From Exploration to Reuse: An Embodied Agent Framework for Manipulation Skill Learning

Mohamed Roshdi
University of Bonn

Alexander Zorn
Fraunhofer IAIS

Jörg Kindermann
Fraunhofer IAIS

Hermann Blum
University of Bonn

***Abstract*—Recent advances in Vision Language Models (VLMs) have enabled agentic robotic frameworks capable of interpreting natural language instructions and generating robot commands in the task-level space or the joint-level space. However, existing approaches predominantly rely on either zero-shot execution with no modular skill reuse methods or task-specific policy learning that must be retrained for each new behaviors. In this paper, we propose an embodied dual-agent architecture that allows agents to acquire new skills by exploring inputs and hyperparameter spaces of model-based controllers and reusing successfully learned skills through a memory skill module in future tasks. The exploratory behavior is enabled through the interaction between VLM planning and monitoring agents, where the former determines a long-term plan and suggests the next skill and the latter explores skill changes that can lead to task success. Thus, we call this framework Model-based Acquisition and Retrieval of Skills (MARS). We showcase MARS across different embodiments in simulation and real-world through a variety of prehensile and non-prehensile, short and long-horizon tasks. Ablation studies on multiple embodiments in real-world and simulation show that the monitoring agent improves success rates from 25% to 77%, while decreasing the number of trials till task success from 6.25 to 1.5. While the skill memory module improves success rates from 39% to 70.5%. The skill memory also allows transferring skills to new objects by reusing strategies learned from previous interactions.**

***Index Terms*—Vision Language Models, Embodied Agentic AI**

I. Introduction & Related Work

The goal of building robots that can understand and respond to free-form natural language instructions has been a longstanding challenge in robotics and AI research [1]. Rather than relying on rigid task identifiers or goal images to specify behaviors, natural language offers a flexible and intuitive interface that scales to a wide variety of tasks without requiring predefined vocabularies or grammars [2]. In this setting, a single agent must execute a diverse series of manipulation tasks each expressed in natural language, performing all tasks sequentially and in any order [2]. This requires the agent to simultaneously understand perception from raw sensory inputs, understand language, and master motor control as a single system [2], [3]. Despite recent progress in Embodied AI and VLM-based robotics, a persistent limitation across most systems is the inability to autonomously acquire new low-level skills, store them in a structured manner, and reuse them effectively in future tasks [3].

Many existing agentic robotic frameworks rely on a fixed set of predefined atomic or composite actions, which constrains their adaptability in dynamic, real-world environments [4]–[6]. While these approaches enable flexible high-level planning, their action spaces are static, preventing the agent from extending its behavioral repertoire through experience. Other methods introduce new skills via imitation learning [7], requiring explicit human demonstrations for each new task [8], [9]. A separate line of work explores learning low-level behaviors through optimization guided by LLM-generated rewards [10], [11], virtual potential fields [12], or model-based controllers [13]. While these methods demonstrate impressive zero-shot generalization, they typically execute skills in an open-loop manner and do not provide mechanisms for reusing the resulting policies. As a result, newly acquired behaviors are discarded after execution, preventing cumulative skill ac-

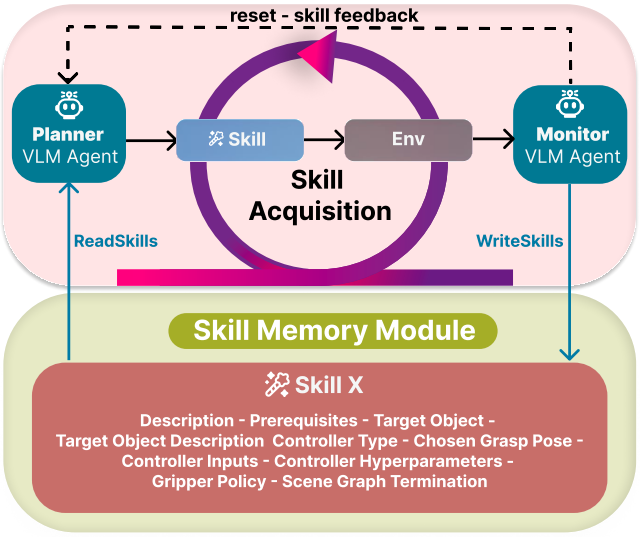

Fig. 1: Overview of the MARS framework and its main components. The Dual VLM Agent design allows the planning agent to focus on skill generation and the monitoring agent to determine success and explore the Skill space to search for possible successful skill configurations that achieve the goal. The interaction between the agents achieves skill acquisition. The monitoring agent has the ability to write successful skill configurations to the Skill Memory Module. Saving the skills as JSON documents allowing persistent sessions, cross-embodiment skill adoption, and human interpretation of skills.

