ImageInThat: Manipulating Image and Language to Convey User Intent to Robots

***Abstract*—Robots are becoming increasingly capable of responding to humans' natural language instructions to perform everyday manipulation tasks such as wiping a table or meal preparation. However, natural language presents challenges, such as ambiguity in spatial instructions (e.g., choosing a specific apple from a basket) and requiring users to mentally track how the robot's state evolves during long-horizon tasks. In this work, we propose using images as an alternative paradigm to instruct robots. We introduce ImageInThat, a specific instantiation of this paradigm, which allows users to perform direct manipulation on images in a timeline-style interface to generate step-by-step robot instructions. Through a user study with twelve participants we evaluated ImageInThat for instructing robots on kitchen manipulation tasks, comparing it to a text-based timeline. The results show that ImageInThat was faster, led to fewer errors, and was preferred by participants over the text-based interface. We also demonstrate that image-based instructions can be translated into robot executable policies, and discuss the potential of combining the strengths of language and images to create multimodal robot instructions.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Robots are becoming increasingly autonomous and capable of performing complex tasks through the advent of robot foundation models. Thus, we expect that robots will aid us in completing everyday tasks in the near-term. While these models are trained on internet-scale data, they do not understand the specific environment they will operate in. Thus, they will still need to attend to instructions from users [1]. Imagine a user asks a robot to put away newly purchased groceries in the kitchen. The robot, following attempt a generic organizing procedure, might randomly store canned goods and cereals together based on the space available inside. However, the user might organize their cabinet in a more nuanced way, such as by placing heavier items on lower shelves due to their cabinet's fragility and lighter items above for easier lifting. They might also want daily-use items such as cereal in the front, even if it blocks other items. As a result, the user would need to intervene and adjust the robot's approach to match their environment and preferences.

As it is expected that most users of future robots will not be experts, natural language has been touted as a modality that people can naturally use to instruct robots [2]. From a technical standpoint, state-of-the-art robot policies often utilize language-conditioning to generate robot actions. Despite the ubiquity of language and its easy-to-use interface, language is not always ideal. Natural language can be abstract and difficult to ground in the physical context. For instance, *"Put the cereal in the cabinet"* is unclear if there are many boxes of cereal and several cabinets to put them away inside of. Hence, the user needs to be more precise to achieve the desired result: *"Put the blue cereal box into the top left corner of the middle cabinet next to the other cereal"*, or alternatively, provide corrections to a simpler instruction by uttering language [3]. Beyond the immediate problem of specifying instructions, the user faces the additional challenge of predicting how the robot's planned actions will affect the environment, particularly for long-horizon tasks.

Instead of solely using natural language as a means for instructing robots, we propose the use of images—both as a means for providing instructions to a robot, and for the robot to communicate what its plans are to the user. Images provide two main benefits over language. First, images are inherently easier to ground, as they visually represent the environment from which they originate. Hence, users should be able to contextualize the robot's current and future states through images. Second, by forming an appropriate representation of the environment as an image, we can enable direct manipulation [4], which could be faster than forming precise language to resolve ambiguity and provide clear instructions. Finally, by supporting direct manipulation, the robot can develop a representation of the user's intent informed by their actions, and use this to intelligently propose future appropriate actions.

We introduce a specific instantiation of this paradigm, ImageInThat, where users can manipulate images of the environment to create instructions for a robot. In a user study with twelve participants, we compared ImageInThat to a language-based method for several instruction following tasks in simulated kitchen environments. We found that participants were able to generate instructions in these tasks faster and with less errors when using ImageInThat, and participants were more confident that their instructions could be understood by a robot when directly manipulating images that our prototype supports. We also demonstrate that instructions created using ImageInThat can be used to perform robot manipulation tasks on a physical robot arm setup. Lastly, we discuss how the two modalities of language and image can be used in a complementary fashion to create *multimodal instructions*.

II. Related Work

There exists a spectrum of robotic systems, ranging from those designed to create reusable, repeatable routines (i.e., using end-user robot programming [1], [5]) to those enabling real-time control (i.e., teleoperation). Our system falls somewhere in between, blending elements of both approaches.

