame from the robot's camera stream at time i, k is the maximum number of historical frames, and t is the current timestep We train a policy $\pi(a_t|S_t, V)$ that infers the underlying task from V and predicts task-relevant action a_t. We need a dataset of paired prompt videos and robot trajectories to train this model. Below, we will discuss in detail how to create paired datasets.

3.3.1 Datasets

We need a dataset of pairs to train a video-conditioned robot policy: prompt videos and robot trajectories that perform the same task. In this work, we explore prompt videos where humans and robots perform the task. To create this dataset, we rely on three classes of data:

1. **Robot-Robot**: We pair existing robot-robot videos of the same task. For this pairing, we consider two videos to match if they perform the same task in different settings. We define "*task*" based on natural language instructions for recording robot trajectories. These instructions typically consist of one or two verbs surrounded by nouns, such as '*place* water bottle upright', '*move* the coke can to the green chip bag' or '*open* top drawer'. Note that we use language instructions only for pairing and as an input to the robot policy. The objective of this pairing is two-fold: first, to take advantage of an already labeled and collected dataset of robot trajectories, and second, to ensure robots can imitate when the same task in a different environment.

2. **Hindsight Human-Robot**: Here, we use the task instructions from the robot trajectories dataset, ask one to five human participants to perform the task, and record a demonstration video from the robot's perspective/view. The instructions are the same as before, but there is a significant embodiment and speed variability due to different humans performing the task with left or right hands and at a randomized robot camera angle. This provides us with a lot of paired data for training the policy with the available set of instructions in the robot dataset, without having to collect new robot trajectories.

3. **Co-located Human-Robot**: In this case, humans and robots perform the same task in the same workspace. We used this approach to collect human demonstrations and robot trajectories in diverse spaces such as a living space with sofas, a meeting room with whiteboards, a hardware workstations with toy tools, a kitchen with countertop, a refrigerator and a sink, a storage supplies area, and more.

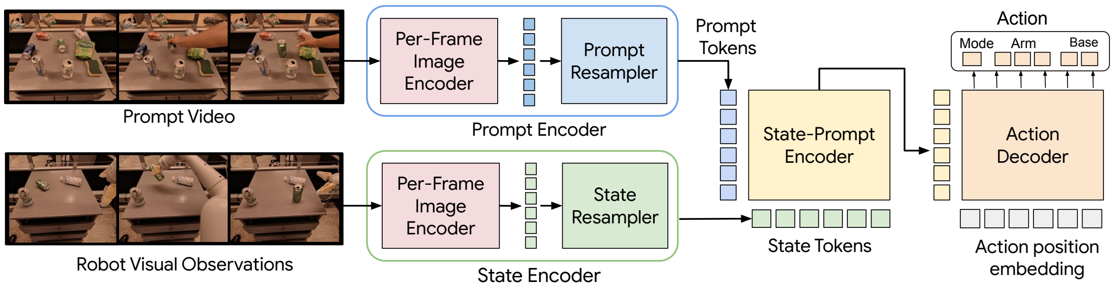


FIGURE 3.3: **Architecture.** Our model takes the prompt video and the robot's current observations as the input, encodes those into token embeddings for the prompt video and the robot's state, cross-attends to produce state-prompt encoding, and translates it into the expected robot action at the current timestep. More details in Section 3.3.2).

We show examples of paired videos of the prompt and robot demo from each of the three datasets in Figure 3.2. As can be seen, there is a considerable difference in the backgrounds and distractor objects in the Hindsight Human-Robot and Co-located Human-Robot datasets. A different complexity arises when comparing the first approach (Robot-Robot), where the actor is a robot with the same morphology, to the other two cases where the human is the actor in the prompt videos.

Each source represents a different level of difficulty and time to collect. Pairing existing robot datasets requires no extra data collection but does not involve any human demonstrations. Our second data involves asking humans to mimic existing robot trajectories. Hindsight human videos made data collection easier as they do not need robot teleoperation data. However, this did not increase the diversity of tasks in the data set. Lastly, we collect data with humans and robots in the same environment. While collecting co-located paired data is a presumed gold standard, it is very time and labor-intensive compared to the previous two approaches. Thus, it forms a small fraction of our overall training set. After combining all the datasets, we have $\sim$100k robot videos and $\sim$10k human videos covering the tasks introduced in RT-1 [19] and RT-2 [241]. The robot-robot dataset makes up more than 90% of the entire dataset. This dataset is publicly available [150]. We provide a Python Notebook in Supplementary Material for accessible visualization of the paired videos used in training.

3.3.2 Model Architecture

Our policy takes the prompt video and the current robot state as inputs and outputs robot actions. It consists of four modules: (1) prompt video encoder, (2) robot state encoder, (3) state-prompt encoder, and (4) robot action decoder. The entire architecture is illustrated in Figure 3.3, and each of the modules is detailed below:

(1) Prompt Video Encoder encodes the video demonstration provided as a reference to convey the desired task semantics. The prompt video encoder implicitly learns to infer what task to perform and how to do it. The prompt encoder consists of a per-frame Image encoder ϕ_p (ViT [52]) followed by a Perceiver Resampler [6, 86] ψ_p. The output of the prompt encoder $\psi_p(\phi_p(V)) = z_{prompt}$ is a set of N tokens of d-dimension to condition the policy with the task-relevant attributes from the video.

(2) Robot State Encoder encodes the current state of the robot given the current frame and last k frames as input. Note that this module also encodes information about the objects and environment visible to the robot. The architecture is similar to the prompt encoder: a per-frame Image encoder ϕ_s followed by a Perceiver Resampler ψ_s. Identical to the prompt encoder's outputs, the output of the state encoder is $\psi_s(\phi_s(S_t)) = z_{state}$ that encodes the latent environment and robot state information from the history of recent observations. We use the same image encoder weights for both (1) and (2), that is, $\phi_p = \phi_s = \phi$. The role of the image encoder ϕ is to capture spatial visual information in each frame. The perceiver resampler enables temporal learning across frames and reduces the number of video tokens passed into the action decoder.

(3) State-Prompt Encoder takes the prompt video encoding z_{prompt} and robot state encoding z_{state} and outputs a task encoding relevant for action prediction, which we call prompt-aware state tokens $z_{state|prompt}$. Here, the state encoding acts as queries, and the prompt video encoding acts as keys and values. Intuitively, the state-prompt encoder enables the fusion of the state and prompt information. Suppose a prompt video shows picking up an apple in the basket, and the current state contains an apple, a banana, and an orange. The State-Prompt Encoder cross-attends and learns which object to attend to in the state based on the prompt video. Capturing interdependencies between prompt and state is crucial for the next step of action decoding.

(4) Robot Action Decoder predicts the action vector a_t for the current state S_t such that it completes the task shown in the prompt video V_p. The action decoder is a transformer decoder architecture that uses the fixed action position tokens [235] as input queries and the prompt-aware state tokens $z_{state|prompt}$ for keys and values. The size of the action position embedding is $N \times d$ where N is the number of action dimensions, and d is the transformer embedding dimension. More details on action vector in §3.3.2.

The action position embeddings cross-attend to the prompt-aware state tokens to predict the target binned action values as output. Each output token of the action decoder corresponds to an action dimension for the mode, arm, and base. We project each token embedding to 256 dimensions, and a softmax layer is applied on the top to obtain the bin corresponding to the target action vector. Unlike prior work [19, 241] that use autoregressive action decoding that requires multiple forward passes during inference, we use action position embeddings for one forward pass prediction like in ACT [235].

Instead of predicting one action for the next timestep, we follow the approach outlined in [96, 235] and train the policy with a prediction horizon of four steps. We always use the action bin with the highest probability, that is, `argmax` over predicted probabilities, to choose the action value for execution.

Cross-Attention Layers. In the Vid2Robotrchitecture, we use Cross-Attention Transformer layers extensively in the following modules: Prompt Resampler, State Resampler, State-Prompt Encoder, and Action Decoder. Compared to standard self-attention layers, which require more memory to process the same video, cross-attention layers help manage the high number of tokens and the resulting large attention matrices when processing prompt and robot state videos. For example, when using ViT-B/16, the total number of video tokens for a 16 frame reference video and a 8 frame robot state video at 224×224 resolution would be $8 \times 196 + 16 \times 196 = 4704$. An entire self-attention operation would lead to an attention matrix with $4704^2 \sim 22$M entries. However, using two Perceiver Resamplers with 64 latent tokens, we train with attention matrices of the size $8 \times 196 \times 64 + 16 \times 196 \times 64 \sim .3$M. Thus, cross-attention layers in Vid2Roboteduce attention computation and enable training with paired videos.

Preprocessing To efficiently train videos of varying lengths, we randomly sample N=16 frames. We always include the first and last frames and sort them in increasing order of time. We sample a robot state S_t during training by sampling a random timestep. We then select the preceding $k-1$ frames to create a robot state video comprising a total of k=8 frames before. If there are less than $k-1$ frames before the current time step, we repeat the first frame to create a fixed-size robot state video. We normalize the pixel values in each frame between 0 and 1 and resize each frame to $(224, 224)$. During training, we apply photometric distortions like cropping, brightness, contrast, hue, and saturation.

The action at that time consists of the three components: *Mode*: (m) whether to terminate episode, move only arm, move only base, or both. *arm*: gripper position (x, y, z), orientation (rotation along xy, yz, zx), and the degree of closedness (c). *Base*: displacement (x, y) and rotation. Overall, the action $a_t = [m, g_x, g_y, g_z, \theta_{xy}, \theta_{yz}, \theta_{zx}, c, b_x, b_y, b_\theta]$ is an 11-dimensional vector. Each value has different ranges, which we first use to scale the values between 0 and 1. We further discretize the values into 256 bins each. In this study, we train and evaluate scenarios where the base remains stationary.

3.3.3 Training

Action Prediction Loss. We train Vid2Robot end-to-end with behavior cloning. We use a classification loss on actions discretized and tokenized into N=256 bins. Given the robot trajectory for performing a task with current visual observations x_t, we have the

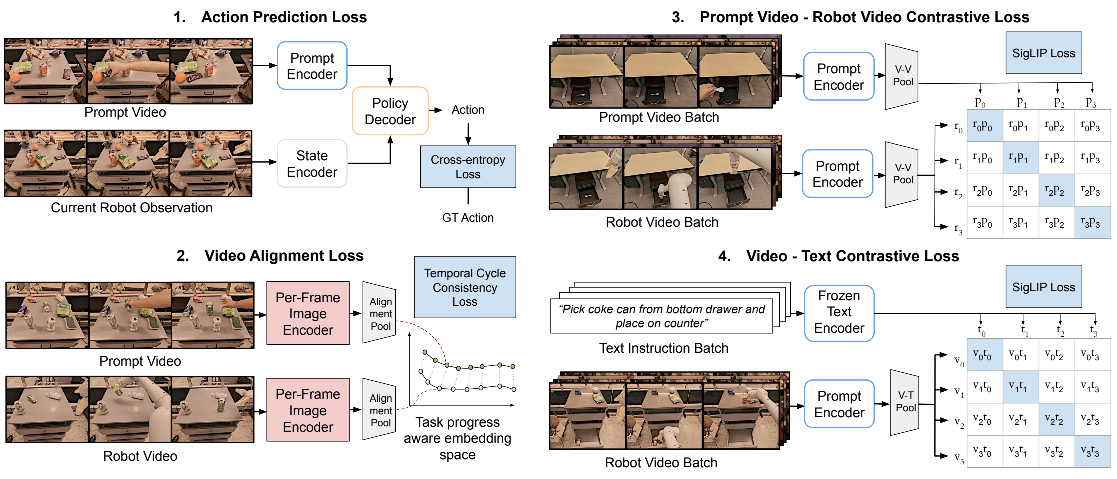


FIGURE 3.4: **Training Setup.** We show all the losses used for training Vid2Robot, particularly how each loss connects to its different modules. Along with (1) the main action prediction loss, we apply three auxiliary losses: (2) temporal video alignment loss, (3) a contrastive loss between the prompt and robot video performing the same task, and (4) a contrastive loss between a prompt/robot video with the language embedding. More details are in Section 3.3.3.

corresponding expert action a_t. The action prediction loss is Cross Entropy between the predicted action and the expert action as: $L_{CE}(a_t, \hat{a}_t) = \sum_{\tau} a_t \log \hat{a}_t$. This loss trains all the model parameters, as shown in Fig 3.3.

Auxiliary Losses. Although our dataset size is substantial, it is insufficient for training large transformer-based models. To prevent over-fitting on the training set, we add three auxiliary losses to learn features that understand semantics in the prompt videos.

Video Alignment Loss: We want to encourage temporal alignment between prompt videos and robot videos performing that show the same task. By aligning prompt videos with the robot videos, we want the image encoder to learn to be invariant to different embodiments, lighting, backgrounds, view angles, and distractor objects while still encoding features relevant to predicting task progress.

Our choice of loss is the temporal-cycle consistency loss introduced in [58]. This loss can encode the task progress when trained on videos of different agents performing the same task [230]. This loss is applied on per-frame image embeddings of the prompt V_p and robot V_r videos during training. To apply the loss, we average pool the per-frame embeddings output in spatial dimensions from image encoder ϕ and apply a projector head of 2-layer MLP [36]. We call this as *alignment pooling layer* Φ on the per-frame image embeddings, as shown in Fig 3.4. For each video V_i, this results in a sequence of embeddings $E_i = \{\Phi(v_i^1), \Phi(v_i^2), ...\Phi(v_i^{L_i})\}$, where L_i is the number of frames in the i^{th} video. We apply TCC loss on encoding E_p and E_r for prompt and robot video, respectively. Intuitively, the TCC loss ensures that the representation of every frame of

E_p should correspond to E_r and vice versa. Applying TCC in Vid2Robot involves two steps: First, we compute soft neighbor of t^{th} frame of E_p in E_r and call it $\widetilde{E^t_{pr}}$.

$$\widetilde{E^t_{pr}} = \sum_{k}^{L_r} \alpha_k E^k_r, \quad \text{where} \quad \alpha_k = \frac{e^{-\|E^t_i - E^k_j\|^2}}{\sum_k^{L_j} e^{-\|E^t_i - E^k_j\|^2}} \tag{3.1}$$

Second, we find the corresponding frame for this newly computed soft-neighbor in E_p. This is called *cycle-back* in [58] and it involves similar soft-neighbour computation as in Equation 3.1 to obtain say $\widehat{E^t_{pr}}$, which ideally should be same as t, that is, $(\widehat{E^t_{pr}} - t)^2$ should be minimized. TCC loss minimizes such mean squared error between all frames for prompt and robot video encodings, and vice-versa, that is,

$$\begin{aligned} L_{TCC}(E_p, E_r) &= \sum_{t \in V_p} (\widehat{E^t_{pr}} - t)^2 \\ L_{TCC} &= \frac{L_{TCC}(E_p, E_r) + L_{TCC}(E_r, E_p)}{2} \end{aligned} \tag{3.2}$$

Prompt-Robot Video Contrastive Loss (VVCL): We want to encourage the prompt encodings to learn task semantics from video tokens in a self-supervised manner. While we pair prompt and robot video using natural language, this does not effectively capture the visual similarity of low-level motions like reaching for objects and rotating the robot arm. For this, we apply contrastive loss between the latent features of the robot and the prompt videos. We use an Attention Pooling layer to merge features from the N prompt tokens to produce a single embedding for each video. We apply the SigLIP [233] loss between video-video pairs to encourage videos showing the same task, involving similar motions and interacting objects, to be close to each other while being away from other videos in the batch. A batch contains the same number of robot and prompt videos, say B. We use the prompt encoder $\psi_p(\phi(\cdot))$ to obtain a batch of full robot video embeddings Z_{robot} and prompt video embeddings Z_{prompt}, each of size $B \times d$. We multiply them, $Z_{robot} \cdot Z^T_{prompt}$ to obtain a $B \times B$ matrix. Adding a learnable temperature τ and bias b, we have our logit matrix as $\hat{Y} = (Z_{robot} \cdot Z^T_{prompt}) * \tau + b$. We consider the videos of robot and prompt performing the same task as positives and assign them a label of 1 along the diagonal and -1 for off-diagonal pairs, that is, the label matrix $Y = 2\mathrm{I}_B - 1$. SigLIP loss is the negative loglikelihood $\sigma'(Z_1, Z_2) = -\sum \log \sigma(Y \cdot (Z_1 \cdot Z_2^T) * t + b)$. The video-video contrastive loss is as follows:

$$L_{VVCL} = \sigma'(Z_{prompt}, Z_{robot}) \tag{3.3}$$

Video-text Contrastive Loss (VTCL): We want to encourage a part of the embedding space to be aware of object names and verbs, as shown in the prompt and the robot videos. We apply a contrastive loss between prompt tokens produced by the robot video

TABLE 3.1: Task Success Rate for Robot and Human prompts.

Prompter	Model	*pick*	*pick-place on*	*place into*	*open*	*close*	*move near*	*knock over*	*place upright*	Overall
Robot	BC-Z	75.0%	50.0%	**61.5%**	16.7%	66.7%	**44.0%**	**58.3%**	**50.0%**	52.6%
	Vid2Robot	75.0%	**58.8%**	50.0%	**91.7%**	**100.0%**	33.3%	41.7%	16.7%	**54.9%**
Human	BC-Z	50.0%	12.5%	12.5%	0.0%	50.0%	43.8%	12.5%	**50.0%**	30.6%
	Vid2Robot	**100.0%**	**50.0%**	**50.0%**	**62.5%**	**87.5%**	**43.8%**	**25.0%**	12.5%	**52.8%**

and the text instructions of the task. A version of this loss has been applied before by BC-Z [96] as auxiliary language regression loss. We use an Attention Pooling layer [227] with one latent query to merge features from the N prompt tokens to produce a single embedding for each video. We retrieve B pairs of video and text embeddings as a batch. Similar to Equation 3.3, we apply SigLIP [233] loss as L_{VTCL} to encourage every video to have similar embeddings to their textual description embeddings, and be different from other text embeddings in the batch.

$$L_{VTCL} = (\sigma'(Z_{prompt}, Z_{text}) + \sigma'(Z_{robot}, Z_{text}))/2 \tag{3.4}$$

Overall, we apply the mean of all four losses for training, that is,

$$L = \frac{1}{4}(L_{CE} + L_{TCC} + L_{VVCL} + L_{VTCL})$$

.

Implementation We trained the model (implemented in Jax) for 200K iterations. We use AdamW optimizer with an initial learning rate of 8e-5 using a cosine learning rate schedule with warmup steps 2,000 and a final learning rate of 1e-6. We use 2 Perceiver Resampler layers with 64 latent tokens for both the Prompt and State Resamplers. Both state-prompt encoder and action decoder are 4-layer deep cross-attention transformers.

3.4 Experiments

We present results with real robot evaluations for our multi-task video-conditioned policy. One of the fundamental questions we tackle in this work is how well robots can imitate humans performing manipulation tasks. Because of differences in embodiments, humans perform manipulation tasks at a different speed and style. We study the effect of using robots and human videos as prompts.

Metrics. We refer to a *rollout* as a sequence of actions inferred from the policy and executed on the robot from an initial state observation and prompt video until the policy terminates or takes the maximum number of steps, whichever is lower.

We define *success* for a rollout when the policy executes the task instruction as shown in the prompt video. A successful rollout involves correct actions to be taken successively in

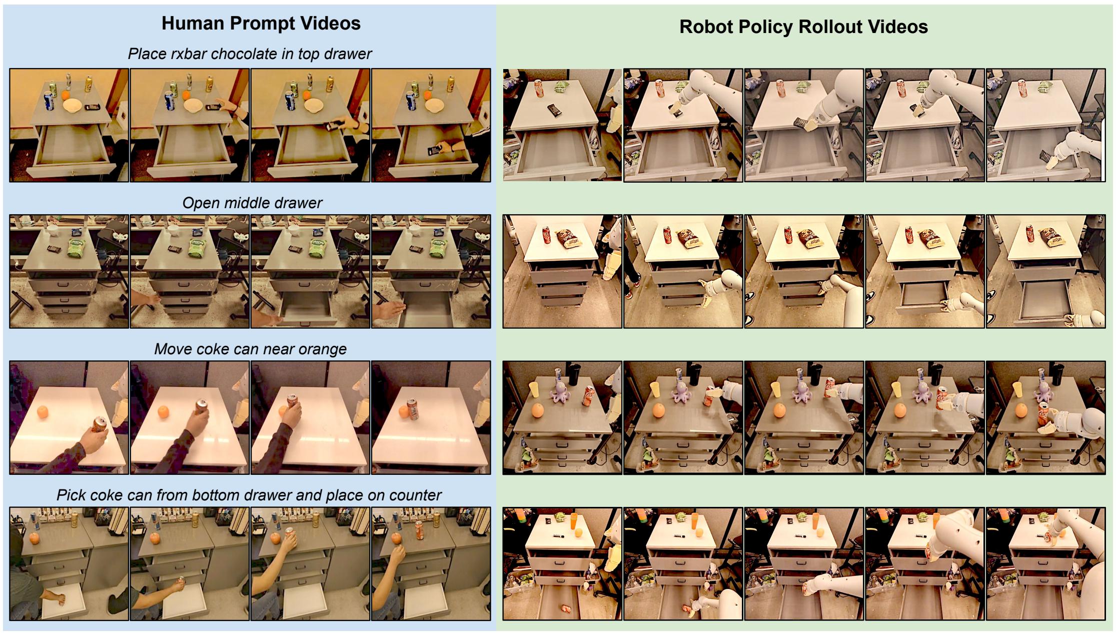

Human Prompt Videos
Robot Policy Rollout Videos
Place rxbar chocolate in top drawer
Open middle drawer
Move coke can near orange
Pick coke can from bottom drawer and place on counter

FIGURE 3.5: **Policy Rollouts.** Each row shows a prompt video of a human doing a task on the left, and on the right, we show the corresponding successful robot rollouts using Vid2Robot. Note how visually different the prompts are, while the policy rollouts have different lighting and backgrounds, as well as the number and placement of the distractor objects.

the environment without any assistance for resets or recovery. We ask a human evaluator to observe whether: (1) whether the robot *reached* the correct location?, (2) *grasped* the correct object?, (3) *released* the object at the correct location?, and (4) *terminated* the task correctly?

If the answer to any question is "no", the answer to subsequent questions is assumed to be "no". If all questions are answered "yes", only then the rollout is considered "successful". For each task instruction, we record a few rollouts per policy with different distractor objects, background, and lighting conditions. We take the average success recorded across all the rollouts for a task and call it that task's *Success Rate*. We also report aggregated success rate across tasks as *Overall* Success Rate. We also analyze the partial success across the four milestones in § 3.4.1.

We looked into automating success detection with concrete criteria or decision rules for evaluation but found that this automation can overlook the *process* of performing the task. Consider the task of "knocking the water bottle over". *Case 1:* The robot successfully grasps the bottle, turns it, and places it on the table. *Case 2:* The robot fails to grasp, pinches the bottle away, and lands in a knocked-down orientation. While Case 1 is desired and expected behavior according to the training data, Case 2 is a failure as it is unintended. With rule-based verification of the final state, we would have deemed both Case 1 and 2 successful. With human evaluators, we focus on the entire process of achieving the task instead of the final state.

Setup. We evaluate the policies by varying the object placement, lighting conditions, background surfaces, and distractor objects. When evaluating a set of policies, we ensure comparable initial object configurations for rollouts. We randomize the initial state only after all policies' rollouts have been performed. For all rollouts, we sample prompt videos not seen during training. In all the experiments, we use a mobile manipulator, the Google Robot. It has an ego-centric camera view, a single arm with seven degrees of freedom, and a two-fingered soft gripper. Refer to [19] for more details.

Baselines. We compare our model with BC-Z [96], a video-conditioned policy using a ResNet-18 encoder. BC-Z [96] processes demonstration-observation pairs via a FiLM [161] conditioned ResNet encoder and feds into a ResNet-based policy network to predict robot actions. We use the same data to train the BC-Z and Vid2Robot for a fair comparison. BC-Z does not have a terminate action, so we run these rollouts for a fixed maximum number of steps.

Key Questions. We address the following questions in this work:

- What is the gap in success due to prompt embodiment (robot vs. human)? (§3.4.1)
- How do video-conditioned policies perform with unseen task videos? (Fig 3.5, §3.4.1)
- Is Vid2Robot's overall success significantly better than BC-Z baseline? (§3.4.1)
- Can the learned motion representations handle out-of-distribution objects? (§3.4.2)

3.4.1 Task-based success

We compare our Vid2Robot model and baseline BC-Z with robot- and human-performed prompt videos in Table 3.1. We train both Vid2Robot and BC-Z on the same data mixture containing robot-robot and human-robot paired data. The prompt videos cover a subset of the training tasks. However, the videos are unseen by the models. In this evaluation, we investigate each model's ability to infer the task specification from the prompt video as well as the current observed state of the robot.

To test the model's capabilities in different settings on real robots, we assess rollouts on the following nine tasks: 'knock water bottle over', 'move rxbar chocolate near coke can', 'move green jalapeno chip bag near coke can', 'pick green rice chip bag', 'place coke can upright', 'pick coke can from bottom drawer and place on counter', 'open middle drawer', 'close middle drawer', and 'place apple into top drawer'.

We ask four evaluators to carry out two rollouts per task for a prompt video dataset and policy setting (a row in Table 3.1). Overall, we have eight trials per task to evaluate a policy's task success rate. We report an overall success rate per row over nine tasks with eight trials per task, that is, $9 \times 8 = 72$ trials. In total, we required $72 \times 4 = 288$ rollouts for Table 3.1.

What is the gap in success rate due to embodiment difference in prompt videos? When prompted with robot and human videos, we compare our model with BC-Z, which is a strong baseline for our comparisons. The overall success rate of our model Vid2Robot outperforms BC-Z for Human prompt videos by 20%, and is comparable for Robot prompt videos. Note that there is an order of magnitude more training samples for robot trajectories than human videos in our training mixture. Hence, there isn't a significant gap in performance for robot prompt videos. Our model outperforms BC-Z in most tas for human prompt video, showing that Vid2Robot captures the task semantics from prompt videos better than the baseline. Our model outperforms in tasks like picking something from a drawer, placing it on the counter, and opening/closing drawers by a large margin. The most challenging task is *placing upright* and *knocking over*. We analyze the failure reasons in §3.5 Fig 3.9.

How well do video-conditioned policies perform when shown a task in an unseen video? In addition to marking a rollout as a success, we recorded partial success annotations per rollout. In Fig 3.6, we observe that our model *reaches* to the correct object 78%, about 8% more than baseline. The policies sometimes fail to get the correct object and go towards a distractor instead. Next, *grasping* errors happen, particularly with small and deformable objects and collision-prone areas like drawer handles or counter's edges. Here, our model (65%) outperforms BC-Z (45%) by a large margin of 20%. A successful grasp is often the most challenging part of a rollout and crucial for success. After grasping, most tasks require *releasing* at a correct location. Both models slightly drop in success rate due to incorrect *release* during the rollouts. While BC-Z runs for a fixed number of steps, our policy Vid2Robot predicts when to terminate. We observe that the rate of *release* and *terminate* is almost identical, about 57% for our model, which implies that after releasing at the correct location, Vid2Robot mostly terminates successfully.

Tasks with More Rollouts To comment on the statistical significance of our results, we conducted more trials while limiting the evaluation to two tasks, namely 'place coke can upright' and 'close middle drawer' for real robot policy evaluations, and reported mean success rate with confidence intervals. In total, we conducted 314 real robot rollouts for results reported in Table 3.2.

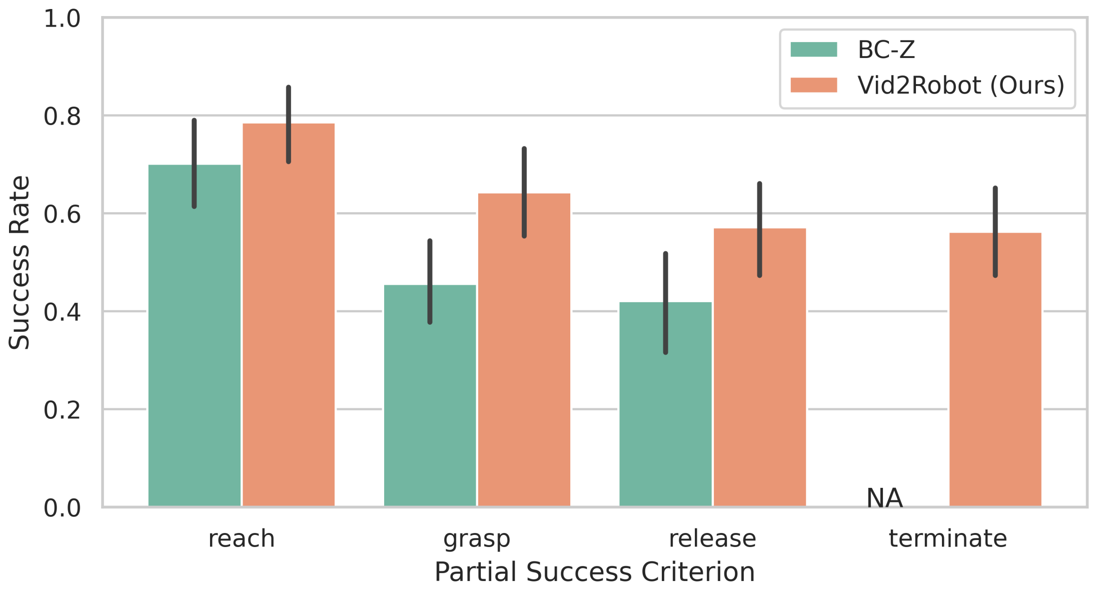


FIGURE 3.6: **Partial Success Rate for BC-Z and Vid2Robot.** Our policy Vid2Robot outperforms BC-Z in terms of *reaching* the correct object, *grasping* it, *releasing* it at the correct location and then *terminating* the episode correctly. Note that BC-Z does not have terminate control.

TABLE 3.2: Real Robot Evaluation of Vid2Robot and BC-Z with more trials to ascertain the statistical significance of the results.

Model	place coke can upright	close middle drawer	Overall
BC-Z	19.4 ± 9.5%	39.2 ± 10.8%	30.2 ± 7.1%
Vid2Robot	**39.1** ± 9.9%	**48.7** ± 11.2%	**43.4** ± 7.2%

3.4.2 Cross-object motion transfer

We trained our Vid2Robot and baseline with paired videos as discussed in § 3.3.1. Due to the pairing, the training data included only those scenarios where the interaction object shown in the prompt is present in the current robot observations. But *what if we provided a prompt video of one object and tested on other objects? Does it make the same motion as shown in the prompt video?* Interestingly, we found our model to perform learned manipulation actions on objects not seen in the prompt video. We call this emergent behavior as *cross-object motion transfer.*

We compare Vid2Robot with BC-Z for cross-object motion transfer ability with five prompt videos, namely, 'knock water bottle over', 'pick green rice chip bag', 'place coke can upright', 'pick coke can from bottom drawer and place on counter', and 'place apple into top drawer'. We evaluate each case of a prompt video by placing unrelated objects in robot's initial observation. The objects used for evaluation are *'orange', 'green*

Table 3.3: Cross-object motion transfer success.

Model	*pick*	*pick-place on*	*place into*	*place upright*	*knock over*	Overall
BC-Z	45.8%	0.0%	29.2%	12.5%	0.0%	17.5%
Vid2Robot	45.8%	**25.0%**	**54.2%**	**16.7%**	**29.2%**	**34.2%**

can', 'chips bag', 'banana', 'pink piggy soft toy', 'wrist watch'. We selected objects with different shapes, sizes, and deformability to evaluate situations requiring different grasps for success.

The evaluation setup is similar to § 3.4.1. Here, the evaluator sets up one of the objects for a task and records rollouts for each model. We compare two models on five tasks with six objects, so every evaluator runs $2 \times 5 \times 6 = 60$ rollouts. We repeat the evaluation with four raters, thus reporting results in Table 3.3 on $4 \times 60 = 240$ rollouts.

In Fig 3.7, we show the above experimental setup qualitatively. We use a prompt video to 'place coke can upright' and observe that the policy can transfer the action of 'placing upright' to objects, like a green can, a chips bag, a stapler, and a soft toy. The policy shows an implicit notion of learned pragmatics, by selecting green can over other objects.

In Table 3.3, we observe that BC-Z is often unable to complete the tasks when testing cross-object motion transfer. In contrast, our model (34%) performs better than BC-Z (17%) in this setting and performs the motion indicated in the prompt video. Our model is comparable to BC-Z with a 45% success rate on *picking* out-of-distribution objects. More importantly, tasks involving placing into drawers demonstrate significant improvement (29% $\rightarrow$ 54%). For specific tasks like *picking from drawers*, *placing on counters*, and *knocking over*, Vid2Robot completes the task 25% – 29% of the time, whereas BC-Z is unable to perform.

3.4.3 Ablations

We analyze our proposed approach to understand the following: (a) Can the policy learn possible interactions with the environment from state observations alone, without needing prompt videos? (b) How do the auxiliary loss functions impact the model's performance?

What is the impact of the prompt for task inference? First, we motivate the importance of prompt videos with an example. Consider a Coke can and other objects on the countertop as the robot's state observation. If a Coke can exists in the robot's view, it is hard to infer whether the task is to "pick a Coke can", "move a Coke can close to a chocolate bar", "move a Coke can near an orange can," or "knock over Coke can". Furthermore, once the robot starts taking action, it can end up in new states, like being close to rxbar, which make it especially difficult to predict tasks from the current state only. Below, we empirically measure the success rate of a policy that does not have access to a suitable prompt video.

We evaluated the "no-prompt" case, in which both models see blank frames as input prompt videos. In this setup, we evaluated three tasks. For each task, we rolled out the policy 20 times. In total, we ran $2 \times 20 \times 3 = 120$ actual robot rollouts for this experiment. Here, the success rate is 23% for Vid2Robot and 5% for BC-Z over 20 rollouts per task per policy. We find that performance improves when we condition the policies on the prompt videos. The success rate improves from 5% to 52.6% for BC-Z and from 23% to 54.6% for Vid2Robot. (Refer Table 3.1). This experiment underlines the importance of prompt videos for task success.

What is the role of auxilliary losses? Second, we analyze the role of additional loss functions in the overall success rate. In § 3.3.3, we presented action prediction loss and three auxiliary losses. We investigate the impact of (1) not using any auxiliary loss and (2) adding auxiliary language loss. We consider the tasks similar to those described in § 3.4.1, 9 tasks for evaluating each policy. We have 3 model variants: the original Vid2Robot, the one without video-text contrastive loss (CL), and the one with only action prediction loss. We ask three human evaluators to run a model variant for two rollouts each. In total, we report results with $3 \times 3 \times 9 \times 2 = 162$ rollouts in Fig 3.8. The error bars show the standard deviation for success.

What is the impact of not using any auxiliary loss?

We observe that the performance of our model (61%) improves significantly by enforcing representation constraints through auxiliary losses, compared to using only action prediction loss (45%). It highlights the importance of the proposed auxiliary losses in § 3.3.3.

What is the impact of the auxiliary language loss?

BC-Z proposed to use language representations to improve video representations for conditioning the policy. We compare our policy with another variant trained with all losses but the Video-Text CL. We observe only a borderline improvement of 1-2% in the success rate when using language loss. This implies that video alignment and video contrastive loss contribute significantly towards performance improvement.

3.5 Limitations and Future Directions

In Sec 3.4, we show that our approach has improved over previous work but there is a gap in performance for video-conditioned policies. Below we discuss the limitations of our work and provide insights for the future.

First, we qualitatively investigate some reasons for the failure of a policy rollout. In Fig 3.9, we illustrate and explain three examples of how self-occlusion, grasping errors, and the presence of distractors can lead to failure during any rollout.

Second, we observe a significant drop in the grasping success in Figure 3.6. While we use robot camera observation to estimate the state and implicitly learn depth estimation, it is often incomplete when occlusion or the robot gripper is out of camera view. Enhancing the state information with multimodal sensor fusion may improve the grasp success rate.

Third, we consider carefully collected short task instruction demonstrations from three different sources as shown in Section 3.3.1, all of which are 5 to 20-second videos. To test our models on long-horizon demonstrations or 'in-the-wild' videos online, we need effective pairing strategies for videos and robot trajectories to train the policy.

3.6 Conclusion

We present Vid2Robot, an end-to-end video-conditioned robot policy. Our proposed system trains on paired videos such that both videos demonstrate the same task but differ in diverse settings of lighting, background, and distractor objects. We use cross-attention (i) to learn the joint latent representations from prompt and state encodings and then (ii) to decode the action. We train the entire model for action prediction with cross-entropy loss and three auxiliary losses that encourage learning of generalizable latent representations to infer tasks directly from raw pixels to suitable actions. Vid2Robot outperforms BC-Z by over ∼20% when prompted with human videos. Further, Vid2Robot outperforms BC-Z by ∼17% for cross-object motion transfer; that is, if the prompt video didn't have the exact object as the object the robot is manipulating now, the model still produces valid actions for the same verb but different objects. Cross-object motion transfer is a promising direction for further extending pragmatic transfer learning of motion to new objects. We hope Vid2Robot enables bootstrapping data collection and human-robot interaction with rapid adaptation to new skills.

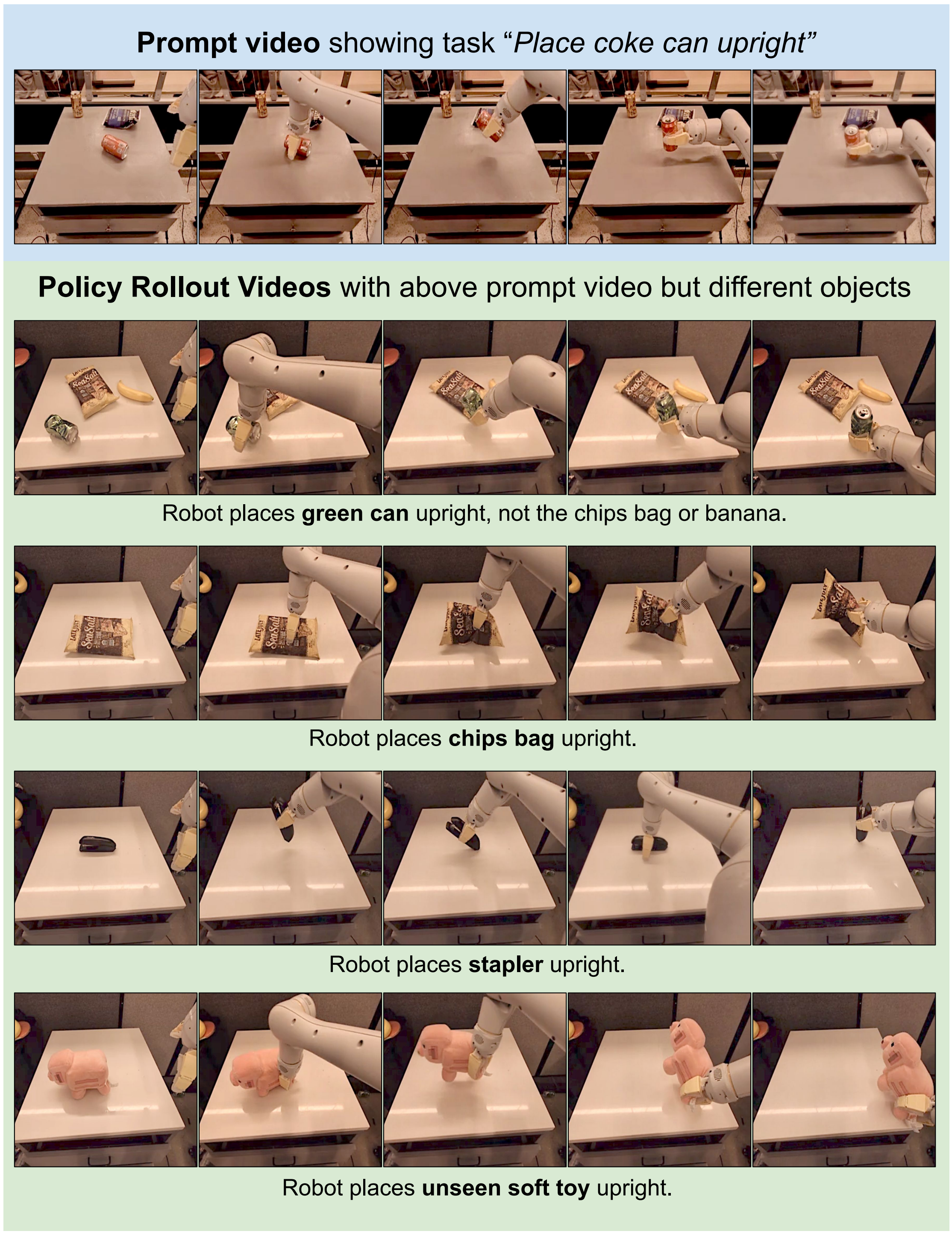

Prompt video showing task *"Place coke can upright"*
Policy Rollout Videos with above prompt video but different objects
Robot places **green can** upright, not the chips bag or banana.
Robot places **chips bag** upright.
Robot places **stapler** upright.
Robot places **unseen soft toy** upright.

