

# Vocal Sandbox – Supplementary Material

## Overview

In the appendices below, we provide additional details around the implementation of the various components of *both* Vocal Sandbox systems (i.e., for both the collaborative gift-bag assembly and the LEGO stop-motion settings), including details around the rest of the system architecture (e.g., speech recognition, text-to-speech). We then extend our discussion from the main body of paper, building on additional opportunities provided by our framework for future research.

A large portion of our paper is grounded in user studies and extended interactions; all those videos are best viewed in (1) the attached supplementary video, and (2) on our anonymous project website: <https://vsandbox-corl-2024.github.io>.

Furthermore, our work involves prompting GPT-3.5 Turbo with function calling [v11-06; 1, 2] through the public OpenAI API; we provide these prompts directly (as code) on our website as well: <https://vsandbox-corl-2024.github.io/#language-prompts>

An overview of each appendix can be found below:

---

### Appx. A – *Motivating Questions*

We index a list of “motivating” questions that may arise from reading the main text and that we expand on further here (e.g., “why adopt a modular vs. an end-to-end approach?”).

Our answers here are *direct*, linking to concrete sections further on in the appendices.

### Appx. B – *Implementing Vocal Sandbox*

We provide complete implementation details for both Vocal Sandbox systems we present in the main body of the paper; we additionally include details around the rest of the system architecture (e.g., speech-to-text, robot control, etc.):

#### §B.1 – *System Architecture*

Additional system details around implementing real-time speech recognition and text-to-speech, with latency and pricing statistics.

#### §B.2 – *Language Models for Task Planning and Skill Induction*

GPT-3.5 Turbo prompting details, generation hyperparameters, and latency statistics.

#### §B.3 – *Visual Keypoint-Conditioned Policy Implementation*

Additional implementation details and static evaluations for our visual keypoint-conditioned policy (for robust object manipulation).

#### §B.4 – *Learning Discrete Dynamic Movement Primitives from Demonstration*

Formalism of discrete dynamic movement primitives (DMP) and learning algorithm; additional details about the DMP features we employ (re-timing and goal editing).

#### §B.5 – *Robot Platform & Low-Level Controller Implementation*

Robot platform and controller implementation; ensuring compliant control and safety.

### Appx. C – *Extended Discussion & Future Work*

We provide an extended discussion on themes from the paper, from the importance of modularity and transparency in developing Vocal Sandbox, the broader impact of Vocal Sandbox in the context of human-robot collaboration, and directions for future work.

## A Motivating Questions

**Q1.** *If I wanted to implement a Vocal Sandbox from scratch, what other components would I need? How do the current experiments handle real-time speech-to-text and text-to-speech? What about pricing – what was the cost of running the Gift-Bag Assembly User Study ( $N = 8$ )?*

Beyond the language model task planner that uses GPT-3.5 Turbo with function calling [v11-06; 1, 2], and the (lightweight) learned skill policy, we use a combination of Whisper [3] for real-time speech recognition (mapping user utterances to text), and the OpenAI text-to-speech (TTS) API [4] for vocalizing confirmation prompts and querying users for teaching feedback. All models, cameras, and API calls are run through a single laptop equipped with an NVIDIA RTX 4080 GPU (12 GB).

For the gift-bag assembly user study ( $N = 8$ ), the total cost of all external APIs (Whisper, OpenAI TTS, GPT-3.5 Turbo) amounted to  $\$0.47 + \$0.08 + \$1.24 = \$1.79$ . For the entirety of the project, GPT-3.5 API spend was  $\$5.79$ , with  $\sim \$4.00$  spent on Whisper and TTS ( $< \$10.00$  total).

**Q2.** *Given the use of powerful closed-source foundation models such as GPT-3.5/GPT-4, why adopt a modular approach for implementing the visual keypoints (and similarly dynamic movement primitives for learning policies)? Why not adopt an end-to-end approach building on top of GPT-4 with Vision, or existing pretrained multitask policies?*

We choose to adopt a modular approach in this work for two reasons. First, existing end-to-end models are still limited when it comes to fine-grained perception and grounding; we quantify this more explicitly through head-to-head static evaluations of our keypoint model vs. pretrained models such as OWLv2 [5, 6] in §B.3. Second, we argue that modularity allows users to systematically *isolate failures* and address them via multimodal feedback, at the right level of abstraction. We expand on this further in §C.1.