End-user robot programming. Prior work can be largely categorized into natural language-based and visual-based methods. Many systems employ speech for programming [6], [7], though challenges remain in recognition and creating complex programs through continuous dialogue. Other approaches use text-based programming with visual scaffolds like blocks [8]–[10] or nodes [11], [12]. Recently, large language models (LLMs) have been employed for text-based interactions, often via chat interfaces [13], [14]. However, language input demands precision, especially for spatial tasks in robotics [15], [16]. Other systems span a range of visual modalities, such as augmented reality [17]–[21], spatial interfaces [22]–[24], sketch-based systems [25], [26], physical demonstrations [27], and tangible interaction [28], [29]. Despite the richness of prior EURP systems, they often require adherence to specific system rules. For instance, AR-based trigger-action programming [17] necessitates precise trigger and action specifications within the user's AR environment. Moreover, many prior systems use intermediate representations (e.g., flow diagrams or blocks) to convey user intent. In contrast, ImageInThat enables users to directly manipulate images to represent a desired world state without an intermediate representation.

Live Robot Control Modern robotic systems often include interfaces for the real-time teleoperation of the robots' joint and end-effector positions. These interfaces are often based on graphical user interfaces (GUIs) or joysticks, offering immediate feedback for the controllers to adjust their input [30], [31]. While effective, GUI- and joystick-based control mechanisms are cognitively and physically taxing for human operators, as they demands mental rotation and managing multiple separate degrees of freedom (DoFs) simultaneously. To alleviate this burden, one thread of recent teleoperation research aims to directly map human motion to robot motion, often from the motion of human hands to robot end-effectors [32]–[34]. Other work opts for a shared-control approach, where some trajectory prior informs a robot to generate high DoF trajectories from low DoF input. Such prior is often derived from inferred operator goals [35]–[37]. Despite the assistance available, live control methods still require operators' continuous mental and physical engagement to manage robots' low-level motion. Therefore, PhotoManipulator, along with other recent research, seeks to enable robots to interpret and execute human operators' high level intents.

Helping robots follow human instructions. With the advent of foundation models (FM), end-users may not need to program their robots from scratch (as with EURP). Recent work has successfully deployed pre-trained LLMs for robotics tasks such as translating language instructions to policy code [38]–[41]. Previous work has demonstrated that language can be used to iteratively guide and correct robots, enabling them to learn from these interactions and apply that knowledge in future tasks [3], [42]. In contrast, other work has attempted to translate human instructions into robot actions directly by training robotic foundation models (RFM [43]). Some of these techniques can generate actions when provided natural language inputs [44], [45], such as, "Bring me the chips from the drawer". Alternatively, images can be used to represent goals, either as sub-steps of a task [46], [47] or the final desired state [48], [49]. There are also attempts at using other modalities, including representing goals as sketches [16], or desired trajectories overlaid on images [50]. Our framework realized through ImageInThat, is agnostic to the method used to provide instructions to the robot. This flexibility is achieved by separating the user's input representation when providing instructions from the representation used by the execute the instructions. For example, images can be captioned and sent as language instructions to a language-conditioned RFM. Alternatively, pre-trained models can translate images into executable code using existing skill primitives (e.g., picking up a coffee mug). The images can also be directly provided to an image-conditioned RFM.

III. Theory and Design Space

A. Why Use Images?

Thus far, we have motivated the use of images to serve as goals for policies enabled by robot foundation models. Here, we motivate why images are useful as a representation for both visualizing their desired goals as well as to manipulate them.

As an output representation. For the user, visually being able to inspect the robot's environment (either in physical form or as images) can make it easier to ground themselves when providing instructions. Beyond this, being able to quickly inspect sub-steps of a program could help them spot errors before they occur, when compared to a step-by-step language procedure which may be more difficult to scan especially if the steps contain long instructions.

As an input representation. When instructing a robot using language, user intent may be hard to convey due to ambiguities associated with language [15], [16]. For instance, when setting a table, simply saying "put the plate and utensils on the table" is not sufficient because it does not detail where the items should be placed (in absolute terms) or with respect to each other (in relative terms). In this case, the user needs to precisely phrase their instruction to describe the task but in a way that can be understood by the system (e.g., "put the fork to the left of the plate but center it vertically").

In contrast, images more concretely describe a desired outcome with less room for interpretation (e.g., putting the fork to the left of the plate and aligning it vertically). However, a major challenge lies in how to generate goal images for the robot to execute [16]. While prior work has shown that this can be achieved through autonomous methods (e.g., [46]), such methods currently suffer from generating images with artifacts. Further, text-to-image models still struggle with the same problems of ambiguity along with needing to specify what parts of an image need to change and what parts need to stay the same.