FIGURE 3.7: **Qualitative results for *cross-object motion transfer*.** (Top-blue) Prompt video of *placing coke can upright*; (Green) Policy rollouts with a *green can*, *chips bag*, *stapler* and a *soft toy* in front of the robot. Vid2Robot infers the motion of *place upright* in the prompt video and applies it to other objects. The policy adheres to the prompt video by picking the green can instead of the chips bag or banana.

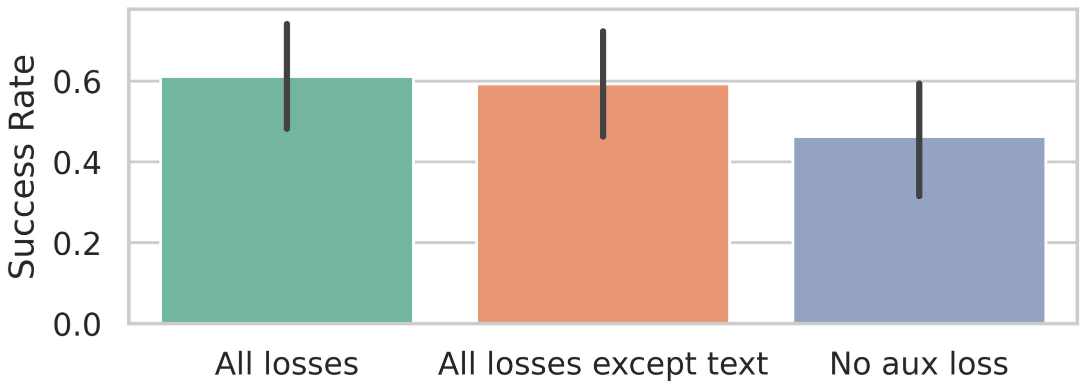


Figure 3.8: **Ablation for auxilliary losses used in Vid2Robot.** We compare our proposed approach that has all auxiliary losses (green, left) with a variant without language contrastive loss that was originally proposed in BC-Z (orange, middle) and a version with no auxilliary losses (blue, right). More details in (§3.4.3)

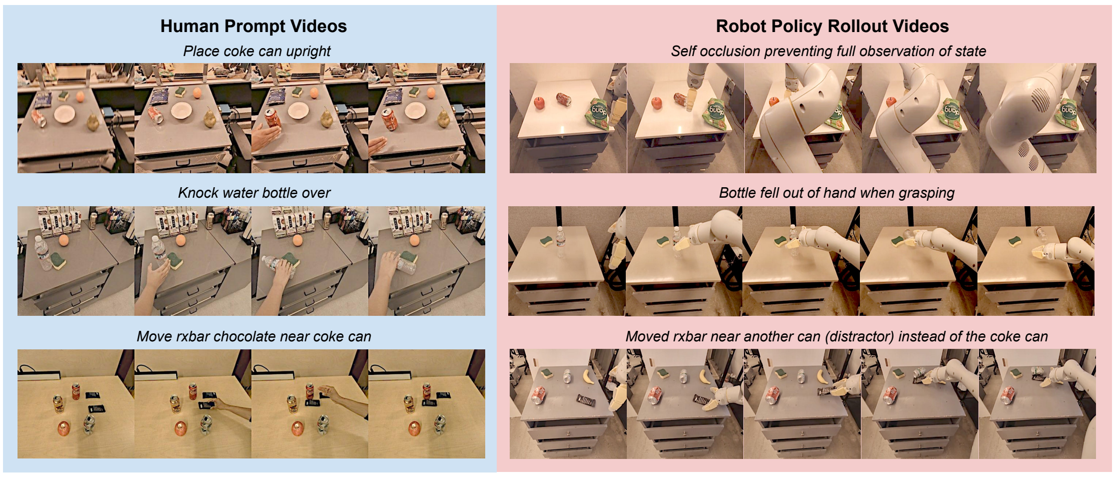

Human Prompt Videos
Place coke can upright
Knock water bottle over
Move rxbar chocolate near coke can
Robot Policy Rollout Videos
Self occlusion preventing full observation of state
Bottle fell out of hand when grasping
Moved rxbar near another can (distractor) instead of the coke can

Figure 3.9: **Failure analysis with policy rollouts.** (Top) Policy predicts gripper pose and depends on the IK solver to move the arm. Sometimes, the IK solution can block the robot's camera view. (Middle) Grasping failures happen, especially with transparent and deformable objects. (Bottom) Distractor objects and differences in lighting and background may cause recognition errors, where policy might perform the correct motion but with an incorrect object(s).

Chapter 4

From high-level instruction to sub-tasks for mobile manipulation

Publication Information
Priyam Parashar, Vidhi Jain, Xiaohan Zhang, Jay Vakil, Sam Powers, Yonatan Bisk, and Chris Paxton. “SLAP: Spatial-Language Attention Policies.” In 7th Annual Conference on Robot Learning at Atlanta, USA. November 2023.

Despite great strides in language-guided manipulation, existing work has been constrained to table-top settings. Table-tops allow for perfect and consistent camera angles, properties are that do not hold in mobile manipulation. Task plans that involve moving around the environment must be robust to egocentric views and changes in the plane and angle of grasp. A further challenge is ensuring this is all true while still being able to learn skills efficiently from limited data. We propose Spatial-Language Attention Policies (SLAP) as a solution. SLAP uses three-dimensional tokens as the input representation to train a single multi-task, language-conditioned action prediction policy. Our method shows an 80% success rate in the real world across eight tasks with a single model, and a 47.5% success rate when unseen clutter and unseen object configurations are introduced, even with only a handful of examples per task. This represents an improvement of 30% over prior work (20% given unseen distractors and configurations). We see a 4x improvement over baseline in mobile manipulation setting. In addition, we show how SLAPs robustness allows us to execute Task Plans from open-vocabulary instructions using a large language model for multi-step mobile manipulation. For videos, see the website: `https://robotslap.github.io`.

4.1 Introduction

Transformers have demonstrated impressive results on natural language processing tasks by being able to contextualize large numbers of tokens over long sequences, and even show substantial promise for robotics in a variety of manipulation tasks [19, 190, 3]. However, when it comes to using transformers for *mobile* robots performing long-horizon tasks, we face the challenge of representing spatial information in a useful way. In other words, we need a fundamental unit of representation - an equivalent of a "word" or "token" - that can handle spatial awareness in a way that is independent of the robot's exact embodiment. We argue this is essential for enabling robots to perform manipulation tasks in diverse human environments, where they need to be able to generalize to new positions, handle changes in the visual appearance of objects and be robust to irrelevant clutter. In this work, we propose Spatial-Language Attention Policies (SLAP), that use a point-cloud based tokenization which can scale to a number of viewpoints, and has a number of advantages over prior work.

SLAP tokenizes the world into a varying-length stream of multiresolution spatial embeddings, which capture a local context based on PointNet++ [171] features. Unlike ViT-style [19], object-centric [229, 3], or static 3D grid features [190], our PointNet++-based [171] tokens capture free-form relations between observed points in space. This means that we can combine multiple camera views from a moving camera when making decisions and still process arbitrary-length sequences.

Our approach leverages a powerful skill representation we refer to as "attention-driven robot policies" [232, 189, 80, 190, 94] operating on an input-space combining language with spatial information. Unlike other methods that directly predict robot motor controls [5, 19], these techniques predict goal poses in Cartesian space and integrate them with a motion planner [232, 80, 190] or conditional low-level policy [94] to execute goal-driven motion. This approach requires less data but still has limitations, such as making assumptions about the size and position of the camera in the input scene and long training times [189, 232, 190]. However, these methods fall into a different trap: they make strong assumptions about how big the input scene is [190], where the camera is [189, 232], and generally take a very long time to train [189, 232], meaning that they could not be used to quickly teach policies in a new environment.

SLAP uses a hybrid policy architecture. The *interaction prediction* module determines which parts of the tokenized environment the robot focuses on, and a *relative action* module predicts parameters of continuous motion with respect to the interaction features in the world. SLAP generalizes better to unseen positions and orientations, as well as distractors, while being unrestricted by workspace size and camera placement assumption, using fewer demonstrations and training in roughly a day.

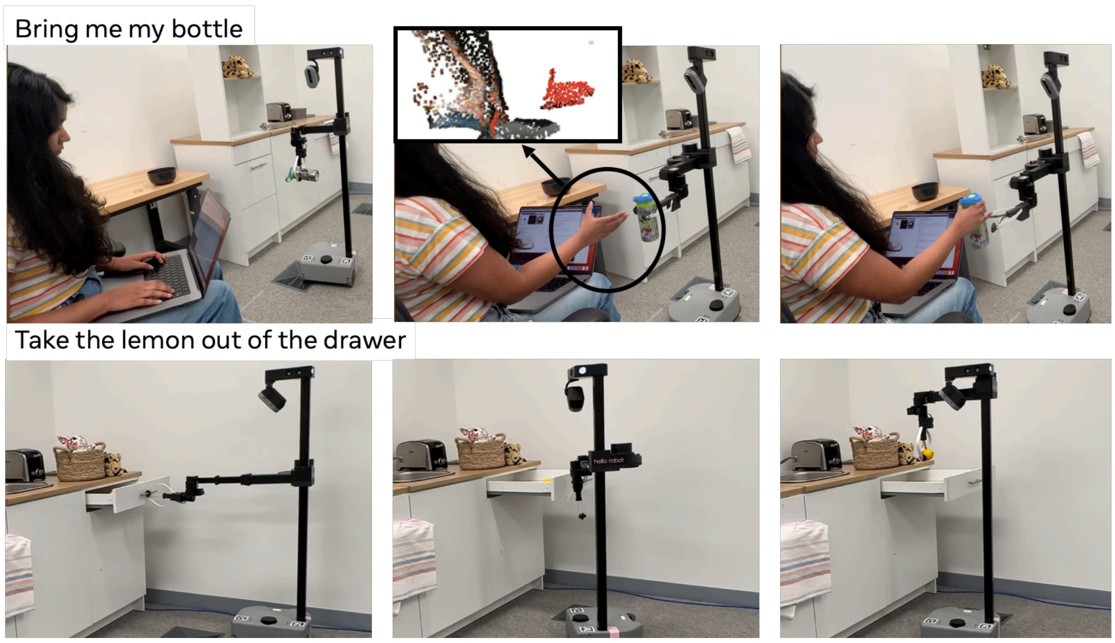


FIGURE 4.1: **Illustration** of SLAP, a framework that allows mobile manipulators to accomplish multi-step tasks given natural language goals in a sample efficient way. The demos are recorded on 'Hello Robot Stretch RE2'. (top) *Bring me a bottle*: Robot navigates to the countertop to pick the blue bottle in its view and then goes to the human to handover. (bottom) *Take the lemon out of the drawer*: Robot opens to drawer and picks up the lemon.

We evaluate SLAP on two robot platforms. First, on a Franka Panda we can perform a direct skills comparison to the current state-of-the-art, PerAct, [190], where we demonstrate better performance with 80% success rate on 8 static real-world tasks on held-out scene configurations and a 47.5% success rate tested with out-of-distribution objects. Second, unlike prior work, we move beyond the stationary camera views and robot arms of a table-top setting, and demonstrate SLAP on the Hello Robot Stretch RE-2 mobile manipulator with an ego-centric camera and 6-DoF end-effector configuration. In this setting, we also include task planning to successfully execute natural language task instructions with ten demonstrations over five learned skills and three heuristic skills (Fig. 4.1). When SLAP is compared to the PerAct baseline for four skills in a controlled setting, we see a $4x$ improvement in the task success rate (Table 4.3).

4.2 Related Work

Attention-Based Policies. Attention-based policies have been widely studied in prior research and have been found to have superior data efficiency, generalization, and the ability to solve previously unsolvable problems [144, 94, 232, 83, 190, 73]. However, these approaches often rely on strong assumptions about the robot's workspace, such as modeling the entire workspace as a 2D image [83, 232, 189, 80] or a 3D voxel cube with predetermined scene bounds [190, 94]. This restricts their applicability and may lead to issues related to camera positioning, workspace location, and discretization size. Additionally, these works can be seen, at least partly, as applications of object detection systems like Detic [238] or 3DETR [141], but they lack the manipulation component.

Compared to previous works, some recent studies focus on unstructured point clouds [144, 213]. These approaches demonstrate improved data efficiency and performance compared to traditional behavior cloning. For instance, Where2Act [144] and VAT-Mart [213] predict interaction trajectories, while UMPNet [223] supports closed-loop 6DoF trajectories. They share a common framework: a generalizable method to predict the interaction location and then predict local motion for the robot.

Training Quickly with Attention-Based Policies. CLIPort [189] and PerAct [190] are attention-based policies similar to Transporter Nets [232]. While fitting our definition of attention-based policies, they confine their workspace, use a rigid grid-like structure and treat action prediction as a discrete classification. While still a limited workspace, SPOT [83], demonstrated the usefulness of 2D attention-based policies for fast RL training, including sim-to-real transfer, and Zeng et al. [232] have shown these policies are valuable for certain real-world tabletop tasks like kitting.

Manipulation of Unknown Objects. Manipulation of unknown objects includes segmentation [217, 215], grasping [147, 201], placement [157], and multi-object rearrangement either from a goal image [173, 70] or from language instructions [124, 123]. These approaches rely, generally, on first segmenting relevant objects out, and then predicting how to grasp them and where to move them using separate purpose-built models, including for complex task and motion planning [44].

Language and Robotics. Language is a natural and powerful way to specify goals for multi-task robot systems. Several recent works [5, 19, 118] use a large-language model for task planning to combine sequential low-level skills and assume to learn the low-level skills with IL or RL. To realistically handle language task diversity, we need to learn these skills quickly. SLAP is more sample-efficient than prior IL or RL approaches. In PaLM-E [3], textual and multi-modal tokens are interleaved as inputs to the Transformer for handling language and multimodal data to generate high-level plans for robotics tasks. Our approach is a spatial extension of this strategy.

Language for Low-Level Skills. A number of works have shown how to learn low-level language-conditioned skills, e.g. [189, 190, 19, 136]. Like our work, Mees et al. [136] predicts 6DoF end effector goal positions end-to-end and sequences them with large language models. They predict a 2D affordance heatmap and depth from RGB; We do not predict depth, but specifically look at robustness and generalization, where theirs is trained from play data in mostly-fixed scenes. Shridhar et al. [190] predict a 3D voxelized

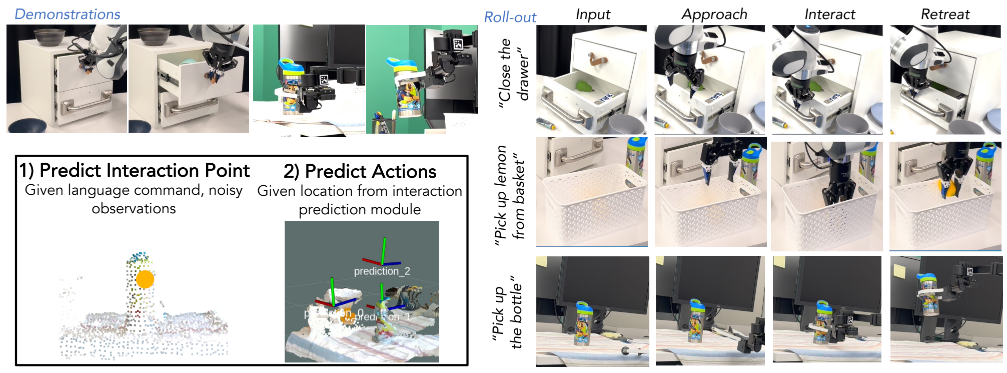

Demonstrations
Roll-out
Input
Approach
Interact
Retreat
"Close the drawer"
"Pick up lemon from basket"
"Pick up the bottle"
1) Predict Interaction Point
Given language command, noisy observations
2) Predict Actions
Given location from interaction prediction module
prediction_2

FIGURE 4.2: **Overview.** Spatial Language Attention Policies (SLAP) learn language-conditioned skills from few demonstrations in a wide variety of cluttered scenes. SLAP has two components: an *interaction prediction module* (IPM) to localize relevant features in a scene, and an *action prediction module* (APM) that uses the local context to predict an executable action.

grid and show strong real-word performance with relatively few examples, but don't look at out-of-domain generalization and are limited to a coarse voxelization of the world.

4.3 Spatial Language Attention Policies

Most manipulation tasks necessarily involve interacting with environment objects [144]. We define an 'atomic skill' as a task that can be specified by an interaction point, and a sequence of relative offsets from this interaction point. For example, `pick('mug')` is an atomic skill as it can be defined in terms of an interaction point on the 'mug' and subsequent relative waypoints for approach, grasp, and lift actions. Similarly, `open('drawer')` is an atomic skill for which the interaction point is on the drawer handle, and relative waypoints from it can be defined for approach, grasp and pull. Although these examples illustrate the concept, SLAP can predict a variable number of waypoints per skill.

We train a two-phase language-conditioned policy $\pi(x, l)$, which takes visual observation x and a language command l as inputs and predicts an *interaction point* p_I, as well as a set of *relative motions*, which are offsets from this point, instead of absolute coordinates.However, any realistic task given to a home robot by a user typically involves more than one atomic skill. Our system breaks down a high-level natural language task description ($\mathcal{T}$) into a sequence of atomic skill descriptions $\{l_j\}$ and uses them to condition the atomic skill motion policies. Our full paradigm is as follows:

$$\begin{aligned}
&\mathcal{T} \rightarrow \{l_0, ..., l_n\} \rightarrow \{\pi_j(x_j, l_j)\}_n \\
&\quad \forall j \in n, \quad \pi_j := (\pi_I, \pi_R), \text{ where:} \\
&\qquad \pi_I(x_j, l_j) \rightarrow p_I \qquad \text{(3D interaction point)} \\
&\qquad \pi_R(x_j, l_j, p_I) \rightarrow \{\mathbf{a}\}_m \qquad \text{(sequence of actions)}
\end{aligned}$$

The interaction point p_I is predicted by an **Interaction Prediction Module** π_I, and the continuous component of the action by a **Relative Action Module** π_R. The Interaction Prediction Module π_I predicts *where the robot should attend to*; it is a specific location in the world, where the robot will be interacting with the object as a part of its skill, as shown in Fig. 4.2. π_R predicts a relative action sequence with respect to this contact point in Cartesian space. These actions are then provided as input to a low-level controller to execute the trajectory. These models are trained using labeled expert demonstrations; a complete overview of the training process is shown in Fig. 4.4. Overall, the system outputs a sequence of end-effector actions $\mathbf{a}$.

Scene Representation The input observation x is a structured point-cloud (PCD) in the robot's base-frame, constructed by combining the inputs from a sequence of pre-defined scanning actions. This point cloud is then preprocessed by voxelizing at a 1mm resolution to remove duplicate points from overlapping camera views. The pointcloud is then used as input into both π_I and π_R.

For π_I, we perform a second voxelization, this time at 5mm resolution. This creates the downsampled set of points P, such that the interaction point $\hat{p_I} \in P$. This means π_I has a consistently high-dimensional input and action space - for a robot looking at its environment with a set of N aggregated observations, this can be 5000-8000 input "tokens" representing the scene.

While SLAP discretizes the world similar to prior work [16, 140, 190], our approach ensures fine resolution even in larger scenes. We couple this with a set-based learning formulation which allows us to attend to fine details in a data-efficient manner.

4.3.1 Architecture

Interaction Prediction Module We use our insight about tasks being shaped around an interaction point to make learning more robust and more efficient: instead of predicting the agent's motion directly, we formulate our π_I to solely focus on predicting a specific point $p_I \in P$, representing a single $5mm$ voxel that is referred to as the "interaction point". This formulation is akin to learning object affordance and can be thought of as similar to prior work like Transporter Nets in 2D [232]. We hypothesize that predicting

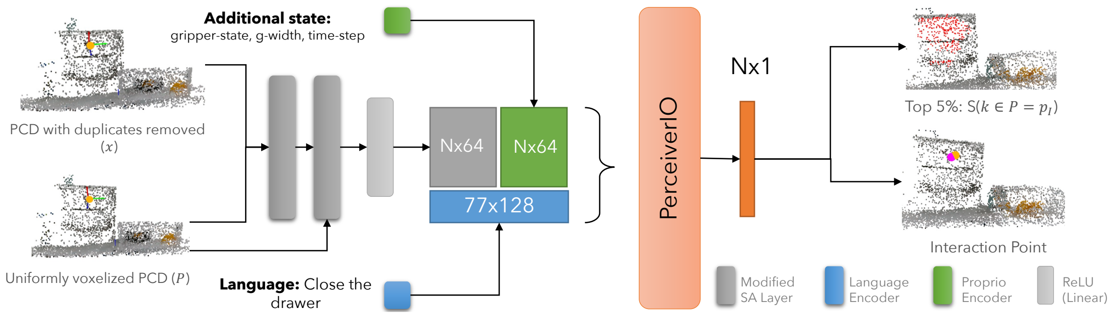

Additional state:
gripper-state, g-width, time-step
PCD with duplicates removed (x)
Uniformly voxelized PCD (P)
Language: Close the drawer
Nx64
Nx64
77x128
PerceiverIO
Nx1
Top 5%: S($k \in P = p_I$)
Interaction Point
Modified SA Layer
Language Encoder
Proprio Encoder
ReLU (Linear)

FIGURE 4.3: **Architecture** of the interaction prediction module. The point cloud is downsampled to remove duplicates and encoded using two modified set-abstraction layers. The SA layers generate a local spatial embedding which is concatenated with proprioceptive features - in our case, the current gripper state. Both spatial and language features are concatenated and input into a PerceiverIO transformer backbone. We then predict an interaction score per spatial feature and the *argmax* is chosen as the interaction site for command l.

attention directly on visual features, even for manipulation actions, will make SLAP more general. We use a PerceiverIO [86] backbone to process the data, based on prior work on language-conditioned real-world policies [190].

We first pass our input point cloud through two *modified set abstraction* layers [171] which result in a sub-sampled point-cloud with each point's feature capturing the local spatial structure around it at two different resolutions. This encourages the classifier to pay attention to *local structures* rather than a specific point that may not be visible in real-world settings. We concatenate the CLIP [174] tokenized natural language command with the encoded point cloud to create an input sequence.

Each point $i \in P$ in the point cloud is assigned a score with respect to task τ_j which results in the interaction point for that task, $p_I^j := \underset{x,y,z}{\text{argmax}}(S(i = p_I^j | l, x, P, \mathcal{D}^j))$, where $\mathcal{D}^j$ is the set of expert demonstrations provided for task τ_j. The IPM architecture overview is provided in Fig. 4.3. Note we also use binary semantic features from Detic in the Stretch experiment for training SLAP as an additional feature channel apart from the color-channels.

Modified Set Abstraction Layer. The default SA layer as introduced by Qi et al. [171] uses farthest point sampling (FPS) to determine which locations feature vectors are created. FPS ensures that subsampled point-cloud is a good representation of a given scene, without any guarantees about the granularity. However, it's very sensitive to the number of points selected - in most PointNet++-based policies, a fixed number of points are chosen using FPS [171]. However, SLAP must adapt to scenes of varying sizes, possibly with multiple views, and still not miss small details.

We propose an alternative PointNet++ set abstraction layer, which computes embeddings based on the original and an *evenly* downsampled version of the point-cloud, P. This

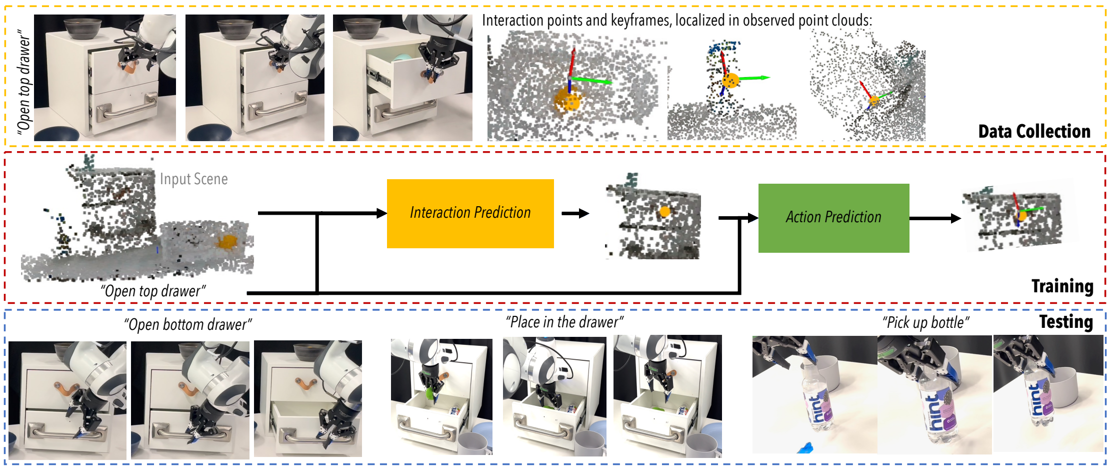


FIGURE 4.4: **Training** SLAP. Demonstrations are collected and used to train the Interaction Prediction module and the Action Prediction Module separately.

results in a denser spatial embedding by considering a subset of all points within a certain radius of each-point in the downsampled point-cloud. This downsampled set of points guarantees we can attend to even small features, and allows us to predict an interaction point p_I from the PointNet++ aggregated features.

Relative Action Module The relative action module relies on the interaction point predicted by the classifier and operates on a cropped point cloud, x_R, around this point to predict the actions associated with this sequence. As in the interaction prediction module, the model uses a cascade of modified *set abstraction* layers as the backbone to compute a multi-resolution encoding feature over the cropped point cloud. We train three multi-head regressors (described further below) over these features to predict the actions for the overall task. Specifically, π_R has three heads, one for each component of the relative action space: gripper activation g, position offset δp, and orientation q. Our LSTM-based architecture (details in C.2) can predict skills with variable number of actions (3,4 in our experiments).

Also note that the cropped input point-cloud is not perfectly centered at the ground truth interaction point $\hat{p_I}$, but rather with some noise added: $\hat{p_I}' = \hat{p_I} + \mathcal{N}(0, \sigma)$. This is done to force the action predictor to be robust to sub-optimal interaction point predictions by the interaction predictor module during real-world roll-outs. Thus, for each part of the action sequence, the keyframe position is calculated as: $p = p_I + \delta p$. When acting, the robot will move to (p, q) via a motion planner, and then will send a command to the gripper to set its state to g.

4.3.2 Training

To collect data, an expert operator guides the robot through a trajectory, pressing a button to record *keyframes* representing crucial parts of a task. At each keyframe, we record the associated expert action $\hat{a} = (\hat{\delta p}, \hat{q}, \hat{g})$. We assume that low-level controllers exist - in our case, we use Polymetis [121] for the Franka arm and Hello Robot's controllers[1] for Stretch. Example tasks are shown in Fig. 4.5.

Interaction Prediction Module. We train π_I with a cross-entropy loss, predicting the interaction point p_I from the downsampled set of coarse voxels P. We also apply what we call a *locality loss* (L_{loc}), as per prior work [166]. Conceptually, we want to penalize points the further they are from the contact point, both to encourage learning relevant features as well as to aid in ignoring distractors. To achieve this, we define the locality loss as: $L_{loc} = \sum_{k \in P} \text{softmax}(f_k) \|\hat{p}_I, k\|^2$, where f_k is the output of the transformer for point $k \in P$. The $softmax$ turns f_k into attention over the points, meaning that L_{loc} can be interpreted as a weighted average of the square distances. Points further from $\hat{p}_I$ are therefore encouraged to have lower classification scores. Combining our two losses, we obtain $L_I = CE(P, \hat{p}_I) + \frac{w}{|P|} L_{loc}$, where w is a scaling constant that implicitly defines how much spread to allow in our points.

Relative Action Module. To train π_R, we use the weighted sum of three different losses. We train $a = (p, q, g) = \pi_R(x_R)$ with an L2 loss over the δp, a quaternion distance metric for the loss on q based on prior work [158] and binary cross-entropy loss for gripper action classification (Sec. C.1).

4.3.3 Task Planning

Consider a natural language instruction from a user such as 'put away the bottle on a table'. We decompose it to a sequence of atomic skills as: `goto('bottle')`, `pick_up('bottle')`, `goto('table')`, and `place_on('table')`. We procedurally generate natural language and code templates for 16 task families. Refer to Appendix C.3. We use LLaMA [206] models for in-context learning [22, 2] and adapter fine-tuning [77] to learn the mapping between the instructions of the natural language task to the corresponding sequence of atomic skills.

4.4 Experiments

We report the success rate of our model for 8 real-world manipulation tasks in Table 4.2, and compare it against prior baselines trained using the same labeling scheme. Overall, we see an improvement of 1.6x over our best comparative baseline, PerAct [190]. Our robotics code was implemented using the HomeRobot framework [224][2]. We test each model under two different conditions: *Seen setting assumption;* i.e. those with seen distractor

[1]https://github.com/hello-robot/stretch_ros
[2]https://github.com/facebookresearch/home-robot

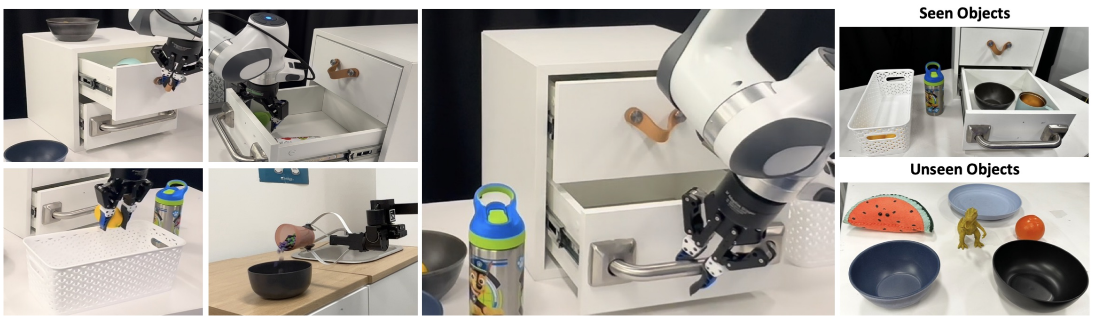


FIGURE 4.5: **Qualitative Results**: Examples of tasks executed on a Franka arm through our trained model in a clean setting. We trained numerous tasks (left) and tested on both seen and unseen objects (right).

	GT	Inferred
Heuristic	66.0	48.5
Learned	80.0	53.2
Total	68.5	58.5

TABLE 4.1: **End-to-end performance.** Learned skills outperform heuristics except when Detic fails.

objects and objects placed roughly in the same range of positions and orientations as in the training data in any relative arrangement (including unseen). Second, we test under *unseen setting assumptions;* i.e. those with unseen distractor objects and the implicated object placed significantly out of the range of positions and orientations already seen. We run five tests per scene setting per skill per model and report the percentage success numbers in Table 4.2. We compare our model with Perceiver-Actor (PerAct) [190]. We train each model for the same number of training steps and choose the SLAP model based on the best validation loss. For PerAct, we use the last checkpoint, according to their testing practices [190].

We also performed a per skill evaluation of SLAP and PerAct on Stretch under the *unseen setting assumption* (see Table. 4.3). This was accomplished by adding unseen distractor objects to the scene and moving the robot base position within reachable distance of the object. We needed to increase the workspace limits of PerAct (1.5 m cube from 1 m) to capture the larger observation area for the mobile manipulator, to keep it consistent with SLAP. Note that demonstrations were collected on a different robot from the one policies were deployed on.

4.4.1 Longitudinal Task Execution on Stretch

We trained a multi-task model for the Stretch robot for five skills using 10 demonstrations each. This model was deployed in an end-to-end system which operates over code-list

	Seen		Unseen	
Skill Name	PerAct	SLAP	PerAct	SLAP
Open bottom drawer	00%	**80%**	00%	**60%**
Open top drawer	60%	**80%**	**40%**	**40%**
Close drawer	**100%**	**100%**	**40%**	**40%**
Pick lemon from basket	60%	**80%**	10%	**40%**
Pick bottle	**60%**	**60%**	**60%**	40%
Place into the drawer	60%	**80%**	40%	**60%**
Place into the basket	40%	**100%**	10%	**60%**
Place into the bowl	40%	**60%**	00%	**40%**
Average Success Rate	50%	80%	27.5%	47.5%
Improvement		1.6x		1.7x

Table 4.2: SLAP and PerAct [190] performance on real world Franka manipulation tasks. We evaluate both seen scenes (seen object positions and distractors), but in different arrangements, and unseen scenes with previously-unseen object positions and distractors. SLAP is notably better overall in both conditions.

generated by a task-planner (as in Sec. 4.3.3). We ran five prompts end-to-end with four to eight skills each, using ground truth plans - we verify the viability of generating these task plans in §4.4.2. These experiments are performed in the *unknown setting* with the robot starting anywhere with respect to the objects.

For fair evaluation in the low-data regime, we add some structure by specifying an orthogonal viewing direction for objects. Once the agent finds the object of interest, it fires an initial prediction using SLAP to find most promising interaction point. This prediction happens under any dynamic viewing angle of the object, however, we assume the object remains in robot's view. This dynamic prediction and pre-programmed viewing angle is used to *approach* the object at an orthogonal viewing angle where the model fires an actionable prediction for the full skill execution. See §C.3. We observe that learned skills suffer significantly when inferred plans are used to create language prompts for SLAP. This is due to discrepancies between object descriptions that the LLM generates and Detic's object detection capabilities. When SLAP does not get the necessary semantic masks, its predictions suffer. On the other hand, when the semantic features of Detic are available, the performance of IPM improves significantly even with unseen distractors (80% versus 47.5% in Table 4.2). We still see failures when the relative position is significantly perturbed. See §C.5 for ablation analysis of our design against PerAct.

4.4.2 Task planning with in-context learning and fine-tuning LLaMa

Previous work has shown the strength of language models as zero-shot planners [81], a result strengthened by improved techniques for "in-context learning" or prompting [177].

Skill Name	PerAct	SLAP
Open Drawer	0%	60%
Close Drawer	40%	100%
Take bottle	0%	80%
Pour into bowl	40%	80%

TABLE 4.3: SLAP on a mobile manipulator using a multi-task model across 4 skills, over 5 tries. With semantic predictions added to our feature space, we see the model perform better on unseen scenes with new distractors and unseen relative position of the robot with respect to the scene

					Task	Lat.
LLaMa		Verb	Noun	Acc.	Corr.	(sec.)
IC	7B	83	81	76	61	16.4
IC	30B	81	81	76	62	27.6
FT	7B	100	98	99	91	19.5

TABLE 4.4: Fine-tuning (FT) outperforms in-context learning (IC) for same latency.

To verify that models can produce task plans with the skills we defined, we experiment with both in-context learning (IC) [212] of LLaMA [206] and adapter fine-tuning (FT) [77]. Table 4.4 presents the models' verb, noun, and combined accuracies. Task Correctness is a binary score if the entire prediction was correct, and finally, latency is measured in seconds on a single A6000 with 16 GB RAM.[3] High Task Correctness from a small model verifies the compatibility of our skills with LLM task planning.

Our experiments show that SLAP has a number of advantages over previous work. We also observe that SLAP predicted consistently successful actions for opening the top and bottom drawers. While the top drawer handle is a small leather loop, the bottom drawer handle is a slender cylindrical rod. PerAct produces more inaccurate predictions for the small loop on the drawer, while failing completely to grasp the bottom handle. See Fig. 4.6 for an example. We note that our π_I has a finer granularity than PerAct's representation.

Locality Loss. We observed reduced object confusion with locality loss when comparing to a version of SLAP trained without it. Before adding the locality loss, the attention for a skill would be spread farther around the scene – spreading to the bottom drawer from top and vice-versa for example. This led to confusion and lower success rates on drawer-related tasks (previously, 0% on bottom drawer and 70% on top drawer from a 9-task model trained with data augmentation). However, the locality loss encouraged

[3]Adaptor fine-tuning increases the model size by ∼6%, which accounts for the additional latency compared to IC. We use standard inference libraries so results are comparable, but not optimized for runtime [61].

attention to be concentrated tightly on the relevant surfaces of the implicated object, as seen in Fig. 4.7.

4.5 Limitations

SLAP has high variance in out-of-distribution situations, resulting in complete failure if π_I fails to correctly identify the context. For π_R, multimodal or noisy data still poses issues; replacing π_R with a policy which can better handle this data, e.g. Diffusion Policies [39]. Overall system has multiple points-of-failure due to heuristic policies, unaligned language and vision models; end-to-end trainable architectures and cross-modal alignment could help.

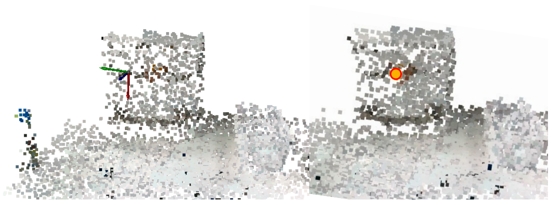


FIGURE 4.6: Output for interaction action predicted by PerAct (left) and interaction point predicted by SLAP (right) for the same input. We see a trend where π_I predictions for finer details are consistently more optimal

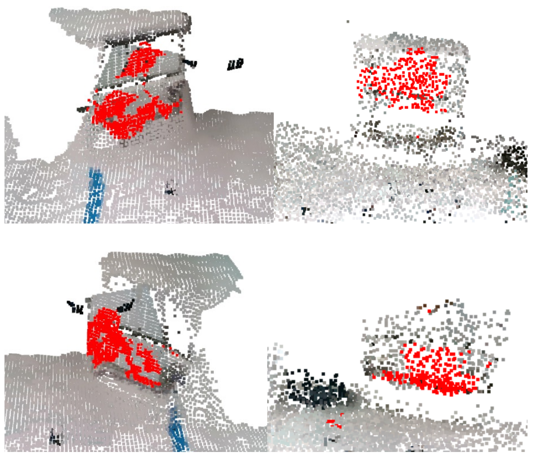


FIGURE 4.7: A comparison of attention masks with (right) and without (left) the locality loss. Attention is more focused on relevant structures and less spread to irrelevant structures in the right image. The top row shows the attention mask for opening the top drawer: the previous model (left) spreads attention to the bottom handle due to its association with the "open" skill, while the new model (right) focuses only on the top drawer. Attention is more concentrated on handle for the "open bottom drawer" skill in the bottom row.

4.6 Conclusion

We propose a new method for learning visual-language policies for decision making in complex environments. SLAP is a novel architecture which combines the *structure* of a point-cloud based input with *semantics* from language and accompanying demonstrations to predict continuous end-effector actions for manipulation tasks. We demonstrate SLAP on two hardware platforms, including an end-to-end evaluation on a mobile manipulator, something not present in prior work. SLAP also outperforms previous state-of-the-art, PerAct, on both mounted and mobile robot setups.

Part II

Understanding implicit intents and constraints

Chapter 5

From visuals and listener's distance to acoustic noise of robot's motion

Vidhi Jain, Rishi Veerapaneni, and Yonatan Bisk. "ANAVI: Audio Noise Awareness using Visuals of Indoor environments for NAVIgation". In 8th Conference on Robot Learning 2024, Munich, Germany.

Everyone and everything makes noise, but while humans are aware of their sound impact on everyone around them, robots are not. This ability is particularly important as robots become members of our households and need to know to whisper when the baby has fallen asleep or that it is ok to make noise during a party. The challenge of audio awareness starts with understanding the geometry and material composition of rooms. In this work, we propose the awareness of audio noise through visual of indoor environment for NAVIgation (ANAVI[1]). We generate data on how loud an 'impulse' sounds at different listener locations in simulated homes, and train our Acoustic Noise Predictor (ANP). Next, we collect acoustic profiles corresponding to different actions for navigation. Unifying ANP with action acoustics, we demonstrate experiments with wheeled (Hello Robot Stretch) and legged (Unitree Go2) robots so that these robots adhere to the noise constraints of the environment. We record the decibel levels of the audio produced at the receiver's location and report the $\epsilon-$threshold accuracy of our model vs. distance-based baseline.

[1]"anavi" also implies peace-loving, kind to people.

5.1 Introduction

Humans are very aware of the noise that their movements create. In homes and offices, we avoid being noticed if a person is having a video call or a child who has just fallen asleep. Home robots need that same social awareness when planning their actions in indoor environments. Unfortunately today robot vacuums may be as loud as 70 decibels, which concerns many people with sensitive ears. The existing solution is to use the "do not disturb" mode [85] which completely stops the robot from functioning so that it does not make noise. Integrating more complex robots, e.g., quadrupeds, in homes, needs more sophisticated methods of addressing noise levels while maintaining efficiency.