**Q3.** *How does Vocal Sandbox fit into the context of prior human-robot interaction works? What are the new capabilities Vocal Sandbox is bringing to the table?*

While the main body of the paper situates our framework against prior work in task planning and skill learning from different modalities, Vocal Sandbox builds on a rich history of work that develops systems for different modes of human-robot interaction. We provide an extended treatment of related work, as well as directions for future work in §C.2.

## B Implementing Vocal Sandbox

Implementing a system in the Vocal Sandbox framework requires not only the learned components for language-based task planning and low-level skill execution, but broader support for interfacing with users via automated speech-to-text, text-to-speech for vocalizing failures or confirmation prompts, as well as a screen for visualizing the graphical user interface. We describe these additional components, as well as provide more detail around the implementation of the learned components of the systems instantiated in our paper over the following sections.

### B.1 System Architecture

For robust and cheap automated speech recognition (mapping user utterances to text), we use Whisper [1, 3], accessed via the OpenAI API endpoint. Whisper is a state-of-the-art model meant for natural speech transcription, and we find that the latency for a given transcription request ( $< 0.5$ s round-trip) is more than enough for all of our use-cases. API pricing is also affordable, with the Whisper API (through OpenAI) charging  $\$0.006$  / minute of transcription (less than  $\$0.50$  to run our entire gift-bag assembly user study). Note that we implement speech-to-text via explicit “push-to-talk” interface, rather than an alternative “always-listening” approach; we find that this not only allows us to keep cost and word-error rate down, but improves user experience. By gating the listening and stop-listening features with explicit audio cues, users are more aware of what the system is doing, and can more quickly localize any failures stemming from malformed speech transcriptions.

In addition to automated speech-to-text, we adopt off-the-shelf solutions for real-time text-to-speech; this is mostly for implementing confirmation prompts (“does this plan look ok to you?”) and for vocalizing the system state, but also includes an adaptive component when probing users to teach new visual concepts or behaviors (“I’m sorry, I’m not sure what the ‘jelly-candy thing’ looks like, could you teach me?”). For these queries, we use the OpenAI TTS API [4] with a similarly affordable pricing scheme of \$15.00 per 1M characters (or approximately 200K words); to run our gift-bag assembly study, this cost fewer than \$0.08. For hardware (for both speech recognition and text-to-speech), we use a standard USB speaker-microphone (the Anker PowerConf S3).

To visualize the graphical user interface to users, we use a standard external monitor (27 inches), placed outside of the robot’s workspace. We drive the GUI, all API calls (speech recognition, text-to-speech, and language modeling via GPT-3.5 Turbo), ZED 2 camera, and all our learned models – including our visual keypoint-conditioned policy, FastSAM [7], and XMem [8] – from a single Alienware m16 laptop with an NVIDIA RTX 4080 GPU with 12 GB of VRAM; this laptop was purchased matching the DROID platform specification [9].

**Modifications for Gift-Bag Assembly User Study.** For the gift-bag assembly user study ( $N = 8$ ) we implement the “push-to-talk” speech recognition interface with physical buttons placed on the table; users are provided two buttons – one for “talking” and one for “cancelling” the prior actions (which serves a dual function as a secondary, software-based emergency stop when the robot is moving). These buttons are placed on the side of the user’s non-dominant hand, always within reach.

**Modifications for LEGO Stop-Motion Animation.** For the LEGO stop-motion animation study, we use the same components as above, with two additions. As the expert user is directing and framing individual camera shots during the course of the collaboration, they add an additional laptop (a Macbook, running Stop Motion Studio) to the workspace (disconnected from the rest of the system). As the user requires both hands free for this study (for articulating LEGO minifigures and structures, or editing the clip on their laptop), we replace the tabletop “push-to-talk” buttons with a USB-connected foot pedal with two switches with the same recognition and cancel functionality.