We think that the direct manipulation could make it possible to quickly create images that more accurately capture a user's desired goals. For instance, through a simple drag-and-drop,

the user can position a fork to the left of a plate to convey both absolute and relative positioning of the objects. Further, we think that the user's interactions with task-relevant objects of the environment through an image could make it possible to make inferences about their intent. For instance, dragging a dirty bowl from the dining table to the sink could signal that other utensils nearby might also need cleaning.

B. Program Representation as Images

C. Model of Program Creation

IV. ImageInThat Interactions

Our system assumes a set-up phase where the robot builds an internal representation of the user's environment. This representation consists of objects that can be stateless or fixtures that are stateful (i.e., take on states). For instance, some items, like bowls, are *objects* because they do not have states (but can vary in location or contents). Other items, like furniture (e.g., drawers) or appliances (e.g., stovetops) are *fixtures* because they can be in one or more states (e.g., open or closed in the case of drawers). In ImageInThat, the user interact with objects and fixtures in different ways.

Defining steps. In ImageInThat, a *step* represents a desired change requested by the user to update the environment's state. Suppose the user wants the robot to put a dirty dish on the counter into the sink for rinsing. Steps here are defined as changes from the user's perspective. Thus, a possible next step could be for the dirty dish to be on the counter. Note that for a robot, executing this single step requires several actions: the robot needs to first pick up the dirty dish, and then place it in the sink.

Viewing steps. Each step in ImageInThat is represented as an image inside a timeline. The timeline shows each step starting from the left and moving to the right. When the user starts ImageInThat, a step representing the current world state populates the timeline. All future steps represent changes from this initial step.

Editing and creating steps. The user can modify steps in the timeline by hovering them to reveal a copy and delete command (but does not apply to the initial step). The user can add instructions by clicking any step in the timeline. This highlights the step and populates it inside an image editor which appears in the middle of the screen displaying the selected step. The editor allows standard pixel manipulation operations, including selection and manipulation.

Interactable objects. Once the editor appears, the user can perform simple drag and drop manipulations on one or many detected objects. When the user manipulates an object in the editor, it automatically populates a step in the timeline to the right of the current step. This also causes the current step to change to the newly created step which now appears in the editor. Continuously manipulating the same object does not create additional steps in case the user makes a mistake when they initially positioned the object during a manipulation.

Interactable fixtures. In ImageInThat, the user can manipulate the state of detected fixtures by clicking them when they are highlighted by the mouse. This triggers the creation of a new step where the image represents the environment with the fixture's state having updated.

Language-based editing. In addition to direct manipulation, the user can also use natural language to request state changes. When a step is selected in the timeline and appears in the editor, the user can input an instruction using natural language. Upon submitting their input, ImageInThat creates additional steps that attempt to address the changes the user requests.

Tracking changes between steps. Since the user may create many steps while providing instructions to the robot, it may be cumbersome for them to keep track of changes particularly when the changes might be subtle. For instance, if the only change between two steps is a plate moving, they user might struggle to notice it. Hence, ImageInThat allows the user to track changes between a selected step and any step they hover over in the timeline. This creates a looping animation that shows the changes occurring on the selected step in the timeline.

Predicting future steps. As the user adds new steps to the timeline, ImageInThat attempts to recognize the user's end goals (e.g., to clean up after meal preparation). Using this knowledge, it automatically proposes steps to the user in the timeline. Proposed steps are visualized as semi-transparent to the user and can be added to the timeline by clicking them or rejected by pressing the "x" button that appears on the top-left.

V. Implementation

High-level system design. Our system comprises two essential components: (1) a back-end server that orchestrates various machine learning models to enable photo manipulation, and (2) an intuitive user interface designed for viewing and editing images, which allows users to create and refine robot instructions through these manipulated images. The server is realized as a Flask application that enables two-way communication between the models and the user interface. The models are hosted as TCP sockets on workstations within the local network and communicate with the Flask server to respond to requests. The user interface is built using ReactJS. We realize ImageInThat in simulation and the real-world. In simulation, we utilize Robocasa [51] to model a mobile robot manipulator in various kitchen environments. In the real-world, we model a tabletop environment with a fixed based manipulator.