Although sound is inevitable with robot movement, it can be mitigated. In isolation, e.g. without acoustic reflections or echoes, unoccluded sound intensity decays quadratically with distance. As a robot moves away from humans or sound-sensitive areas, its sounds, such as motor noise or beeps, become less likely to be heard. Thus, a simplistic approach to reducing this noise is for the robot to move slowly and steer clear of people and pets in its environment. While this might initially seem like a sensible strategy, it can lead to increased time and energy consumption for task completion and may not always be practical or feasible. Importantly, sound intensity depends on many other factors beyond distance. The intensity at the same distance of 1 meter will be reduced if separated by a wall or in a carpeted room, whereas, conversely, halls create echoes. Architectural geometry and material properties affect how the sound reflects, diffracts, and is absorbed.

Existing work on audio for robotics focuses on finding the source of an audio signal, guiding navigation [65, 59]; which leverages realistic 3D simulators to learn how to navigate to the sound of dripping water in the sink or a fire alarm. In contrast, we focus on awareness of self-generated sounds; a related yet orthogonal problem: how loud will the robot's actions be at a listener's location?

To this end, we propose Audio Noise Awareness by Visual Interaction (ANAVI), a framework that allows a robot to learn about the unintended noise made by its actions and thereby adapt its actions to the noise constraints of the environment. We train an Acoustic Noise Prediction (ANP) model using audio impulse response simulation (SoundSpaces) in 3D real-world scans (Matterport). Intuitively, ANP learns to predict how an impulse generated at the source (robot's location) will be heard at a listener's location, using the relative distance and direction of the listener and, more importantly, using the visual features around the robot. ANAVI framework makes use of ANP model for estimating audio noise costs for planning a quieter robot path.

The contributions of our work are as follows. (a) Our Acoustic Noise Prediction (ANP) model with visual features performs better than distance-based heuristics and learned baselines. We show that ANP predictions overlap well with the data distribution and

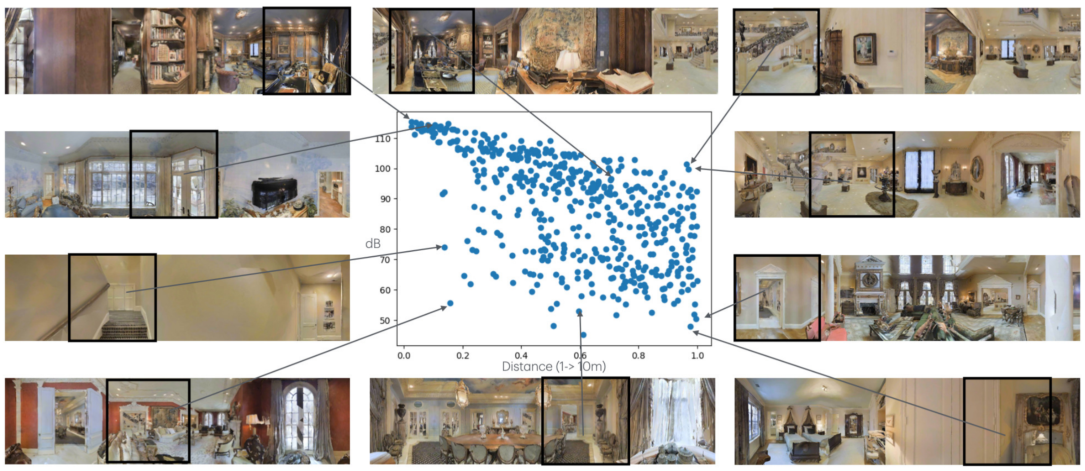


FIGURE 5.1: **Overview**. We generate data for acoustic impulse response in simulated 3D scenes from real-world environment scans. x-axis shows the relative distance of the listener from the source agent, y-axis shows the max decibels of a simulated Impulse Response (IR) at listener. The box area on the panaroma shows the listener's direction relative to the agent. Note the complex response pattern as materials, objects, and geometry have non-linear interactions with the sound.

have higher epsilon thresholded accuracy (a metric analogous to PR curves) than distance-based baselines. (b) We show qualitative analysis of the ANP model's prediction for fixed robot location (while assuming the listener at every point on the map) and fixed listener locations (while assuming the robot at every point on the map). (c) Further, we perform real-world experiments to compare real audio decibels with our model's predictions and propose ANAVI framework for adapting robot's navigation plans for quieter paths.

5.2 Related Work

Acoustics in Learning and embodied AI. Sound and vibrations have been extensively studied in physics, architectural designs, and acoustic material properties. Recent works in multimodal machine learning have studied the alignment of audio and visual observations [153], acoustic synthesis for novel view [30], and how actions sound [154, 32, 142]. Improved acoustic simulation, with bi-directional ray tracing techniques [111], has led to realistic room audio simulation and further research in audio-visual matching [34], spatial alignment [146], learning dereverberation [29], audio-goal navigation [31] and audio based learning [132]. [38] present a movable microphone and camera rig to record realistic room impulse response, and show sim2real transfer. We leverage realistic acoustic and visual simulation frameworks to solve an underexplored but practical problem that requires agent to predict acoustic noise loudness at specified location in the environment.

Acoustics in Robotics. Existing works [41, 204, 63, 53] consider acoustics as a sensory signal that informs about the robot-object or object-object interactions in the environment. Attending to the audio signal improves the robustness in manipulation tasks;

especially where the robot cannot see due to self-occlusion or low-lighting conditions. To simulate human robot interaction, [65] proposed a multisensory platform in a audio-visual simulator similar to ours. In ego-noise prediction task for robots, very few works exist, for example [164] studies self-generated audio imitation. In contrast, our work investigates how the robot can learn to anticipate how the sound generated by its movement at other locations. Additionally, audio is actively studied for easier spatial exploration [79, 56], verbal task specification for improved human-robot interaction [188, 27, 234]. Our work provides tools to further advance the robot's ability to act based on how the audio will be perceived by the listeners.

5.3 Acoustic Noise Prediction (ANP)

A robot navigating indoor environments produces sound, either from its motors or its speakers trying to communicate with others in its environment. Given the robot in an environment and a listener elsewhere, we want to predict how loud the robot is at the listener's location. Sound travels from the robot's location to a listener's location depending on the relative distance, architectural geometry, materials, and hearing sensitivity of the listener. In our simulated environments, the most dramatic differences happen at short distances blocked by walls or furniture, versus long echoing hallways.

Simulator Setup. We create a simulated dataset of sound decibels perceived at the listener's location when the robot is located in different parts of home environments. We use the Matterport3D dataset, which consists of 85 real-world scans of indoor environments. The simulator assumes a sound impulse at the source location and calculates a response (aka Room Impulse Response or RIR) at the listener location. For this, we use Sound-Spaces 2.0 [33] to simulate how audio waves travel from the source to arrive at the listener as an impulse response. To simulate any sound, the room impulse response is convolved with the audio at the source. As convolution is linear, the max decibel of audio at the listener is linearly proportional to the decay of the max db of the RIR from that of the original impulse. We assume that an impulse generates the sound pressure level of $1W/m^2$ at the source and that the listener is within 10 m of the source. More details are in Appendix D.2.2.

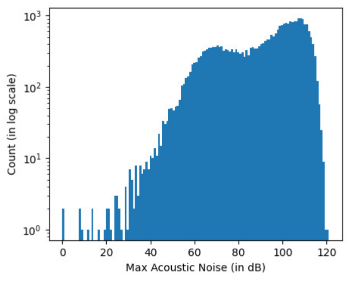


FIGURE 5.2: Histogram of max decibel values from simulated data

Data. We approximate the max dB of an audio at listener = (max dB of the RIR / max dB of the impulse at source) * max dB of audio source. Sound decibels heard by the listener as an impulse response generated at the robot's location vary as shown in Fig. 5.2.

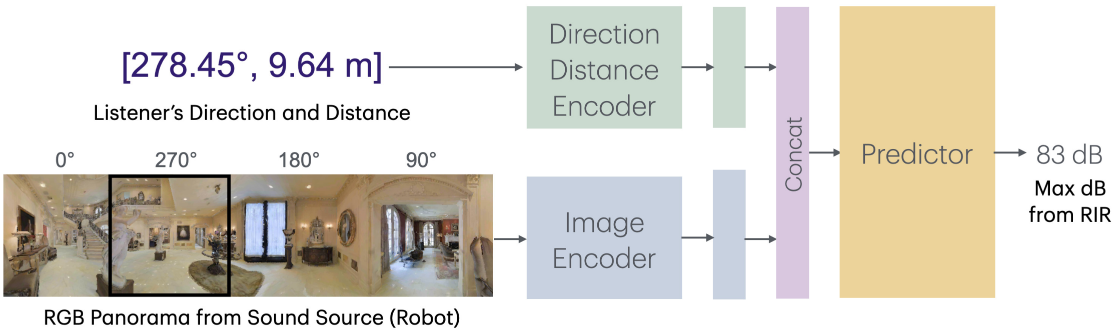

[278.45°, 9.64 m]
Listener's Direction and Distance
0°
270°
180°
90°
RGB Panorama from Sound Source (Robot)
Direction Distance Encoder
Image Encoder
Concat
Predictor
83 dB
Max dB from RIR

FIGURE 5.3: **Architecture.** Our Acoustic Noise Prediction (ANP) model consists of *Image encoder*, *Direction-Distance encoder* and *Predictor* modules. The inputs are the 360°RGB panorama view at the robot's location, and the relative polar coordinates of the listener. The square box on the panorama highlights robot's current facing direction, and is drawn for illustration purposes only. The output is the max decibel (dB) of the Room Impulse Response (RIR) at the listener's location.

Note the two modes in the distribution at 68dB and 110 dB. From each map, we sample a dataset of $\mathcal{D} = \{\boldsymbol{s}_i, \boldsymbol{l}_i\}_{i=1}^{N}$, where N is the number of samples. For each pair of robot and listener locations $\{\boldsymbol{s}_i, \boldsymbol{l}_i\}$, we record the visual observation at the robot V_s, relative polar coordinates of the listener $\{r_{sl}, \theta_{sl}\}$ and the max decibels of the impulse response generated at the listener's location y. The relative distance r_{sl} is an important feature in predicting sound intensity heard by the listener based on spherical spreading of sound waves. To render realistic sounds, the simulator uses additional privileged information about 3D mesh geometry and material sound properties for ray tracing [111].

Visual observations of the surroundings contain partial information, and we aim to train a model that learns to predict the audio at the listener based on the latent visual features. Recall our example in Sec. 5.1, where a 1m distance in the same room versus separated by a wall affects loudness at the listener's locations. A robot should learn to perceive visual features like a wall, open area, corridors, etc., to make an informed estimate about the perceived loudness at the listener's location. For panoramic visual input, the direction of the sound-sensitive listener's location θ_{sl} is important to infer what geometry and materials are in the area the sound waves will most likely pass through on their way to the listener. We also consider ego-centric visual features with robot facing in the direction of the listener. More details comparing the ego- and pano- visual features in Sec. 5.4.2.

Architecture We learn an acoustic noise predictor function f such that $y = f(r_{sl}, \theta_{sl}, V_s)$. Fig. 5.3 shows our proposed architecture for Acoustic Noise Prediction (ANP). Our model consists of three components. *Image Encoder* extracts the visual features using pretrained visual encoders like ResNet-18. We apply adaptive average pooling and normalize it to get visual encoding e_{visual}. *Direction-Distance Encoder* is a simple linear projection to obtain an encoding e_{dirdis}. *Predictor* takes the concatenated vector of e_{visual} and e_{dirdis}, and processes it through a series of blocks. Each block contains a linear layer, a batch norm layer, and GeLU activation. The output size of each successive linear layer shrinks to act as an "information bottleneck". The final output is either a scalar real number for regression or $m-$ binned logits for classification.

Training Our code uses PyTorch and torchvision ResNet-18 checkpoint. We apply ResNet-18 pre-processing on V_s for mean 0 and std 1. We normalize the r, θ and y between 0 and 1. We use Huber Loss[82] with $\delta = 0.1$ and the AdamW optimizer, which has a learning rate of 0.01 and a step schedule with 0.95 decay after 10 epochs. We use a batch size of 64 on a 24GB RAM GPU.

5.4 Simulation Experiments

We assess our acoustic noise prediction model in simulation and demonstrate its applicability to real-world environments. Our simulation experiments are conducted in the Habitat 2.0 simulator [202] on the Matterport3d dataset [26]. We inherit the train/val/test splits for Matterport3d scenes from the PanoIR [33] dataset - 59 maps for train, 11 for val, and 15 for test. For training, we create 5,000 samples per map, where each sample consists of robot location $\boldsymbol{s}$, listener location $\boldsymbol{l}$, the robot's panoramic view V_s, and the listener's Impulse Response w_l. We post-process the impulse response to calculate the max dB heard at the listener's location as y. For val and test, we create 500 samples per map. We evaluate $500 \times 15 = 7500$ samples in all simulation experiments.

Metrics. We compare acoustic noise prediction models with data distribution coverage plots. To understand the coverage of predicted values with respect to the true labels, we include a distance-decibel plot in Fig.5.4. The higher the overlap of the red overlay over the blue, the better the model's performance. Note, the highly diverse data distribution of decibels y for a given distance r.

Another, more quantitative, view of the regression analysis is via ϵ-thresholded accuracy curves. Our ϵ-thresholded accuracy curves are inspired by Precision-Recall curves used in classification. This comparison allows us to see the effect of cross-entropy style training, both on performance and how it skews the data distribution. See Fig. 5.5. Let y be the normalized max decibel value of the simulated impulse response (IR)s, and $\hat{y}$ be the predicted value by the model. Let there be N samples, $\mathbf{1}(\cdot)$ is the indicator function, which is 1 if the condition inside is true and 0 otherwise, and $N = 7500$. For ϵ-accuracy, we compute $\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(|y_i - \hat{y}_i| \leq \epsilon)$. An $\epsilon = 1/128 = 0.007$ captures whether the model can learn to accurately predict within a single decibel. Humans typically can barely differentiate the 1 dB difference for sounds in 1000-5000 Hz frequencies, also known as Just Noticeable Difference (JND) [119]. JND increases to 3-5 dB at very low or very high frequencies and changes with larger initial loudness levels.

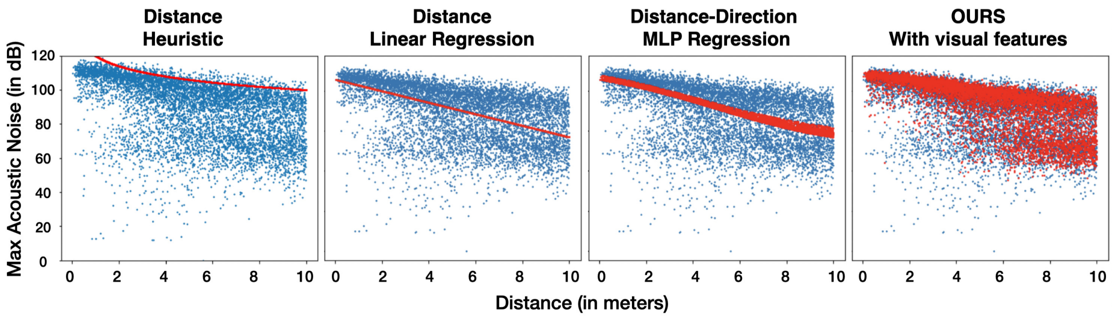


FIGURE 5.4: **Main Results.** Predicted Data Distribution plots for (left to right) Heuristic, DisLinReg, DirDisMLP, **Ours**(Pano)-VisDirDis.

Baselines. We compare the performance of our Acoustic Noise Prediction (ANP) model with a distance-based heuristic and two learned baselines: (i) *Heuristic Distance based (Heuristic)* Sound intensity decays by a factor of inverse squared distance, that is $\frac{1}{r^2}$. Thus, the predicted max dB IR can be calculated by $-20\log_{10}(r) + 120$. (ii) *Regression with Distance (DisLinReg)* We train a linear regression model to match the average shape and curvature of the data as a function of the data. Note that this more accurately captures the shifting mass distribution than the heuristic; however, it is unimodal in its predictions. (iii) *MLP Regression with Distance and Direction (DirDisMLP)* We train a multi-layer perceptron model with blocks of linear layers, batch norm, and ReLU activation, where layer sizes of 2, 8, 8, and 1. (iv) *Visual features with Distance and Direction (VisDirDis)* One concern with the DirDisMLP is that it does not incorporate visual information. Therefore, it cannot distinguish between scenarios with a wall between the robot and the listener versus instances in open spaces where sound may travel collision-free. In VisDirDis model, we use a panoramic RGB view at the agent's location as the visual feature of 512-dim. We include direction as input to localize and attend to how the loud listener hears the robot. The direction and distance are encoded by a linear projection to 16-dim. We concatenate and pass the resulting 528-dim vector to the predictor. The predictor consists of 4 blocks with layers sizes: 528, 256, 64, 8 and outputs as a scalar.

5.4.1 Results

We show significant performance improvement by incorporating visual information. Visual information conveys the architectural geometry and materials used in the space, which affects the perceived sound intensity at the listener's location. Fig. 5.4 shows the predicted data distribution (in red), overlayed on the true test data (in blue). The ideal model would directly match the blue data distribution, and it is apparent that our model most closely aligns with the ground truth. Additionally, Figure 5.5 shows $\epsilon-$accuracy for learned models. On the far left of Figure 5.4, the geometry spreading-based distance heuristic (Heuristic) only predicts correctly for free space and doesn't account for reflections in corridors or absorption by walls in home environments. Second, we plot the linear regression prediction (DisLinReg) based on distance. While it improves the overall $\epsilon-$accuracy curve in Fig. 5.5, it doesn't cover the plot. The learned MLP (DirDisMLP) slightly improves performance and begins to spread out the generated data distribution. But it lacks sufficient signal about the spatial context that could inform if there are possible reflections, absorptions, and diffractions of the sound. These observations validate our approach on the right, where we use visual features along with distance and direction (Pano-VisDirDis) and observe the best $\epsilon-$accuracy and data distribution coverage.

5.4.2 Ablations

We evaluate modeling design choices for loss functions and visual representations. Refer to Fig. 5.6 for the data distribution coverage results for all our model variants in this section. (i) *Loss function: Regression vs Classification.* We use $m-$bins to discretize the prediction and convert it from regression into a classification problem. The granularity of the prediction changes depending on the bin size. For example, a 16-binned model considers 81 and 87 to be in the same bin, while a 128-bin prediction considers 87 and 88 as separate bins. We observe that the cross-entropy models train faster to reach their peak performance, especially with coarser granularity like 16-binned labels. While in many cases, 16 bins could be enough to tell if the robot is too loud, it may not help with sensitive listeners.

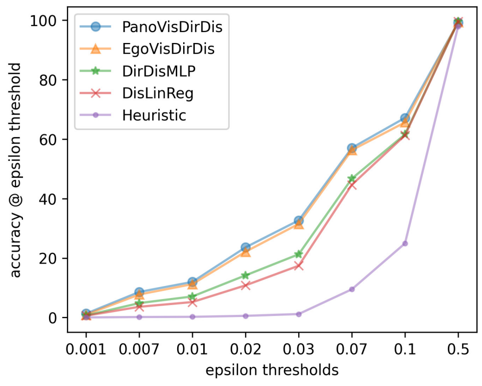


FIGURE 5.5: $\epsilon-$threshold accuracy for Heuristic, DisLinReg, DirDisMLP, EgoVisDirDis, and PanoVisDirDis

(ii) *Visual input: Ego-centric vs Panoramic view.* For ego-centric images, we extract the 90°FoV from the panoramic image at the robot's location in the direction of the receiver. We use ResNet-18 as the visual encoder for RGB features. We encode the distance and direction with a linear projection to a 16-dim vector. In Fig. 5.6, we observe that the learned distributions with ego-view miss outliers with very high values for the longer distances. In Fig. 5.5, the overall $\epsilon-$accuracy is lower than our Pano- version. While ego-centric is desirable for simplified real-time perception with a camera, it has a limited field of view and lacks the information about architectural geometry in the immediate vicinity of the robot; which can affect how sound travels to the listener.

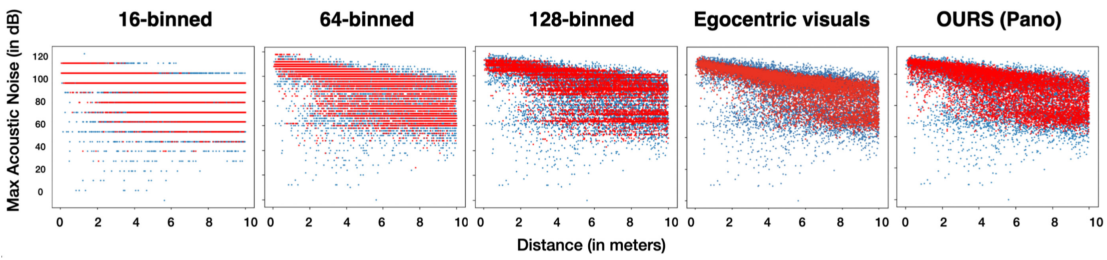


FIGURE 5.6: **Ablations**. Data distribution plots for classification models trained with the cross-entropy loss with (a) 16, (b) 64, (c) 128 binned dB values, (d) regression model with an ego-centric view and (e) our proposed model on the right. We observe that the cross-entropy models are easier to train and capture outliers but provide a coarser prediction granularity. The learned distributions with ego-view miss outliers with very high values for the longer distances.

5.4.3 Analysis

To better understand the model's performance, we create visual representations of two scenarios by mapping the expected max dB value at each location. See Appendix D.2.3 for visualizations. (i) *Fixed Robot Acoustics Map:* In this scenario, we initialize the agent at a location and compute the model's predictions for different listener locations. The input to the model consists of the same image, with variations in the distance and direction of the listener. This acoustics map helps to predict how many possible listeners in the environment perceive the robot's actions as loud in Sec. 5.5. (ii) *Fixed Listener Acoustics Map:* In this scenario, we fix the listener's location and compute the max dB heard by the listener when the robot is at different locations in the environment. We construct this acoustic map to estimate the noise cost incurred at each state for planning in Sec. 5.5.

5.5 Real world experiments

We propose to evaluate how well the sound loudness is perceived at different listener locations in its environment. Unlike the audio simulators, which have privileged information about the scene mesh, material properties, etc., the agent can only see in its immediate vicinity and knows the relative polar coordinates of the listener. Ideally, we would move the robot to multiple different rooms/apartments and then record the audio. However, practically, robots can be hard to move across apartments and this limits data collection. Thus, an easier method to gather audio responses is to record the sound of robot's actions onto a device (e.g. phone or laptop). Then instead of moving the robot to other locations, we take our device and play back the sound. An added benefit of using recorded audio is that we reduce the variability of the source sound across measurements at different listener locations.

Our experiments use the recorded sounds from a Unitree Go2 and Hello Robot Stretch performing different actions. We use a laptop speaker to play the robot action's audio;

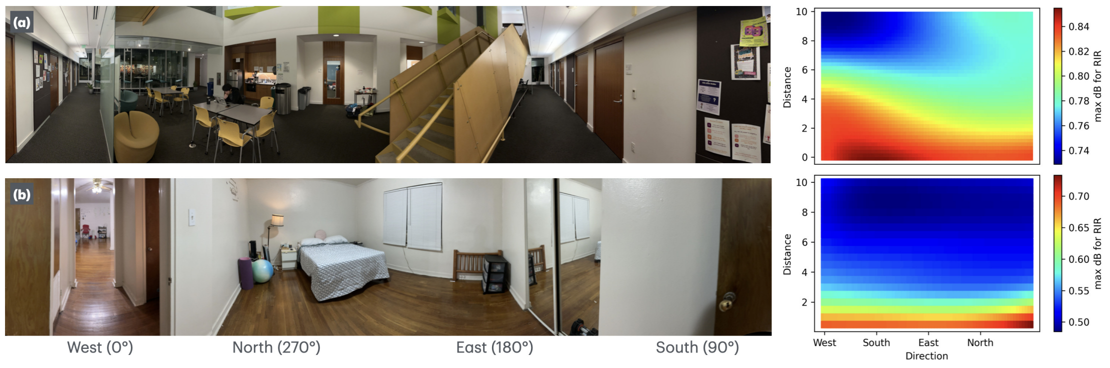


FIGURE 5.7: We present our ANP model's predictions for maximum decibels of room impulse response (max dB RIR) with real-world panorama scans. Note that max dB RIR is normalized between 0 to 1. **Top**: a large space in an office building. Here the north faces an open space area, with a corridor from west-to-east direction. There is a staircase close enough in the northeast direction that blocks most of the sound. In the south direction is a wall, and office rooms have closed wooden doors and bulletin boards. **Bottom**: a bedroom of an apartment. The west direction leads to a corridor and then the living area till 9.5m from the robot's location. The bed and a side table are placed in the north, with a wall at about 3.5m. Towards the north are windows and some furniture at about 3 m relative to the robot. In the south direction, a wall and an opened wooden door at 0.5m. The sound measurements are reported in Table 5.1. Our model's predictions align with that sound decays slowly in the open space with distance and peaks at shorter distances in corridors due to reverberations. The decay is much faster in a smaller room than in an open office space.

acting as the sound source. We use mobile phones to record panoramic images at the robot's location, measure relative distances with Augmented Reality apps, and record audio at the listener location [103].

How loud will the robot be in my home? We want to evaluate our model's prediction for the loudness measured at different listener locations from a fixed robot location. Recall in Sec. 5.4.3; we discussed acoustic maps for fixed robot and fixed listener locations. We visualize the dense prediction of our model for a few real-world panoramas at a fixed robot location. In Fig.5.7, we show ANP in (a) a large open office space, and (b) a bedroom. We observe several sim-to-real gaps, as noted in Appendix D.1; highlighting the distribution shift between sim and real.

Table 5.1 reports the real decibel sound pressure level (dB SPL) calculated from the audio recorded at different distances in the bedroom environment (as shown in Fig. 5.7 (b)). We notice that the error is greater at greater distances, and the model underestimates dB specifically in the long corridor leading to the open area in this instance; indicating a wide sim2real gap for audio and vision.

Can the robot plan for a quieter path? We want to measure the perceived loudness at a fixed listener location as the robot moves around to reach the target location. A naive solution to reduce robot audio is to increase the distance from the listener. While this heuristic is applicable in open settings, it is most likely sub-optimal in indoor homes.

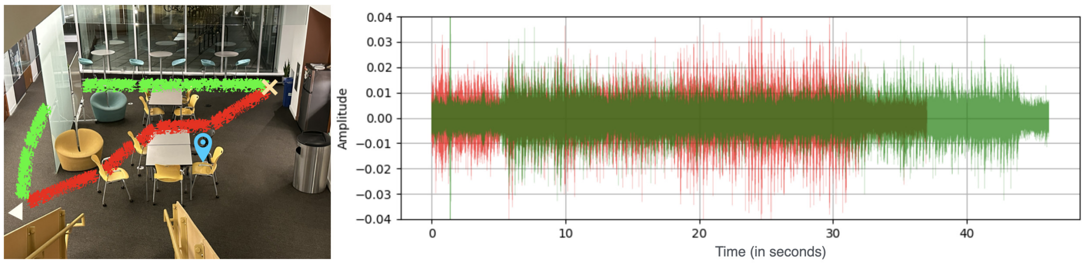


FIGURE 5.8: Real-world audio recorded at the listener location (in blue) as the robot starts (from the triangle) and reaches the target (cross). The red path is approximately the shortest distance (32 seconds), but here, the robot goes quite close to the listener. A longer yet quieter path (46 seconds) is shown in green as it navigates behind the separator, thereby diffusing the robot's movement sound that reaches the listener. Best viewed in color.

TABLE 5.1: Acoustic measurements of Stretch and Unitree Go2 robot actions sounds, over the distance. The audio recordings were taken in a real Bedroom in an Apartment.

Distance	Predicted		Stretch (fast move forward)			Unitree Go2 (running)		
Direction	max dB IR	normalized	Real dB	Pred dB	Error	Real dB	Pred dB	Error
0m, origin	120 dB	1	76	-		98	-	
0.5m S	87.9 dB	0.68	52	51.68	0.32	70	66.64	3.36
1m N	84.8 dB	0.66	49	50.16	-1.16	63	64.68	-1.68
5m W	68.5 dB	0.53	47	40.28	6.72	54	51.94	2.06

Our main idea with the ANAVI framework is that we can use the ANP to estimate the robot's noise level at the listener and incorporate this as a cost alongside the time to find a path that balances the noise cost and path length. This balance is context dependent, e.g. a person working with or without headphones would change the relative impact of the robot's noise. Using existing LLMs or VLMs with prompting can assign a numeric weight on the noise cost which can be used for the planner. To better understand how the robot's movements are perceived by a listener, we consider the following setting. In Fig 5.8, we record the panorama at the different locations in the environment. As discussed in Sec. 5.4.3, we sample a few locations in the environment to record the panorama for acoustic noise predictions at the listener's location. Based on this, we select a quiet path and teleoperate the Unitree Go2 along this path. We also record the shortest path traversed as a control set. For more details, refer to Appendix D.1.1.

5.6 Limitations

Acoustics is inherently complex and often noisy (pun unintended), whether in simulation or the real world. In simulation, the holes in mesh reconstructions often reduce the ray tracing efficiency. In the real world, ambient noise and microphone sensitivity usually interfere with reliable and reproducible acoustic measurements. Material absorption, scattering, transmission, and dampening properties are complex, and the resulting reflections and reverberations are hard to measure and predict.

This work uses simulated environments (Matterport3D scans) and audio impulse response simulation (SoundSpaces). Our approach enables controlled experiments and large-scale data collection. However, as discussed in Sec. 5.5, it may not fully capture the complexities and nuances of real-world environments and acoustic properties. Our current framework does not address the effects of ambient noise. In this work, we focus on first-order principles that directly involve the robot's audio in relation to the audio. However, in real scenarios, the effect of other ambient noises (e.g. a loud TV on) changes how the robot's noise effects the listener. Incorporating other noise sources requires more complex reasoning like sound source separation and noise processing.

Loudness is a subjective and psychological sound pressure; differing based on demographics [23] and prior duration of noise exposure. We use max dB from the impulse response $w(t)$; it is an important yet insufficient aspect to truly measure loudness. In the future, we hope to train models that capture frequency, duration, and other characteristics relevant to perceived sound from room impulse response in simulated and real-world environments.

5.7 Conclusion

We want our future robots to reason about auditory disturbances when operating in human environments like households, hospitals, and offices. Robot assistants in homes and indoor environments generate sound due to their movements or speech through robot's speakers. For any sound produced, robots should be aware of the loudness perceived by the other listeners in the environment.

We present an Acoustic Noise Prediction (ANP) model that uses visual features, along with the listener's distance and directions, to predict the max decibel value of impulse response at the listener. The model enables us to predict the robot's perceived loudness in the environment and to plan routes that lead to less noise perceived by the listener. We show the real-world applicability of our framework to mitigate noise to the listeners. We hope that future robots utilize ANP modeling to adjust their path, velocity, and speaker's volume to adhere to the environmental noise constraints.

Part III

Enabling evaluation and scaling for intent-awareness

Chapter 6

Active visual querying

This manuscript is in preparation for peer review.

6.1 Introduction

We effortlessly adjust viewpoints and object configurations to reveal the information we need. For instance, to read the expiration date on a food package, we instinctively reorient it; to locate a vacuum's charging port, we subtly shift our body and the device until the right angle emerges. Although trivial for us, these seemingly simple acts pose a significant challenge to embodied agents, that must coordinate perception and control to respond to a visual query.

Existing Visual Question Answering (VQA) [71, 57] systems excel at reasoning over diverse images, but they assume the human user supplies perfectly framed views and cannot act to resolve occlusions. We shift this responsibility to the embodied agent itself, to actively manipulating objects and its cameras to uncover hidden details. This capability shift - from passive perception to active information gathering - raises a fundamental question: how should an agent learn to plan and execute physical interactions so as to acquire the precise observations needed to answer a query?

Despite advances in both virtual and robotic benchmarks, none address this full cycle of visual querying plus manipulation. Web-based environments like WebArena [237] or SWE-bench [99] focus on symbolic reasoning within structured documents, while robotics platforms (e.g., RLBench [95], COLOSSEUM [168]) concentrate on narrow manipulation primitives. While mobile-agent benchmarks (such as OpenEQA [131], ALFRED [191], EXCALIBUR [239]) evaluate navigation and interaction, it does so in largely static scenes and does not require the fine-grained manipulation needed to, for instance, orient a can to read its label. What is missing is an integrated testbed where agents must combine active camera control with grounded object handling to satisfy open-ended visual queries.

We present **ActVQ-Arena**, a physically realistic arena for visually grounded query tasks. Given instructions like "find the expiry date", an agent must autonomously move its camera and objects to capture the requested detail in its view. ActVQ-Arena supports a spectrum of embodiment: from a no-robot tabletop setting, through a single-arm Franka Emika Panda platform, to a dual-arm ALOHA system. Our object library spans YCB, Objaverse, and custom photogrammetry scans. Agents are rewarded for efficiently revealing the target information, measured by the visibility and clarity of the queried element in the captured frame.

Our key contributions are:

1. We present ActVQ-Arena, a novel task suite that combines active perception and control to capture relevant information and perform a given task.
2. We design an annotation pipeline that requires a single annotation for an object-task pair and allows reward calculation across varying scene configurations.
3. We study and examine performance of imitation learning, reinforcement learning, and zero-shot visual-language models, highlighting how the tasks require semantic-focus for low-level control.
4. We provide empirical evaluations across diverse objects (selected from YCB, Objaverse) and embodiments (robot-free setting, single-arm (franka) and bimanual (Aloha) environments).

6.2 Related Work

Embodied visual querying sits at the crossroads of VQA, active perception and robotic manipulation. Early VQA methods assume a perfectly framed view and answer free-form questions about visible content (e.g., [9, 7]), while Embodied QA and ALFRED combine vision with navigation and planning but focus on spatial reasoning ("find the object") rather than fine-grained label reading and use fixed instruction sets [45, 191]. Active vision and next-best-view techniques [109] optimize camera motion for geometric reconstruction under known 3D models [42], and visual servoing brings objects into view via hand-engineered controllers, in contrast to our RGB-only learned policies.

Learning-based control includes behavior cloning with diffusion policies [39] and Pi-0–style VLA networks [15], which excel in static settings but lack robustness to novel objects

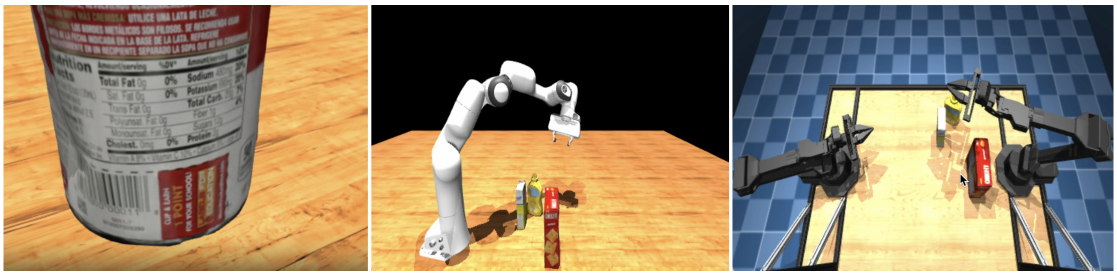


FIGURE 6.1: Environment tiers. (left) robot-free for direct control on camera and object pose, (middle) Single-arm Franka, (right) Bimanual ALOHA

and instructions. Recent work leverages human teleoperation for end-to-end multimodal policies that outperform passive BC models [220], and integrates learned gaze policies into perception–action loops for improved manipulation [105]. Large VLMs like GPT-4V offer powerful semantic priors for zero-shot planning, though translating high-level plans into low-level robot control remains challenging [152].

We bring these threads together by proposing an embodied visual querying benchmark that evaluates diffusion policies, VLA finetuning, RL agents and VLM planners across consistent splits, embodiment tiers and generalization regimes.

6.3 Problem Setting

We model our task as a partially observable Markov decision process (POMDP). This formulation captures key challenges such as occlusions, sensor noise, and limited fields of view. Each episode unfolds in a scene containing one or more moving objects and begins with a language instruction l_i (e.g., 'find the expiration date' or 'find the ingredients').

Formally, the environment is a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{R}, \gamma)$, where the true state $s_t \in \mathcal{S}$ is not directly observable. At each timestep t, the agent receives one or more RGB images $v_t^c \in \mathbb{R}^{H \times W \times 3}$ from camera streams indexed by $c \in C$. It may also observe proprioceptive inputs or a short history of past poses. The action $a_t \in \mathcal{A}$ consists of 6-DoF pose commands for at least one camera, denoted x_{cam}, and one or more object poses $x_{\text{obj},k}$. The reward $\mathcal{R}(s_t, a_t)$ is positive when the requested information (e.g. expiry date) is fully visible in the current view v_t, and is otherwise zero or negatively shaped to penalize extra steps. The agent's goal is to maximize the cumulative reward by efficiently aligning camera and object poses to reveal the visual information queried in as few time steps as possible. More details in Appendix E.1.

6.4 ActVQ-Arena

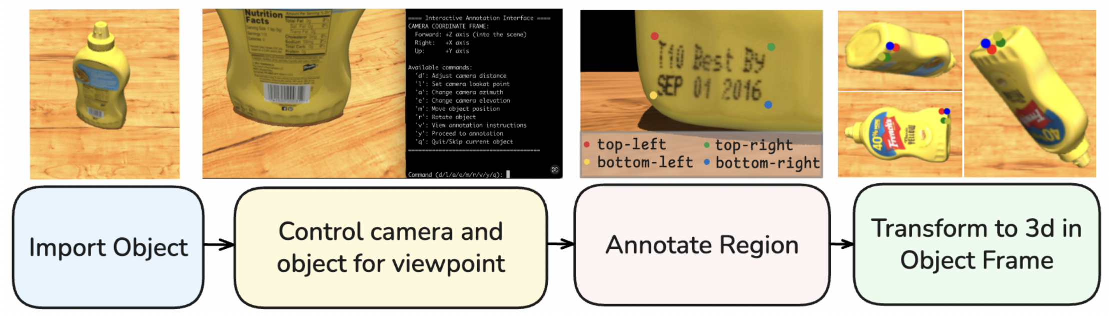
ActVQ-Arena is designed to support diverse embodiments, task complexities, and evaluation setups. We focus on modularity, scalability, and reuse of annotation across tasks, robots, and camera configurations.

Environment Tiers. We define three tiers of embodiment to evaluate agents under increasingly challenging conditions. The *No-Robot* (Easy) tier involves static tabletop scenes where only the camera can be moved; this setting is ideal for abstract perception planning agents and is not tied to low-level manipulation capabilities of a robot like grasping size, payload, etc. The *Single-Arm* (Medium) tier introduces a Franka Emika Panda robot, equipped with wrist-mounted and/or external cameras. Here, the agent controls the robot joints with a 7-dim action vector, navigating moderate physical constraints to capture the information on either of the cameras. The *Bimanual* (Hard) tier uses a dual-arm ALOHA setup with primarily left and right wrist camera views, and optionally overhead and worm-eye view for lesser occlusion. The action output involves 14-dim joint control.

Object Sets. Our benchmark includes diverse object assets from three sources. The set *YCB objects* consists of movable items from the real world with well-defined geometry and physical properties. *Objaverse* provides large-scale 3D scans with rich visual diversity, capturing variation in texture and shape. We select a few ‘boxed’, ‘bagged’, and ‘canned’ objects from Objaverse for the initial setting.

Annotation Pipeline. To enable precise reward computation, we annotate each object-task pair once by identifying four 3D points - *top-left, top-right, bottom-right*, and *bottom-left* - that define the spatial region containing the relevant information (e.g., nutrition label, expiration date). These annotated points are stored in the local frame of the object and reused in scenes, regardless of the object or the pose of the camera. This annotation strategy enables rapid expansion of the benchmark by supporting consistent, reusable supervision signals across scene variations. Refer to Figure 6.2

At runtime, we first load the simulation scene with arbitrary initial poses for both the object and the camera. Depending on the task, we then retrieve the four pre-computed 3D corner annotations (top-left, top-right, bottom-right, bottom-left) defined in the object’s local frame. Given the depth and segmentation mask for the objects, we determine whether the four points are indeed visible to the camera. Next, we transform these points into world coordinates and project them through the camera matrix onto the 2D image plane. Finally, if all projected points lie within the image bounds, we compute the quadrilateral area using Shoelace’s Theorem, normalize it by the total image area and issue a reward proportional to the fraction of the label region that remains visible.


top-left
top-right
bottom-left
bottom-right
Import Object
Control camera and object for viewpoint
Annotate Region
Transform to 3d in Object Frame

```
==== Interactive Annotation Interface ====
CAMERA COORDINATE FRAME:
  Forward: +Z axis (into the scene)
  Right:   +X axis
  Up:      +Y axis

Available commands:
  'd': Adjust camera distance
  'l': Set camera lookat point
  'a': Change camera azimuth
  'e': Change camera elevation
  'm': Move object position
  'r': Rotate object
  'v': View annotation instructions
  'y': Proceed to annotation
  'q': Quit/Skip current object
========================================

Command (d/l/a/e/m/r/v/y/q):
```

FIGURE 6.2: Annotation pipeline for ActVQ-Arena, showing the four stages: (1) Object Import into MuJoCo, (2) Navigation to desired view by controlling the camera distance, look-at, azimuth, elevation and object's translation and rotation, (3) Region Annotation with top-left, top-right, bottom-right and bottom-left points marked by the user, and (4) Transform these to 3D points using depth and camera intrinsics, finally to Object Frame for later use in reward at runtime computation.