## B.2 Language Models for Task Planning and Skill Induction

As described in the main body of the paper, we use GPT-3.5 Turbo with function calling [v11-06; 1, 2] as our base language model for the task planner. This was the latest, most affordable, and highest latency language model at the time we began this work (prior to the release of GPT-4 and GPT-4o), with a response time between 1-3s on average, and a cost of \$2.00 / 1M tokens; for this work, the total cost we spent on GPT 3.5 API calls (including development) was \$5.79, with the gift-bag assembly user study itself only amounting for \$1.24 of the total spend.

We use the GPT-3.5 function calling capabilities throughout our work, requiring formatting our API specification following a custom JSON schema set by OpenAI; we provide these function calling prompts (and all GPT-3.5 prompts for our work) on our supplementary website for easy visualization: <https://vsandbox-cori-2024.github.io/#language-prompts>. All language model outputs were generated with low-temperature sampling (0.2).

## B.3 Visual Keypoint-Conditioned Policy Implementation

As described in the main body of the paper, we use three components to implement Vocal Sandbox’s low-level visual skills: (1) a learned language-conditioned keypoints model, (2) a pretrained mask propagation model [XMem; 8], and (3) a point-conditioned segmentation model [FastSAM; 7].

Our learned keypoint model predicts object centroids from language, enabling us to generalize across object instances where XMem struggles. Given an RGB image  $o_t \in \mathbb{R}^{H \times W \times 3}$  and natural language literal  $c_{\text{ref}}$  from the high-level language planner, it predicts a matrix of per-pixel scores  $\mathcal{H} \in [0, 1]^{H \times W}$ . We take the coordinate-wise argmax of  $\mathcal{H}$  as the predicted keypoint. We implement our model with a two-stream architecture following Shridhar et al. [11] that fuses pretrained CLIP [12] textual embeddings with a fully-convolutional architecture. We train this model on a small, cheap-to-collect dataset of 25 unique images each annotated with 3 keypoints (75 examples total). To fit our model, we create heatmaps from each ground-truth label, centering a 2D Gaussian around

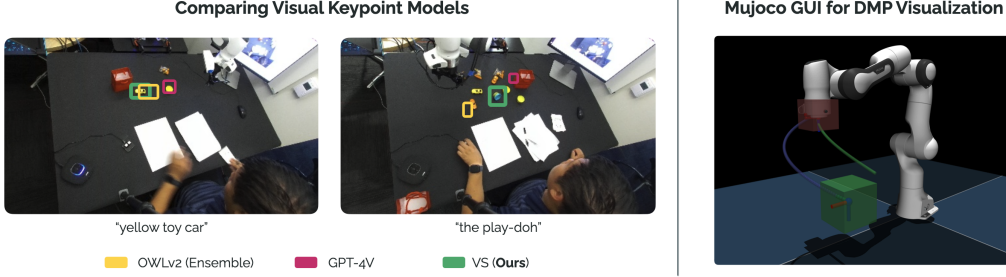


Figure 1: **Visual Keypoint Static Evaluation & DMP Visualizer.** We visualize visual keypoint predictions for object locations across a clean environment [Left] and a more difficult, cluttered environment [Middle]. We also highlight the trace generated by our Dynamic Movement Primitive (DMP) skills, rendered in a Mujoco [10] simulation environment to display the robot path before execution.

each keypoint with a fixed standard deviation of 6 pixels; we train our model by minimizing the binary cross-entropy between model predictions and these heatmaps, augmenting images with various label-preserving affine transformations (e.g., random crops, shears, rotations).

Our mask propagation model, XMem [8] tracks object segmentation masks from one image frame to the next; we provide a brief overview here. XMem is comprised of three convolutional networks (a query encoder  $e$ , a decoder  $d$ , and a value encoder  $v$ ) and three memory modules (a short-term sensory memory, a working memory, and a long-term memory). For a given image  $I_t$ , the query encoder outputs a query  $q = e(I_t)$  and performs attention-based memory reading from working and long-term memory stores to extract features  $F_{\text{ref}}$ , where  $c_{\text{ref}}$  is the language utterance (e.g., “candy”). The decoder  $d$  then takes as input  $q$ ,  $F$ , and  $h_{t-1}$  (the short-term sensory memory) to output a predicted mask  $M_t$ . Finally, the value encoder  $v(I_t, M_t)$  outputs new features to be added to the memory history  $h_t$ . The query encoder  $e$  and value encoder  $v$  are instantiated with ResNet-50 and ResNet-18 [13] respectively. The decoder  $d$  concatenates the short-term memory history  $h_{t-1}$  with the extracted features  $F$ , upsampling by a factor of 2x until reaching a stride of 4. While upsampling, the decoder fuses skip connections from the query encoder  $e$  at every level. The final feature map is passed through a  $3 \times 3$  convolution to output a single channel logit which is upsampled to the image size. See Cheng and Schwing [8] for additional details.