Paper	Learn New Skills Independently	Save & Reuse Skills	Memory	Graph Based Planning
[5] SayPlan	–	–	–	X
[9] ROS-LLM	–	X	–	–
[8] SayCan	–	X	X	–
[15] RoboReflect	–	(Reflections)	X	–
[4] CognitiveOS	–	(Action Sequence)	X	–
[6] Pragmatist Robot	–	(Reflections)	X	–
[16] Language2Rewards	X	–	–	–
[12] VoxPoser	X	–	–	–
[11] IKER	X	–	–	X
[13] Meta-Control	X	–	–	–
[14] Uni-Skill	(Video Retrieval)	(Static Corpus)	–	–
MARS (ours)	**X**	**X**	**X**	**X**

TABLE I: Comparison of embodied agentic frameworks .

quisition and adaptation. Concurrently, Uni-Skill [14] presents few-shot skill generalization based on annotated robotic video demonstrations, but lacks independent and memory-persistent closed-loop skill acquisition. Table I shows a summary of the features and limitations of relevant agentic approaches.

To address these limitations, we propose **MARS** (**M**odel-based **A**cquisition and **R**etrieval of **S**kills), an embodied dual-agent architecture in which two cooperating VLM agents jointly explore the parameter spaces of model-based controllers to acquire new manipulation skills and reuse successfully learned skills through a persistent skill memory module, as shown in Figure 1. A *planning agent* determines a long-horizon plan and proposes the next skill to execute, while a *monitoring agent* evaluates execution outcomes, explores corrective skill configurations, and commits successful skills to memory. In this work, a *skill* is defined as an atomic behavior that induces a targeted change in the environment, represented as a goal in the scene graph. Each skill is paired with a low-level model-based controller, whose hyperparameters and inputs determine the behavior and success of the skill. For each task, the system can explore multiple candidate skill configurations and retain the most reliable configuration based on execution feedback. We rely on a semantic 3D Scene Graph for both short and long-horizon planning, where nodes represent objects and edges represent spatial and physical relationships. The proposed formulation avoids voxel-based representations, static skill libraries, and imitation-learned policies, while remaining few-shot, reusable, and agnostic to specific environments and robots.

The main contributions of this paper are as follows:

- A dual-agent VLM architecture that acquires new manipulation skills online through few-shot exploration of model-based controller parameter spaces.
- A persistent skill memory module that stores learned skills augmented with semantic and relational metadata, enabling retrieval and reuse across tasks and object instances.
- Experimental validation across three 6DOF robot embodiments in simulation and real-world tasks.

By comparing the full MARS framework to different baselines in an ablation study across different embodiments and custom short and long-horizon tasks, we show the dominant performance of dual-agent design and the skill memory retention module across CALVIN-style [17] tasks and custom tabletop manipulation tasks.

II. Methodology

A. Problem Statement

Given a natural language instruction I, we consider the problem of generating a series of consecutive skills $\{\sigma_t\}$ that achieve the target task, where a skill σ_{t+1} is generated and executed immediately after σ_t terminates. A skill is defined as a high-level Bauplan of an environment-changing robotic behavior: **name**, **description**, **target object** and **target-object description**, **controller type**, **controller inputs**, **controller hyperparameters**, and **preconditions**. The name provides a symbolic identifier for referencing and reuse, while the description captures the strategy of the behavior. The target object specifies the entity to be manipulated, its description disambiguates that entity using visual attributes, and the chosen grasp pose describes the SE(3) pose selected for contacting the object. The controller type defines the low-level control family, controller inputs define the task-level goal variables passed to that controller, and controller hyperparameters define execution characteristics. The gripper policy meanwhile can be considered a separate controller structure with only binary conditions that either open or close the gripper. Constraints—including preconditions, timeout, and termination conditions—are integral parts of the skill; preconditions are critical for safe, context-correct execution, while termination conditions may be explicitly empty (`None`) when timeout- or state-based completion is sufficient for achieving the task. The problem inputs are augmented with the following: a library of predefined low-level controllers $\{C_1, C_2, ..C_n\}$, a Scene Graph $S_t =< V, E >$, where V is a set of detected object nodes and E is the set of inter-object relationships, and access to a library of pre-learned skills $\{\sigma_p\}$ if available. Thus, the system π