Metrics. Agent performance is evaluated across two main axes. *Success Rate* measures the fraction of episodes in which the queried information is successfully revealed. *Time-to-Success* tracks the normalized number of steps taken before success is achieved, compared to the shortest distance path. In addition, we measure generalization gaps with unseen objects $\Delta_{obj} = SR_{train} - SR_{unseen-objects}$ and instructions $\Delta_{instr} = SR_{train} - SR_{unseen-instructions}$.

6.5 Baselines

We evaluate four classes of control policies for each task type under the same train–test split (see Appendix E.4). Depending on the environment suite, control outputs may be end-effector velocities, joint targets, or camera-pose adjustments. Each model is assessed in three generalization regimes: (i) seen objects with novel initial configurations, (ii) unseen objects under instructions similar to training, and (iii) unseen objects with unseen instructions.

Random. Actions (or camera movements) are sampled uniformly at random within the allowed range, providing a lower-bound reference on task success rates.

Behavior Cloning (DP, VLA). We collect expert demonstrations pairing target camera views with robot actions across a diverse set of object–task instances. From this dataset we train two variants: (1) *Diffusion Policies* (DP), which iteratively denoise coarse trajectory seeds into executable control sequences [39]; and (2) a Vision–Language Action (VLA) model, instantiated as a finetuned Pi-0 network that maps visual and textual inputs directly to motor commands [15]. Both variants are evaluated on all three generalization regimes to measure robustness to novel object configurations and instructions.

Table 6.1: Quantitative results for all methods in the No-Robot tier. **SR**: Success Rate, **T2S**: Time-to-Success, Δ **Unseen-Objects** : Change in Success Rate with Unseen Objects, Δ **Unseen-Instructions (%)**: Change in Success Rate with Unseen Instructions (%)

Method	SR (%)	T2S (steps)	Δ-obj (%)	Δ-instr (%)
Random	1.2	–	–	–
GPT-4V Planner	41.3	35.8	-6.1	-3.0
PPO (RL)	55.2	25.1	-17.8	–
Pi-0 VLA (BC)	63.7	16.7	-11.5	-7.0
Diffusion Policies	**78.4**	**12.3**	-8.3	-5.0

Reinforcement Learning (PPO). We train Proximal Policy Optimization (PPO) [183] agents in dense and sparse reward formulations. Due to the high sample complexity, these agents are presumed to have experienced all configurations of seen objects during training and are therefore only evaluated in regimes (ii) and (iii), which involve unseen objects.

VLM-based Zero-Shot Planner (e.g., GPT-4V, Gemini 2.0 Flash). We use the frontier models for single-query action based on the visual context and control API as function-calling tools (GPT-4V [151], Gemini 2.0 Flash [69]). Additionally, we designed multi-stage prompting workflows that decompose each task into perception, reasoning, planning, and backtracking queries to Vision–Language Models (VLMs). These models leverage extensive world knowledge and require no task-specific data collection, but they frequently fail to convert high-level plans into precise, low-level action commands.

Implementation Details. All behavior cloning are trained with 100 trajectories per task for 500k gradient updates and RL policies upto 10 M (PPO) steps using Adam with a learning rate of 3×10^{-4} and minibatch size of 256. VLM prompting pipelines perform about six queries per action step, with backtracking triggered if model confidence falls below 0.7.

6.6 Results and Discussion

Quantitative comparison across methods. We begin by comparing all four classes of control policies on the core "find expiry date" and "read ingredients" perceptual tasks under the No-Robot tier. As shown in Table 6.1, Diffusion Policies (DP) achieve the highest overall success rate (78.4%), significantly outperforming both the finetuned Pi-0 VLA model (63.7%) and the RL baselines (PPO: 55.2%). The GPT-4V zero-shot planner attains a modest 41.3% success rate, reflecting its strong semantic reasoning but limited low-level control precision. The random control remains near zero, which

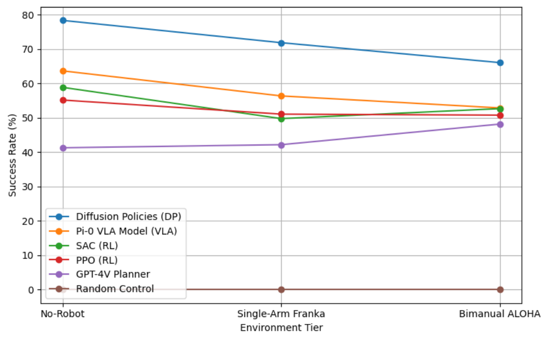


FIGURE 6.3: Success Rate based on Environment Tiers

confirms the non-triviality of the task. Across all learned methods, behavior cloning variants demonstrate substantially faster Time-to-Success (mean 12.3 steps for DP, 16.7 for VLA) compared to RL (25.1 for PPO) and VLM planners (35.8).

Breakdown by environment tier. When moving from No-Robot to the Single-Arm Franka setting, all methods see a moderate drop in performance due to added manipulation constraints (Figure 6.3). DP retains a strong 71.9% success rate, whereas VLA falls to 56.4% and RL methods to approximately 50%. In the Bimanual ALOHA tier, DP still leads at 66.1%, but the gap narrows: GPT-4V improves to 48.2%, likely benefiting from its ability to reason over multiple camera streams. These results highlight that active control methods generalize reasonably across embodiments but that physical constraints increasingly challenge both imitation and RL approaches.

Generalization to novel objects and instructions. For the unseen-objects, we use test items from Objaverse rather than YCB. Behavior cloning exhibits robust transfer; DP's success rate only declines by 8.3 percentage points, and VLA by 11.5. In contrast, RL policies degrade more sharply (PPO: –17.8), suggesting overfitting to training object geometries. Zero-shot VLM planners maintain a relatively stable performance (–6.1), underscoring their strong semantic priors but continued struggles with fine-grained control. For unseen instructions (e.g., "locate the barcode" rather than "find expiry date"), DP and VLA drop by 5–7%, while GPT-4V surprisingly holds within 3% of its baseline, reflecting its flexible language understanding.

Qualitative examples (success/failure frames). Figure 5 presents representative frames. In successful DP trials, the agent smoothly reorients both object and camera to center the target label before issuing "stop." VLA models occasionally over-rotate, leading to partial occlusions. RL agents often "hesitate", alternating small corrective movements around the correct view. GPT-4V plans exhibit semantically coherent multi-stage sequences ("lift object $\rightarrow$ tilt camera $\rightarrow$ read region") but suffer from execution noise, causing the camera to drift outside the annotation box. These examples illustrate the complementary strengths of data-driven control and language-driven planning, and point to avenues for hybrid approaches.

6.7 Discussion

Our evaluation is currently limited to single task per policy and no clutter in the scene, and metrics aggregated across tasks. While these are initial reuslts, the actual benefit of the benchmark is to combinatorially scale up the evaluation and data for training in simulation and enable real world testing.

We showcase distinct strengths and limitations across agent classes. Behavior cloning with diffusion policies excels in familiar, static scenarios but fails to generalize to novel objects or instructions. This is a common issue noted in large-scale, multi-task benchmarks [107]. Vision–Language–Action models like Pi-0 match fine-tuned performance with minimal data and maintain robustness on unseen tasks, suggesting that combining diffusion's fine-grained control with Pi-0's semantic generality could yield more versatile agents. End-to-end RL adapts to new situations but remains sample-inefficient in sparse-reward settings, while VLM planners provide powerful high-level guidance yet still require effective grounding to reliable low-level controls.

ActVQ-Arena's simulation focus offers two key advantages: rapid integration of diverse, real-world object meshes and reusable annotations across task variations, enabling extensive, statistically significant benchmarking far more efficiently than physical trials. Nonetheless, sim-to-real transfer will demand careful mitigation of sensor and actuator noise discrepancies; future work should iterate on digital-twin fidelity and incorporate onboard perception uncertainties to close this gap. By highlighting the trade-offs between imitation, reinforcement, and language-guided approaches, ActVQ-Arena lays the groundwork for developing agents that actively seek and accurately perceive visual information in both virtual and real environments.

6.8 Conclusion and Future Work

We present ActVQ-Arena, a unified benchmark for embodied visual querying, evaluating diffusion policy, vision language action (VLA) fine-tuning, end-to-end RL and VLM planners under consistent splits, embodiment tiers, and generalization regimes. Our results reveal that while BC models excel on seen objects, they falter on new ones; RL adapts but is sample-hungry; and VLMs offer strong semantics yet need bridging to low-level control. By highlighting the value of end-to-end multimodal training, learned viewpoint selection, and tighter planning–execution integration, ActVQ-Arena paves the way for robots that actively seek and read visual information.

ActVQ-Arena is designed for extensibility. New task types and language instructions can be added by associating them with annotated regions. Additional object models can be incorporated by following the annotation pipeline. The benchmark also accommodates new embodiment setups, robot platforms, and sensor modalities, making it suitable for a wide range of research in vision-language robotics. Future work can explore more goal-conditioned tasks as the object assets scale in diversity and improve mesh quality, for example, "find the can with the highest sodium content", and "sort the clothes according to washing instructions".

However, extensibility should not be mistaken for completeness. Although designed to be extensible, the benchmark has not yet captured several aspects of multimodal perception that are critical in real-world interactions. Currently, we do not reliably simulate objects with varying mass distributions (e.g., a half-filled can). We also omit tasks based on material properties like tactile sensing and contact-rich dynamics, and those requiring audio cues arising from physical interactions for perception and reasoning. Bridging this gap will require advances in fast rendering and simulation for physics-based modeling, tactile sensing, and audio–visual integration.

Chapter 7

Conclusion

Parts of the Chapter 7.3 have appeared in the survey:
Yafei Hu*, Quanting Xie*, **Vidhi Jain***, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim et al. "Toward general-purpose robots via foundation models: A survey and meta-analysis." arXiv preprint arXiv:2312.08782 (2023).

7.1 Summary of Contributions and Next Steps

This thesis addresses three fundamental challenges in enabling robots to understand and act upon human intent in embodied settings. Intent-aware embodied agents require: (i) methods that can follow explicit human intents (preferences, demonstrations, instructions), (ii) mechanisms that respect implicit social constraints, and (iii) evaluation settings that force agents to actively seek information rather than passively receive it. Concretely, the thesis contributes: TTP (Transformer Task Planner) for preference-aware sequencing; Vid2Robot for end-to-end video-conditioned manipulation; SLAP for spatial-language policies over point clouds; ANP for acoustic-awareness; and ActVQ-Arena, a benchmark for active visual querying across embodiments.

First, we investigate methods for learning from explicit intent. In Chapter 2, we introduce the **Transformer Task Planner (TTP)** [92], a preference-based sequential policy that infers both ordering and placement preferences from a single visual demonstration. We validate TTP on a simulated dishwasher-loading task characterized by diverse user preferences, showing preference-adherent task planning in simulation and deployment in a Franka robot.

Future directions include: (i) scaling beyond tabletop semantics. The object-centric representation is handcrafted for the dishwasher domain; and the TTP approach can be extended to capture spatial-temporal preferences in long-horizon household workflows (laundry, pantry). (ii) handling language preferences. Currently, preferences are only conveyed via a visual prompt; and we should move to language-described and multimodal preferences. (iii) improving robustness to partial/failed steps. The loop failure mode suggests the value of self-correction (detect no-progress, ask for help, or re-plan). Also, the steep drop on increased object counts suggests capacity/credit assignment limits indicates that we need a curriculum on object density or memory mechanisms.

Chapter 3 presents **Vid2Robot** [93], an end-to-end video-conditioned policy that maps raw pixel observations directly to actions. By training in paired human and robot demonstrations – varying in lighting, background, and object distractors – Vid2Robot employs cross attention to learn a shared latent representation that supports robust action decoding. Empirically, Vid2Robot outperforms the BC-Z baseline by approximately 20% when prompted with human videos and maintains a 17% advantage in cross-object transfer, underscoring its capacity to generalize motions to novel object instances.

Future directions include: (i) studying how the data mixture affects the performance. Currently, we used a 50-50 mix of human and robot data, but these changes in this mixture across tasks are not well understood. (ii) developing alignment strategies for human and robot videos, (iii) improving action space for contact: The policy outputs discretized action bins which is helpful for stable training but possibly constraining fine contact; a follow-up with continuous diffusion action heads will be useful for dexterous tasks.

In Chapter 4, we develop **SLAP** [155], a structured fusion model that integrates point-cloud geometry with linguistic semantics to predict continuous end-effector trajectories for spatial language–conditioned manipulation. We evaluated SLAP on both a fixed manipulator and a mobile robotic platform, constituting the first end-to-end mobile manipulation study of its kind, and demonstrated that SLAP surpasses the state-of-the-art PerAct system in both settings.

Future directions include: (i) scaling with reduced dependence on detectors (e.g., Detic), which likely bottlenecks recall; end-to-end semantic tokenization of point clouds would make the system more graceful under open-set clutter. (ii) studying long-horizon chaining, especially on error accumulation and online re-planning thresholds. (iii) developing systematic cost/latency/accuracy benchmarks, that inform choices, like adapter fine-tuning vs. in-context approaches.

Second, we explore the incorporation of implicit human constraints into robot planning. Chapter 5 introduces the **Acoustic Noise Prediction (ANP)** [87] model, which uses visual context and listener geometry to estimate the maximum decibel level perceived by nearby humans. By integrating ANP into path and velocity planning, robots can select quieter trajectories and adjust speaker volume to minimize disturbance, thus exhibiting more considerate behavior in domestic environments.

Future directions include: (i) developing metric alignment with human perception. The peak dB (max amplitude) may not capture perceived annoyance. (ii) better profiling for material acoustics to bridge the multimodal sim-to-real gap, and (iii) evaluating closed-loop cost for end-to-end task performance under a noise budget to quantify real trade-offs.

Third, we establish a framework for a human-centered, scalable evaluation of embodied intelligence. In Chapter 6, we present **ActVQ-Arena** [88], a unified benchmark for embodied visual querying in which agents must locate fine-grained information, such as expiration dates or ingredient labels, under partial observability and varying viewpoints. ActVQ-Arena provides consistent evaluation splits and generalization regimes for diffusion policies, vision-language action models, end-to-end reinforcement learning, and vision–language model planners. Our results reveal that behavior cloning excels on familiar objects but degrades on novel ones; reinforcement learning adapts effectively but at significant sample complexity; and vision–language planners offer strong semantic reasoning yet require tighter integration with low-level control. ActVQ-Arena thus highlights the importance of joint perception–planning–execution training and charts a clear path for future benchmarks.

Future directions include: (i) handling sim-to-real, fidelity gaps, and adding sensor noise models and latency would increase external validity, (ii) training hybrid agents like explicit VLM-planner combined with a diffusion-controller baseline to test the "best of both" hypothesis. (iii) training RL with options or skills to balance between not having to hard-code strategies and learning everything from scratch.

7.2 Key Takeaways

The work presented in this thesis demonstrates that end-to-end multimodal learning, from raw sensory input to motor output, is critical to achieve flexible generalization in robotic systems. Models such as TTP, Vid2Robot and SLAP illustrate how single demonstration prompting, video conditioning, and structured point cloud and language fusion enable robots to adapt to new preferences, environments, and object instances without reliance on brittle, hand-crafted pipelines.

Furthermore, capturing implicit contextual cues, exemplified by the Acoustic Noise Prediction model, allows robots to anticipate and respect unspoken human constraints, moving beyond literal command execution toward more socially aware behavior.

Active perception emerges as another central theme: Effective embodied intelligence requires not only selecting the correct action but also the appropriate viewpoint from which to acquire necessary information. The ActVQ-Arena benchmark underscores that coupling viewpoint selection with downstream reasoning is indispensable for tasks demanding fine-grained visual understanding.

Finally, we need to expand our notions of generalization. While the community agrees that leveraging diverse training signals (e.g., paired videos, language priors) and inductive biases in modeling (e.g., attention mechanisms, spatial structure) is generally important, we don't have thorough benchmarks to compare these contributions with other classical approaches. Like ActVQ-Arena, we need scalable evaluation suites that mirror real human needs and expectations, like long-horizon goals, fine-grained queries, understanding explicit and implicit intents and asking relevant clarifications. Designing open and dynamic evaluations for these objectives will enable our community to build robust hybrid policy designs that combine the strengths of behavior cloning, reinforcement learning, and vision–language planning, diagnose weaknesses systematically and drive rapid progress towards training collaborative robot policies for homes.

7.3 Broad Open Challenges

Despite huge strides in large language models (LLMs) like ChatGPT and GPT-4V, robotics has not seen a comparable rate of progress. Even before the rise of AI, computing evolved rapidly, from warehouse-sized machines in the 1960s to personal devices in every pocket and on every wrist by 2024. So why haven't we seen a similar transformation in robotics? What is holding us back from building the equivalent of a "RoboGPT"?

Limited Data. Deep learning success has been mostly about scaling up data. However, robotic data is very scarce as compared to the text, images and videos on the internet. We see two paradigms evolving to address the data scarcity in robotics. First, simulation is improving to offer more versatile 3d virtual worlds, faster rendering, and realistic physics. Many works [205, 133, 143, 180, 202, 167] propose generating synthetic data in simulated environments for factory automation, home robots and self-driving. While synthetic data might enable training, it does not always enable realistic transfer to the real world. As the second paradigm, therefore, we are seeing larger concerted efforts for collecting real world robot demonstrations. These datasets are collected for self-driving research [67, 135, 200], as well as general-purpose robot manipulation [100, 114, 76, 150].

Collecting robot data from human demonstration is expensive [19] but it ensures a well trained policy on such data is directly deployable in the real world. Overall, the evaluation of a robot policy can be thought of in two phases. First, simulation offers indefinite amounts of controlled data to test whether what modeling approach works the best compared to the baselines. Second, real-world is useful to collect more data for fine-tuning, online adaptation, and often for training policies on broad diversity of tasks, where simulated assets can be time-consuming to design, have inaccurate physics and material properties. Real-world evaluation is gold-standard for evaluating whether the policy can be deployed and how robustness it is environmental perturbations.

Limited Out-of-distribution Generalization. Robots often struggle with accurate perception and corresponding motion. Training with behavior cloning is the most widely used approach, but the error in this case grows quadratically with timesteps [196]. The recent works evaluate robot policies for generalization with different positional changes but largely confined within same workspace as in the training data. These evaluation setups are often not reproducible, limiting the ways to compare and extend the existing work. More importantly, the evaluations do not reflect what we care about - that is, whether the robot policies can adapt to realistic changes in (i) objects for interaction and those that are mere distractors, (ii) lighting, background, reflections and shadows.

Limited Adaptability to Physical Changes in Embodiment. From flipping a light switch with a pen to maneuvering down a staircase with a broken leg, the human brain demonstrates versatile and adaptable reasoning. It is a single unit that controls perceptual understanding, motion control, and dialogue capabilities. For motion control, it adapts to the changes in the embodiment, due to tool use or injury. This adaptability extends to more profound transformations, such as individuals learning to paint with their feet or mastering musical instruments with specialized prosthetics. Recent work shows that foundation models can be deployed for navigation on various robot platforms [185], including wheeled robots and quadrupedal robots. We also witnessed the manipulation foundation model used in different manipulators [241, 11] which can be used across different robotic platforms, ranging from tabletop arms to mobile manipulators.

One of the key open research questions is how the foundational models of robotics should enable motion control in different physical embodiments. Robot policies deployed in homes and offices must be robust to mechanical motion failures, such as sensor malfunctions or actuator breakdowns, ensuring continued functionality in challenging environments. Furthermore, robotic systems must be designed to adapt to a variety of tools and peripherals, mirroring the human capability to interact with different instruments for specific tasks and physical tool uses. While some works [172, 207, 170] have explored learning representations for diverse tool use, these approaches are yet to be scaled up with foundation models.

Limited Grounding. Grounding is an effective medium or interface that bridges concepts and robot actions. Existing interfaces use natural language [5, 3] and code [116, 193], are limited in their applicability for low-level robot actions. While concepts can be articulated through language and code, they are not universally applicable to nuances such as dexterous body movements. Furthermore, these interfaces often depend on predefined skill libraries that are not only time-intensive to develop but also lack generalization to new environments. Using reward as an interface [228, 218, 128] may alleviate some of the generalization issues in simulations by acquiring skills dynamically. However, the time-consuming and potentially unsafe nature of training RL in the real world makes this method quite tedious, if not impractical.

Current work focuses on mostly unimodal notion of grounding, like mapping the word to meaning to a more holistic grounding of multiple sensory modalities. Approaches that rely solely on visual data [64] may capture certain physical properties such as material, transparency, and deformability. Yet, they fall short in grasping concepts like friction, which requires interactive data with proprioceptive feedback, or the scent of an object, which cannot be acquired without additional modalities such as olfaction.

Grounding can also be viewed from the embodiment perspective. The same task may necessitate distinct actions based on the robot's embodiment; for example, opening a door would require drastically different maneuvers from a humanoid robot compared to a quadruped. Current research on grounding often emphasizes environmental adaptation while giving less consideration to how embodiment shapes interaction strategies.

Limited Ways for Task Specification Understanding the task specification and grounding it in the robot's current understanding of the world is a critical challenge to obtaining generalist agents. Often, these task specifications are provided by users with limited understanding of the limitations on the robot's cognitive and physical capabilities. This not only raises questions about what the best practices are for providing these task specifications, but also about the naturalness and ease of crafting these specifications. Understanding and resolving ambiguity in task specifications, conditioned on the robot's understanding of its own capabilities, is also challenging. Foundation models, again, are a promising solution for this challenge: task specification can be formulated as language prompts [19, 241, 5], goal images [43], and rewards for policy learning [203, 128].

7.4 Future Directions

Below I discuss four research areas relevant for lowering barriers in human-robot communication, namely: (i) integrating multimodal signals, (ii) grounding in pragmatic reasoning, (iii) adapting to individuals, and (iv) optimizing for human-centric outcomes.

Multimodal prompting. Integrating diverse sensory modalities beyond video and language into how we prompt the robot policies will be a key enabler in human-like communication. For instance, tactile feedback can reveal information about material properties, texture, or resistance, while audio perception can provide cues about context (e.g., footsteps, clattering dishes, or alarms). Unifying these channels into coherent multimodal prompting would allow robots not only to parse user preferences expressed in natural language, but also to fuse them with subtle sensory cues from the environment. This could support higher-level inference about tasks that require sensitivity to both the physical and social environment.

Pragmatic grounding. Developing mechanisms for end-to-end grounding into pragmatics will be crucial for autonomous robots around humans. Today's systems often excel at syntactic or semantic alignment as in Chapter 4 and 5, but struggle with pragmatic reasoning: balancing competing objectives, interpreting indirect requests, and knowing when to seek clarification. Robust human-AI collaboration requires models that can recognize underspecified goals, resolve them by actively engaging with the user, and gracefully handle trade-offs between efficiency, safety, and user comfort. Such pragmatic reasoning capabilities would transform robots from passive executors of commands into active partners in achieving shared goals.

Personalization. Unlike static deployments, real-world household robots must operate in diverse homes, with users who differ in preferences, routines, and communication styles. Adaptation on-the-fly, whether through few-shot updates in Chapter 3, preference learning in Chapter 2, or lightweight reinforcement from user feedback, could enable systems that become more attuned over time. It is not just about execution efficiency in homes, but establishing trust: by recognizing personal habits and tailoring behaviors accordingly. Overall, personalization and customization strategies in robot learning can enhance long-term interaction quality and foster a sense of reliability and companionship.

Human-centric metrics. Finally, progress will depend on defining and adopting metrics that reflect human-centric criteria such as readability of robot behavior, comfort of interaction, and alignment with actual household utility. Traditional benchmarks often emphasize technical performance, but scaling robots into homes requires measuring what matters to people: *Do interactions feel natural? Are tasks completed in a way that minimizes disruption? Can users easily interpret and anticipate robot actions?* Establishing such metrics will not only drive more user-aligned evaluation but will also accelerate iterative improvements in dynamic, real-world environments.

These directions point to a vision of robots that are not only technically capable but socially intelligent and contextually grounded. Addressing the challenges will be central to realizing the long-standing goal of robots that truly understand, anticipate, and fulfill human intent in the complexity of everyday life.

Part IV

Appendix

Appendix A

Appendix: From demonstrations to preferences for dishwasher loading

A.1 Hardware Experiments

Real-world prompt demonstration

Here we describe how we collected and processed a visual, human demonstration in the real-world to treat as a prompt for the trained TTP policy (Fig. A.1). Essentially, we collect demonstration pointcloud sequences and manually segment them into different pick-place segments, followed by extracting object states. At each high-level step, we measure the state using three RealSense RGBD cameras[106], which are calibrated to the robot frame of reference using ARTags [62]. The camera output, extrinsics, and intrinsics are combined using Open3D [236] to generate a combined pointcloud. This pointcloud is segmented and clustered to give objects' pose and category using the algorithm from [214] and DBScan. For each object point cloud cluster, we identify the object pose based on the mean of the point cloud. For category information we use median RGB value of the pointcloud, and map it to apriori known set of objects. In the future this can be replaced by more advanced techniques like MaskRCNN [75]. Placement poses are approximated as a fixed, known location, as the place action on hardware is a fixed 'drop' position and orientation. The per step state of the objects is used to create the input prompt tokens used to condition the policy rollout in the real-world, as described in Section 2.4.2.

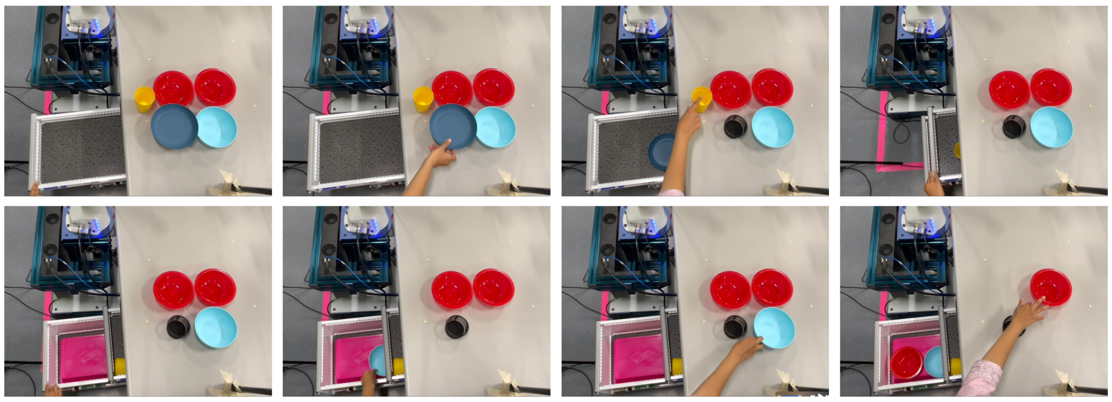


FIGURE A.1: Human demonstration of real-world rearrangement of household dishes.

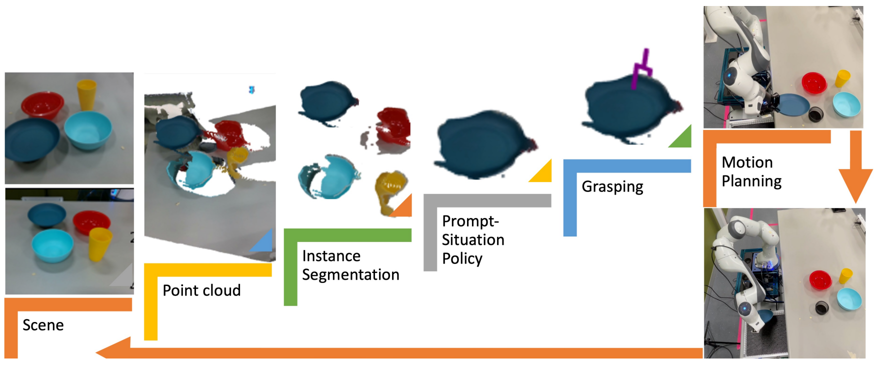


FIGURE A.2: Pipeline for Real Hardware Experiments

Hardware policy rollout

We zero-shot transfer our policy π trained in simulation to robotic hardware, by assuming low-level controllers. We use a Franka Panda equipped with a Robotiq 2F-85 gripper, controlled using the Polymetis control framework [121]. Our hardware setup mirrors our simulation, with different categories of dishware (bowls, cups, plates) on a table, a “dishwasher” (cabinet with two drawers). The objective is to select an object to pick and place it into a drawer (rack) (see Fig. A.1).

Once we collect the human prompt demonstration tokens, we can use them to condition the learned policy π from simulation. Converting the hardware state to tokens input to π follows the same pipeline as the ones used for collecting human demonstrations. At each step, the scene is captured using 3 Realsense cameras, and the combined pointcound is segmented and clustered to get object poses and categories. This information along with the timestep is used to generate instance tokens as described in Section 2.3 for all objects visible to the cameras. For visible already placed objects, the place pose is approximated as a fixed location. The policy π, conditioned on the human demo, reasons about the state of the environment, and chooses which object to pick. Next, we use a grasp generator from [60] that operates on point clouds to generate candidate grasp locations on the chosen object. We filter out grasp locations that are kinematically

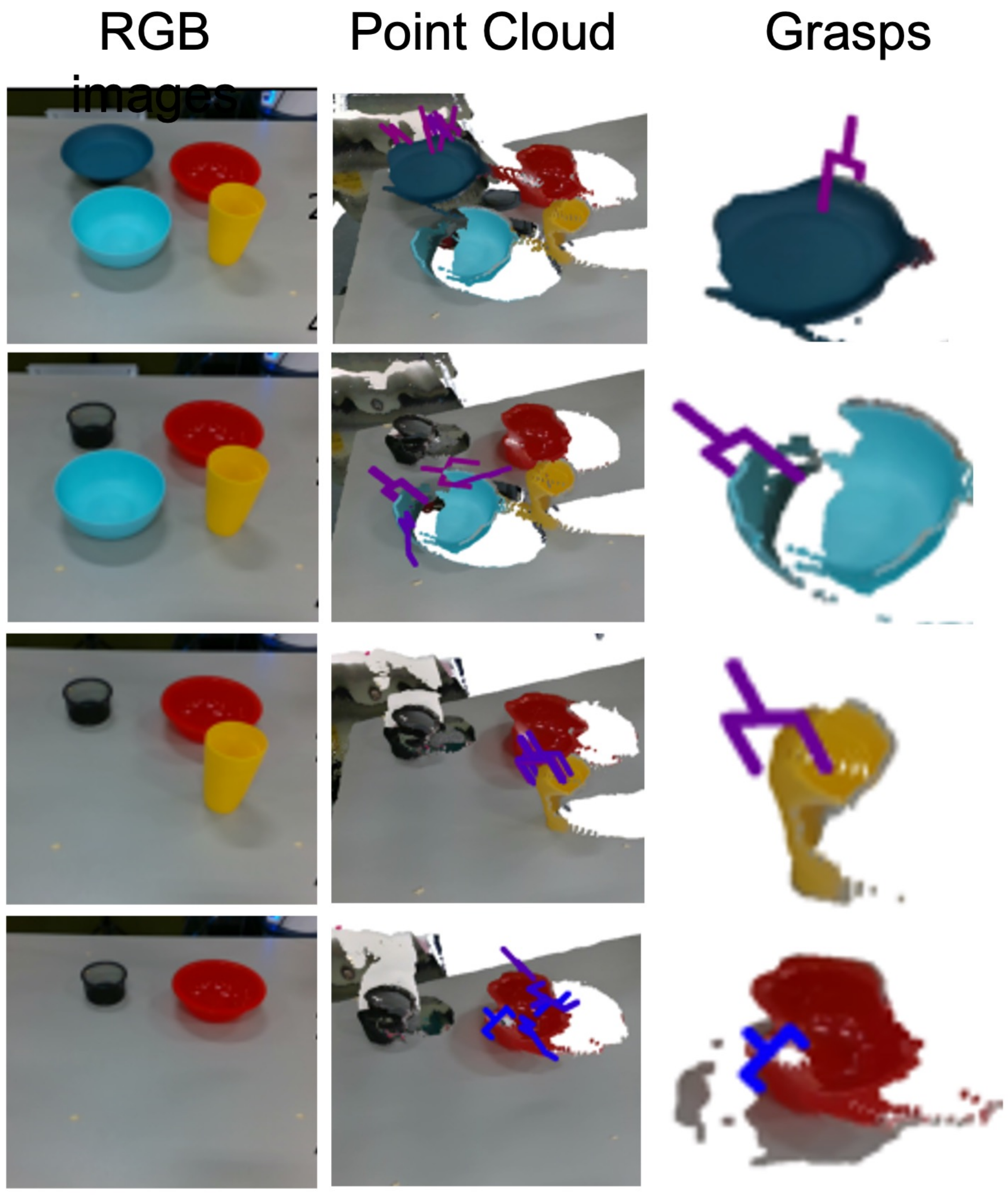


Figure A.3: Point cloud and grasps for different objects during policy rollout.

not reachable by the robot, as well as grasp locations located on points that intersect with other objects in the scene. Next, we select the top 5 most confident grasps, as estimated by the grasp generator, and choose the most top-down grasp. We design an pre-grasp approach pose for the robot which is the same final orientation as the grasp, located higher on the grasping plane. The robot moves to the approach pose following a minimum-jerk trajectory, and then follows a straight line path along the approach axes to grasp the object. Once grasped, the object is moved to the pre-defined place pose and dropped in a drawer. The primitives for opening and closing the drawers are manually designed on hardware.

The learned policy, conditioned on prompt demonstrations, is applied to two variations of the same scene, and the predicted pick actions are executed. Fig.A.3 shows the captured image from one of the three cameras, the merged point cloud and the chosen object to pick and selected grasp for the same. The policy was successful once with 100% success rate, and once with 75%, shown in Fig.2.1. The failure case was caused due to a perception error – a bowl was classified as a plate. This demonstrates that our approach (TTP) can be trained in simulation and applied directly to hardware. The policy is robust to minor hardware errors like a failed grasp; it just measures the new state of the environment and chooses the next object to grasp. For example, if the robot fails to grasp a bowl, and slightly shifts the bowl, the cameras measure the new pose of the bowl, which is sent to the policy. However, TTP relies on accurate perception of the state. If an object is incorrectly classified, the policy might choose to pick the wrong object, deviating from the demonstration preference. In the future, we would like to further evaluate our approach on more diverse real-world settings and measure its sensitivity to the different hardware components, informing future choices for learning robust policies.

Transforming hardware to simulation data distribution

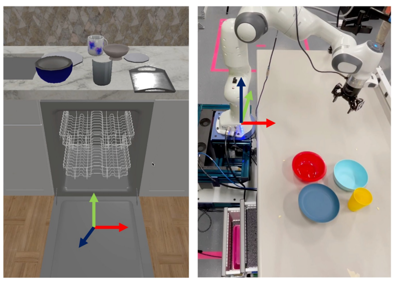


FIGURE A.4: Coordinate Frame of reference in simulation (left) and real world setting (right). Red is x-axis, green is y-axis and blue is z-axis.

The policy trained in simulation applies zero-shot to real-world scenarios, but it requires a coordinate transform. Fig. A.4 shows the coordinate frame of reference in simulation and real world setting. Since our instance embedding uses the poses of objects, it is dependant on the coordinate frame that the training data was collected in. Since hardware and simulation are significantly different, this coordinate frame is not the same between sim and real. We build a transformation that converts hardware measured poses to the simulation frame of reference, which is then used to create the instance tokens. This ensures that there is no sim-to-real gap in object positions, reducing the challenges involved in applying such a simulation trained policy to hardware. In this section we describe how we convert the real world coordinates to simulation frame coordinates for running the trained TTP policy on a Franka arm.

We use the semantic work area in simulation and hardware to transform the hardware position coordinates to simulation position coordinates. We measure the extremes of the real workspace by manually moving the robot to record positions and orientations that define the extents of the workspace for the table. The extents of the drawers are measured by placing ARTag markers. We build 3 real-to-sim transformations using the extents for counter, top rack and bottom rack: Let $X \in \mathbb{R}^{3\times N}$ contain homogeneous $xz-$

coordinates of a work area, along its column, as follows:

$$X = \begin{bmatrix} x^{(1)} & x^{(2)} & \cdots \\ z^{(1)} & z^{(2)} & \cdots \\ 1 & 1 & \cdots \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \cdots \end{bmatrix} \tag{A.1}$$

As the required transformation from real to simulation involves scaling and translation only, we have 4 unknowns, namely, $\boldsymbol{a} = [\alpha_x, \alpha_y, x_{trans}, z_{trans}]$. Here α_x, α_z are scaling factors and x_{trans}, z_{trans} are translation offset for x and z axis respectively. To solve $X_{sim} = AX_{hw}$, we need to find the transformation matrix $A = \hat{\boldsymbol{a}} = \begin{bmatrix} \alpha_x & 0 & x_{trans} \\ 0 & \alpha_z & z_{trans} \\ 0 & 0 & 1 \end{bmatrix}$.

$$X_{sim} = \hat{\boldsymbol{a}} X_{hw} \tag{A.2}$$

$$\text{Rewriting the system of linear equations,} \tag{A.3}$$

$$\implies \begin{bmatrix} x_{sim}^{(1)} \\ z_{sim}^{(1)} \\ x_{sim}^{(2)} \\ z_{sim}^{(2)} \\ \vdots \end{bmatrix} = \begin{bmatrix} x_{hw}^{(1)} & 0 & 1 & 0 \\ 0 & z_{hw}^{(1)} & 0 & 1 \\ x_{hw}^{(2)} & 0 & 1 & 0 \\ 0 & z_{hw}^{(2)} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \boldsymbol{a}^T \tag{A.4}$$

$$\tag{A.5}$$

Let the above equation be expressed as $Y_{sim} = Z_{hw} a^T$ where $Y_{sim} \in \mathbb{R}^{2N \times 1}$, $Z_{hw} \in \mathbb{R}^{2N \times 4}$, and $a^T \in \mathbb{R}^{4 \times 1}$. Assuming we have sufficient number of pairs of corresponding points in simulation and real world, we can solve for $\boldsymbol{a}$ by least squares $a = (Z_{hw}^T Z_{hw})^{-1} Z_{hw}^T Y_{sim}$. The height y_{sim} is chosen from a look-up table based on y_{hw}. Once we compute the transformation A, we store it for later to process arbitrary coordinates from real to sim, as shown below.

```
def get_simulation_coordinates(xyz_hw: List[float], A: np.array) -> List:
    xz_hw = [xyz_hw[0], xyz_hw[2]]
    X_hw = get_homogenous_coordinates(xz_hw)
    X_sim_homo = np.matmul(A, X_hw)
    y_sim = process_height(xyz_hw[1])
    X_sim = [X_sim_homo[0]/X_sim_homo[2], y_sim, X_sim_homo[1]/X_sim_homo[2]]
    return X_sim
```

The objects used in simulation training are different from hardware objects, even though they belong to the same categories. For example, while both sim and real have a small

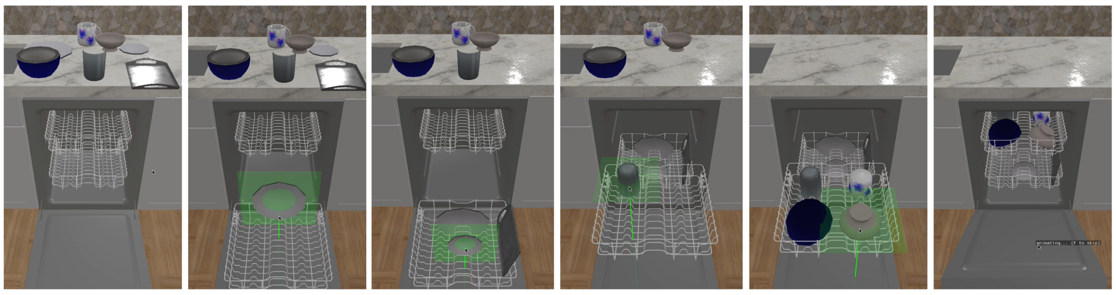


Figure A.5: Human demonstration with point and click in simulation

plate, the sizes of these plates are different. We can estimate the size of the objects based on actual bounding box from the segmentation pipeline. However, it is significantly out-of-distribution from the training data, due to object mismatch. So, we map each detected object to the nearest matching object in simulation and use the simulation size as the input to the policy. This is non-ideal, as the placing might differ for sim versus real objects. In the future, we would like to train with rich variations of object bounding box size in simulation so that the policy can generalize to unseen object shapes in the real world.