Finally, our point-conditioned segmentation model, FastSAM [7], is used to extract an object mask from a predicted keypoint. It has two components: a YOLOv8 [14] segmentation model  $s$  for all-instance segmentation, and a point prompt-guided selection for identifying the object mask in which the point lies. From a given predicted keypoint  $p$ , the segmentation model outputs the mask  $M$  from  $s$  that encompasses  $p$ . We refer to [7] for additional details. This predicted mask  $M$  is subsequently added to the XMem memory storage after being passed through the value encoder  $v$ .

Robot actions are coded as parameterized primitives (i.e., `pick_up` or `go_to`) that take object locations as input and output trajectories.

**Static Evaluations – Robust Object Grounding.** To highlight the need for a data-efficient, domain-specific vision system, we evaluate the performance of our vision module implementation (as described above) compared to existing closed-source foundation models such as OWLv2 and GPT-4V. To compare, we consider an (image, annotation) dataset of all visual queries from the  $N = 8$  user study, where the annotation is where the user confirmed or manually selected a correct object location. We report measures for accuracy and precision – keypoint mean squared error in pixel distance and success counts for predictions within a toy-car radius (14 pixels) from the annotation. For the Vocal Sandbox predicted mask, the centroid of the mask is used for these point-to-point calculations. We observe that while the mean squared error across all three methods are comparable, our Vocal Sandbox vision module greatly outperforms the foundation model baselines in the precision metric. This is because all the objects are clustered together on a table (Fig. 1) – randomly selecting between

these objects yields low MSE predictions, however a nearby prediction is not sufficient to identify and isolate the correct object for grasping.

	Keypoint MSE (px)	Precision
OWLv2 Ensemble (ViT-L/14)	$35.3 \pm 1.01$	$1.83 \pm 0.91$
GPT-4-Turbo (w/ Vision)	$36.39 \pm 1.73$	$15.94 \pm 2.55$
Vocal Sandbox (Ours)	$30.46 \pm 3.61$	$69.41 \pm 3.12$

#### B.4 Learning Discrete Dynamic Movement Primitives from Demonstration

For our LEGO stop-motion animation setting, we implement our low-level skill policy as a library of discrete Dynamic Movement Primitives [15, 16]. We adopt the traditional discrete DMP formulation from Ijspeert et al. [16], defining a second-order point dynamical system in terms of the system state  $y$ , a goal  $g$ , and phase variable  $x$  such that:

$$\tau \ddot{y} = \alpha_y (\gamma_y (g - y) - \dot{y}) + f(x, g); \quad \tau \ddot{x} = -\alpha_x x$$

where  $\alpha$  and  $\gamma$  define gain terms,  $\tau \in (0, 1]$  denotes a temporal scaling factor, and  $f(x, g)$  is the *learned forcing function* that drives a DMP to follow a specific trajectory to the goal  $g$ ;  $f(x, g)$  is implemented as a learned linear combination of  $J$  radial basis functions and the phase variable  $x$  such that:

$$f(x, g) = \frac{\sum_{j=1}^J \psi_j w_j}{\sum_{j=1}^J \psi_j} x (g - y_0); \quad \psi_j = \exp(-h_j (x - c_j)^2)$$

where  $c_j$  and  $h_j$  are the heuristically chosen centers and heights of the basis functions, respectively. We fit the DMP weights  $\beta = \{w_1, w_2 \dots w_J\}$  with locally-weighted regression (LWR) from the provided kinesthetic demonstration. For all DMPs in this work, we use  $J = 32$ , with gain values  $\alpha_y = 25$ ,  $\gamma_y = \frac{25}{4}$  and basis functions parameters set following prior work [16].