$$\pi(\sigma_t \mid \sigma_{\{0,\ t-1\}},\ \{C_1, C_2, ..C_n\},\ S_t,\ \{\sigma_p\})$$

samples a skill σ_t given the inputs and the history of previously applied skills.

B. MARS Dual-Agent Skill Acquisition

The complete skill acquisition loop is formalized in Algorithm 1. Let I denote the user instruction, x_t the RGB observation at step t, S_t the scene graph, and $\mathcal{C} = \{C_1, \ldots, C_n\}$ the controller library. Let $\mathcal{M}$ denote the skill memory.

The planning agent P proposes the next skill (or a stop token $\varnothing$), where h_t is the interaction history including the initial user message and agent intermediate outputs.

The monitoring agent M evaluates the execution given the skill and the before and after scene graphs and images, returning a structured outcome

$$(o_t, f_t, \tilde{\sigma}_t, r_t, m_t) = M(I, x_t, S_t, \sigma_t, x_{t+1}, S_{t+1}),$$

where $o_t \in \{\texttt{success}, \texttt{failure}\}$ is skill success, f_t is text feedback for re-planning, $\tilde{\sigma}_t$ is an optional refined skill variant, $r_t \in \{0, 1\}$ indicates whether to reset the environment, and $m_t \in \{0, 1\}$ indicates whether the (possibly refined) skill should be written to memory. The monitoring agent is instructed to reset if and only if the environment has been corrupted and hinders further learning or if too many skill trials have been explored, causing the history and context window to grow larger.

Algorithm 1 MARS Skill Acquisition and Execution Loop

Input: instruction I, initial scene graph S_0, initial image x_0, controller library $\mathcal{C}$, skill memory $\mathcal{M}$, Planning Agent P, Monitoring Agent M
Initialize: $t \leftarrow 0$, history $h_0 \leftarrow \emptyset$
while true do
  $\sigma_t \sim P(\cdot \mid I, x_t, S_t, \mathcal{C}, \mathcal{M}, h_t)$
  if $\sigma_t = \varnothing$ **then**
    break
  end if
  Execute skill σ_t
  Observe updated state (x_{t+1}, S_{t+1})
  $(o_t, f_t, \tilde{\sigma}_t, r_t, m_t) \sim M(I, x_t, S_t, \sigma_t, x_{t+1}, S_{t+1})$
  if $m_t = 1$ **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{\tilde{\sigma}_t\}$ if skill refined, else $\mathcal{M} \leftarrow \mathcal{M} \cup \{\sigma_t\}$
  end if
  if $r_t = 1$ **then**
    Reset environment
  end if
  $h_{t+1} \leftarrow \texttt{UpdateHistory}(h_t, \sigma_t, o_t, f_t)$
  $t \leftarrow t + 1$
end while