A.2 Simulation Setup

Dataset

“Replica Synthetic Apartment 0 Kitchen” consists of a fully-interactive dishwasher with a door and two sliding racks, an adjacent counter with a sink, and a “stage” with walls, floors, and ceiling. We use selected objects from the ReplicaCAD [202] dataset – seven types of dishes (cups, glasses, trays, small bowls, big bowls, small plates, big plates). Fig. A.5 shows a human demonstration recorded in simulation by pointing and clicking on the desired object to pick and place.

We initialize every scene with an empty dishwasher and random objects placed on the counter. Next, we generate dishwasher loading demonstrations, adhering to a given preference, using an expert-designed data generation script. Expert actions include opening/closing dishwashers/racks and picking/placing objects in feasible locations or the sink if there are no feasible locations left. Experts differ in their preferences and might choose different object arrangements in the dishwasher. A demonstration is a state-action sequence $\tau = \{(S_0, a_0), (S_1, a_1), \cdots, (S_{T-1}, a_{T-1}), (S_T)\}$. Here S_i is the set of object and place instances and a_i are the pick-place actions chosen by an expert at time i. At every time step, we record the state of the objects, the pick instance chosen by the expert, place instances for the corresponding category, and the place instance chosen by the expert. Expert actions are assumed to belong to the set of visible instances in S_i. However, different experts can exhibit different preferences over a_i. For example, one expert might choose to load bowls first in the top rack, while another expert might load bowls last in the bottom rack. In the training dataset, we assume labels for which preference each demonstration belongs, based on the expert used for collecting the demonstration. Given K demonstrations per preference $m \in \mathcal{M}$, we have a dataset for preference m: $\mathcal{D}_m = \{\tau_1, \cdots, \tau_K\}$. The complete training dataset consists of demonstrations from all preferences: $\mathcal{D} = \bigcup_{m=1}^{M} \mathcal{D}_m$. During training, we learn a policy that can reproduce all demonstrations in our dataset. This is challenging, since the actions taken by different experts are different for the same input, and the policy needs to disambiguate the preference. At test time, we generalize the policy to both unseen scenes and unseen preferences.

Expert Preferences

In Section 2.4, we define preference, that is, a set of 'properties' of the demonstration trajectory, like which rack is loaded first with what objects? Individual task preferences differ in the sequence of expert actions, but collectively, preferences share the underlying task semantics. For example, the user always opens the dishwasher rack before loading it for all preferences. By jointly learning overall preferences, our policy can benefit from cross-preference data to learn task structure, and sparse per-preference demonstration data to learn to follow the desired preference in the current situation.

There are combinatorially many preferences possible, depending on how many objects we use in the training set. Given 7 categories of dishes and two choices in which rack to load first, the hypothesis space of possible preferences is $2 \times 7!$. Our dataset consists of 12 preferences (7 train, 5 held-out tests) with 100 sessions per preference. In a session, $n \in \{3, ..., 10\}$ instances are loaded in each rack. The training data consists of sessions with 6 or 7 objects allowed per rack. The held-out test set contains 5 unseen preferences and settings for $\{3, 4, 5, 8, 9, 10\}$ objects per rack. Thus, there are 2×7^n unique possible session demonstrations, i.e. different prompts and situations.

Dynamically appearing objects

To add additional complexity to our simulation environment, we simulate a setting with dynamically appearing objects later in the episode. During each session, the scene is initialized with $p\%$ of maximum objects allowed. The policy/expert starts filling a dishwasher using these initialized objects. After all the initial objects are loaded and both racks are closed, new objects are initialized one per timestep to the policy. The goal is to simulate an environment where the policy does not have perfect knowledge of the scene and needs to reactively reason about new information. The policy reasons on both object configurations in the racks, and the new object type to decide whether to 'open a rack and place the utensil' or 'drop the object in the sink'.

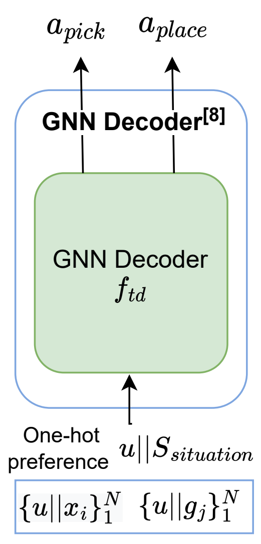
A.3 Simulation Experiment Details: Baselines, Training and Metrics

GATs and Transformers are closely related network architectures. In theory, a GAT architecture over a fully-connected graph with an additional time-based feature would be equivalent to our Transformer-based model. We are certainly not the first to apply attention-based architectures to planning problems, but unlike [120, 101], we consider sequential planning over multiple timesteps, which is critical to understanding human preferences [53]. Another difference from GNN for sequential planning is that we use cross-attention to condition the policy on the prompt, instead of adding preferences as an input feature. We use a slot attention architecture to learn a preference representation, and re-use instance encoders across prompt and situation. While these architectural choices are possible in GNNs, they are more intuitive and faster to implement in Transformers.

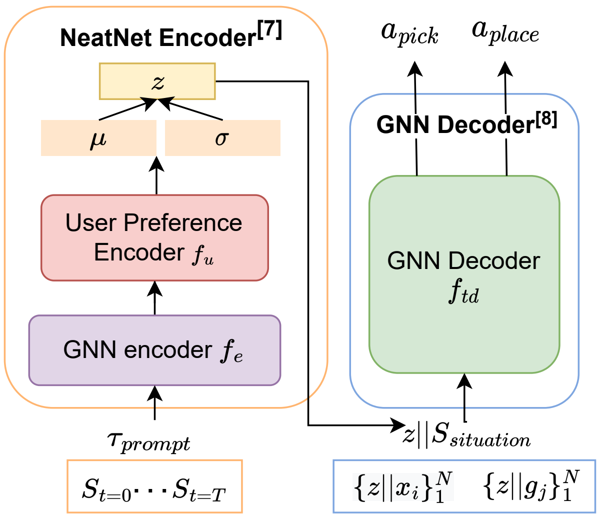
Baselines

Random We sample the action, that is, pick or place instance from the set of visible instances in the current state/observation with equal probability. Similarly, the place pose is sampled randomly from all possible poses for the chosen category. The performance of a random policy reduces exponentially with the length of the action sequence (typically around 20-30 steps long), and number of visible objects in the scene (between 6 to 20). SPA is low if the policy either (1) fails to adhere to the physical constraints, or (2) violates the spatial preference. Even if we ignore the physical constraint violations and consider the task of selecting the preferred rack for each object, the probability of a SPA of 1.0 for a random policy would be around $(1/2)^{N_{objects}}$, since it has to choose either top or bottom rack per object.

GNN-IL We use GNN with attention-based message passing. The input consists of 12 dimensional attributes (1D-timestep, 3D-category bounding box extents, 7D-pose, 1D-is object bool) and 12 dimensional one-hot encoding for the preference u. This forms the input node features, with each node representing a visible object in the environment. All the nodes are fully connected with equal edge weight of 1. The preference vector u is concatenated with each node $x_{i}{}_{i=1}^{N}$ of the state from the situation, resulting in 24 dimensional vector representation per node, which is in turn fed into the GNN decoder as used in [120]. Refer figure A.6a for the overall architecture. The policy is trained using



(A) GNN-IL



(B) GNN-VAE-IL

FIGURE A.6: Model workflow for GNN-IL and GNN-VAE-IL inspired from [120, 101]. The one-hot encoding per preference is passed as input to GNN-IL or GNN-GT-IL (there is no unseen preference for GNN-GT-IL). GNN-VAE-IL infers the preference from prompt through reconstruction and KL loss on preference embedding space.

supervised learning to minimize the cross-entropy loss over the expert demonstration (choose the same object and place pose as selected by the expert of given preference).

GNN-RL We train the GNN decoder with Proximal Policy Optimization (PPO) [183]. Each node is represented similarly 12 instance attributes and ground-truth one-hot encoded preference (12-dimensional), like in GNN-IL. Reward function for the RL policy is defined in terms of preference. The policy gets a reward of +1 every time it predicts the instance to pick that has the category according to the preference order and whether it is placed on the preferred rack.

GNN-VAE-IL Note that GNN-IL gets privileged information about the preference encoding and not from the prompt demonstrations. So, we train a preference encoding based policy using VAE loss as per [101] and call this variant GNN-VAE-IL. The prompt is passed through the GNN encoder, global add pool and the feed-forward layers to predict mean μ and log variance σ of preference latent z as per [101]. This is used to sample a z from preference embedding space to concatenate with the visible object instances in the situation. The concatenated preference to instance features is passed into the state-action policy modelled as GNN decoder in [120] to predict pick and place instances. GNN-VAE-IL is trained to minimize the VAE reconstruction loss over the actions taken by an expert of given preference in a situation.

Architectural details

TTP We use a 2-layer 2-head Transformer network for encoder and decoder. The input dimension of instance embedding is 256 and the hidden layer dimension is 512. 100 slots and 3 iterations for Slot Attention. Instance attributes are encoded as follows: $\texttt{Category} : 3 \rightarrow 64, \texttt{Pose} : 7 \rightarrow 128, \texttt{Timestep} : 1 \rightarrow 32, \texttt{Is object marker?} : 1 \rightarrow 32$. For preference encoding, we use the slot attention layer at the head of Transformer encoder with 100 slots and 3 slot iterations.

GNN Decoder We use three layer GATConv, each followed by MLP for the GNN decoder with 128-dimensional hidden layer as per [120]. At the output, each node feature is reduced to a logit. We select the pick and place instance using the input 'is object marker', and consider the softmax from each respective set.

GNN Encoder We use the GNN encoder as in [101] with 128 hidden dim GATConv layer followed by elu activation and 256 dimensional MLP. This is passed to a preference encoder, which is 2-layer MLP with leaky ReLU and that outputs 100-dimensional mean and log variance corresponding to the assumed Gaussian distribution of the inferred preference encoding. During training, the preference encoding is sampled from this distribution. While evaluation, we use the mean as the best representation of preference encoding.

Training

We use a batch-size of 64 sequences. Within each batch, we use pad the inputs with 0 upto the max sequence length. Our optimizer of choice is SGD with momentum 0.9, weight decay 0.0001 and dampening 0.1. The initial learning rate is 0.01, with exponential decay of 0.9995 per 10 gradient updates. We used early stopping with patience 100. Fig.A.7 shows the accuracy achieved for picked category on uniformly sampled input states from the training data, unlike rollout where the next state depends on the previous action taken. We show runs grouped by batch sizes 64, 128 and 256.

Metrics

In Section 2.4, we presented Spatial Preference Adherence (SPA) and Temporal Preference Adherence (TPA) metrics collected on a policy rollout. Here we discuss additional metrics about training progress and rollouts for GNN-VAE-IL.

Category-token Accuracy indicates how well the policy can mimic the expert's action, given the current state. We monitor training progress by matching the predicted instance to the target chosen in the demonstration (Fig. A.7). We see that TTP is able to predict the same category object to pick perfectly (accuracy close to 1.0). However, this is a simpler setting than sequential decision-making. During the rollout, any error in a state could create a setting that is out-of-distribution for the policy. Thus, category token accuracy sets an upper bound for rollout performance, that is, while having high category token accuracy is necessary, it is not sufficient for high packing efficiency and inverse edit distance.

Policy Rollouts for Evaluation We evaluate trained policies on rollouts in the simulation, a more complex setting than the accuracy of prediction. Rollouts require repeated decisions in the environment, without any resets. A mistake made early on in a rollout session can be catastrophic, and result in poor performance, even if the prediction accuracy is high. For example, if a policy mistakenly fails to open a dishwasher rack, the rollout performance will be poor, despite good prediction accuracy.

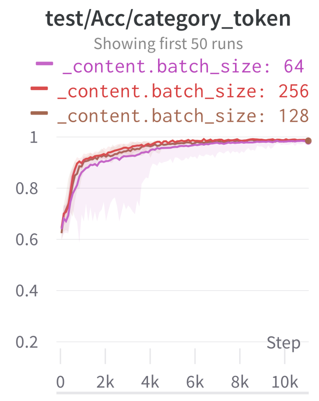
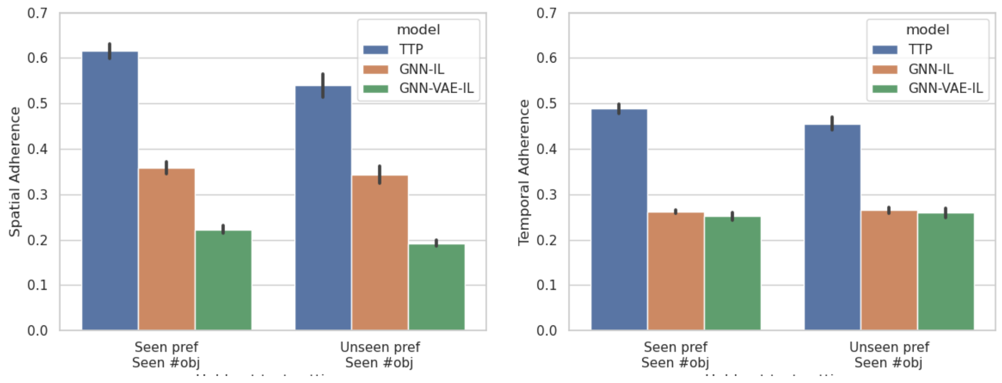


FIGURE A.7: Category level accuracy grouped by batch size for prompt-situation training.

Note that the simulated dishwasher loading is a challenging task. The performance of the random agent is bad because we evaluate the SPA at the end of an episodic rollout. The performance of a random policy reduces exponentially with the length of the action sequence (typically around 20-30 steps long), and the number of visible objects in the scene (between 6 to 20). SPA is low if the policy either (1) fails to adhere to the physical constraints, or (2) violates the spatial preference. Even if we ignore the physical constraint violations and consider the task of selecting the preferred rack for each object, the probability of a SPA of 1.0 for a random policy would be around $(1/2)^{N_{objects}}$ since it has to choose either top or bottom rack per object.

We compare TTP, GNN-IL, and GNN-VAE-IL on seen and unseen preferences in Fig. A.8. GNN-IL performs better than GNN-VAE-IL in terms of SPA but both are similar in terms of TPA. Intuitively, GNN-VAE-IL solves a harder learning problem of inferring preference from the trajectory, unlike GNN-IL which uses a privileged ground-truth label for the unseen preferences.

Temporal efficiency Just like SPL [8] for navigation agents, we define the efficiency of temporal tasks in policy rollout, in order to study how efficient the agent was at achieving the task. For episode $i \in [1, ..N]$, let the agent take p_i number of high-level interactions



(A) Spatial Preference Adherence (SPA)

(B) Temporal Preference Adherence (TPA)

FIGURE A.8: Comparisons of TTP, GNN-IL, and GNN-VAE-IL simulation across two metrics: SPA and TPA. TTP shows good performance at the task of dishwasher loading on seen and unseen preferences and outperforms the GNN-IL and GNN-VAE-IL baselines. Note that GNN-IL has ground-truth training data on unseen preferences.

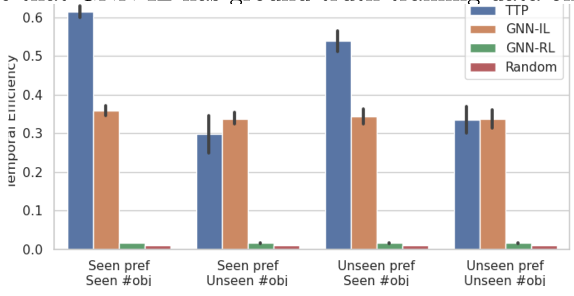


FIGURE A.9: TE metric for held-out test settings.

to execute the task, and the demonstration consists of l_i interactions for the initial state. We scale the spatial preference adherence SPA_i of the policy by the ratio of steps taken by the expert versus the learned policy. Temporal efficiency is defined between 0 to 1, and higher is better. This value will be equal to or lower than the packing efficiency. This especially penalizes policies that present a 'looping' behavior, such as repeatedly opening/closing dishwasher racks, over policies that reach a low SPA in shorter episodes (for example, by placing most objects in the sink). Fig. A.9 shows the temporal efficiency over our 4 main held-out test settings (as shown in Fig. 2.5).

A.4 Failure Analysis

We present the failure analysis in terms of error counts over 63 rollouts. Figure A.10 shows that TTP learns physical constraints of the problem perfectly, and never takes an action that violates them. Instead, the failure cases stem from either (1) not following user preferences, or (2) repeating certain actions which don't make progress towards the task, like repeatedly picking and placing the same object. In contrast, our baselines like continuous pose prediction suffer from violating task constraints, like attempting to place

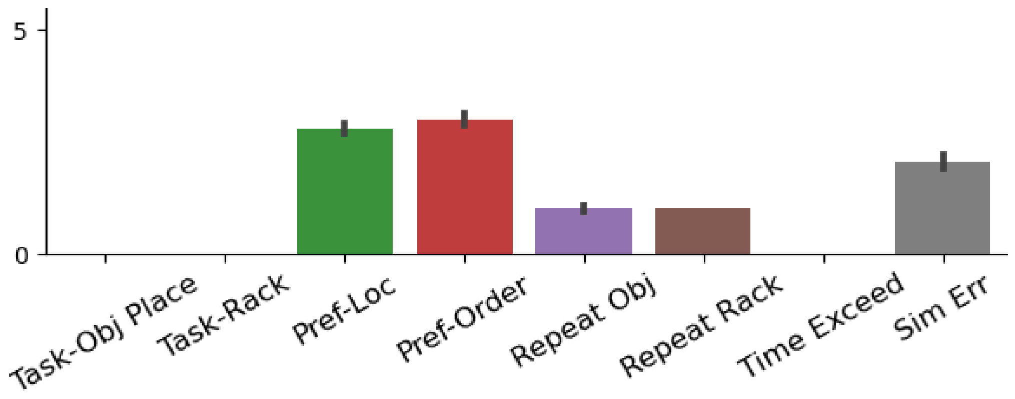


FIGURE A.10: Failure analysis for TTP. X-axis indicates enlists different error types concerning physical constraints (**Task-Obj Place**: object place pose is infeasible, **Task-Rack**: not placing when in top when its open, not placing in bottom when top is closed and bottom is open, placing object when both racks are closed), preference constraints (**Pref-Loc**: location violation in terms of incorrect rack, **Pref-Order**: order violation of preference) and No-ops (**Repeat Obj**: picking an already placed object, **Reposition**: Adjust the placement of an object within the same rack, **Repeat Rack**: picking a dishwasher part repeatedly; leads to termination, **Time Exceed**: leads to termination). Y-axis is the error count per rollout. Most errors are due to failing in the location-aspect and order-aspect of the preference. The **repeat-*** pattern triggers the termination in most rollouts, indicating that failure scenario usually obeys physical constraints but does not make progress towards the task, thereby resulting in lower SPA and TPA.

an object in an already occupied location. Fig. A.11 provides distribution per error type shown here, with leftmost bar near zero indicating that that error never occurs, and the right extreme denotes that that error occurs in high frequency in some rollouts. Both (1) and (2) stem out of the problem of sequential decision making. An open problem is to learn policies that can adapt from interaction and correct for such errors.

A.5 Additional Ablation Experiments

Towards the end of the section 2.4.1, we presented ablation experiments over the number of demonstrations per preference used for training, and the number of unique preferences used. In this section, we present additional ablation experiments over (A.5) the design of instance encodings in TTP, (A.5) the effect of increasing the temporal context of TTP on performance, the comparison between the choice of (A.5) discrete category tokens vs continuous box extent representation in TTP, and (A.5) continuous placement pose vs discrete alternative used in TTP.

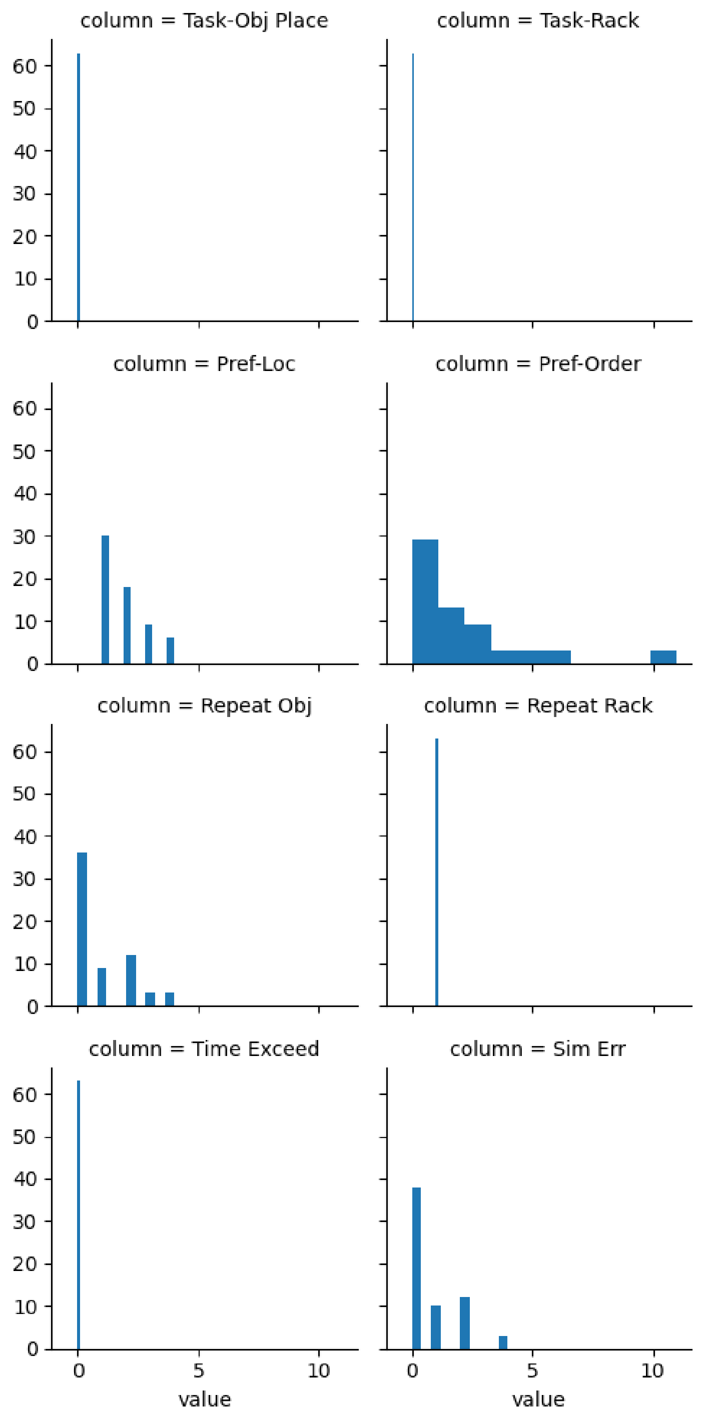


Figure A.11: Histogram per error type to estimate the frequency (Fig A.10 shows only mean and std). X-axis shows the value of the error count. Y-axis shows the number of times that value occurs in all of the rollouts. Most rollouts have 0 or very few preference based errors. Error count show peak near 0 and gradually reduces of towards higher numbers. This is an extended analysis of the error counts in figure A.10 showing the failure profiles for the different error types.

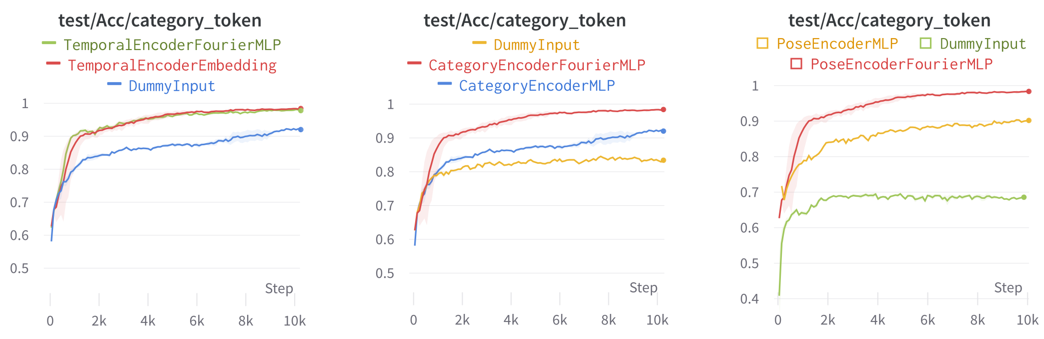
Design of Instance Encoding

Why object attributes are important? The pose describes where an object is. For this task, it depends on whether the object is on the counter or in the dishwasher.

The category is important to follow the order-based preference and to group the placement instances. A big bowl may have very limited placement instances in a dishwasher rack compared to a plate or a small cup.

The timestamp is important for encoding the prompt – it allows the neural network to reason over prior instances without explicit tracking. Tracking an object across frames is difficult, especially with partially observed scenarios (such as an bowl in the top rack is not visible if the top rack is closed).

The r marker is used to denote whether it is a pick or place instance. Marker acts as a flag and allows us to use the same architecture input to process a variable number of both pick and place instance embeddings.

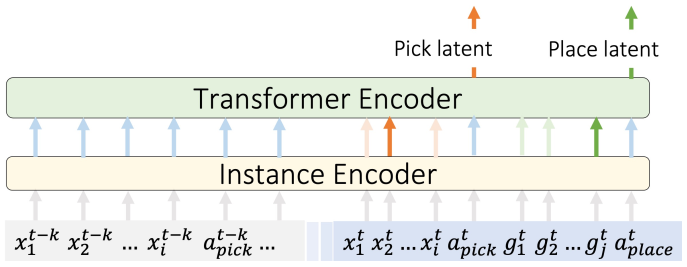


(A) Temporal encoding (B) Category encoding (C) Pose encoding

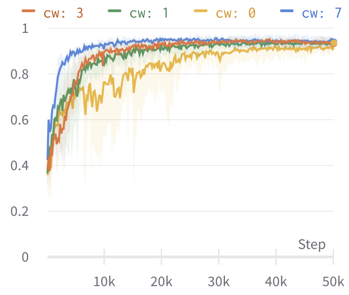
FIGURE A.12: [Left-to-Right] Comparing different design choices of attribute encoders in terms of category token accuracy on held-out test prompt-situation session pairs.

How much does temporal encoding design matter? Fig. A.12a shows that learning an embedding per timestep or expanding it as Fourier transformed vector of sufficient size achieves high success. On the other hand, having no timestep input shows slightly lower performance. Timestep helps in encoding the order of the prompt states. The notion of timestep is also incorporated by autoregressive masking in both the encoder and the decoder.

How much does category encoding design matter? In our work, we represent category as the extent of an object's bounding box. An alternative would be to denote the category as a discrete set of categorical labels. Intuitively, bounding box extents


Pick latent
Place latent
Transformer Encoder
Instance Encoder
x_1^{t-k} x_2^{t-k} ... x_i^{t-k} a_{pick}^{t-k} ...
x_1^t x_2^t ... x_i^t a_{pick}^t g_1^t g_2^t ... g_j^t a_{place}^t

(A) Model workflow for context History k



(B) Plot with context history of 0,1,3,7

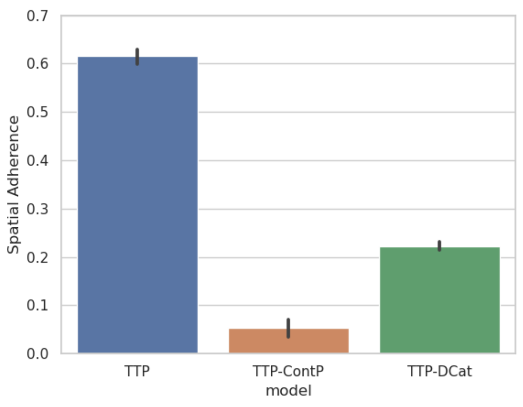
FIGURE A.13: Incorporating Context History instead just current observation. **Left**: Modified TTP for processing with previous Context History k. **Right**: Plot showing category level accuracy for the held-out test sessions for single preference training with context windows. While larger context window size learns faster, the asymptotic performance for all context windows converges in our setting.

capture shape similarity between objects and their placement implicitly, which discrete category labels do not. Fig. A.12b shows that Fourier transform of the bounding box achieves better performance than discrete labels, which exceeds the performance with no category input.

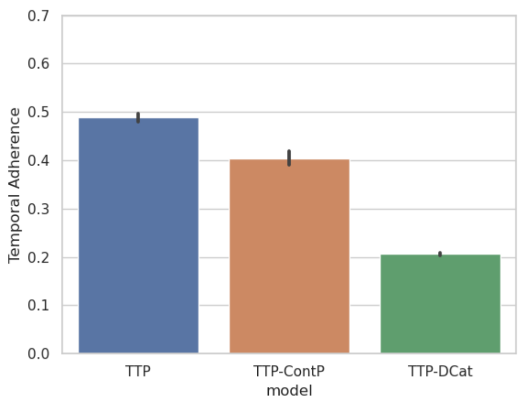
How much does pose encoding design matter? We encode pose as a 7-dim vector that includes 3d position and 4d quaternion. Fig. A.12c shows that the Fourier transform of the pose encoding performs better than feeding the 7 dim through MLP. Fourier transform of the pose performs better because such a vector encodes the fine and coarse nuances appropriately, which otherwise either require careful scaling or can be lost during SGD training.

Markov assumption on the current state in partial visibility scenarios

Dynamic settings, as used in our simulation, can be partially observable. For example, when the rack is closed, the policy doesn't know whether it is full or not from just the current state. If a new object arrives, the policy needs to decide between opening the rack if there is space, or dropping the object in the sink if the rack is full. In such partially observed settings, the current state may or may not contain all the information needed to reason about the next action. However, given information from states in previous timesteps, the policy can decide what action to take (whether to open the rack or directly place the object in the sink). With this in mind, we train a single preference policy for picking only with a different context history. As shown in Fig. A.13a, the context window of size k processes the current state as well as k predecessor states, that is, in total $k + 1$ states.



(A) Spatial Preference Adherence (SPA)



(B) Temporal Preference Adherence (TPA)

FIGURE A.14: Comparisons of TTP, Continuous Place Pose Prediction (TTP-ContP), and Discrete Category Token Input (TTP-DCat), in simulation across two metrics: SPA and TPA (on Y-axis), on seen preferences and objects during training. Continuous Pose prediction achieves much lower SPA due to infeasible object place pose prediction and thereby violates the *location-aspect* of the preference. As the TPA is based on the pick order sequence, TPA is high indicating that the policy learns the *order-aspect* of the preference. Discrete category prediction achieves low SPA and TPA due to as few as 7 categories mapped to the embedding of size 64; similar to original TTP with continuous category representation in terms of the bounding box.

Let context history k refer to the number of previous states included in the input. Then the input is a sequence of previous k states' instances (including the current state), as shown in Fig. A.13a.

Fig. A.13b shows that TTP gets $> 90\%$ category-level prediction accuracy in validation for all context windows. While larger context windows result in faster learning at the beginning of the training, the asymptotic performance of all contexts is the same. This suggests that the dataset is largely visible and a single context window captures the required information. In the future, we would like to experiment with more complex settings like mobile robots, which might require a longer context.

Discrete Category Tokens

Category embeddings should be more descriptive than just a bounding box and include some language labels like knife versus fork. In our current work, we assumed that similar-sized objects have similar occupancy footprints, and this was enough to reason about feasible placing locations in a rack. This was conducive to learning the physical constraints of objects in the scene but ignores other relationships between objects. Figure A.14 shows TTP-DCat that indicates the performance with the discrete category is much lower and requires hyperparameter tuning for the embedding size. In future work, we would like to explore additional multi-modal object features, including language descriptions along with the geometry information.

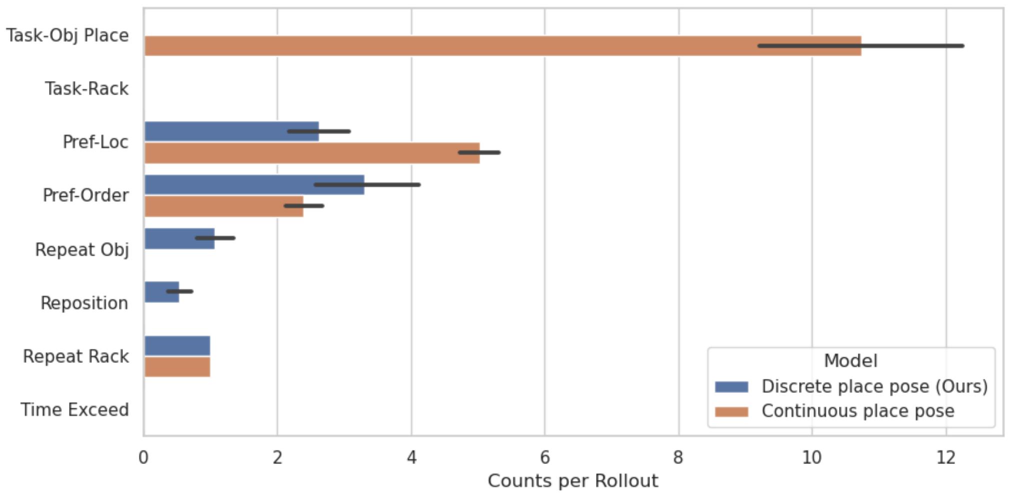


FIGURE A.15: Failure case analysis for continuous place pose prediction. The X-axis indicates the error count per rollout. The Y-axis enlists different error types concerning physical constraints (Task-Obj Place: object place pose is infeasible, Task-Rack: not placing when in the top when its open, not placing in the bottom when the top is closed and the bottom is open, placing object when both racks are closed), preference constraints (Pref-Loc, Pref-Obj) and No-ops (Repeat Obj: picking an already placed object, Reposition: Adjust the placement of an object within the same rack, Repeat Rack: picking a dishwasher part repeatedly; leads to termination, Time Exceed: leads to termination).

Continuous Placement Pose

In TTP, we consider two solution spaces: decision over 'what to pick', and 'where to place'. The pick solution space is naturally discretized by the objects of interest visible in the environment (dishwasher, racks, dishes, etc.), and does not limit real-world transfer.

Note that while we consider discrete objects, their features like location are continuous. An object segmentation pipeline, as used in our real-world experiments, can provide a discrete set of objects to interact with, and their continuous features. On the other hand, our choice of discretized 'where to place' solution space can be seen as a hurdle to real-world transfer. However, in our experiments, dense discrete poses generalized to unseen scenarios better than continuous prediction. Figure A.14 shows that discrete placement pose instances achieve higher SPA and TPA than continuous place pose prediction. The TPA looks comparable to the original TTP but the SPA is much lower. This is because the TTP-ContP policy learns to pick objects in the correct order like TTP but suffers in placement pose prediction. Figure A.15 shows the count of error types over 9 rollouts for each of the 7 preferences (=total 63 rollouts). Here lower error count is better. Error for continuous place pose policy occurs mostly in the prediction of infeasible place pose for the object.

The geometry of a dishwasher is well-suited for discrete positions, as small prediction errors can make a target place position infeasible. For example, a plate can only fit in a particular pose, and small orientation errors make placing infeasible. In the real world, one could imagine creating a dense place pose dictionary based on the model of the dishwasher in the user's home. However, we agree that for other receptacles, like a shelf, this discretization is not necessary, and a continuous prediction approach like our baseline might be better suited.

Finally, we note that while we use discrete place poses, they are densely sampled, and not mutually exclusive. Hence, several discrete poses overlap, and if one pose is occupied by an object, there might be several others that are also occupied by the same object. TTP has to reason about the size of an object and distances between place poses to decide which pose is empty to place the next object, while maximizing packing efficiency. Hence, our policy is able to reason about 3D space occupancy, even if working with discretized poses.

A.6 Limitations and Future scope

In Section 3.5, we briefly discussed the limitations and risks. Here we enlist more details and highlight future directions.

Pick grasping depends on accurate segmentation and edge detection Grasping policy depends on the quality of segmentation and edge detection of the selected object. Due to noise in calibration, shadows, and reflections, there are errors in detecting the correct edge to successfully grasp the object. For example, it is hard to grasp a plate in a real setting. The plate is very close to the ground and the depth cameras cannot detect a clean edge for grasping. As the depth edges are not often not accurate due to noisy point clouds, the collision prediction with the candidate grasps and the object is not very robust and conservatively results in no feasible grasps. Therefore, in our work, we place the plate on an elevated stand for easy grasping. Grasping success also depends on the size and kind of gripper used.

Placement in real setting For placement, the orientation of the final pose is often different from the initial pose and may require re-grasping. The placement pose at the final settlement is different from the robot's end-effector pose while releasing the object from its grasp. Similar to picking, placement accuracy will largely depend on the appropriate size and shape of the gripper used. Due to these reasons, placement in the real world is an open challenging problem and we hope to address this in future work.

Hardware pipeline issues due to calibration The resulting point cloud has noisy depth edges due to two reasons. First, incorrect depth estimation due to camera hardware, lighting conditions, shadows, and reflections. Second, any small movements among cameras affect calibration. If we have a noisy point cloud, it is more likely to have errors in subsequent segmentation and edge detection for grasp policy. Having sufficient coverage of the workspace with cameras is important to mitigate issues due to occlusions and incomplete point clouds.

Undesired motion due to IK and grasping We use the Pinnochio library [4] for planning robot joints given the end-effector as this is fast and easy to use. But it assumes collision-free plans, which is difficult with relaxed IK limits to enable robot arm is reachable to the base of the drawers. Also, the robot faces difficulty in grasping certain objects that are closer to the base and occluded in the camera images.

We use GraspNet [60] for obtaining analytical candidate grasps on the segmented point cloud for a dish. The best candidate grasp may not be always semantically appropriate, for example, the best candidate grasps on a cup or wine glass are closer to the pinched at the rim (due to clearer edge detection) but many humans prefer to hold the curved cylinder stem. Incorporating preferences in low-level grasping is an open direction for future work.

Incomplete information in prompt The prompt session may not contain all the information to execute the situation. For example, in a prompt session, there might be no large plates seen, which is incomplete/ambiguous information for the policy. This can be mitigated by ensuring complete information in the prompt demo or having multiple prompts in slightly different initialization.

Appendix B

Appendix: From pixels of videos to low-level actions for manipulation

In our video provided in the supplementary materials, we show several examples of successful rollouts, and model architecture with prompt video and robot observations. We provide specific examples of task-based success, and cross-object motion transfer. Additionally, we provide qualitative results on long-horizon task composition and rollouts for tasks that are rare in our training dataset. Finally, we provide rollouts of noted failures like case of self-occlusion, grasping errors and ambiguous interpretation of the task in the prompt video.

B.1 Training Details

We will release the model code and trained checkpoints. In Table B.1, we present the detailed version of the Vid2Robotodel architecture. In Table B.2, we present the hyperparameters used while training. We also add data augmentations to the videos while training as per the settings in Table B.3. We normalize the videos as required for the pre-trained ViT model.

B.2 Dataset

We created the dataset mixture from the three sources. We collected 120k robot trajectories, 5k human trajectories, and 5k co-located human and robot trajectories.

Table B.1: Detailed Architecture for Vid2Robot

Module	Layer	Output Size	Notes
Prompt Encoder	Input	16×224×224×3	Reference Video of 16 frames
	Image Encoder	16×196×768	Each frame passes through ViT-B/16
	Reshape	3136×768	Reshapes all space-time tokens to be in one dimension
	Perceiver Resampler	64×768	2 layers with 768 dims and 12 attention heads
State Encoder	Input	8×224×224×3	Robot Observation Video of 8 frames
	Image Encoder	8×196×768	Each frame passes through ViT-B/16
	Reshape	1568×768	Reshapes all space-time tokens to be in one dimension
	Perceiver Resampler	64×768	2 layers with 768 dims and 12 attention heads
State Prompt Encoder	Cross-attention Transformer Layer	64×768	4 layers of cross-attention with 768 dim and 8 attention heads Uses prompt tokens as keys and state tokens as queries
Action Decoder	Cross-attention Transformer Layer	11×768	4 layers of cross-attention with 768 dim and 8 attention heads Uses prompt-aware state tokens as keys and action position embeddings as queries
Action Head	Linear	11×256	Projection of action decoder output to 256 bins

Table B.2: Hyperparameters for Vid2Robot

Hyperparameters	Values
Batch size	2048
Learning rate	8e-5
Optimizer	AdamW
Num training steps	200,000
Warmup steps	2000
Image size	224
Num prompt frames	16
Num robot frames	8
Prediction Horizon	4

Some of the data was not useful due to incomplete videos, not showing the full task demonstration.

To create pairs, we sampled 3 videos as prompts per robot trajectory. This gives 360k pairs for Robot-Robot, about 15k pairs for Hindsight Human-Robot and 5k pairs for Co-located Human-Robot datasets.

To ensure that all pairs are approximately sampled equally, we create the mixture proportions as 90% Robot-Robot, 5% Hindsight Human-Robot and 5% Co-located Human-Robot.