We choose (discrete) DMPs to implement skill learning as they permit efficient learning from a kinesthetic demonstration, and have two properties that enable rich generalization to 1) new goals (by specifying a new  $g$ ) and 2) arbitrary temporal scaling (by rescaling  $\tau$ ). This lets us induce a simple algebra for parameterizing our policy  $\pi_{d,\beta} : (c_{\text{ref}}, l, N)$ , indexing each learned DMP with a learned referent  $c_{\text{ref}}$ , a new goal location  $l$ , and a number of waypoints  $N$  (used to set  $\tau$ ) – in other words, allowing us to learn a new DMP – `track(loc: Location)` – that we can call with arbitrary new locations (from novel initial states) with arbitrary timing parameters (e.g., “can you track around Loki in 30 frames” or “I need a tracking shot around the tower... let’s try 2 seconds”).

**Visualizing DMP Rollouts.** Another advantage of using DMPs for parameterizing control is that they allow us to visualize entire trajectories *prior to execution*. Similar to how we visualize the keypoints and object segmentation masks in the collaborative gift-bag assembly setting, we provide a GUI that shows the robot and the planned path (and end-effector poses) via a simple Mujoco-based viewer. Fig. 1 (Right) provides an example – we plot the original kinesthetic demonstration relative to the current robot pose in green (for reference), and the planned DMP trajectory in blue, along with the end-effector orientation frames at the beginning and end of the trajectory. Users additionally can dynamically advance the simulation to visualize the entire rollout (at the actual speed of execution).

#### B.5 Physical Robot Platform & Controller Parameters

We use a Franka Emika Panda 7-DoF robot arm with a Robotiq 2F-85 parallel jaw gripper following the platform specification from DROID [9]. The robot and its base are positioned at one side of a 3’ x 5’ table, across from the user, such that the user and robot share the tabletop workspace. We use an overhead ZED 2 RGB-D camera with known intrinsics and extrinsics. For robot control, we use a modified version of the DROID control stack based on Polymetis [17]. Low-level policies command joint positions at 10 Hz to the joint impedance controller from [17] which runs at 1 kHz. We implement two compliance modes: a *stiff* mode which is activated when the robot is executing a low-level skill, and a *compliant* mode for when the user provides a kinesthetic demonstration.



**Safety.** We include multiple safeguards to ensure user safety. Users have the option to cancel any proposed behavior when an interpretable trace is presented with a physical Cancel button as described in §B.1 – this prevents execution and immediately backtracks the Vocal Sandbox system. Second, during execution of any low-level skill, the user can interrupt the robot’s motion with this button as well. This halts the robot’s motion and it immediately becomes fully compliant. Lastly, during user studies, both the user and proctor have access to the hardware emergency stop button which cuts the robot’s power supply and mechanically locks the robot arm.

## C Extended Discussion & Future Work

The following sections expand on the discussion from the main body of the paper, with a specific focus on the benefits of Vocal Sandbox’s modular design, before providing an extended treatment of our contributions and future directions in the broader context of systems for human-robot collaboration.

### C.1 On Modular vs. End-to-End Approaches

We develop Vocal Sandbox as a *modular* framework; the decoupled nature of the high-level language behavior planner from the low-level skill policies is explicit, and characterizes the rest of our contributions. Yet, this choice poses an important question – *why not an end-to-end approach?*

An initial answer stems from limitations in current models; our user studies show that “flat” planning with language models has several failure modes when it comes to reliability, while our static evaluations in §B.3 indicate deficiencies in ability for current multimodal models (e.g., GPT-4 Turbo with Vision) for high-precision language grounding in cluttered scenes. Yet, even if we consider a future where we have stronger end-to-end approaches that unify language, vision, and action [e.g., building on top of RT-2 or RT-H; 18, 19], we argue that modularity is an important feature in allowing users to isolate system failures and localize their feedback at the right level of abstraction.