A. Representing the User's Environment

ImageInThat creates an initial representation of the user's environment by extracting information about the objects and fixtures present.

Background image generation. Using the knowledge about the initial world state and possible states that each fixture can take, all possible combinations of fixture states are generated. Then, images representing the *background* image pertaining to each possible state in the user's environment are generated. These images can be generated using a variety of methods.

Specific to the simulation environment, we generate images by using LLMs to write code via to modify the state of individual fixtures that represent a given state. We envision that this could be possible in the future given that many existing applications of robots assume the existence of digital twins [todo: CITE]. However, ImageInThat can also generate images representing a state change based on a language instruction in the absence of a simulation environment, made possible by fine-tuning diffusion models [46], [52]. At the end of this process, ImageInThat creates a background image for all possible states, and sets the background image to the one that represents the current state.

B. Enabling user manipulation of images

Generating interactable objects and fixtures. In order to enable the direct manipulation of images, ImageInThat begins by detecting items of interest. The items of interest can be provided as list of objects and fixtures (e.g., such as those provided by Robocasa). Or, ImageInThat can take the current background image representing the world state state to generate the items of interest. Next, ImageInThat generates bounding boxes for any detected items using an open vocabulary object detector, OWLv2 [53]. Then, information about the detected objects (not fixtures) are passed to a segmentation model, Segment Anything (SAM [54]), to generate masks that can be manipulated by the user. In simulation, the objects can be hidden when generating the background image so there is no white space and direct manipulation is possible at this stage. However, for real-world tasks where objects are usually in front of the background, manipulating them can leave white space behind. Hence, ImageInThat performs an inpainting step to eliminate white spaces behind the objects before the user manipulates them [55].

User manipulation of objects and fixtures through the editor. After processing the initial world state, the background image, generated masks, and bounding boxes for the fixtures are sent to user interface and rendered as a an SVG image inside a step. When the user clicks the initial step, the underlying representation is rendered inside an image editing application ([todo: CITE website]). The editor enables simple interactions such as the selection, deselection, and manipulating individual objects. When the user performs direct manipulations on one or more objects, the system creates a new step by copying the representation of the previous step while updating data about any objects that moved. To enable interactable fixtures, bounding boxes containing the location of fixtures are utilized to detect mouse clicks within this region, which activates a state change. When changing state, the current state variable is updated and the corresponding background image is retrieved and utilized when creating the step.

Predicting future steps (autocomplete)

VI. Evaluation

We evaluated ImageInThat through a controlled user study with twelve participants recruited using university and professional networks. Specifically, we compare an instance of ImageInThat to a language-based method. The choice of a language-based method for comparison is based on its widespread deployment to instruct robots [3], [48].

Conditions. To enable a fair comparison of each modality, we omitted all the language features of ImageInThat, including the ability to generate new images using language instructions or modify them using captions. For the language condition, we re-purposed ImageInThat, replacing all image-based interactions with text. Instead of populating images in the timeline, the user populates the timeline with steps that consist of text. For the image-based method in ImageInThat, we omit all language-based features (e.g., modifying images with captions) as well as autocomplete at the manipulation and step levels. In both the conditions, participants provided instructions pertaining to one object at a time, reflecting the current capabilities of robots, which typically handle single-object manipulation. Further, most language-conditioned robot policies, including foundation models, follow a similar approach, executing single language instructions at a time. While large language models can interpret more abstract instructions and decompose them, they are prone to errors and often require corrections [3].

Study design. The study utilizes a within-subjects design with two conditions–image and text–that are counterbalanced to minimize ordering effects (see Figure X for a full diagram). Within each condition, participants complete four tasks where they instruct a robot to complete kitchen manipulation tasks. The tasks include *Organizing Pantry*, *Sorting Fruits*, *Preparing Stirfry*, and *Washing Dishes*. Within each condition, the four tasks were randomly assigned. After both condition blocks, participants complete a freeform task to experiment with the features that were excluded from ImageInThat. Details of the individual tasks can be found in the appendices ([todo: link]) and the website.

Measures. In the study, we collected data about participants' performance when using both methods. Quantitative measures include task completion time and number of errors, which were determined by comparison to an *oracle* representation of the task established a priori by two researchers. We also measured subjective perceptions of the prototypes, including participants' confidence in correctly communicating their intent to the robot, workload (NASA TLX [56]), and usability (SUS [57]).