Table B.3: Data augmentation for training Vid2Robot

Hyperparameters	Ranges
Height crop range	(0.95, 1.)
Width crop range	(0.95, 1.)
Brightness range	(0.9, 1.1)
Contrast range	(0.8, 1.2)
Hue range	(0, 0.03)
Saturation range	(0.8, 1.2)

B.3 Hardware setup

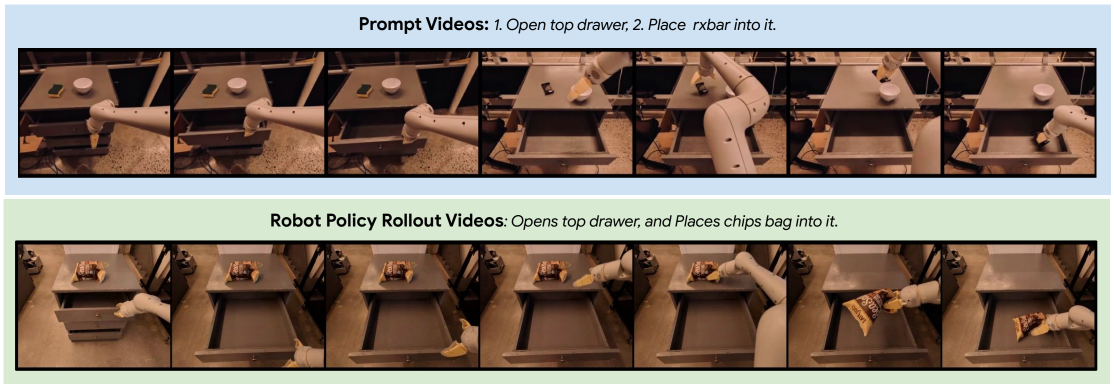
In all the experiments in this paper, we use mobile manipulators. These robots have a 7 degree-of-freedom arm, a two-fingered gripper, and a mobile base. As the focus of our experiments is on manipulating objects, we keep the base fixed for each episode.

We test the policy with client-server setup, where the policy is deployed as server, which on-robot client can query. The observations from the head camera of the robot are recorded on the client side, to maintain a recent history of 8 frames. Based on the average response time from the policy, we can execute predicted actions at approximately 5-7 Hz.

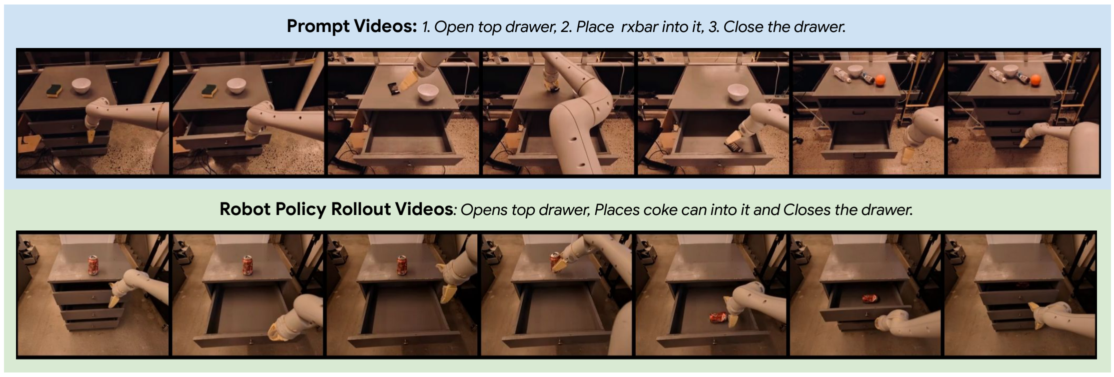
B.4 Vid2Robotor Long Horizon Tasks

We show Vid2Robotan perform long horizon tasks by using multiple prompt videos showing different tasks. To do so, we chain different prompt videos together to create a long horizon task. For example, Vid2Robotan be used to prompt a robot to *clean up the objects on a table into the drawer* by chaining prompt videos of 3 tasks: *open the drawer*, *place object on table into drawer*, and *close the drawer*.

When Vid2Robotutputs terminate episode for the current prompt video, we change the prompt video to be the next sub-task without resetting the robot. In this manner, the robot can complete long horizon tasks with Vid2Robotithout explicitly being trained on long horizon videos. We show successful examples in Fig. B.1. For videos of the robot completing long-horizon tasks, check the summary video in the supplementary material.


Prompt Videos: 1. Open top drawer, 2. Place rxbar into it.
Robot Policy Rollout Videos: Opens top drawer, and Places chips bag into it.

(a) Two step chaining of prompt videos with (1) Open drawer, (2) Place object into it.


Prompt Videos: 1. Open top drawer, 2. Place rxbar into it, 3. Close the drawer.
Robot Policy Rollout Videos: Opens top drawer, Places coke can into it and Closes the drawer.

(b) Three step chaining of prompt videos. (1) Open drawer, (2) Place object into it, (3) Close drawer.

FIGURE B.1: Qualitative results showing how Vid2Robot can perform long horizon tasks with chain prompt videos.

B.5 Qualitative Results on Rare Tasks

We also test SLAPn some tasks which are rare in the training set and find that video-conditioned models are also able to complete them. Some examples of these tasks are: *opening glass jar, picking up green micro-fiber cloth and pulling out napkin.* We show examples in Fig. B.2. For the video version of these results please take a look at the supplementary video.

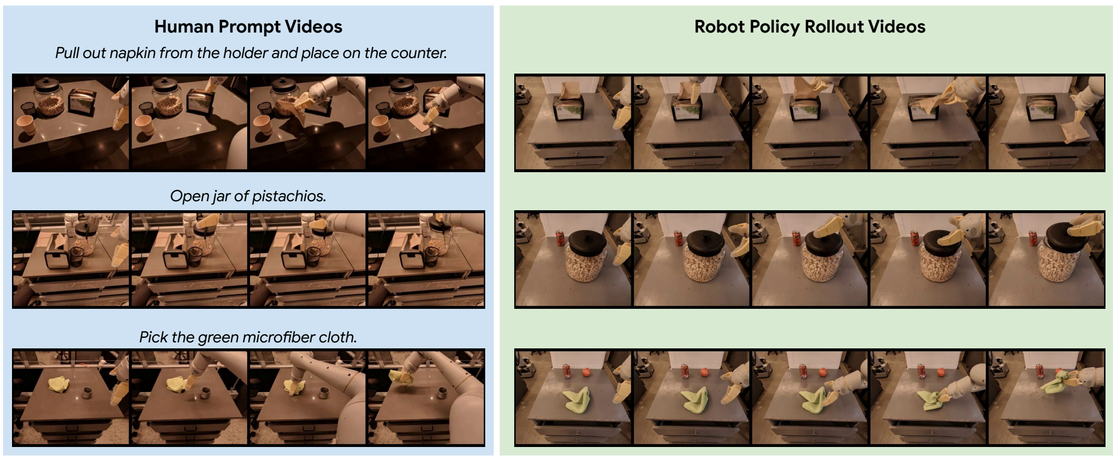

Human Prompt Videos
Pull out napkin from the holder and place on the counter.
Open jar of pistachios.
Pick the green microfiber cloth.
Robot Policy Rollout Videos

FIGURE B.2: Qualitative results on rare tasks.

Appendix C

Appendix: From high-level instruction to sub-tasks for mobile manipulation

C.1 Training

Below is expanded information on our training, algorithm, and data processing to improve reproducibility.

Data Collection and Annotation

When collecting an episode with the Franka arm, we first scan the scene with a pre-defined list of scanning positions to collect an aggregated x. In our case, we make no assumption as to what or how many these are, or how large the resulting input point cloud x is. With the Hello Robot Stretch [104], we collect data based on exactly where the robot is looking.

Then, we collect demonstration data using kinesthetic teaching for the Franka arm (demonstrator physically moves the robot) and via controller teleoperation for the Stretch robot. The demonstrator moves the arm through the trajectory associated with each task, explicitly recording the *keyframes* [1] associated with action execution. These represent the salient moments within a task – the bottlenecks in the tasks' state space, which can be connected by our low-level controller.

Data Processing

We execute each individual skill open-loop based on an initial observation. We use data augmentation to make sure even with relatively few examples, we still see good generalization performance.

Data Augmentation. Prior work in RGB-D perception for robotic manipulation (e.g. [217, 74]) has extensively used a variety of data augmentation tricks to improve real-world performance. In this work, we use three different data augmentation techniques to randomize the input scene x used to train $p_I = \pi_I(x, l)$:

- *Elliptical dropout:* Random ellipses are dropped out from the depth channel to emulate occlusions and random noise, as per prior work [130, 217]. Number of ellipses are sampled from a Poisson distribution with mean of 10.
- *Multiplicative Noise:* Again as per prior work [130, 217, 157], we add multiplicative noise from a gamma process to the depth channel.
- *Additive Noise:* Gaussian process noise is added to the points in the point-cloud. Parameters for the Gaussian distribution are sampled uniformly from given ranges. This is to emulate the natural frame-to-frame point-cloud noise that occurs in the real-world.
- *Rotational Randomization:* Similar to prior work [190, 157, 124], we rotate our entire scene around the z-axis within a range of ± 45 degrees to help force the model to learn rotational invariance.
- *Random cropping:* with $p = 0.75$, we randomly crop to a radius around $\hat{p}_I + \delta$, where δ is a random translation sampled from a Gaussian distribution. The radius to crop is randomly sampled in $(1, 2)$ meters.

Data Augmentation for π_R. We crop the relational input $x_R \subset x$ around the ground-truth p_I, using a fixed radius $r = 0.1m$. We implement an additional augmentation for learning our action model. Since p_I is chosen from the discretized set of downsampled points P, we might in principle be limited to this granularity of response. Instead, we randomly shift both p_I and the positional action δp by some uniformly-sampled offset $\delta r \in \mathbb{R}^3$, with up to $0.025m$ of noise. This lets π_R adapt to interaction prediction errors of up to several centimeters.

Action Prediction Losses

Following [158] for the orientation, we can compute the angle between two quaternions θ as:

$$\theta = \cos^{-1}(2\langle \hat{q}_1, \hat{q}_2 \rangle^2). \tag{C.1}$$

We can remove the cosine component and use it as a squared distance metric between 0 and 1. We then compute the position and orientation loss as:

$$L_R = \lambda_p \|\delta p - \hat{\delta p}\|_2^2 + \lambda_q (1 - \langle \hat{q}, q \rangle) \tag{C.2}$$

where λ_p and λ_q are weights on the positional and orientation components of the loss, set to 1 and $1e - 2$ respectively.

Predicting gripper action is a classification problem trained with a cross-entropy loss. For input we use the task's language description embedding and proprioceptive information about the robot as input, i.e. $s = (l, g_{act}, g_w, ts)$ where g_{act} is 1 if gripper is closed and 0 otherwise, g_w is the distance between fingers of the gripper and ts is the time-step. The gripper action loss is then:

$$L_g = \lambda_g CE(g, \hat{g}) \tag{C.3}$$

where λ_g is the weight on cross-entropy loss set to 0.0001. The batch-size is set at 1 for this implementation.

We train π_I and π_R separately for $n = 85$ epochs. At each epoch, we compare validation performance to the current best - if validation did not improve, we reset performance to the last best model.

Skill Weighting

In Stretch experiments, we used a wide range of skills with different error tolerances and corresponding variances. As a result, we needed to use two different sets of weights for learning position, orientation and gripper targets. A more forgiving weight-set for noisier tasks like pouring and handover, and a tighter weight-set for task with more consistent trajectories like opening and closing the drawer. These weights were empirically determined but can be further optimized via hyper-parameter tuning methods.

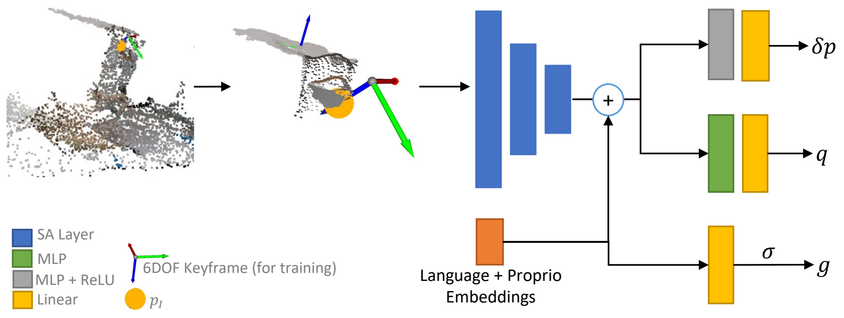


FIGURE C.1: Regression model architecture with separate heads for each output. The point-cloud is cropped around the interaction point with some perturbation and passed to a cascade of set abstraction layers. Encoded spatial features are then concatenated with language and proprioception embeddings to predict position offset of action from interaction point, absolute orientation and gripper action as a boolean.

Training for PerAct and SLAP

In all our experiments we ensure PerAct and SLAP are trained on the same *data volume*. Data volume is defined as total number of augmented samples per collected sample. Note this results in different number of training steps per model. This is due to the LSTM-based model updating once per trajectory while PerAct updates once per sample in a trajectory.

C.2 Relative Action Module

In our work, the Relative Action Module π_R is assumed to be some *local* policy which predicts end-effector poses. In our case, we implement two different versions of this policy, one which was used on the static Franka manipulator and one which was implemented on the Stretch. In both cases:

- The policy predicts an *end effector pose* relative to the predicted interaction point from π_I
- The policy is conditioned on a *local crop* around this interaction point.

MLP Implementation

Fig. C.1 gives an overview of the MLP version of the regression model. The model takes in the cropped point cloud (augmented during training as discussed in Sec. C.1. We saw that injecting random noise to the interaction point during training allowed the policy to, at test time, recover from failures (because it predicted an interaction point *near* the correct area, instead of at the correct position).

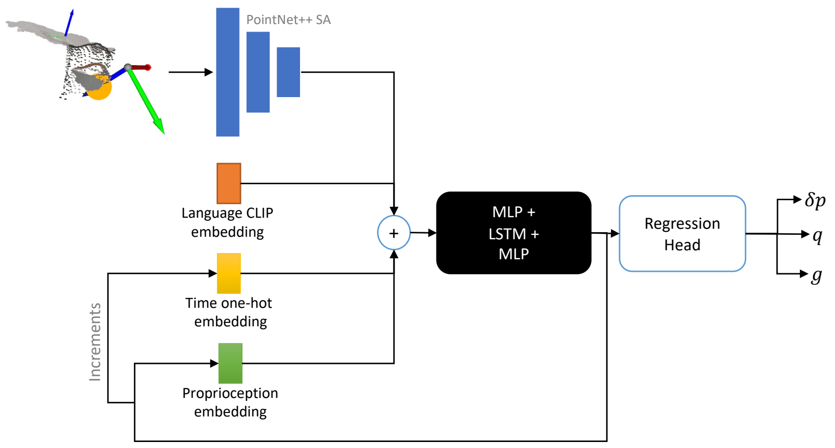


FIGURE C.2: LSTM-based regression model architecture based on the regression head and PointNet++ embeddings introduced in Fig. C.1. LSTM-based architecture shows higher stability in learning action distribution with wider distribution due to the conditioning effect.

LSTM-based Implementation

We observed the MLP architecture suffer when positional distribution of actions varied widely with respect to the interaction point position across tasks due to the multi-modality this introduced. Thus we modified the above model by adding an LSTM to condition each task and action with a hidden-state. This model exhibited better performance in learning wider action distribution, based on our initial experiments with the outlined Stretch tasks.

C.3 High-level Task Planning and Execution

Dataset

We procedurally generated a dataset consisting of more than 500k tuples of language instructions and the corresponding sequence of atomic skills. The data is created for 16 task families and can be extended further in the future. For each task family, 10% samples are held-out for evaluation. The distribution of samples across task families is shown in C.1.

For each task family, we define a corresponding template containing the sequence of atomic skills. This means that the sequence of atomic skill “verbs” is the same among the samples of a task family. Each sample within a task family differs in terms of language instructions and object(s) of interaction.

To populate these templates and generate the data, we create a list of more than 150 movable objects kitchen objects, surfaces like `table`, `kitchen counter` and articulated

Table C.1: Data Distribution of procedurally generated samples across task families.

Task Family	**Total number of samples**
Bring X From Y Surface To Pour In Z Then Place On W Surface	35640
Bring X From Y Articulated To Wipe Z	612
Move X From Y Surface To Z Surface	16092
Move X From Y Surface To Z Articulated	301400
Take X From Human To Z Articulated	19300
Bring X From Y Surface To Human	5933
Bring X From Y Surface To Wipe Z	3168
Take X From Human To Pour In Z	2184
Take X From Human To Z Surface	8050
Take X From Human To Wipe Z	1458
Move X From Y Articulated To Z Articulated	97923
Bring X From Y Articulated To Human	13617
Bring X From Y Surface To Pour In Z	3960
Bring X From Y Articulated To Pour In Z	8712
Move X From Y Articulated To Z Surface	50060
Take X From Human To Pour In Z And Place On Y Surface	19656
	587765

objects like `drawer`, `cabinets`. For `pour` skill, we create a list of "spillable" items such as `cup of coffee`, or `bowl of jelly beans`. Similarly, for wipe skill, we have a list of items to wipe with such as `sponge`, or `brush`.

Models

Table 4.4 shows that in-context learning with 5 examples from the same task family achieves close to 76% accuracy. There is no training involved with in-context learning, so it can't overfit. For finetuning, our evaluation consists of a held-out dataset with unseen variations in the phrasing of the language instruction and/or object(s) of interaction. The goal of the fine-tuned model is to demonstrate that it is possible to achieve improved task planning performance with lower latency than in-context learning with larger models. We do not evaluate the generalization of the fine-tuned model with unseen task families in this work. The remaining results in this work use the fine-tuned model for task planning.

System Architecture and Plan Executionr

We use SLAP with three heuristic policies: SearchForObject (π_{search}), PickUpObject (π_{pick}) and PlaceOn (π_{place}). π_{search} uses Detic, frontier-based exploration policy and a language query to explore the map until object described by the language query is found in the view of the robot. This is also one of the primary points of failure when LLM is integrated into the pipeline. LLM expects Detic to be able to handle any freeform query of the object (for example, “detect open drawer” is a typical output from the LLM however always fails as Detic has no notion of an open drawer).

π_{pick} is a heuristic picking policy which always grabs given object from the top given its mask from Detic and depth from camera. This is a generally robust policy but fails when *contextual, task-oriented grasps* are required in a task. For example consider a bottle placed within a cabinet, the only pick policy which will solve this scene is one where gripper grasps bottle laterally and not from the top. π_{place} places whatever object is in robot’s gripper on the surface previously detected by Detic. Another point of failure, since up-close Detic is not able to detect objects like table, counter surfaces, etc.

SLAP is used *after* the object is successfully detected by Detic, given language query, and is in the robot’s field of view. Note the robot at this point can be at any unconstrained orientation and position with respect to the object, the only condition sufficed here is that the object is within sight. We use SLAP’s interaction prediction module to estimate the affordance over this object with Detic’s mask as an additional feature, and predict an interaction point on the object. The robot then uses hand-engineered standoff orientations to move to a head-on position with respect to the object, where we use the full SLAP system to predict the action trajectory.

The standoff orientations are used so that SLAP can be tested fairly within reasonable bounds of the state distribution it was trained for. Training data for all our skills is recorded from a very narrow range of robot positioning with respect to the objects (see Appendix C.4). This means the action prediction module’s sense of action orientation is not robust to huge rotational variations over object’s position around the robot’s egocentric frame. Interaction prediction module on the other hand is very robust as it does not need to consider directionality, just the local structure of object and related affordance. See Fig. C.3 for details on how regions are assigned to specific objects, and related stand-off orientations provided to the robot so that it is always facing the object head-on after estimating the interaction point. Given predicted interaction point p_i, pre-determined standoff orientation vector $vec_{standoff}$ and pre-determined 3D standoff distance vector $dist_{standoff}$, the robot moves to an orientation $vec_{standoff}$ and position:

$$pos_{next} = p_i + dist_{standoff} * (-1 * vec_{standoff})$$

The orientation vector is of the form $(1, 0, 0)$ or $(0, 1, 0)$ in this evaluation.

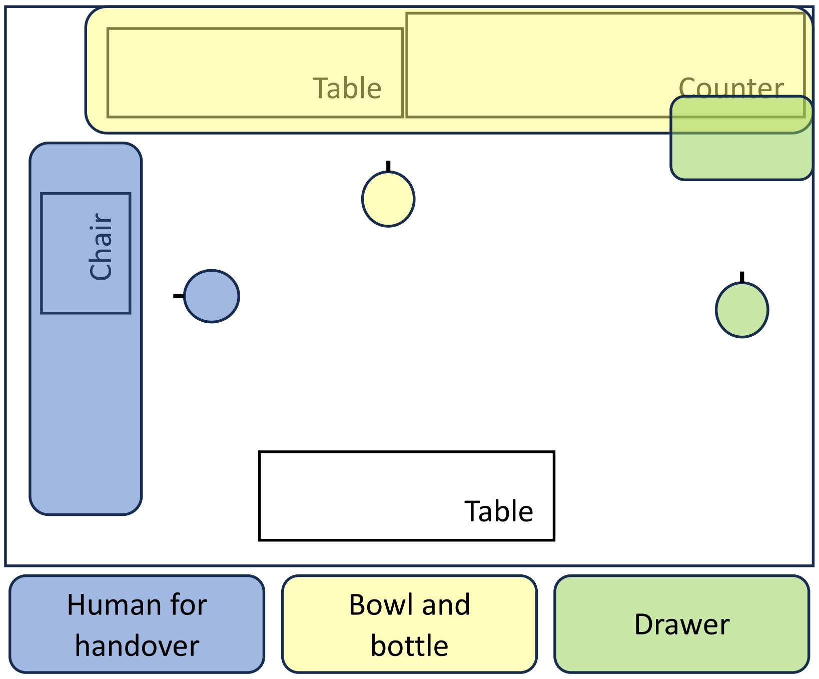


FIGURE C.3: Diagram showing where immovable objects were placed in the environment (note that chair can move anywhere within the blue region). The three colored regions signify the placement assignment for different artifacts involved in the tasks. The circular symbol signifies the robot's pre-determined orientation where the beak represents where the robot will be facing. Position for robot's placement is determined based on predicted p_i

C.4 Experimental Setup for Skills

Here we refer to atomic skills learned by SLAP as simple tasks or "tasks". This allows us to discuss corresponding "actions" that are defined in terms of the relative offset from the interaction point.

In vs. Out Of Distribution

We used a number of objects for our manipulation experiments, which included both in- and out-of-distribution objects (see Fig. C.4 and Fig. C.5). One goal of SLAP is to show that our methods generalize much better than others to different types of scenes and different levels of clutter.

We also randomized the objects with seen and unseen clutter around them, as well as placed them in unseen environments. Fig. C.7 and Fig. C.8 show the extent of variation captured in test scenes against training scenes for Stretch experiments. Fig. C.6 shows a range of test scenes from Franka evaluations.

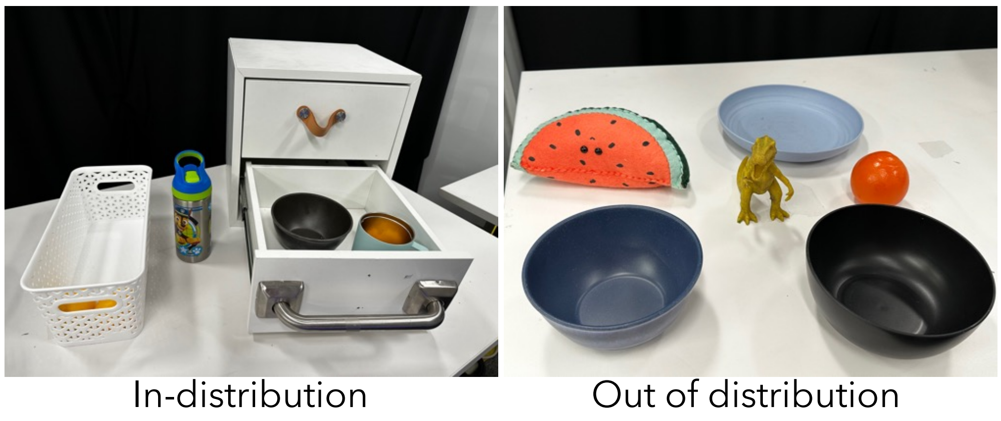


Figure C.4: Within distribution objects used at training time and out-of-distribution objects introduced during testing in our experiments.

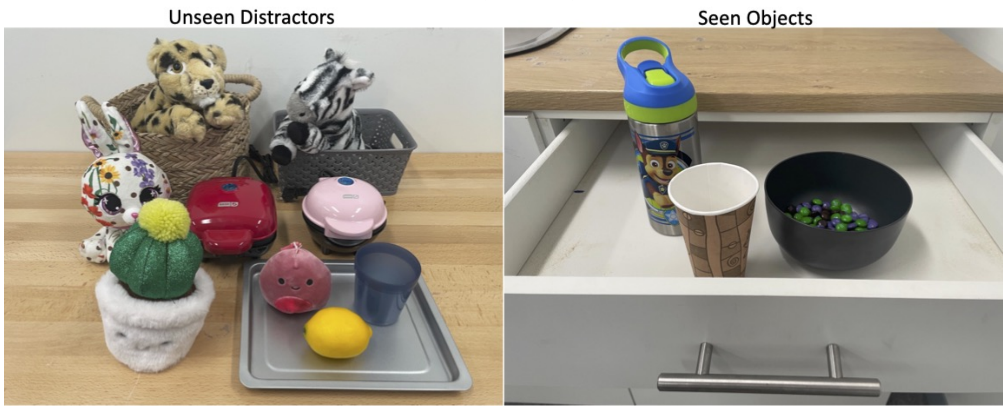


Figure C.5: Seen objects and unseen distractors used in longitudinal experiments with Stretch.

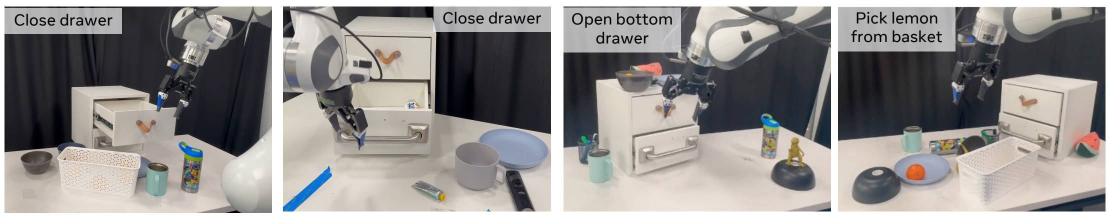


Figure C.6: Snapshot of test scenes from Franka evaluations to show the range of variation at test-time

Skill Definitions and Success Conditions

Every real-world task scene had a sub-sample of all within-distribution objects. Following describes skills and their success condition for Franka experiments:

1. Open the top drawer
   - *Task:* Grab the small loop and pull the drawer open. Drawer configuration within training data is face-first with slight orientation changes

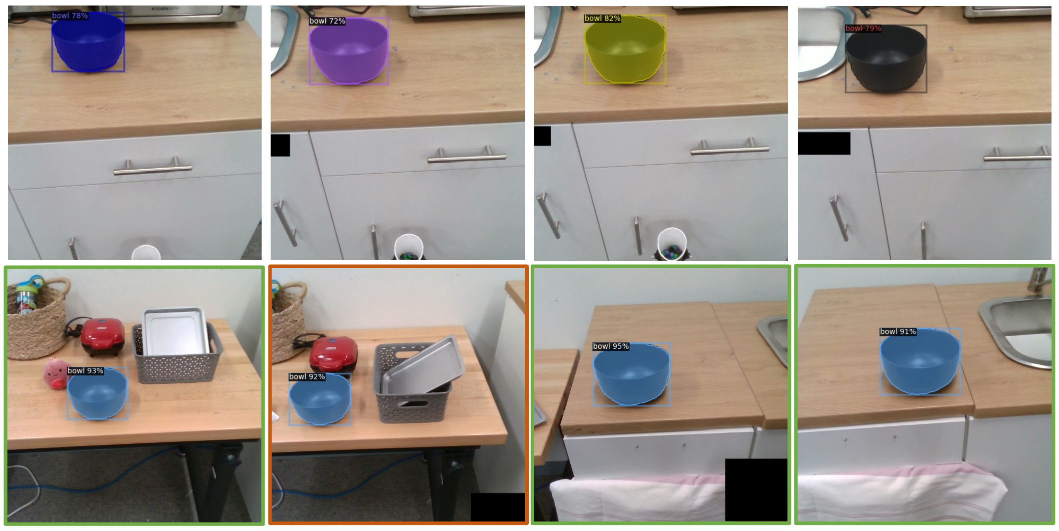


FIGURE C.7: Pour into bowl task: Showing variability and out-of-domain distribution covered by test against training samples. Top row: Training scenes. Note that the bowl was always placed somewhere on this particular region (right of sink) on the counter. Bottom row: Test scenes. Note bowl is surrounded by unseen clutter, placed in novel unseen environment and at relatively different positioning with respect to the camera and robot. Green boundary signifies successful episode, red a failed episode.

- *Action labeling:* Approach the loop, grab the loop, pull the drawer out
- *Success metric:* When the drawer is open by 50% or more

2. Open the bottom drawer
    - *Task:* Grab the cylindrical handle and pull the drawer open. Drawer configuration within training data is face-first with slight orientation changes. Note significantly different grasp is required than for top drawer
    - *Action labeling:* Approach the handle, grab it, pull the drawer out
    - *Success metric:* When the drawer is open by 50% or more
3. Close the drawer
    - *Task:* This task is unqualified, i.e. the instructor does not say whether to close the top or bottom drawer instead the agent must determine which drawer needs closing from its state and close it. Align the gripper with the front of whichever drawer is open and push it closed. The training set always has only one of the drawers open, in a front-facing configuration with small orientation changes
    - *Action labeling:* Approach drawer from the front, make contact, push until closed
    - *Success metric:* When the drawer is closed to within 10% of its limit or when arm is maximally stretched out to its limit (when the drawer is kept far back)

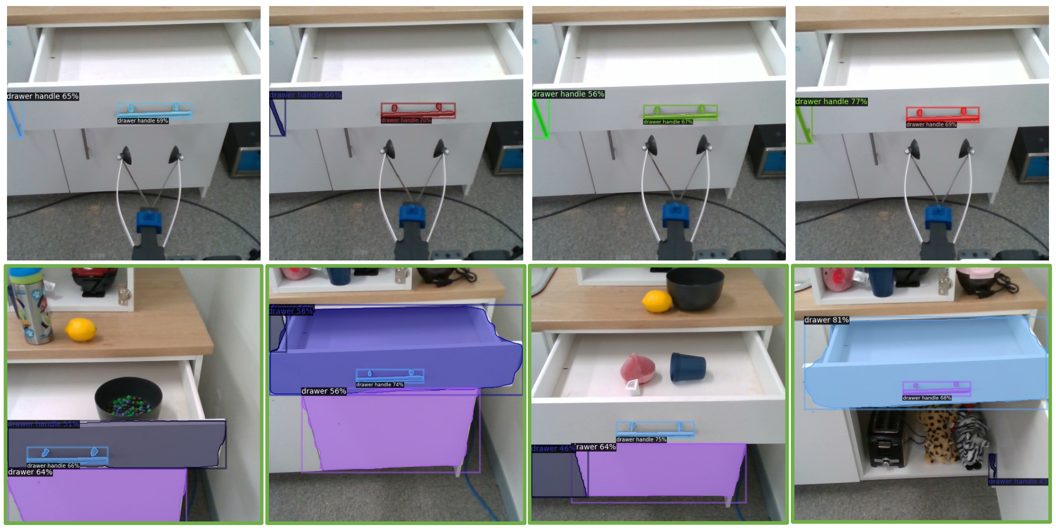


FIGURE C.8: Close drawer task: Showing variability and out-of-domain distribution covered by test against training samples. Top row: Training scenes. Note the absence of any clutter and narrow range of relative positioning of drawer with respect to the camera and robot. Bottom row: Test scenes. Note presence of objects used in other tests in the same frame. Green boundary signifies successful episode, red a failed episode.

4. Place inside the drawer
   - *Task:* Approach an empty spot inside the drawer and place whatever is in hand inside it
   - *Action labeling:* Top-down approach pose on top of the drawer, move to make contact with the surface and release the object, move up for retreat
   - *Success metric:* Object should be inside the drawer
5. Pick lemon from the basket
   - *Task:* Reach into the basket where lemon is placed and pick up the lemon
   - *Action labeling:*
   - *Success metric:* Lemon should be in robot's gripper
   - *Considerations:* Since the roll-out is open-loop and a lemon is spherical in nature, a trial was assigned success if the lemon rolled out of hand upon contact after the 2nd action. This was done consistently for both PerAct and SLAP.
6. Place in the bowl
   - *Task:* Place whatever is in robot's hand into the bowl receptacle
   - *Action labeling:* Approach action on top of the bowl, interaction action inside the bowl with gripper open, retreat action on top of the bowl
   - *Success metric:* The object in hand should be inside the bowl now

7. Place in the basket
   - *Task:* Place the object in robot's hand into the basket
   - *Action labeling:* Approach action on top of the free space in basket, interaction action inside the basket with gripper open, retreat action on top of the basket
   - *Success metric:* The object is inside the basket
8. Pick up the bottle
   - *Task:* Pick up the bottle from the table
   - *Action labeling:* Approach pose in front of the robot with open gripper, grasp pose with gripper enclosing the bottle and gripper closed, retreat action at some height from previous action with grippers closed
   - *Success metric:* The bottle should be in robot's gripper off the table

Notably, success for opening drawers is if the drawer is 50% open after execution; this is because sometimes the drawer is too close to the robot's base for it to open fully with a fixed-base Franka arm.

Following describes the skills undertaken in Stretch experiments and their success conditions:

1. Open the drawer
   - *Task:* Grab the handle of the drawer and pull the drawer open
   - *Action labeling:* Approach the handle, grab the handle, pull the drawer out and open grasp
   - *Success metric:* When the drawer is open by 100%
2. Close the drawer
   - *Task:* Align with the surface of drawer's face and push the drawer close
   - *Success metric:* When the drawer is closed within 10% of fully-closed configuration
3. Pour into bowl
   - *Task:* Skill starts with a cup filled with candies already in robot's gripper. Align cup with the bowl and turn the cup in a pouring motion
   - *Success metric:* When $\geq$ 50% of candies are in the bowl
4. Take bottle

- *Task:* Approach bottle with gripper orientation in the right configuration, grasp the bottle, lift the bottle up and retract keeping the grasp
- *Success metric:* The bottle is in robot's gripper in a stable configuration at the end of execution

5. Handover to person

   - *Task:* Approach hand of the person with object in hand, align with hand's surface and release the object, finally retracting the gripper back
   - *Success metric:* The object is in human's hand at the end of execution

Note that we count the success for pouring $\geq 50\%$ of the candies because we are comparing this task to pouring a liquid. Liquid would pour out completely in the intended final configuration due to different dynamics.

Language Annotations

In the following, we include the list of language annotations used in our experiments. Table C.2 shows the language that was used to train the model; we're able to show some robustness to different language expressions. We performed a set of experiments on held-out, out-of-distribution language despite this not being the focus of our work; this test language is shown in Table C.3.

Out of distribution Results from SLAP

We show more results for the attention point predicted by π_I in Fig. C.9. For the placement task, the agent has never seen a heavily cluttered drawer inside before, but it is able to find flat space that indicates placement affordance. For the bottle picking task, this sample has a lemon right next to the bottle which changes the shape of the point-cloud around the bottle. We see that π_I is able to find an interaction point albeit with placement different from expert and lower down on the bottle. Similarly, the open-top drawer sample has more heavy clutter on and around the drawer to test robustness.

Fig. C.10 shows the prediction and generated trajectory for picking up a previously unseen bottle. Note that while the models are able to detect the out of distribution bottle, the trajectory actually fails because the bottle is much wider and requires more accuracy in grasping. For the mobile manipulator domain, we observe SLAP performed better than vanilla PerAct on every count. Our hypothesis is that SLAP's better performance is due to the addition of semantic features, more efficient training, and higher resolution due to a non-grid point-cloud representation.

Motion Planning Failures

Our evaluation system has a simple motion planner which is not collision aware as a result we saw a number of task failures for both the models. However, we note that the frequency of task failures due to motion planning problems was higher for PerAct. We think it is because PerAct predicts each action of the same task as an entirely separate prediction trial, while SLAP forces continuity on the relative motions for the same task by centering them around the interaction point (see Fig. C.10). That said, we also

Task Name	Training Annotations
pick up the bottle	pick up a bottle from the table
	pick up a bottle
	grab my water bottle
pick up a lemon	pick the lemon from inside the white basket
	grab a lemon from the basket on the table
	hand me a lemon from that white basket
place lemon in bowl	place the lemon from your gripper into the bowl
	add the lemon to a bowl on the table
	put the lemon in the bowl
place in the basket	place the object in your hand into the basket
	put the object into the white basket
	place the thing into the basket on the table
open bottom drawer	open the bottom drawer of the shelf on the table
	pull the second drawer out
	open the lowest drawer
close the drawer	close the drawers
	push in the drawer
	close the drawer with your gripper
open top drawer	open the top drawer of the shelf on the table
	pull the first drawer out
	open the highest drawer
place in the drawer	put it into the drawer
	place the object into the open drawer
	add the object to the drawer

Table C.2: Examples of language used to train the model.

Task Name	Held-Out Test Annotations
Pick up the bottle	Grab the bottle from the table
	Pick up the water bottle
Open the top drawer	Pull top drawer out
	Open the first drawer
Place into the drawer	Add to the drawer
	Put inside the drawer

Table C.3: Examples of out-of-distribution language annotations used for evaluation

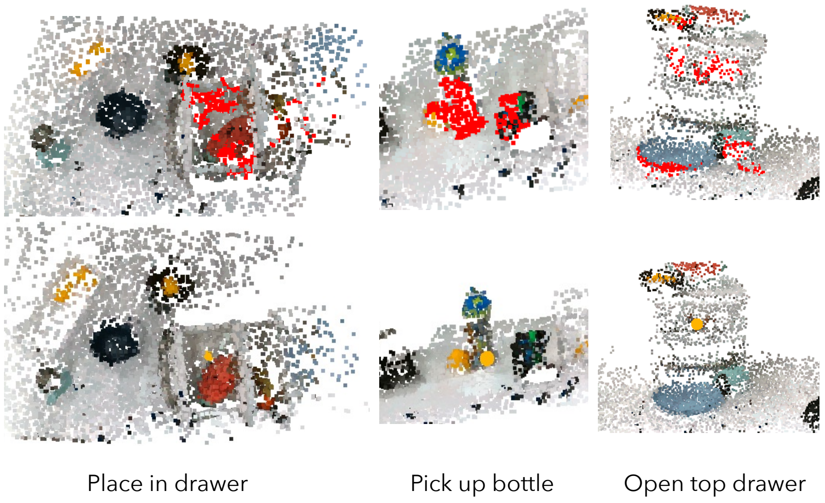


FIGURE C.9: Examples of out of distribution predictions made by π_I. We show that it is able to handle heavy clutter around the implicated object to predict interaction points. Note that the prediction for bottle picking is sub-optimal in this example.

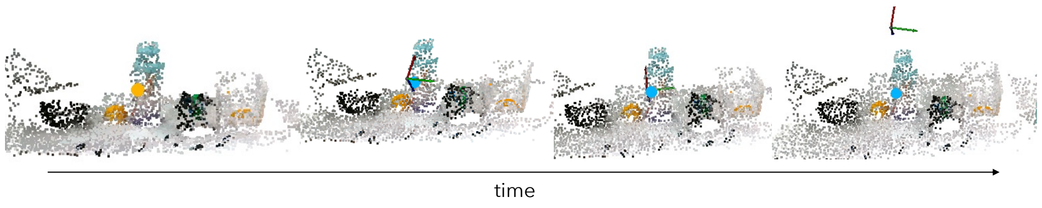


FIGURE C.10: A generalization example of success for our model. The new bottle has same shape as the within distribution bottle but is much taller, different in color and wider in girth. The model is able to predict the interaction site and a feasible trajectory around it. We note though the execution of this trajectory was a failure; due to wider girth of the bottle the predicted grasp was not accurate enough to enclose the object.

note with a collision-aware motion planner PerAct may not run into such issues as seen during our evaluations. However, the setting and conditions of the planner were the same between the two models in these evaluations. The authors note that they rely on good motion planning solutions in [190].