Consider a common failure mode of end-to-end policies learned from data: visual robustness. Different degrees of distribution shift (e.g., introducing new distractor objects, or even perturbing the scene in small ways) not only hurt success rate [20], but they also affect the closed-loop execution in arbitrary ways [21], leading to suboptimal or unsafe trajectories. Worse is that errors only *cascade* as the robot or scene go further out of distribution, leading to even more unpredictable behavior.

Conversely, one of the more salient observations from our user study was how quickly users were able to not only identify failure modes in our system, but *co-adapt* to them. For example, within assembling the first two gift bags in the study, many users identified that the learned keypoint model was especially poor at predicting one category of object (Play-Doh). Rather than let this failure completely derail the task, users leaned on the modularity of our system to *isolate* this failure to the specific module (i.e., the visual keypoints-based policy) and sequence their feedback – teaching new high-level behaviors while *expecting* when and where the robot would fail. Specifically, users opted to teach a high-level behavior “assemble\_bag()” that would always attempt to pack the Play-Doh into the gift bag as the final step, affording them the ability to maximally “disengage” from actively supervising the robot for the bulk of execution, only intervening at the last step. In other words, modularity in Vocal Sandbox gives users the leverage to quickly understand the robot’s capabilities, as well as the power to meaningfully build on its strengths, and adapt around its weaknesses.

### C.2 Broader Context and Future Work

While the main body of our paper situates our framework against methods that use language models for task planning and learning low-level skills from multimodal feedback, Vocal Sandbox builds off a much larger body of work that build systems for different forms of human-robot interaction. The systems differ in the *modes* of collaboration they enable, from explicit human-robot teaming in situated environments [22–26], to learned methods for shared autonomy [27–29], to platforms for assistive robotics [30–32], amongst many others [33].

While Vocal Sandbox is heavily inspired by this prior work, especially those that learn language interfaces for grounding user intent to low-level robot behavior [34–36], this is only the beginning. Future iterations of our framework will build on the types of interactions and learning we permit (e.g., multi-robot teaming or integrating modalities such as touch or nonverbal feedback), all driving towards general and seamless human-robot collaboration.

## References

- [1] OpenAI. Introducing ChatGPT and Whisper APIs. <https://openai.com/index/introducing-chatgpt-and-whisper-apis/>, 2022.
- [2] OpenAI. GPT-3.5 – Function calling and other updates. <https://openai.com/index/function-calling-and-other-api-updates/>, 2023.
- [3] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever. Robust speech recognition via large-scale weak supervision. *ArXiv*, abs/2212.04356, 2022. URL <https://api.semanticscholar.org/CorpusID:252923993>.
- [4] OpenAI. Text-to-speech models. <https://platform.openai.com/docs/guides/text-to-speech>, 2023.
- [5] M. Minderer, A. A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, X. Wang, X. Zhai, T. Kipf, and N. Houlsby. Simple open-vocabulary object detection with vision transformers. In *European Conference on Computer Vision (ECCV)*, 2022.
- [6] M. Minderer, A. A. Gritsenko, and N. Houlsby. Scaling open-vocabulary object detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [7] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang. Fast segment anything, 2023.
- [8] H. K. Cheng and A. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *European Conference on Computer Vision*, 2022.
- [9] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Y. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J.-P. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. B. Simpson, Q. H. Vuong, H. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Zhao, C. Agia, R. Baijal, M. G. Castro, D. L. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O’Neill, R. M. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024.
- [10] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012.
- [11] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2021.
- [12] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, volume 139, pages 8748–8763, 2021.
- [13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [14] G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics YOLO, Jan. 2023. URL <https://github.com/ultralytics/ultralytics>.