Procedure. Participants first provided consent and completed a pre-study questionnaire assessing their familiarity with robots and instructing them. After watching a video tutorial introducing ImageInThat and the text-based method and brief experimentation with both, participants began one of the two study condition blocks. Between tasks, participants rated how confident they were that the robot could understand their instructions unambiguously. At the end of each condition block, two questionnaires (NASA TLX and SUS) were administered to assess workload and usability, respectively. At the end of the study, participants rated their preference for the text-based method compared to ImageInThat. Lastly, we conducted a brief interview probing participants about various

aspects of both methods.

Hypotheses. We formulated three hypotheses: Participants will complete tasks faster using the ImageInThat compared to the text-based method (**H1**); Participants will make fewer errors using the ImageInThat compared to the text-based method (**H2**); Participants will feel more confident that a robot unambiguously understands their instructions when using ImageInThat compared to the text-based method (**H3**).

Findings.

VII. Discussion

VIII. Conclusion

Appendix

References

[1] G. Ajaykumar, M. Steele, and C.-M. Huang, "A survey on end-user robot programming," **ACM Computing Surveys (CSUR)**, vol. 54, no. 8, pp. 1–36, 2021.

[2] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, "Robots that use language," **Annual Review of Control, Robotics, and Autonomous Systems**, vol. 3, no. 1, pp. 25–55, 2020.

[3] L. Zha, Y. Cui, L.-H. Lin, M. Kwon, M. G. Arenas, A. Zeng, F. Xia, and D. Sadigh, "Distilling and retrieving generalizable knowledge for robot manipulation via language corrections," **arXiv preprint arXiv:2311.10678**, 2023.

[4] B. Shneiderman, "Direct manipulation: A step beyond programming languages," **Computer**, vol. 16, no. 08, pp. 57–69, 1983.

[5] H. Lieberman, F. Paternò, M. Klann, and V. Wulf, "End-user development: An emerging paradigm," in **End user development**. Springer, 2006, pp. 1–8.

[6] M. Cakmak and L. Takayama, "Teaching people how to teach robots: The effect of instructional materials and dialog design," in **Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction**, 2014, pp. 431–438.

[7] J. F. Gorostiza and M. A. Salichs, "End-user programming of a social robot by dialog," **Robotics and Autonomous Systems**, vol. 59, no. 12, pp. 1102–1114, 2011.

[8] J. Huang and M. Cakmak, "Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts," in **Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction**, 2017, pp. 453–462.

[9] J. Huang, T. Lau, and M. Cakmak, "Design and evaluation of a rapid programming system for service robots," in **2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)**. IEEE, 2016, pp. 295–302.

[10] D. Weintrop, A. Afzal, J. Salac, P. Francis, B. Li, D. C. Shepherd, and D. Franklin, "Evaluating coblox: A comparative study of robotics programming environments for adult novices," in **Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems**, 2018, pp. 1–12.

[11] S. Alexandrova, Z. Tatlock, and M. Cakmak, "Roboflow: A flow-based visual programming language for mobile manipulation tasks," in **2015 IEEE International Conference on Robotics and Automation (ICRA)**. IEEE, 2015, pp. 5537–5544.

[12] D. Porfirio, A. Sauppé, A. Albarghouthi, and B. Mutlu, "Authoring and verifying human-robot interactions," in **Proceedings of the 31st annual acm symposium on user interface software and technology**, 2018, pp. 75–86.

[13] U. B. Karli, J.-T. Chen, V. N. Antony, and C.-M. Huang, "Alchemist: Llm-aided end-user development of robot applications," in **Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction**, 2024, pp. 361–370.

[14] Y. Ge, Y. Dai, R. Shan, K. Li, Y. Hu, and X. Sun, "Cocobo: Exploring large language models as the engine for end-user robot programming," **arXiv preprint arXiv:2407.20712**, 2024.

[15] D. Masson, S. Malacria, G. Casiez, and D. Vogel, "Directgpt: A direct manipulation interface to interact with large language models," in **Proceedings of the CHI Conference on Human Factors in Computing Systems**, 2024, pp. 1–16.