C.5 Additional Analysis

Ablations

Hybrid vs Monolithic Architecture (Table-top). We train SLAP and PerAct such that they observe same amount of data. SLAP outperforms PerAct on six of eight tasks when tested in in-distribution settings and five of eight tasks in out-of-distribution settings on Franka. PerAct performs equally well as our model for two of eight tasks on our in-distribution scenes. Similarly, for our "hard" generalization scenes, PerAct

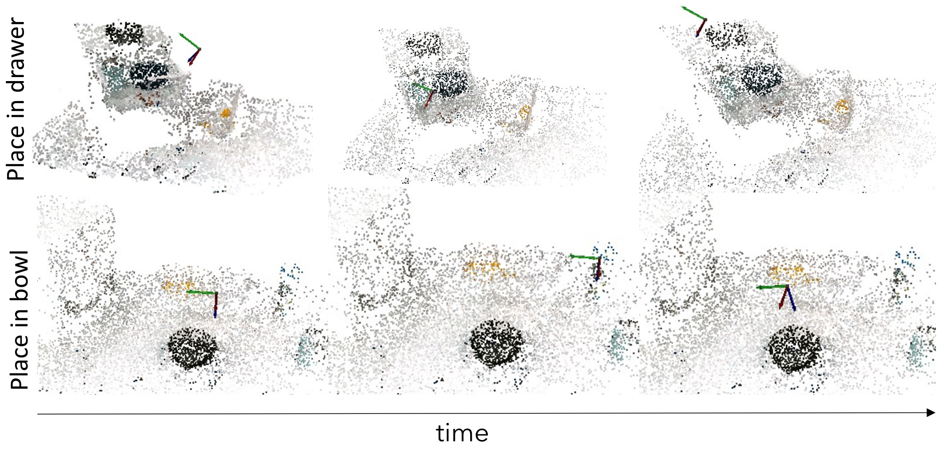


FIGURE C.11: Examples of failure cases for our baseline, PerAct, for the "place in drawer" and "place in bowl" tasks. In the top example, the gripper is moved from drawer's side towards inside, instead of from the top as demonstrated by expert. The gripper ends up pushing off the drawer to the side as our motion-planner is not collision-aware. Note that SLAP does not exhibit such behaviors as π_R implicitly learns the collision constraints present in demonstrated data. In the bottom example, each action prediction is disjointed from previous and semantically wrong.

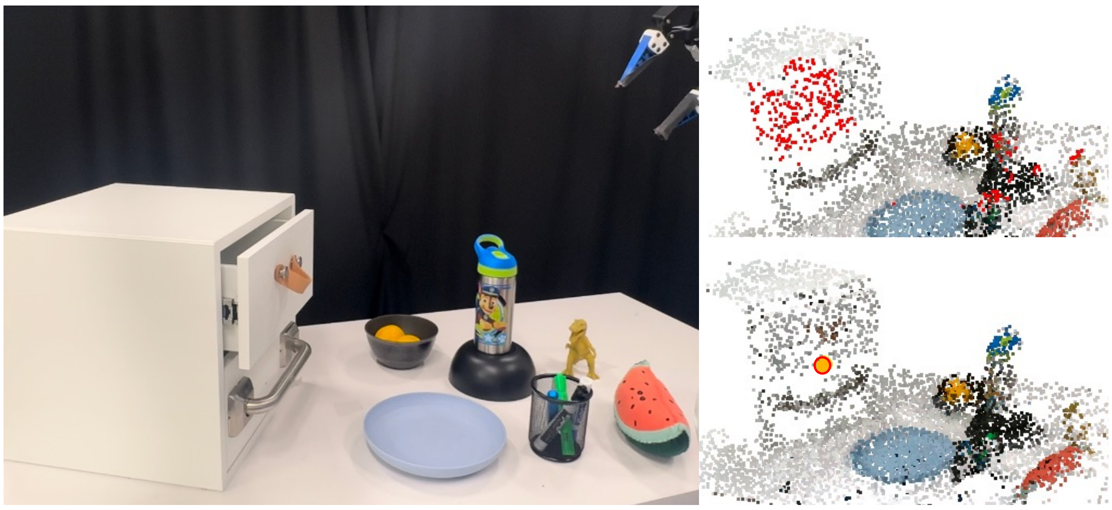


FIGURE C.12: An example out-of-distribution SLAP failure where an extreme sideways configuration of the drawer is paired with unseen distractors for the "open top drawer" skill. Note that the attention mask ranks other distractors in its top 5% and fails to choose an optimal interaction point.

performed equally well in two cases, and actually outperformed SLAP when picking up a bottle. Under similar experimentation conditions, SLAP outperforms PerAct in all four tasks in cluttered scenes for the mobile manipulator environment on Stretch. In failure cases, π_R predicted the correct trajectory, but not with respect to the right part of the object.

Unseen Scene Generalization. We see a drop in the success rate for both PerAct and SLAP when tested on out-of-distribution settings. PerAct would often predict the correct approach actions, but then it would fail to grasp accurately. With SLAP, however, we saw that p_I was predicted fairly accurately, but the regressor would fail for out-of-distribution object placements specifically because of bad orientation prediction. When π_I failed, it was because the position and orientation of the target object was dramatically different, *and* unseen distractors confused it. We see better results for SLAP under Stretch setting due to the addition of semantic features from Detic.

Visualizing the Learned Attention

Since we use scores to choose the final interaction point, our classifier model is naturally interpretable, being able to highlight points of interest in a scene. We visualize this attention by selecting the points with the highest 5% of interaction score given a language command l.

Language Generalization

By using pretrained CLIP language embeddings to learn our spatial attention module π_I, our model can generalize to unseen language to some extent. We tested this by running an experiment where we evaluate performance on in-distribution scene settings, prompted by a held-out list of language expressions. We choose three representative tasks for this experiment and run 10 tests with 2 different language phrasings.

C.6 Additional Related Work

We note some other related work related to the larger language-conditioned, mobile manipulation domain that SLAP is situated in, but not as directly relevant.

Vision-Language Navigation. Similar representations are often used to predict subgoals for exploration in vision-language navigation [16, 140, 80, 184, 17]. HLSM builds a voxel map [16], whereas FiLM builds a 2D representation and learns to predict where to go next [140]. VLMaps proposes an object-centric solution, creating a set of candidate objects to move to [80], while CLIP-Fields learns an implicit representation which can be used to make predictions about point attentions in responds to language queries [184], but does not look at manipulation. Similarly, USA-Net [17] generates a 3D representation with a lot of semantic features including affordances like collision. Such a representation can naturally be incorporated for collision-aware action plans at prediction time.

Appendix D

Appendix: From visuals and listener's distance to acoustic noise of robot's motion

D.1 Real-world Experiments

D.1.1 More details on ANAVI Framework

In Section 5.5, we provide experiment for robot planning a quieter path. For adapting robot path plans, we detail the ANAVI framework here. See Figure D.1.

(a) **Record the environment**: Localize and cache the visual observations. This is similar to the Matterport scans of a physical space that records the localization and panoramic views from each location. Let each location be a node in the graph.

(b) **Run ANP**: Given known (1) location of listeners and (2) the panoramas of the discrete locations of the map: For each node in the graph, use the panorama and the relative polar coordinates of the listener from this node as inputs to the ANP model to predict the max RIR dB. Use the max RIR dB with the robot's action audio files to estimate max dB for each action.

(c) **Time vs. Audio Noise Tradeoff**: Given the social scenario, use LLM prompting or heuristics to decide a weighting factor between the audio noise cost and the time-to-go cost. For example, if a person is having a video call, then the audio noise cost is high with respect to time taken to goal, versus if someone is listening to headphones, then the noise cost is not too high and the robot can optimize for time. With multiple listeners, we can estimate their sound sensitivity, and then incorporate their relative weighted plan into the overall cost.

(d) Plan: Use the weighted overall cost with an A* planner (or any other cost informed motion planner) to plan over what actions and path and velocity to take. We discretize slow/normal/fast velocities and encode them as actions with corresponding audio and time costs.

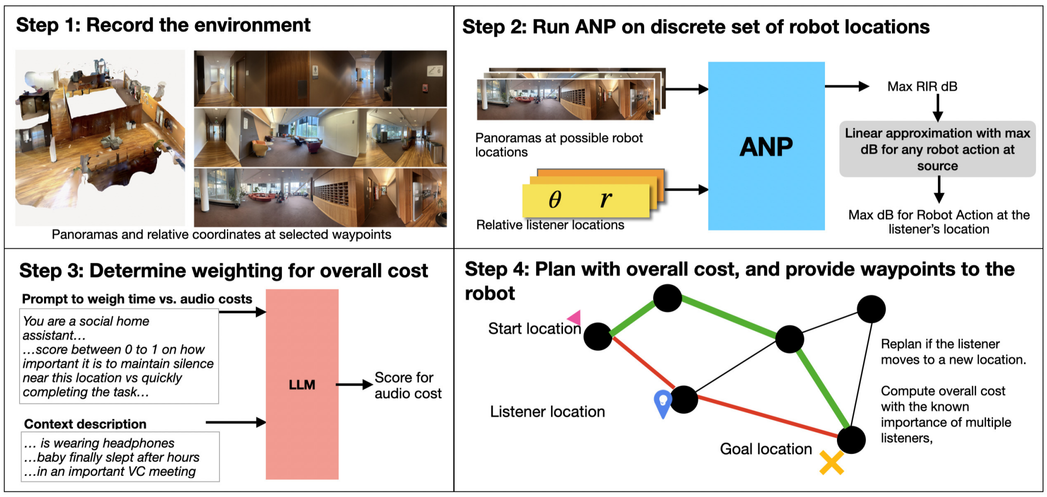

Step 1: Record the environment
Panoramas and relative coordinates at selected waypoints
Step 2: Run ANP on discrete set of robot locations
Panoramas at possible robot locations
Relative listener locations
θ r
ANP
Max RIR dB
Linear approximation with max dB for any robot action at source
Max dB for Robot Action at the listener's location
Step 3: Determine weighting for overall cost
Prompt to weigh time vs. audio costs
You are a social home assistant… …score between 0 to 1 on how important it is to maintain silence near this location vs quickly completing the task…
Context description
… is wearing headphones …baby finally slept after hours …in an important VC meeting
LLM
Score for audio cost
Step 4: Plan with overall cost, and provide waypoints to the robot
Start location
Listener location
Goal location
Replan if the listener moves to a new location.
Compute overall cost with the known importance of multiple listeners,

FIGURE D.1: Overview of ANAVI Framework: (a) Scan Environment, (b) Run ANP for listeners, (c) Weigh the audio noise vs. time costs, multiple listeners, and (d) plan with the overall cost.

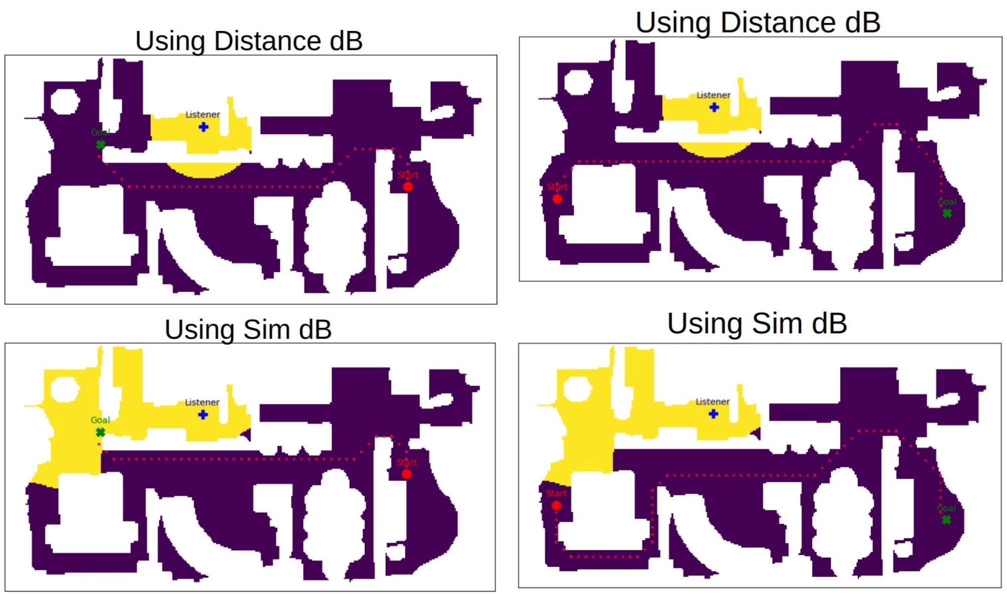


FIGURE D.2: We visualize how accurately estimating the noise cost (using the simulator as opposed to a distance heuristic) can enable better audio-aware paths.

D.1.2 Collecting Real-world Audio

Setup We compare the model's prediction with how loud the robot would be in the real world in Section 4. We describe how we collected the data and provide real-world acoustic measurements in Table D.1. In this experiment, we fix the robot at a location and vary the listener locations.

We want accessible commodity hardware to ensure easy replicability for researchers so they can use existing resources. We also want to ensure that an easy setup on a mobile robot for future research. To this end, we use mobile phones to capture images, record audio and measure distances.

Here we are the steps that we followed after we had recorded our the Stretch and Unitree audio onto our laptop. Our laptop acts as a proxy for the two robots in our experiments, and we place it at the location where we want to pretend the robot is at.

1. Decide an origin (robot location), and take a panorama picture. We select indoor locations like bedrooms, living area, dining area, open and closed working spaces, corridors, stairs, auditoriums and more.
2. Place speaker device with the recorded robot audio at the origin. We used our laptop and set the volume to its maximum. The speaker volume is fixed across all acoustic measurements.
3. Measure distance and direction for a listener location.
4. Place a microphone for recording at the listener location. We used an iPhone for recording.
5. Start the recording on the device and walk back to the origin to play the robot action sound. We note that the walk to the origin and back needs to be trimmed from the audio file so that it only contains the main audio. We thus found it helpful to first say 'Action', then play the audio file for robot action (at max volume) and then say, 'Cut'. After 'Cut' we walked back to the microphone (iPhone in our case) and stopped the recording. Saying 'Action' and 'Cut' helps to trim the audio faster by looking at the audio waveform to trim and only saving the audio in between. Researchers working in pairs can skip this complication and just use hand signals to denote starting and stopping the audio recording and speaker sound.
6. Extract the max decibels from the recorded audio files.

To estimate the max decibels at the origin (sound source), we need to know the volume gain adjustment on the original waveform. We guesstimate it to be a factor of 10, that is 20 dB. Note, the model is likely to have sim-to-real gap and our estimated value also accounts for the systematic error in the model's predictions. To obtain the model's

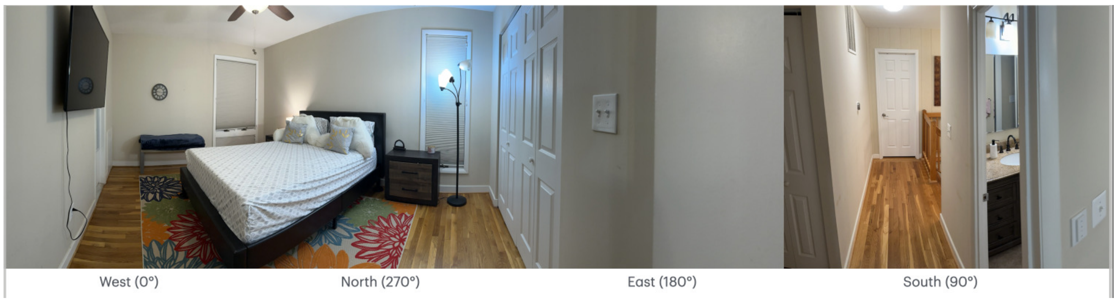


FIGURE D.3: Panorama taken from the sound source location. Note the directions west, north are in the room, facing east is a wall and the south direction leads out of the room into the hallway.

TABLE D.1: Real Audio for a Bedroom in a Single Family House.

Distance-Direction	Line-of-sight	Outside the room	Stretch Real dB	Unitree Go2 Real dB
1m-west	yes	no	54.2	69.7
1m-south	yes	yes	52.1	67.1
1m-south-1m-east	no	yes	49.7	66.1
2m-north	yes	no	51.7	67.3
2m-south	yes	yes	50.5	66.7
2m-north-2m-west	no	no	45.3	61.1
3m-west	yes	no	48.6	65.4
3m-south	yes	yes	48.2	65.4

prediction for the max decibel level by each robot's action, we use a linear approximation, that is, multiplying the normalized max dB output with the estimated max decibels at the origin (sound source). We report the error between this value obtained with linear approximation and that from the recorded audio files in the main paper.

Discussion Table D.1 shows the real audio recorded for Stretch at the fastest forward velocity and Unitree Go2 in running mode using our set-up. We keep the speaker at the origin corresponding to Fig. D.3 and vary our listener location. As we can see in the figure, moving west or north stays in the room while moving south or east corresponds to moving outside the room. We first compare the dB between "1m-west" and "1m-south" and observe how moving the sensor out the room decreases dB (as expected). We see this difference consistently although with smaller values as the distance from the source increases. This highlights how architectural geometries can have non-trivial impacts on audio (as expected). Additionally, we observe how line-of-sight between the source and listener affects the dB values. In case of obstacles like the bed or wall, the dB values decrease faster as compared to longer distances with line-of-sight (e.g. "1m-south-1m-east" is lower than "2m-north").

D.1.3 Qualitative Evaluations on Real-World Panoramas

We collect real-world panorama of indoor environments to visualize the model's predictions. We focus on sim-to-real performance of the audio prediction, as once accurately estimated, these values can be weighted against distance for audio-informed planners.

Setup To collect real world panoramas, we requested graduate students to contribute these panoramic images, by capturing their surrounding indoor environment for acoustic profiling. Contributors used their mobile phones at zoom level 1× and took a single panoramic image which we then resized to 256 × 1024. The images are then fed into a neural network that predicts the maximum decibel level at a given distance and direction from the robot. The ANP neural network is trained on a dataset of simulated Matterport renderings, and we qualitatively evaluate its performance on a dataset of real-world panoramas. Below we use Figure D.4 to explain our visualizations. The first row shows the real-world panorama. The left- and right-most edges correspond to 45°, and we change angles in clockwise direction from left-to-right. The reason for non-increasing order of angles on x-axis is that panorama's are taken left-to-right, that is, clockwise, whereas angles are measured anti-clockwise. Note that the cardinal directions indicated on the plots does not imply that the panorama starting direction and are only for illustration purposes.

The second row shows heatmap plot with the model's prediction for different distance and direction values. The direction is shown on x-axis, covering 360°, and the distances on y-axis range from 0 to 10 meters. Note that the direction is aligned with the visual image, i.e. it starts from -45°to 0, then reduces 270°, 180°, 90°, and finally to 45°.

In the third row, we plot the difference from the distance-based baseline. Since the distance-based baseline does not depend on the image, we visualize the variation in our model's prediction from the baseline. Here, the green indicates where the model predicts significantly higher normalized max dB values than the distance-based linear regression model, and purple indicates significantly lower normalized max dB values than naive baseline. Note that the dB values are normalized from 0 to 1.

Discussion We see that the model mostly predicts that the max dB values monotonically decrease with distance, though it may vary with direction. This is an important initial sign of life, as image features are very high-dimensional as compared to a scalar number used for relative distance. Nevertheless, the model largely learns to attend to the distance as input.

Does our model understand acoustic features beyond distance? In Figure D.5, we show cherry-picked examples of (D.5a) an open office area , (D.5b) a bedroom in a single family house. In Fig. D.5a, we observe that the model predicts high dB values at larger distances in open spaces, as in north-east direction. In terms of the difference with the

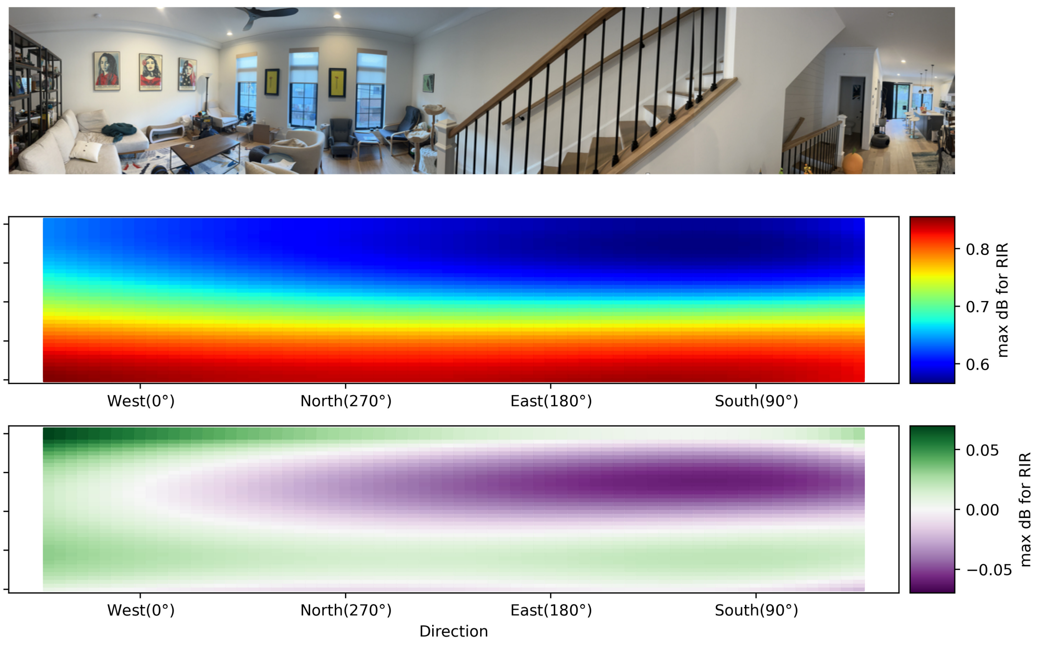
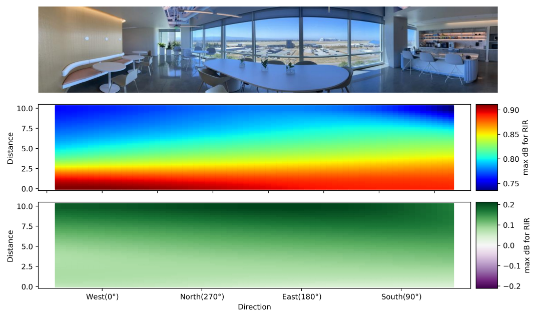


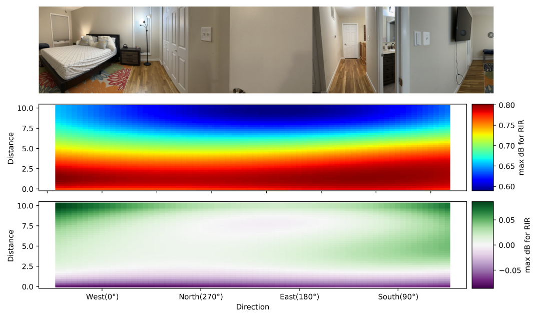
FIGURE D.4: **Setup for ANP Predictions on Real World Images**. This image visualizes ANP model predictions similar to Figure 7 from the main paper. The top row depicts a real-world panaromic image. The second row plots the maximum decibels of room impulse response $\hat{y}_{\text{ANP}}$ (max dB RIR). Note again that max dB RIR is normalized between 0 to 1. The third row plots the difference between the model's prediction and the linear regression prediction, i.e. $\hat{y}_{\text{ANP}} - \hat{y}_{linreg}$.

baseline, we observe that the model predicts higher sound intensity than the baseline at larger distances. In Fig. D.5b, we observe that the model predicts low values at larger distance for some areas where there is a wall. Walls acts as an obstacle for sound intensity (see east direction). Additionally, we observe that the model predicts higher values at larger distances in open spaces, within the room (see west and north directions), than in the outside corridor (see south direction). In the second subplot, we note that our model predicts higher max dB RIR than the baseline at larger distances, and lower values near the sound source.

Through the qualitative real-world data evaluation, we note several sim-to-real gaps. First, the real images are higher quality than the Habitat environment's Matterport3D renderings. This includes differences in how visually environment attributes as inferred, such as depth, material texture, and architectural geometry. Second, lighting changes and variations are more drastic in real-world than the static Matterport renderings in simulation. This has implications on acoustic noise prediction models deployed on the home robots, and we need to improve the robustness to lighting variations for the same scene. Third, the height perspective of our images and the relative angle to the ground differed between contributors, while this was a fixed constant in simulation. Differences in camera height, pan and tilt requires slightly different interpretation of the distances from visual features.



(A) **Open office area**. High dB values predicted at larger distances for open spaces (see east), and at shorter distances in the corners, near walls (see west and south-west). ANP predicts higher sound intensity than the baseline, especially at larger distances, as shown in green.



(B) **Bedroom in a single family house** Low dB values predicts near wall as an obstacle (see east) and outside the room (see south). High dB values at larger distances in open spaces (see west). ANP predicts higher values than the baseline at larger distances (green), and lower values near source (purple).

FIGURE D.5: We show cherry-picked examples of model's prediction in (a) an open office area, and (b) a bedroom in a single family house.

We observe that the current model performs poorly on these diverse set of real-world panoramic images. The first row (Figures D.6a and D.6b) shows two panoramas of corridors where we expect corridors to have high predicted db regardless of sensor distance while the corresponding wall area have small values when the distance is large (as then the sensor would be behind the wall). Instead, we did not see any noticeable correlation to the corridors. Upon closer inspection on predictions, the model seems to struggle with identifying walls and their impact on sound. We expect high dB predictions in rooms when the distance falls in the room, but then a large drop-off to low dB prediction outside rooms. But in Fig D.6c, D.6e and D.6g, we do not see the drop-off.

Our results show that zero-shot generalization to real panoramas is not easy. It is unclear if this issue lies in simulation inaccuracies, an inadequate ANP model, or the distribution shift to real-world panoramas. Unfortunately, the model is unable to consistently capture meaningful visual features like walls or corridors that would effect dB. Future work should address these sim-to-real differences to enable using an ANP model effectively in real environments. In future, we hope to bridge the sim2real gap with (1) effective domain randomization for camera poses, lighting and robot's viewing angles, (2) improving model training with auxiliary losses, and (3) fine-tuning model with real-world measurements for acoustics, distances and visuals, collected from diverse indoor environments.

D.2 Simulation

D.2.1 Training Data Generation

We generate training data for ANP in simulation using SoundSpaces 2.0. Here we provide additional details for the data generation process: (i) *Uniform sampling across the map.* We first divide the map into grids to get 100 points. These points serve as circle centers to sample a suitable navigable point within 20 meters radius for the robot's location as the source. We then use the robot's location as the circle center and sample a listener location within 10 meters radius. (ii) *Visual panorama for the robot's source location.* We sample RGB camera observations 4 times with rotations of 90 degrees at 256×256 resolution and 90 degrees field-of-view. We discuss the audio data generation below as part of room impulse response (RIR).

D.2.2 More details on Room Impulse Response (RIR)

An impulse response is the output signal (sound) that results when an audio system or environment is excited by an idealized impulse signal at a specific location. An impulse signal has a very short duration and theoretically contains all frequencies.

We notionally define the simulated impulse response at the listener's location as $w(t)$, a time-domain waveform. We obtain an impulse response using standard techniques that we define below. Let an impulse be generated at the robot's location $\boldsymbol{s} = \{s_x, s_y\}$ and heard by a listener at location $\boldsymbol{l} = \{l_x, l_y\}$. Then, using bi-directional ray tracing in simulation, we obtain the maximum sound intensity at the listener location. First, we convert the simulated IR waveform $w(t)$ from the time domain to the frequency domain $W(f)$ with Fast Fourier transform. Second, we compute the sound intensity $I(f) = W(f)^2/(\rho * c)$, where ρ is air density and c is the speed of sound in air.

We extract the maximum sound intensity as $I_{max} = \max I(f)$. Since the faintest sound human ears can hear is considered $I_o = 10^{-12} W/m^2$, the sound intensity is converted into to decibels by $dB_{max} = 10 \log_{10} I_{max} + 120$. As we make a modeling design choice to assume an impulse of $1 W/m^2$ in simulation, we define 120 dB sound pressure level at the source. A large impulse value ensures a good signal-to-noise ratio while avoiding distortion. To ensure that the decibel values are normalized for training, we assume the highest decibel value as 128 and normalize it between 0 and 1 to get target labels y. For simplicity, we assume that the listener captures a mono-channel audio.

More concretely, we do the following steps:

$$w(t) = \text{SimulateIR}(\text{from} = p_{source}, \text{at} = p_{receiver}) \tag{D.1}$$

$$W(f) = \text{Fast Fourier Transform}(w(t)) \tag{D.2}$$

$$I(f) = (W(f)^2)/(\rho * c) \tag{D.3}$$

$$I_{max} = \max I(f) \tag{D.4}$$

$$I_{max} = \text{clip}(I_{max}, \text{min} = 10^{-12}, \text{max} = 10^{0.8}) \tag{D.5}$$

$$dB_{max} = 10 \log_{10} I_{max} + 120 \tag{D.6}$$

$$y = dB_{max}/128 \tag{D.7}$$

Here $w(t)$ is the time-domain waveform generated at the receiver's location, $W(f)$ is the frequency domain waveform, $I(f)$ is the frequency domain sound intensity through air computed by root mean square. The ρ is air density and c is speed of sound in air. We use the maximum sound intensity and convert it to decibels. To ensure that the decibel values are normalized for training, we assume the highest value of decibel as 128 and compute target labels y. Since the faintest sound human ears can hear is considered $I_o = 10^{-12} W/m^2$, we convert the max sound intensity into to decibels by $dB_{max} = 10 \log_{10}(I_{max}/I_o) = 10 \log_{10} I_{max} + 120$.

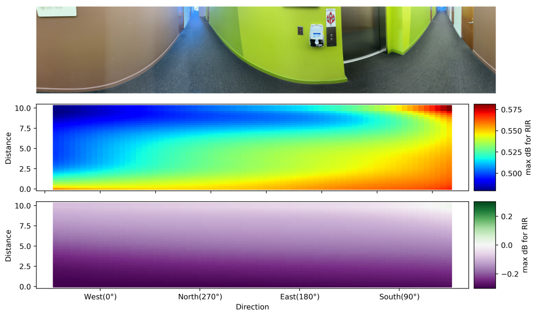
D.2.3 Visualization of Fixed Robot and Fixed Listener maps

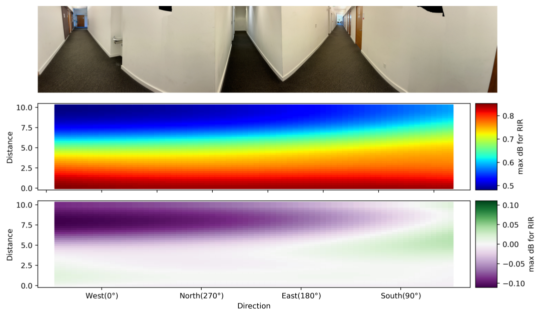
In Section 5.4.3, we discuss the fixed listener and fixed robot prediction maps for analysis. Figure D.7 shows a few of the fixed listener and robot maps, and highlights the potential failure cases of the ANP model.

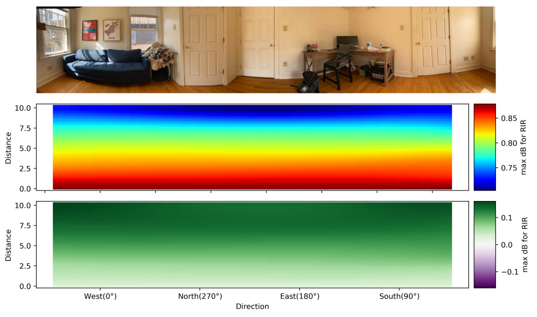
D.3 Discussion and Broader Scope

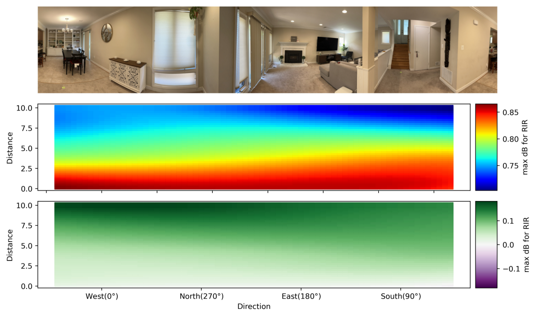
With audio-level prediction, we are one step closer to future home robots. Nevertheless, we must develop reliable and calibrated loudness-aware text-to-speech response and navigation policies. Current acoustic map predictions must be calibrated to real-world audio and appropriate weighting relative to time cost. Learning from visual features is prone to overfitting. Collecting audio-visual measurement data over a larger, diverse set of environments can help alleviate this. This will eventually facilitate context-aware robot policies for different loudness sensitivities.

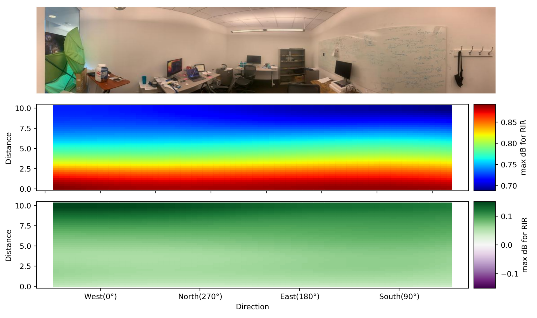
Our current framework does not address the effects of ambient noise. In this work, we focus on first-order principles that directly involve the robot's audio in relation to the audio. However, in real scenarios, the effect of other ambient noises (e.g. a loud TV on) changes how the robot's noise effects the listener. Incorporating other noise sources requires more complex reasoning like sound source separation and noise processing that is less robotics related. For example, this could involve predicting that a refrigerator makes ambient noise and thus the robot can be louder near it, but this has been studied in acoustic literature and was not the main focus of our work. Future work could extend ANAVI framework to handle this in two ways: (i) by adjusting the "Time vs Audio Noise Tradeoff" (See D.1.1 (c)) based on the prompt (e.g. the TV is on) that LLM can use to adjust the audio noise tradeoff. (ii) by modifying ANP to include ambient noise as 'signal' and robot movement sounds as 'noise' to guide the robot's path and velocity planning.

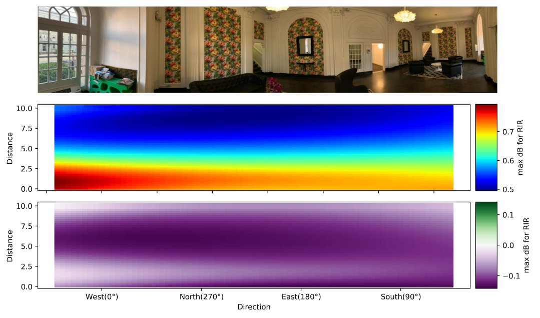

(A) Single Corridor: ANP model predicts softer

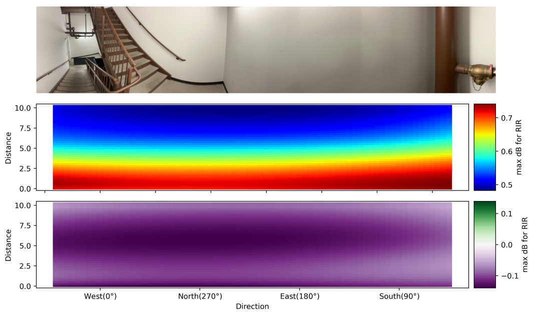

(B) Multiple Corridors: Mixed


(C) Single Room: ANP model predicts louder


(D) Open Dining Area: ANP model predicts louder


(E) Confined Room: ANP model predicts louder


(F) Open Living Room: ANP model predicts softer


(G) Confined Staircase: ANP model predicts softer

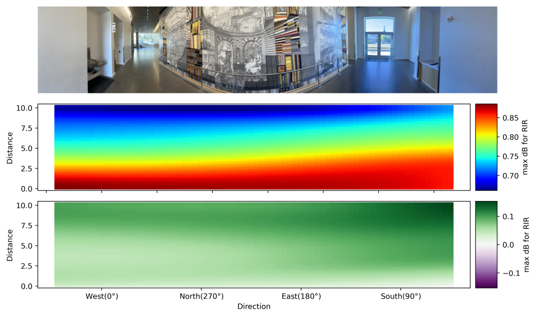

(H) Wide Hallway: ANP model predicts louder

Figure D.6: Sim-to-real gap when applying the ANP model to real world panoramas. For each plot, the top row shows a real world panorama, second row the ANP predictions depending on the distance and angle to the sensor location, and the third row the difference compared a basic linear regression estimate (green = ANP predicts louder).

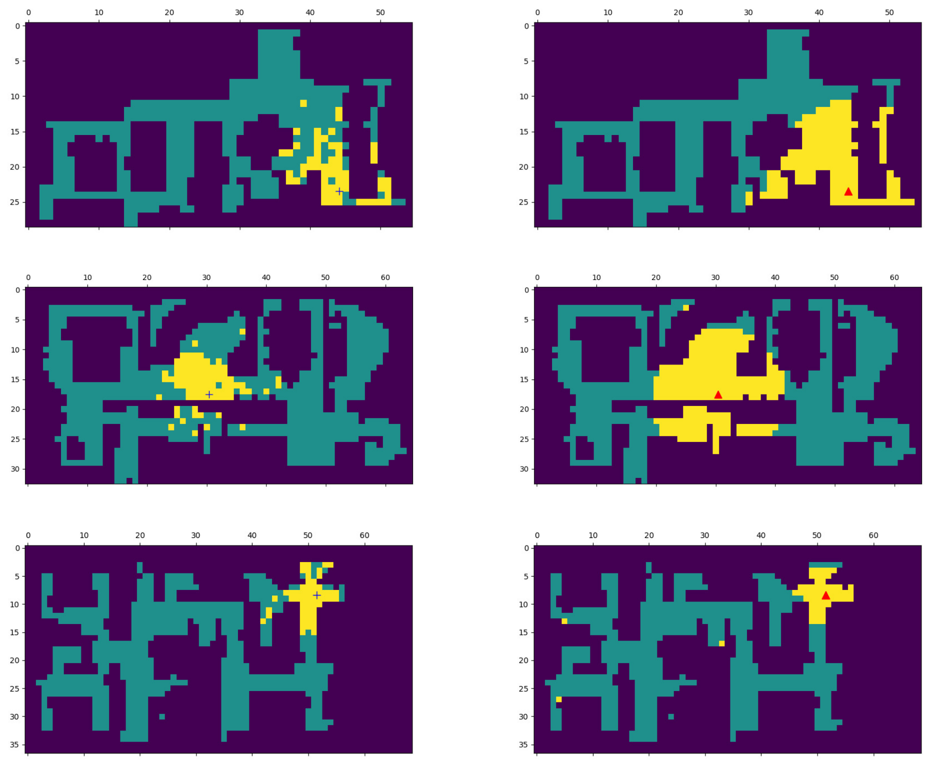


FIGURE D.7: ANP Predictions over the Birds Eye View (BEV) map for (Left) Fixed listener (indicated with blue plus sign) and (right) Fixed robot (indicated with red triangle). Note that ANP uses panorama from robot's point of view, implying that the fixed listener maps use different visual features whereas fixed robot uses the same visual for each cell. The relative distance and direction from robot to the listener changes for each cell. We show all locations with 96 dB (more than 85%) in yellow, rest in blue and non-navigable regions at height 0 in purple. **Top**: The fixed listener predictions are spread more along the corridor (up-down) than (left-right). The fixed robot plot is more smoothly and uniformly spread across the map. **Middle**: The fixed listener predictions align with intuition that the sound should be louder in the open space compared to the area at same distance but separated by a wall. The fixed robot predictions plot follows the distance heuristic but not the common intuition. **Bottom**: Both fixed listener and fixed robot maps show the expected spread of sound at the cross-intersection. It is interesting to see that the fixed listener plots are quite noisy while the fixed robot one is very circular (and doesn't seem to care too much about walls)

Appendix E

Appendix: Active visual querying

E.1 POMDP details

Formally, we model the environment as a POMDP

$$(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{R}, \gamma),$$

where:

- $\mathcal{S}$ is the (hidden) state space of the object and camera poses. This represents all possible true configurations of the environment. Since it's not directly observable, the agent doesn't know its exact contents, but it would include the precise 6-DoF poses of all objects, the camera(s), and the location of the requested visual information within the scene.
- $\mathcal{A}$ is the action space that effectively changes the 6-DoF pose for the camera(s) and object(s), either directly or through the robot embodiment.
- $\mathcal{T}(s_{t+1} \mid s_t, a_t)$ defines state transitions under the action a_t.
- $\mathcal{O}(o_t \mid s_t, a_{t-1})$ is the observation distribution: at each step t, the agent receives $o_t = \{v_t^c\}_{c \in C} \cup \{\text{proprio}\}$, where $v_t^c \in \mathbb{R}^{H \times W \times 3}$ are one or more RGB frames and 'proprio' denotes the optional pose history or joint readings.
- $\mathcal{R}(s_t, a_t)$ gives a positive reward if the requested information (e.g., expiration date) is fully visible in the current view v_t and zero otherwise. Additional information like the depth and segmentation mask for the view is used as part of s_t for computing visible area in occlusion.
- γ is the discount factor.