- [15] S. Schaal. Dynamic movement primitives - a framework for motor control in humans and humanoid robotics. In H. Kimura, K. Tsuchiya, A. Ishiguro, and H. Witte, editors, *Adaptive Motion of Animals and Machines*, pages 261–280. Springer Tokyo, Tokyo, 2006.
- [16] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25:328–373, 2013. URL <https://api.semanticscholar.org/CorpusID:2431443>.
- [17] Y. Lin, A. S. Wang, G. Sutanto, A. Rai, and F. Meier. Polymetis. <https://facebookresearch.github.io/fairo/polymetis/>, 2021.
- [18] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023.
- [19] S. Belkhale, T. Ding, T. Xiao, P. Sermanet, Q. Vuong, J. Tompson, Y. Chebotar, D. Dwibedi, and D. Sadigh. Rt-h: Action hierarchies using language. *ArXiv*, abs/2403.01823, 2024. URL <https://api.semanticscholar.org/CorpusID:268249108>.
- [20] K. Burns, Z. Witzel, J. I. Hamid, T. Yu, C. Finn, and K. Hausman. What makes pre-trained visual representations successful for robust manipulation? *arXiv preprint arXiv:2312.12444*, 2023.
- [21] A. Xie, L. Lee, T. Xiao, and C. Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2024.
- [22] G. Hoffman and C. L. Breazeal. Collaboration in human-robot teams. *AIAA 1st Intelligent Systems Technical Conference*, 2004. URL <https://api.semanticscholar.org/CorpusID:1114471>.
- [23] C. L. Breazeal, G. Hoffman, and A. L. Thomaz. Teaching and working with robots as a collaboration. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, pages 1030–1037, 2004. URL <https://api.semanticscholar.org/CorpusID:14275016>.
- [24] J. Y. Chai, L. She, R. Fang, S. Ottarson, C. Little, C. Liu, and K. Hanson. Collaborative effort towards common ground in situated human-robot dialogue. *2014 9th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 33–40, 2014. URL <https://api.semanticscholar.org/CorpusID:7617225>.
- [25] M. Natarajan, E. Seraj, B. Altundas, R. R. Paleja, S. Ye, L. Chen, R. Jensen, K. C. Chang, and M. C. Gombolay. Human-robot teaming: Grand challenges. *Current Robotics Reports*, 4:81 – 100, 2023. URL <https://api.semanticscholar.org/CorpusID:260747359>.
- [26] J. Brawer, O. Mangin, A. Roncone, S. Widder, and B. Scassellati. Situated human-robot collaboration: predicting intent from grounded natural language. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 827–833, 2018. URL <https://api.semanticscholar.org/CorpusID:51998195>.
- [27] A. D. Dragan and S. S. Srinivasa. A policy-blending formalism for shared control. *International Journal of Robotics Research (IJRR)*, 32:790–805, 2013.



- [28] S. Javdani, H. Admoni, S. Pellegrinelli, S. S. Srinivasa, and J. A. Bagnell. Shared autonomy via hindsight optimization for teleoperation and teaming. *International Journal of Robotics Research (IJRR)*, 37:717–742, 2018.
- [29] S. Karamcheti, M. Srivastava, P. Liang, and D. Sadigh. LILA: Language-informed latent actions. In *Conference on Robot Learning (CoRL)*, 2021.
- [30] B. Driessen, H. Evers, and J. A. v Woerden. Manus—a wheelchair-mounted rehabilitation robot. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 215:285 – 290, 2001. URL <https://api.semanticscholar.org/CorpusID:35700443>.
- [31] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *RAS*, 57, 2009.
- [32] D. P. Losey, C. G. McDonald, E. Battaglia, and M. K. O’Malley. A review of intent detection, arbitration, and communication aspects of shared control for physical human-robot interaction. *Applied Mechanics Reviews*, 70, 2018.
- [33] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. O. Albu-Schäffer, K. Kosuge, and O. Khatib. Progress and prospects of the human–robot collaboration. *Autonomous Robots*, 42:957 – 975, 2017. URL <https://api.semanticscholar.org/CorpusID:21722736>.
- [34] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2011.
- [35] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy. Asking for help using inverse semantics. In *Robotics: Science and Systems (RSS)*, 2014.
- [36] H. Wang, K. Kedia, J. Ren, R. Abdullah, A. Bhardwaj, A. Chao, K. Y. Chen, N. Chin, P. Dan, X. Fan, G. Gonzalez-Pumariaga, A. Kompella, M. A. Pace, Y. Sharma, X. Sun, N. Sunkara, and S. Choudhury. Mosaic: A modular system for assistive and interactive cooking. *arXiv preprint arXiv:2402.18796*, 2024.