[16] P. Sundaresan, Q. Vuong, J. Gu, P. Xu, T. Xiao, S. Kirmani, T. Yu, M. Stark, A. Jain, K. Hausman **et al.**, "Rt-sketch: Goal-conditioned imitation learning from hand-drawn sketches," 2024.

[17] B. Ikeda and D. Szafir, "Programar: Augmented reality end-user robot programming," **ACM Transactions on Human-Robot Interaction**, vol. 13, no. 1, pp. 1–20, 2024.

[18] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, "Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces," in **Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems**, 2022, pp. 1–33.

[19] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. M. Van der Loos, and E. Croft, "Robot programming through augmented trajectories in augmented reality," in **2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. IEEE, 2018, pp. 1838–1844.

[20] L. Gong, S. Ong, and A. Nee, "Projection-based augmented reality interface for robot grasping tasks," in **Proceedings of the 2019 4th International Conference on Robotics, Control and Automation**, 2019, pp. 100–104.

[21] Y. Cao, T. Wang, X. Qian, P. S. Rao, M. Wadhawan, K. Huo, and K. Ramani, "Ghostar: A time-space editor for embodied authoring of human-robot collaborative task with augmented reality," in **Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology**, 2019, pp. 521–534.

[22] G. Huang, P. S. Rao, M.-H. Wu, X. Qian, S. Y. Nof, K. Ramani, and A. J. Quinn, "Vipo: Spatial-visual programming with functions for robot-iot workflows," in **Proceedings of the 2020 CHI conference on human factors in computing systems**, 2020, pp. 1–13.

[23] Y. Cao, Z. Xu, F. Li, W. Zhong, K. Huo, and K. Ramani, "V. ra: An in-situ visual authoring system for robot-iot task planning with augmented reality," in **Proceedings of the 2019 on designing interactive systems conference**, 2019, pp. 1059–1070.

[24] K. Mahadevan, Y. Chen, M. Cakmak, A. Tang, and T. Grossman, "Mimic: In-situ recording and re-use of demonstrations to support robot teleoperation," in **Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology**, 2022, pp. 1–13.

[25] D. Sakamoto, K. Honda, M. Inami, and T. Igarashi, "Sketch and run: a stroke-based interface for home robots," in **Proceedings of the SIGCHI conference on human factors in computing systems**, 2009, pp. 197–200.

[26] D. Porfirio, L. Stegner, M. Cakmak, A. Sauppé, A. Albarghouthi, and B. Mutlu, "Sketching robot programs on the fly," in **Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction**, 2023, pp. 584–593.

[27] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "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.

[28] Y. S. Sefidgar, P. Agarwal, and M. Cakmak, "Situated tangible robot programming," in **Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction**, 2017, pp. 473–482.

[29] Y. Gao and C.-M. Huang, "Pati: a projection-based augmented table-top interface for robot programming," in **Proceedings of the 24th international conference on intelligent user interfaces**, 2019, pp. 345–355.

[30] K. Darvish, L. Penco, J. Ramos, R. Cisneros, J. Pratt, E. Yoshida, S. Ivaldi, and D. Pucci, "Teleoperation of humanoid robots: A survey," **IEEE Transactions on Robotics**, vol. 39, no. 3, pp. 1706–1727, 2023.

[31] D. J. Rea and S. H. Seo, "Still not solved: A call for renewed focus on user-centered teleoperation interfaces," **Frontiers in Robotics and AI**, vol. 9, p. 704225, 2022.

[32] D. Rakita, B. Mutlu, and M. Gleicher, "A motion retargeting method for effective mimicry-based teleoperation of robot arms," in **Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction**, 2017, pp. 361–370.

[33] D. Rakita, B. Mutlu, M. Gleicher, and L. M. Hiatt, "Shared control–based bimanual robot manipulation," **Science Robotics**, vol. 4, no. 30, p. eaaw0955, 2019.

[34] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," **arXiv preprint arXiv:2401.02117**, 2024.

[35] C.-M. Huang and B. Mutlu, "Anticipatory robot control for efficient human-robot collaboration," in **2016 11th ACM/IEEE international**

conference on human-robot interaction (HRI). IEEE, 2016, pp. 83–90.

[36] S. Jain and B. Argall, "Probabilistic human intent recognition for shared autonomy in assistive robotics," **ACM Transactions on Human-Robot Interaction (THRI)**, vol. 9, no. 1, pp. 1–23, 2019.