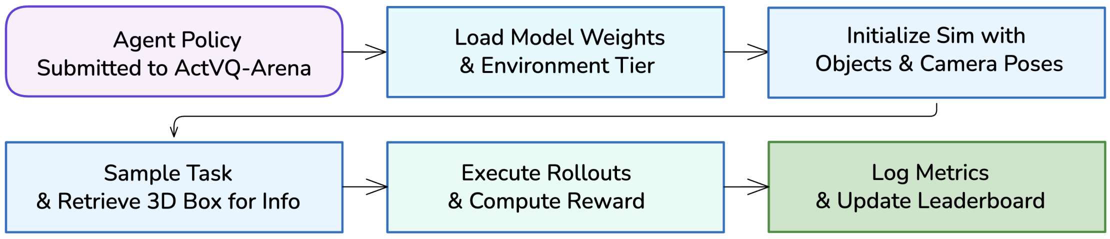


Figure E.1: ActVQ-Arena Policy Submission Workflow: Upon receiving a new agent policy, ActVQ-Arena loads the model weights and configures the policy for its designated environment tier. The system then instantiates simulation with randomized object layouts and camera poses to ensure robust evaluation across diverse scenarios. For each trial, ActVQ-Arena selects a specific task objective and retrieves the corresponding precomputed 3D annotation points that define where relevant information is. The policy predicts action for the agent, and ActVQ-Arena computes a reward at each timestep and number of steps taken. After completing all rollouts, ActVQ-Arena reports the performance metrics - such as task success rate, average visible area, and cumulative reward to the leaderboard, thereby ranking the submission among competing policies.

An episode terminates when the agent issues a "stop" action (no further pose changes), or upon failure: (i) an object moves out of reach, or (ii) a collision exceeds a safety threshold. The agent's objective is to maximize expected cumulative reward by efficiently aligning camera and object poses to reveal the queried information in as few steps as possible.

E.2 Submission Workflow

When a new agent policy is submitted to ActVQ-Arena, it starts with loading the policy's model weights and configuring its designated environment tier. It then launches a fresh simulation with randomized object arrangements and camera poses to challenge the policy across diverse scenarios. For each task, the arena samples the objective and pulls in the precomputed 3D annotation points that define success criteria. The policy drives the agent through the simulation, and the system computes rewards based on how well the agent's actions reveal and align those points in the camera view. Finally, ActVQ-Arena aggregates the performance metrics, logs them, and updates the global leaderboard to reflect the new submission's standing. Refer Figure E.1.

E.3 Reward Calculation

The reward function evaluates a candidate view of an object via the following pipeline:

1. Transform the object's 3D bounding-box corners p_i^{local} ($i =$TL,TR,BR,BL) into world coordinates and then into camera coordinates $p_i = (x_i, y_i, z_i)$.

2. Project p_i onto the image plane to obtain pixel coordinates (u_i, v_i).

3. Compute the projected quadrilateral area via the Shoelace Theorem, with vertices in the fixed order TL(u_1, v_1), TR(u_2, v_2), BR(u_3, v_3), BL(u_4, v_4):

$$A = \frac{1}{2}\left|\sum_{i=1}^{4}(u_i\, v_{i+1} - u_{i+1}\, v_i)\right|, \quad (u_5, v_5) \equiv (u_1, v_1).$$

4. Check visibility and size: - If any depth $z_i \leq 0$ or $A \leq 0$: fail ("not visible"). - If $A < \tau\,(WH)$ for image width W, height H: fail ("too small").

5. (Optional) Check alignment: require box "up"/"right" directions to have positive dot-product with camera axes; otherwise: fail ("misaligned").

6. Assign

$$R = \begin{cases} 1 & \text{if all tests pass,} \\ 0 & \text{otherwise.} \end{cases}$$

E.4 Train test split

We generate 500 initial configurations for 'expiry date' and 'instructions' in No-Robot setting and split by 90-10 split. The shortest path is computed through a cubic interpolation planner. This is not perfect as it is not collision aware but provides a good heurisitc for most scenarios where the object needs to move towards the camera without clutter. The object is always initialized in the camera's field of view.

Bibliography

[1] Baris Akgun et al. "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective". In: *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction.* 2012, pp. 391–398.

[2] Ekin Akyürek et al. *What learning algorithm is in-context learning? Investigations with linear models.* 2023. arXiv: `2211.15661 [cs.LG]`.

[3] Danny Driess et al. "PaLME: An Embodied Multimodal Language Model". In: *ArXiv* abs/2303.03378 (2023).

[4] Justin Carpentier et al. *Pinocchio: fast forward and inverse dynamics for poly-articulated systems.* https://stack-of-tasks.github.io/pinocchio. 2015–2021.

[5] Michael Ahn et. al. "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances". In: *CoRL*. 2022.

[6] Jean-Baptiste Alayrac et al. "Flamingo: a Visual Language Model for Few-Shot Learning". In: *ArXiv* abs/2204.14198 (2022).

[7] Peter Anderson et al. "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2018, pp. 6077–6086.

[8] Peter Anderson et al. "On evaluation of embodied navigation agents". In: *arXiv preprint arXiv:1807.06757* (2018).

[9] Stanislaw Antol et al. "VQA: Visual Question Answering". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* 2015, pp. 2425–2433.

[10] Montserrat Gonzalez Arenas et al. "How to Prompt Your Robot: A PromptBook for Manipulation Skills with Code as Policies". In: *2nd Workshop on Language and Robot Learning: Language as Grounding.* 2023. URL: `https://openreview.net/forum?id=T8AiZj1QdN`.

[11] Montserrat Gonzalez Arenas et al. "How to Prompt Your Robot: A PromptBook for Manipulation Skills with Code as Policies". In: *2024 IEEE International Conference on Robotics and Automation (ICRA).* 2024, pp. 4340–4348. DOI: `10.1109/ICRA57147.2024.10610784`.

[12] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. “Human-to-Robot Imitation in the Wild”. In: *RSS*. 2022.

[13] Shikhar Bahl et al. “Affordances from human videos as a versatile representation for robotics”. In: *Computer Vision and Pattern Recognition*. Apr. 2023.

[14] Yonatan Bisk et al. “PIQA: Reasoning about Physical Commonsense in Natural Language”. In: *Thirty-Fourth AAAI Conference on Artificial Intelligence*. 2020. URL: `https://yonatanbisk.com/piqa`.

[15] Kevin Black et al. “π_0: A Vision–Language–Action Flow Model for General Robot Control”. In: *arXiv preprint arXiv:2410.24164* (2024).

[16] Valts Blukis et al. “A persistent spatial semantic representation for high-level natural language instruction execution”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 706–717.

[17] Benjamin Bolte et al. “USA-Net: Unified Semantic and Affordance Representations for Robot Memory”. In: *arXiv preprint arXiv:2304.12164* (2023).

[18] Konstantinos Bousmalis et al. “RoboCat: A Self-Improving Generalist Agent for Robotic Manipulation”. In: Sept. 2023.

[19] Anthony Brohan et al. “RT-1: Robotics transformer for real-world control at scale”. In: *arXiv preprint arXiv:2212.06817* (2022).

[20] Tim Brooks et al. *Video generation models as world simulators*. 2024. URL: `https://openai.com/research/video-generation-models-as-world-simulators`.

[21] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[22] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *ArXiv* (2020).

[23] Liyu Cao and Joachim Gross. “Cultural differences in perceiving sounds generated by others: Self matters”. In: *Frontiers in psychology* 6 (2015), p. 163665.

[24] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. “Goal-Conditioned Reinforcement Learning with Imagined Subgoals”. In: (July 2021). arXiv: `2107.00541 [cs.LG]`.

[25] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. “Learning Video-Conditioned Policies for Unseen Manipulation Tasks”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 909–916. DOI: `10.1109/ICRA48891.2023.10161336`.

[26] Angel Chang et al. “Matterport3D: Learning from RGB-D Data in Indoor Environments”. In: *International Conference on 3D Vision (3DV)* (2017).

[27] Peixin Chang et al. “Learning visual-audio representations for voice-controlled robots”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 9508–9514.

[28] Devendra Singh Chaplot, Deepak Pathak, and Jitendra Malik. "Differentiable spatial planning using transformers". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 1484–1495.

[29] Changan Chen et al. "Learning Audio-Visual Dereverberation". In: *ICASSP*. 2023.

[30] Changan Chen et al. "Novel-view acoustic synthesis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 6409–6419.

[31] Changan Chen et al. "Sim2Real Transfer for Audio-Visual Navigation with Frequency-Adaptive Acoustic Field Prediction". In: *arXiv preprint arXiv:2405.02821* (2024).

[32] Changan Chen et al. "SoundingActions: Learning How Actions Sound from Narrated Egocentric Videos". In: *arXiv preprint arXiv:2404.05206* (2024).

[33] Changan Chen et al. "SoundSpaces 2.0: A Simulation Platform for Visual-Acoustic Learning". In: *NeuriPS 2022 Datasets and Benchmarks Track* (2022).

[34] Changan Chen et al. "Visual acoustic matching". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18858–18868.

[35] Lili Chen et al. "Decision transformer: Reinforcement learning via sequence modeling". In: *Advances in neural information processing systems* 34 (2021).

[36] Ting Chen et al. "Big self-supervised models are strong semi-supervised learners". In: *Advances in neural information processing systems* 33 (2020), pp. 22243–22255.

[37] Yulong Chen et al. "AdaPrompt: Adaptive Model Training for Prompt-based NLP". In: *arXiv preprint arXiv:2202.04824* (2022).

[38] Ziyang Chen et al. "Real Acoustic Fields: An Audio-Visual Room Acoustics Dataset and Benchmark". In: 2024.

[39] Cheng Chi et al. "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[40] Paul F Christiano et al. "Deep reinforcement learning from human preferences". In: *Advances in neural information processing systems* 30 (2017).

[41] Samuel Clarke et al. "Learning Audio Feedback for Estimating Amount and Flow of Granular Material". In: *Proceedings of The 2nd Conference on Robot Learning*. Ed. by Aude Billard et al. Vol. 87. Proceedings of Machine Learning Research. PMLR, 2018, pp. 529–550. URL: `https://proceedings.mlr.press/v87/clarke18a.html`.

[42] Charles I. Connolly. "The Determination of Next Best Views". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 1985, pp. 432–435.

[43] Yuchen Cui et al. “Can Foundation Models Perform Zero-Shot Task Specification For Robot Manipulation?” In: *Learning for Dynamics and Control Conference*. PMLR. 2022, pp. 893–905.

[44] Aidan Curtis et al. “Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 1940–1946.

[45] Abhishek Das et al. “Embodied Question Answering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 1–10.

[46] Sudeep Dasari and Abhinav Gupta. “Transformers for one-shot visual imitation”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 2071–2084.

[47] Xinke Deng et al. “Self-supervised 6d object pose estimation for robot manipulation”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 3665–3671.

[48] Eadom Dessalene et al. “Forecasting Action Through Contact Representations From First Person Video”. en. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.6 (June 2023), pp. 6703–6714.

[49] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[50] Prafulla Dhariwal and Alexander Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[51] Yiming Ding et al. “Goal-conditioned imitation learning”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., Dec. 2019, pp. 15324–15335.

[52] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations*. 2021. URL: `https://openreview.net/forum?id=YicbFdNTTy`.

[53] Maximilian Du et al. “Play it by ear: Learning skills amidst occlusion through audio-visual imitation learning”. In: *Robotics: Science and Systems* (2022).

[54] Yilun Du et al. “Learning Universal Policies via Text-Guided Video Generation”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: `https://openreview.net/forum?id=bo8q5MRcwy`.

[55] Yan Duan et al. “One-shot imitation learning”. In: *Advances in neural information processing systems* 30 (2017).

[56] Frederike Dümbgen et al. “Blind as a bat: Audible echolocation on small robots”. In: *IEEE Robotics and Automation Letters* 8.3 (2022), pp. 1271–1278.

[57] Debidatta Dwibedi et al. “Flexcap: Describe Anything in Images in Controllable Detail”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 111172–111198.

[58] Debidatta Dwibedi et al. “Temporal Cycle-Consistency Learning”. In: *Computer Vision and Pattern Recognition.* 2019.

[59] Gamaleldin F. Elsayed et al. “SAVi++: Towards end-to-end object-centric learning from real-world videos”. In: *Advances in Neural Information Processing Systems.* 2022.

[60] Hao-Shu Fang et al. “Graspnet-1billion: A large-scale benchmark for general object grasping”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2020, pp. 11444–11453.

[61] Jared Fernandez et al. “The Framework Tax: Disparities Between Inference Efficiency in Research and Deployment”. In: *ArXiv* (2023). URL: `https://arxiv.org/abs/2302.06117`.

[62] Mark Fiala. “ARTag, a fiducial marker system using digital techniques”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05).* Vol. 2. IEEE. 2005, pp. 590–596.

[63] Dhiraj Gandhi, Abhinav Gupta, and Lerrel Pinto. “Swoosh! rattle! thump!–actions that sound”. In: *arXiv preprint arXiv:2007.01851* (2020).

[64] Jensen Gao et al. “Physically Grounded Vision-Language Models for Robotic Manipulation”. In: *arxiv.* 2023.

[65] Ruohan Gao et al. “Sonicverse: A Multisensory Simulation Platform for Training Household Agents that See and Hear”. In: *ICRA.* 2023.

[66] Caelan Reed Garrett et al. “Integrated task and motion planning”. In: *arXiv preprint arXiv:2010.01083* (2020).

[67] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).

[68] Rohan Anil Google et al. *PaLM 2 Technical Report.* 2023.

[69] Google DeepMind. *Gemini 2: Flash.* Whitepaper / Technical Overview. 2025.

[70] Ankit Goyal et al. “Ifor: Iterative flow minimization for robotic object rearrangement”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2022, pp. 14787–14797.

[71] Yash Goyal et al. “Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2017, pp. 6904–6913.

[72] Oliver Groth et al. "Goal-Conditioned End-to-End Visuomotor Control for Versatile Skill Primitives". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2021, pp. 1319–1325.

[73] Pierre-Louis Guhur et al. "Instruction-driven history-aware policies for robotic manipulations". In: *arXiv preprint arXiv:2209.04899* (2022).

[74] Ankur Handa et al. "DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality". In: *arXiv preprint arXiv:2210.13702* (2022).

[75] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[76] Alexander Herzog* et al. "Deep RL at Scale: Sorting Waste in Office Buildings with a Fleet of Mobile Manipulators". In: *Robotics: Science and Systems (RSS)*. 2023.

[77] Edward Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021.

[78] Yafei Hu, Quanting Xie, Vidhi Jain, et al. "Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis". In: (Dec. 2023). arXiv: `2312.08782 [cs.RO]`.

[79] Chenguang Huang et al. "Audio Visual Language Maps for Robot Navigation". In: *Proceedings of the International Symposium on Experimental Robotics (ISER)*. Chiang Mai, Thailand, 2023.

[80] Chenguang Huang et al. "Visual Language Maps for Robot Navigation". In: *arXiv preprint arXiv:2210.05714* (2022).

[81] Wenlong Huang et al. "Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents". In: *arXiv preprint arXiv:2201.07207* (2022).

[82] Peter J. Huber. "Robust Estimation of a Location Parameter". In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73 –101. DOI: `10.1214/aoms/1177703732`. URL: `https://doi.org/10.1214/aoms/1177703732`.

[83] Andrew Hundt et al. """Good robot!": Efficient reinforcement learning for multistep visual tasks with sim to real transfer". In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6724–6731.

[84] Aaron Hurst et al. "Gpt-4o system card". In: *arXiv preprint arXiv:2410.21276* (2024).

[85] iRobot Home Support. *Article 32709 from iRobot Home Support*. `https://homesupport.irobot.com/s/article/32709`. 2024.

[86] Andrew Jaegle et al. "Perceiver IO: A General Architecture for Structured Inputs & Outputs". In: *International Conference on Learning Representations*. 2022. URL: `https://openreview.net/forum?id=fILj7WpI-g`.

[87] Vidhi Jain, Rishi Veerapaneni, and Yonatan Bisk. "ANAVI: Audio Noise Awareness using Visuals of Indoor environments for NAVIgation". In: *8th Annual Conference on Robot Learning* (2024).

[88] Vidhi Jain et al. *ActVQ-Arena: A Platform for Active Visual Querying in Embodied Environments.* Manuscript in preparation. Under submission. 2025.

[89] Vidhi Jain et al. "Learning embeddings that capture spatial semantics for indoor navigation". In: *arXiv preprint arXiv:2108.00159* (2021).

[90] Vidhi Jain et al. "Predicting human strategies in simulated search and rescue task". In: *arXiv preprint arXiv:2011.07656* (2020).

[91] Vidhi Jain et al. "Transformers Are Adaptable Task Planners". In: *6th Annual Conference on Robot Learning.* 2022. URL: `https://openreview.net/forum?id=Eal_1L08v_l`.

[92] Vidhi Jain et al. "Transformers Are Adaptable Task Planners". In: *Proceedings of The 6th Conference on Robot Learning.* Ed. by Karen Liu, Dana Kulic, and Jeff Ichnowski. Vol. 205. Proceedings of Machine Learning Research. PMLR, 2023, pp. 1011–1037. URL: `https://proceedings.mlr.press/v205/jain23a.html`.

[93] Vidhi Jain et al. "Vid2Robot: End-to-end Video-conditioned Policy Learning with Cross-Attention Transformers". In: *Robotics Science and Systems.* 2024.

[94] Stephen James and Andrew J Davison. "Q-attention: Enabling efficient learning for vision-based robotic manipulation". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1612–1619.

[95] Stephen James et al. "RLBench: The Robot Learning Benchmark and Learning Environment". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3019–3026.

[96] Eric Jang et al. "BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning". In: *Proceedings of the 5th Conference on Robot Learning.* 2022, pp. 991–1002.

[97] Michael Janner, Qiyang Li, and Sergey Levine. "Offline Reinforcement Learning as One Big Sequence Modeling Problem". In: *Advances in Neural Information Processing Systems.* 2021.

[98] Yunfan Jiang et al. "VIMA: General Robot Manipulation with Multimodal Prompts". In: *Fortieth International Conference on Machine Learning.* 2023.

[99] Carlos E. Jimenez et al. "SWE-bench: Can Language Models Resolve Real-World GitHub Issues?" In: *International Conference on Learning Representations.* 2024.

[100] Dmitry Kalashnikov et al. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation". In: *CoRL.* 2018.

[101] Ivan Kapelyukh and Edward Johns. “My House, My Rules: Learning tidying preferences with graph neural networks”. In: *Conference on Robot Learning (CoRL).* 2021.

[102] Russell Kaplan, Christopher Sauer, and Alexander Sosa. “Beating atari with natural language guided reinforcement learning”. In: *arXiv preprint arXiv:1704.05539* (2017).

[103] Chucri A. Kardous and Peter B. Shaw. “Evaluation of smartphone sound measurement applicationsa)”. In: *The Journal of the Acoustical Society of America* 135.4 (Mar. 2014), EL186–EL192. ISSN: 0001-4966. DOI: 10.1121/1.4865269. eprint: https://pubs.aip.org/asa/jasa/article-pdf/135/4/EL186/15312422/el186_1_online.pdf. URL: https://doi.org/10.1121/1.4865269.

[104] Charles C Kemp et al. “The design of stretch: A compact, lightweight mobile manipulator for indoor human environments”. In: *2022 International Conference on Robotics and Automation (ICRA).* IEEE. 2022, pp. 3150–3157.

[105] Justin Kerr et al. “Eye, Robot: Learning to Look to Act with a BC-RL Perception-Action Loop”. In: 2025. arXiv: 2506.10968 [cs.RO]. URL: http://arxiv.org/abs/2506.10968.

[106] Leonid Keselman et al. “Intel realsense stereoscopic depth cameras”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops.* 2017, pp. 1–10.

[107] Moo Jin Kim et al. “OpenVLA: An Open-Source Vision-Language-Action Model”. In: *8th Annual Conference on Robot Learning.* 2024. URL: https://openreview.net/forum?id=ZMnD6QZAE6.

[108] W Bradley Knox et al. “Models of human preference for learning reward functions”. In: *arXiv preprint arXiv:2206.02231* (2022).

[109] Michael Krainin, Brian Curless, and Dieter Fox. “Autonomous generation of complete 3D object models using next best view manipulation planning”. In: *2011 IEEE International Conference on Robotics and Automation.* 2011, pp. 5031–5037. DOI: 10.1109/ICRA.2011.5980429.

[110] Jacob Krantz et al. “Navigating to Objects Specified by Images”. In: (Apr. 2023). arXiv: 2304.01192 [cs.CV].

[111] Asbjørn Krokstad, S. Strøm, and Svein Sorsdal. “Calculating the acoustical room response by the use of a ray tracing technique”. In: *Journal of Sound and Vibration* 8 (1968), pp. 118–125. URL: https://api.semanticscholar.org/CorpusID:89608438.

[112] Sateesh Kumar et al. *Graph Inverse Reinforcement Learning from Diverse Videos.* 2022. arXiv: 2207.14299 [cs.LG].

[113] Kimin Lee et al. “B-Pref: Benchmarking Preference-Based Reinforcement Learning”. In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*. 2021. URL: `https://openreview.net/forum?id=ps95-mkHF_`.

[114] Sergey Levine et al. “Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection”. In: *arXiv:1603.02199*. 2016.

[115] Mike Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *ACL*. 2020.

[116] Jacky Liang et al. “Code as policies: Language model programs for embodied control”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 9493–9500.

[117] Xinran Liang et al. “Reward Uncertainty for Exploration in Preference-based Reinforcement Learning”. In: *arXiv preprint arXiv:2205.12401* (2022).

[118] Kevin Lin et al. “Text2motion: From natural language instructions to feasible plans”. In: *arXiv preprint arXiv:2303.12153* (2023).

[119] Weisi Lin and Gheorghita Ghinea. “Progress and opportunities in modelling just-noticeable difference (JND) for multimedia”. In: *IEEE Transactions on Multimedia* 24 (2021), pp. 3706–3721.

[120] Yixin Lin et al. “Efficient and interpretable robot manipulation with graph neural networks”. In: *IEEE Robotics and Automation Letters* (2022).

[121] Yixin Lin et al. *Polymetis*. `https://facebookresearch.github.io/fairo/polymetis/`. 2021.

[122] Pengfei Liu et al. “Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing”. In: *ACM Comput. Surv.* 55.9 (Jan. 2023). ISSN: 0360-0300. DOI: `10.1145/3560815`. URL: `https://doi.org/10.1145/3560815`.

[123] Weiyu Liu et al. “StructDiffusion: Object-Centric Diffusion for Semantic Rearrangement of Novel Objects”. In: *arXiv preprint arXiv:2211.04604* (2022).

[124] Weiyu Liu et al. “Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 6322–6329.

[125] Xiao Liu et al. “P-Tuning: Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 61–68. DOI: `10.18653/v1/2022.acl-short.8`. URL: `https://aclanthology.org/2022.acl-short.8/`.

[126] Yuxuan Liu et al. "Imitation from Observation: Learning to Imitate Behaviors from Raw Video via Context Translation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, May 2018, pp. 1118–1125.

[127] Francesco Locatello et al. "Object-centric learning with slot attention". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11525–11538.

[128] Yecheng Jason Ma et al. "Eureka: Human-Level Reward Design via Coding Large Language Models". In: *2nd Workshop on Language and Robot Learning: Language as Grounding.* 2023. URL: `https://openreview.net/forum?id=WbFBZ3bQUs`.

[129] Yecheng Jason Ma et al. "VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training". In: (Sept. 2022). arXiv: `2210.00030 [cs.RO]`.

[130] Jeffrey Mahler et al. "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics". In: *arXiv preprint arXiv:1703.09312* (2017).

[131] Arjun Majumdar et al. "OpenEQA: Embodied Question Answering in the Era of Foundation Models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2024, pp. 16488–16498. DOI: `10.1109/CVPR52733.2024.01560`.

[132] Sagnik Majumder, Ziad Al-Halah, and Kristen Grauman. "Move2Hear: Active Audio-Visual Source Separation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).* Oct. 2021, pp. 275–285.

[133] Viktor Makoviychuk et al. *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning.* 2021. arXiv: `2108.10470 [cs.RO]`.

[134] Zhao Mandi et al. "Towards more generalizable one-shot visual imitation learning". In: *2022 International Conference on Robotics and Automation (ICRA).* IEEE. 2022, pp. 2434–2444.

[135] Daniel Maturana et al. "Real-time semantic mapping for autonomous off-road navigation". In: *Field and Service Robotics.* Springer. 2018, pp. 335–350.

[136] Oier Mees, Jessica Borja-Diaz, and Wolfram Burgard. "Grounding Language with Visual Affordances over Unstructured Data". In: (2023), pp. 11576–11582. DOI: `10.1109/ICRA48891.2023.10160396`.

[137] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. "Structured World Models from Human Videos". In: 2023.

[138] Antoine Miech et al. "HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips". In: *ICCV.* 2019.

[139] Ben Mildenhall et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *ECCV.* 2020.

[140] So Yeon Min et al. “Film: Following instructions in language with modular methods”. In: *arXiv preprint arXiv:2110.07342* (2021).

[141] Ishan Misra, Rohit Girdhar, and Armand Joulin. “An end-to-end transformer model for 3d object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2021, pp. 2906–2917.

[142] Himangi Mittal et al. “Learning state-aware visual representations from audible interactions”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 23765–23779.

[143] Mayank Mittal et al. *Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments.* 2023. DOI: 10.1109/LRA.2023.3270034.

[144] Kaichun Mo et al. “Where2act: From pixels to actions for articulated 3d objects”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2021, pp. 6813–6823.

[145] Hans Moravec. *Mind children: The future of robot and human intelligence.* Harvard University Press, 1988.

[146] Pedro Morgado, Yi Li, and Nuno Nvasconcelos. “Learning Representations from Audio-Visual Spatial Alignment”. In: *Advances in Neural Information Processing Systems.* Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 4733–4744. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/328e5d4c166bb340b314d457a208dc83-Paper.pdf.

[147] Adithyavairavan Murali et al. “6-dof grasping for target-driven object manipulation in clutter”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA).* IEEE. 2020, pp. 6232–6238.

[148] Ashvin Nair et al. “Visual Reinforcement Learning with Imagined Goals”. In: *NeurIPS.* July 2018.

[149] Suraj Nair et al. “R3m: A universal visual representation for robot manipulation”. In: *arXiv preprint arXiv:2203.12601* (2022).

[150] Open X-Embodiment Collaboration et al. “Open X-Embodiment: Robotic Learning Datasets and RT-X Models”. In: *arXiv 2310.08864* (2023).

[151] OpenAI. *GPT-4 Technical Report.* Tech. rep. OpenAI, 2024.

[152] OpenAI. *GPT-4V(ision): Grounding Large Vision–Language Models in Robotic Control.* Technical Report. OpenAI, 2024.

[153] Andrew Owens and Alexei A Efros. “Audio-visual scene analysis with self-supervised multisensory features”. In: *Proceedings of the European conference on computer vision (ECCV).* 2018, pp. 631–648.

[154] Andrew Owens et al. “Visually indicated sounds”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 2405–2413.

[155] Priyam Parashar et al. “SLAP: Spatial-Language Attention Policies”. In: *7th Annual Conference on Robot Learning*. 2023.

[156] Priyam Parashar et al. “Spatial-Language Attention Policies for Efficient Robot Learning”. In: *Conference on Robot Learning*. 2023. URL: `https://arxiv.org/abs/2304.11235`.

[157] Chris Paxton et al. “Predicting stable configurations for semantic placement of novel objects”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 806–815.

[158] Chris Paxton et al. “Prospection: Interpretable plans from language by predicting the future”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 6942–6948.

[159] Chris Paxton et al. “Representing robot task plans as robust logical-dynamical systems”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 5588–5595.

[160] Xue Bin Peng et al. “SFV: reinforcement learning of physical skills from videos”. In: *ACM Trans. Graph.* 37.6 (Dec. 2018), pp. 1–14.

[161] Ethan Perez et al. “Film: Visual reasoning with a general conditioning layer”. In: *Proceedings of the AAAI conference on artificial intelligence*. 2018.

[162] Karl Pertsch, Youngwoon Lee, and Joseph J Lim. “Accelerating reinforcement learning with learned skill priors”. In: *arXiv preprint arXiv:2010.11944* (2020).

[163] Vladimír Petrík et al. “Learning Object Manipulation Skills via Approximate State Estimation from Real Videos”. In: *Proceedings of the 2020 Conference on Robot Learning*. Ed. by Jens Kober, Fabio Ramos, and Claire Tomlin. Vol. 155. Proceedings of Machine Learning Research. PMLR, 2021, pp. 296–312.

[164] Antonio Pico et al. “How do I sound like? forward models for robot ego-noise prediction”. In: Sept. 2016, pp. 246–251. DOI: `10.1109/DEVLRN.2016.7846826`.

[165] Sören Pirk et al. “Online Object Representations with Contrastive Learning”. In: (2019).

[166] Sam Powers, Abhinav Gupta, and Chris Paxton. *Evaluating Continual Learning on a Home Robot*. 2023. arXiv: `2306.02413 [cs.RO]`.

[167] Xavier Puig et al. *Habitat 3.0: A Co-Habitat for Humans, Avatars and Robots*. 2023. arXiv: `2310.13724 [cs.HC]`.

[168] Wilbert Pumacay et al. “The COLOSSEUM: A Benchmark for Evaluating Generalization for Robotic Manipulation”. In: *Robotics: Science and Systems*. Delft, Netherlands, 2024. DOI: `10.15607/RSS.2024.XX.133`.

[169] Aaron L Putterman et al. *Pretraining for Language Conditioned Imitation with Transformers*. 2022. URL: `https://openreview.net/forum?id=eCPCn25gat`.

[170] Carl Qi et al. “Learning Generalizable Tool-use Skills through Trajectory Generation”. In: *ArXiv* abs/2310.00156 (2023). URL: `https://api.semanticscholar.org/CorpusID:263334247`.

[171] Charles Ruizhongtai Qi et al. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Advances in neural information processing systems* 30 (2017).

[172] Zengyi Qin et al. *KETO: Learning Keypoint Representations for Tool Manipulation.* 2019. arXiv: `1910.11977 [cs.RO]`.

[173] Ahmed H Qureshi et al. “Nerp: Neural rearrangement planning for unknown objects”. In: *Robotics: Science and Systems (RSS).* 2021.

[174] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International Conference on Machine Learning.* PMLR. 2021, pp. 8748–8763.

[175] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *ArXiv* abs/1910.10683 (2020).

[176] Antonin Raffin et al. “Stable-Baselines3: Reliable Reinforcement Learning Implementations”. In: *Journal of Machine Learning Research* (2021).

[177] Ori Ram et al. “In-Context Retrieval-Augmented Language Models”. In: vol. 11. Cambridge, MA: MIT Press, 2023, pp. 1316–1331. DOI: `10.1162/tacl_a_00605`. URL: `https://aclanthology.org/2023.tacl-1.75/`.

[178] Aditya Ramesh et al. “Zero-shot text-to-image generation”. In: *International Conference on Machine Learning.* PMLR. 2021, pp. 8821–8831.

[179] Victor Sanh et al. “Multitask prompted training enables zero-shot task generalization”. In: *arXiv preprint arXiv:2110.08207* (2021).

[180] Manolis Savva et al. “Habitat: A platform for embodied ai research”. In: *Proceedings of the IEEE/CVF international conference on computer vision.* 2019, pp. 9339–9347.

[181] Timo Schick and Hinrich Schütze. “It’s not just size that matters: Small language models are also few-shot learners”. In: *arXiv preprint arXiv:2009.07118* (2020).

[182] Karl Schmeckpeper et al. “Reinforcement Learning with Videos: Combining Offline Observations with Interaction”. In: (Nov. 2020). arXiv: `2011.06507 [cs.LG]`.

[183] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[184] Nur Muhammad Mahi Shafiullah et al. “CLIP-Fields: Weakly Supervised Semantic Fields for Robotic Memory”. In: *arXiv preprint arXiv:2210.05663* (2022).

[185] Dhruv Shah et al. “ViNT: A Foundation Model for Visual Navigation”. In: *arxiv preprint arXiv:2306.14846.* 2023.

[186] Rutav Shah, Roberto Martín-Martín, and Yuke Zhu. "MUTEX: Learning Unified Policies from Multimodal Task Specifications". In: *7th Annual Conference on Robot Learning*. 2023.

[187] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. "Third-Person Visual Imitation Learning via Decoupled Hierarchical Controller". In: *Adv. Neural Inf. Process. Syst.* 32 (2019).

[188] Lucy Xiaoyang Shi et al. "Yell at your robot: Improving on-the-fly from language corrections". In: *arXiv preprint arXiv:2403.12910* (2024).

[189] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "Cliport: What and where pathways for robotic manipulation". In: *Conference on Robot Learning*. PMLR. 2022, pp. 894–906.

[190] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "Perceiver-actor: A multi-task transformer for robotic manipulation". In: *arXiv preprint arXiv:2209.05451* (2022).

[191] Mohit Shridhar et al. "ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10740–10749.

[192] Ishika Singh et al. "Progprompt: Generating situated robot task plans using large language models". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 11523–11530.

[193] Ishika Singh et al. "ProgPrompt: Generating Situated Robot Task Plans using Large Language Models". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023.

[194] Laura Smith et al. "AVID: Learning Multi-Stage Tasks via Pixel-Level Translation of Human Videos". In: *arXiv* (Dec. 2019). arXiv: `1912.04443 [cs.RO]`.

[195] Kaitao Song et al. "MASS: Masked Sequence to Sequence Pre-training for Language Generation". In: *ICML*. 2019.

[196] Jonathan Spencer et al. "Feedback in imitation learning: The three regimes of covariate shift". In: *arXiv preprint arXiv:2102.02872* (2021).

[197] Austin Stone et al. "Open-World Object Manipulation using Pre-trained Vision-Language Models". In: Mar. 2023.

[198] Julian Straub et al. "The Replica Dataset: A Digital Replica of Indoor Spaces". In: *arXiv preprint arXiv:1906.05797* (2019).

[199] Jiankai Sun et al. "PlaTe: Visually-grounded planning with transformers in procedural tasks". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4924–4930.

[200] Pei Sun et al. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[201] Martin Sundermeyer et al. "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 13438–13444.

[202] Andrew Szot et al. "Habitat 2.0: Training Home Assistants to Rearrange their Habitat". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[203] Yujin Tang et al. *SayTap: Language to Quadrupedal Locomotion*. 2023. arXiv: `2306.07580 [cs.RO]`.

[204] Abitha Thankaraj and Lerrel Pinto. "That Sounds Right: Auditory Self-Supervision for Dynamic Robot Manipulation". In: *arXiv preprint arXiv:2210.01116* (2022).

[205] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pp. 5026–5033.

[206] Hugo Touvron et al. "LLaMA: Open and Efficient Foundation Language Models". In: *arXiv preprint arXiv:2302.13971* (2023).

[207] Dylan Turpin et al. "GIFT: Generalizable Interaction-aware Functional Tool Affordances without Labels". In: *ArXiv* abs/2106.14973 (2021). URL: `https://api.semanticscholar.org/CorpusID:235650930`.

[208] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[209] Sai H Vemprala et al. "Chatgpt for robotics: Design principles and model abilities". In: *IEEE Access* (2023).

[210] He Wang et al. "Normalized object coordinate space for category-level 6d object pose and size estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2642–2651.

[211] Xiaofei Wang et al. "Skill preferences: Learning to extract and execute robotic skills from human feedback". In: *Conference on Robot Learning*. PMLR. 2022, pp. 1259–1268.

[212] Jerry Wei et al. *Larger language models do in-context learning differently*. arXiv:2303.03846 [cs]. Mar. 2023. URL: `http://arxiv.org/abs/2303.03846` (visited on 05/25/2023).

[213] Ruihai Wu et al. "VAT-Mart: Learning Visual Action Trajectory Proposals for Manipulating 3D ARTiculated Objects". In: *arXiv preprint arXiv:2106.14440* (2021).

[214] Yu Xiang et al. "Learning rgb-d feature embeddings for unseen object instance segmentation". In: *arXiv preprint arXiv:2007.15157* (2020).

[215] Yu Xiang et al. "Learning rgb-d feature embeddings for unseen object instance segmentation". In: *Conference on Robot Learning*. PMLR. 2021, pp. 461–470.

[216] Tete Xiao et al. “Masked Visual Pre-training for Motor Control”. In: *arXiv preprint arXiv:2203.06173* (2023).

[217] Christopher Xie et al. “Unseen object instance segmentation for robotic environments”. In: *IEEE Transactions on Robotics* 37.5 (2021), pp. 1343–1359.

[218] Tianbao Xie et al. *Text2Reward: Automated Dense Reward Function Generation for Reinforcement Learning*. 2023. arXiv: `2309.11489 [cs.LG]`.

[219] Yiming Xie et al. *SV4D: Dynamic 3D Content Generation with Multi-Frame and Multi-View Consistency*. 2024. arXiv: `2407.17470 [cs.CV]`. URL: `https://arxiv.org/abs/2407.17470`.

[220] Haoyu Xiong et al. “Vision in Action: Learning Active Perception from Human Demonstrations”. In: *arXiv preprint arXiv:2506.15666* (2025).

[221] Mengda Xu et al. “XSkill: Cross Embodiment Skill Discovery”. In: *7th Annual Conference on Robot Learning*. 2023. URL: `https://openreview.net/forum?id=8L6pHd9aS6w`.

[222] Mengdi Xu et al. “Prompting Decision Transformer for Few-Shot Policy Generalization”. In: *Thirty-ninth International Conference on Machine Learning*. 2022.

[223] Zhenjia Xu, Zhanpeng He, and Shuran Song. “UMPNet: Universal manipulation policy network for articulated objects”. In: *arXiv preprint arXiv:2109.05668* (2021).

[224] Sriram Yenamandra et al. “HomeRobot: Open-Vocabulary Mobile Manipulation”. In: *7th Annual Conference on Robot Learning*. 2023. URL: `https://openreview.net/forum?id=b-cto-fetlz`.

[225] Sriram Yenamandra et al. “The HomeRobot Open Vocab Mobile Manipulation Challenge”. In: *Thirty-seventh Conference on Neural Information Processing Systems: Competition Track*. 2023. URL: `https://aihabitat.org/challenge/2023_homerobot_ovmm/`.

[226] Youngrock Yoon, Guilherme N DeSouza, and Avinash C Kak. “Real-time tracking and pose estimation for industrial objects using geometric features”. In: *2003 IEEE International conference on robotics and automation (cat. no. 03CH37422)*. Vol. 3. IEEE. 2003, pp. 3473–3478.

[227] Jiahui Yu et al. “Coca: Contrastive captioners are image-text foundation models”. In: *arXiv preprint arXiv:2205. 01917* ().

[228] Wenhao Yu et al. “Language to Rewards for Robotic Skill Synthesis”. In: *Arxiv preprint arXiv:2306.08647* (2023).

[229] Wentao Yuan et al. “Sornet: Spatial object-centric representations for sequential manipulation”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 148–157.

[230] Kevin Zakka et al. “Xirl: Cross-embodiment inverse reinforcement learning”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 537–546.

[231] Andy Zeng et al. “Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge”. In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 1386–1383.

[232] Andy Zeng et al. “Transporter networks: Rearranging the visual world for robotic manipulation”. In: *Conference on Robot Learning*. PMLRG. 2020, pp. 726–747.

[233] Xiaohua Zhai et al. “Sigmoid Loss for Language Image Pre-Training”. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2023, pp. 11941–11952. DOI: `10.1109/ICCV51070.2023.01100`. URL: `https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.01100`.

[234] Qiwu Zhang. “The Application of Audio Control in Social Robotics”. In: *Proceedings of the 2022 4th International Conference on Robotics, Intelligent Control and Artificial Intelligence*. RICAI ’22. Dongguan, China: Association for Computing Machinery, 2023, 963–966. ISBN: 9781450398343. DOI: `10.1145/3584376.3584548`. URL: `https://doi.org/10.1145/3584376.3584548`.

[235] Tony Z Zhao et al. “Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware”. In: *Robotics: Science and Systems*. 2023.

[236] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A modern library for 3D data processing”. In: *arXiv preprint arXiv:1801.09847* (2018).

[237] Shuyan Zhou et al. “WebArena: A Realistic Web Environment for Building Autonomous Agents”. In: *International Conference on Learning Representations*. 2024.

[238] Xingyi Zhou et al. “Detecting Twenty-thousand Classes using Image-level Supervision”. In: *ECCV*. 2022.

[239] Hao Zhu et al. “EXCALIBUR: Encouraging and Evaluating Embodied Exploration”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. –.

[240] Menglong Zhu et al. “Single image 3D object detection and pose estimation for grasping”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 3936–3943.

[241] Brianna Zitkovich et al. “RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control”. In: *Proceedings of The 7th Conference on Robot Learning*. Ed. by Jie Tan, Marc Toussaint, and Kourosh Darvish. Vol. 229. Proceedings of Machine Learning Research. PMLR, 2023, pp. 2165–2183. URL: `https://proceedings.mlr.press/v229/zitkovich23a.html`.