[37] D. P. Losey, H. J. Jeon, M. Li, K. Srinivasan, A. Mandlekar, A. Garg, J. Bohg, and D. Sadigh, "Learning latent actions to control assistive robots," **Autonomous robots**, vol. 46, no. 1, pp. 115–147, 2022.

[38] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in **2023 IEEE International Conference on Robotics and Automation (ICRA)**. IEEE, 2023, pp. 9493–9500.

[39] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in **2023 IEEE International Conference on Robotics and Automation (ICRA)**. IEEE, 2023, pp. 11 523–11 530.

[40] P. Liu, Y. Orru, C. Paxton, N. M. M. Shafiullah, and L. Pinto, "Ok-robot: What really matters in integrating open-knowledge models for robotics," **arXiv preprint arXiv:2401.12202**, 2024.

[41] K. Mahadevan, J. Chien, N. Brown, Z. Xu, C. Parada, F. Xia, A. Zeng, L. Takayama, and D. Sadigh, "Generative expressive robot behaviors using large language models," in **Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction**, 2024, pp. 482–491.

[42] J. Liang, F. Xia, W. Yu, A. Zeng, M. G. Arenas, M. Attarian, M. Bauza, M. Bennice, A. Bewley, A. Dostmohamed **et al.**, "Learning to learn faster from human feedback with language model predictive control," **arXiv preprint arXiv:2402.11450**, 2024.

[43] K. Kawaharazuka, T. Matsushima, A. Gambardella, J. Guo, C. Paxton, and A. Zeng, "Real-world robot applications of foundation models: A review," **arXiv preprint arXiv:2402.05741**, 2024.

[44] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu **et al.**, "Palm-e: An embodied multimodal language model," **arXiv preprint arXiv:2303.03378**, 2023.

[45] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi **et al.**, "Openvla: An open-source vision-language-action model," **arXiv preprint arXiv:2406.09246**, 2024.

[46] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine, "Zero-shot robotic manipulation with pretrained image-editing diffusion models," **arXiv preprint arXiv:2310.10639**, 2023.

[47] A. Nair, S. Bahl, A. Khazatsky, V. Pong, G. Berseth, and S. Levine, "Contextual imagined goals for self-supervised robotic learning," in **Conference on Robot Learning**. PMLR, 2020, pp. 530–539.

[48] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu **et al.**, "Octo: An open-source generalist robot policy," **arXiv preprint arXiv:2405.12213**, 2024.

[49] I. Kapelyukh, V. Vosylius, and E. Johns, "Dall-e-bot: Introducing web-scale diffusion models to robotics," **IEEE Robotics and Automation Letters**, vol. 8, no. 7, pp. 3956–3963, 2023.

[50] J. Gu, S. Kirmani, P. Wohlhart, Y. Lu, M. G. Arenas, K. Rao, W. Yu, C. Fu, K. Gopalakrishnan, Z. Xu **et al.**, "Rt-trajectory: Robotic task generalization via hindsight trajectory sketches," **arXiv preprint arXiv:2311.01977**, 2023.

[51] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu, "Robocasa: Large-scale simulation of everyday tasks for generalist robots," **arXiv preprint arXiv:2406.02523**, 2024.

[52] T. Brooks, A. Holynski, and A. A. Efros, "Instructpix2pix: Learning to follow image editing instructions," in **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**, 2023, pp. 18 392–18 402.

[53] M. Minderer, A. Gritsenko, and N. Houlsby, "Scaling open-vocabulary object detection," **Advances in Neural Information Processing Systems**, vol. 36, 2024.

[54] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo **et al.**, "Segment anything," in **Proceedings of the IEEE/CVF International Conference on Computer Vision**, 2023, pp. 4015–4026.

[55] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky, "Resolution-robust large mask inpainting with fourier convolutions," in **Proceedings of the IEEE/CVF winter conference on applications of computer vision**, 2022, pp. 2149–2159.

[56] S. G. Hart, "Nasa-task load index (nasa-tlx); 20 years later," in **Proceedings of the human factors and ergonomics society annual meeting**, vol. 50, no. 9. Sage publications Sage CA: Los Angeles, CA, 2006, pp. 904–908.

[57] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," **Intl. Journal of Human–Computer Interaction**, vol. 24, no. 6, pp. 574–594, 2008.