We save the skills on desk as JSON documents and add the growing set of skills to both agents prompts as execution continues. In simulation, MARS uses two low-level controllers in IsaacLab depending on the skill: a model predictive controller (MPC) [18] for goal-directed end-effector motion and an operational space controller (OSC) [19] for task-space wrench force controller. We implement MPC as a nonlinear program in CasADi [20] in a receding horizon manner [21], producing SE(3) way-points that are converted to joint-space commands through the IsaacLab IK controller, while OSC directly tracks target pose–wrench commands with axis-wise stiffness and force gains [22]. In the real world Igus Rebel, we use the Commonplace Robotics IK controller through the CRI Python library [23], so real-world skills are executed as asynchronous end-effector pose commands with velocity and timeout constraints rather than MPC rollouts. Each skill also includes a gripper policy and a termination rule: the gripper state is governed either by an always open/close behavior or by pose-triggered binary conditions of the form $d_t < \tau$ or $d_t > \tau$, while termination is evaluated from scene-graph relations as $\beta_\sigma(S_t) = \texttt{TERMINATE}$ if and only if all required relations hold and all forbidden relations are absent. We build the scene graph using a lightweight manipulation-oriented pipeline built on ConceptGraphs [24]. From RGB-D observations, we detect and segment objects, lift masked pixels into 3D point clouds, fit object-level 3D bounding boxes, infer semantic relations, and output a structured graph whose nodes store object identity, captions, 3D geometry, and heuristic candidate grasp poses (top, left, right), while edges encode manipulation-relevant relations including `on top of`, `inside`, `next to`, and `gripping object`. This scene graph serves as the shared state for both agents.

III. Experimental Design

We evaluate MARS in simulation and the real world across multiple embodiments. In IsaacLab, we use a custom tabletop scene with a cup, basket, and box, and a CALVIN C desk scene [17] with a drawer, slider, button, and cube. Simulation experiments use a UR10 with a vacuum gripper and a UR10E with a Robotiq 2F-85 gripper, on tasks including drawer opening/closing, slider movement, button pressing, cube pick-and-place, and cup placement into a basket. Real-world experiments use a 6DOF Igus Rebel with a vacuum gripper in a tabletop scene, where the tasks are placing a cup or a concave box into a basket; the latter requires side grasps. For planning, monitoring, and scene-graph captioning, we use GPT-4o, GPT-5o, and GPT-5-mini, respectively. In simulation, both IsaacLab and the scene-graph pipeline run on a single NVIDIA RTX 6000 machine. Across embodiments, skills are kept fixed except for a lightweight adapter that transforms target end-effector quaternions into the tool frame. We first evaluate short-horizon skills through two ablation studies isolating the effects of the monitoring agent and the skill memory module. Specifically, we compare the full system against a planning-only baseline, a planning-plus-memory baseline, and a dual-agent variant without skill memory. Performance is measured by success rate S_R and trials to first success T_S over $T \in \{5, 7, 10\}$ trials, with $T_S = T$ indicating failure and $T_S = 0$ indicating zero-shot skill success. To evaluate long-horizon reuse and adaptation, we also test Long-Horizon Multi-Task Language Control (LH-MTLC) [17] by executing valid 5-task CALVIN chains after pre-learning the individual skills and reporting the fraction of successful task chains.

IV. Results

In the first ablation study in Table II, MARS shows a large performance gap relative to the baselines on both metrics, achieving substantially lower T_{S} and markedly higher S_R across almost all tasks, achieving a mean S_R of 77% and T_{S} of 1.5, compared to only 25% S_R and 6.25 T_{S}. In the second ablation study in Table III, the baseline without skill memory achieves the same mean T_{S} as MARS, with both variants reaching 1.5. However, the difference in success rate remains large: the version without skill memory reaches only 39%, whereas MARS achieves 70.5%. This gap indicates that skill memory improves consistency because without it the agent must repeatedly relearn successful skill configurations across trials instead of reusing them.

TABLE II: MARS ablation with and without monitor agent.

Task	Robot	Metric	Planner-Only	Planner-Memory	MARS
Open Drawer	UR10E (sim)	T_S	10	10	2
		S_R	0%	0%	80%
Close Drawer	UR10 (sim)	T_S	10	10	1
		S_R	0%	0%	70%
Move Slider	UR10E (sim)	T_S	10	10	2
		S_R	0%	0%	70%
Press Button	UR10E UR10 (sim)	T_S	5	5	1
		S_R	0%	0%	80%
Pick and Place Cube	UR10 (sim)	T_S	0	0	0
		S_R	100%	100%	100%
Pick and Place Cup in Basket	UR10 (sim)	T_S	10	10	4
		S_R	0%	0%	60%
Pick and Place Concave Box	Igus Rebel (real)	T_S	2	3	1
		S_R	29%	43%	71%
Pick and Place Cup	Igus Rebel (real)	T_S	2	2	1
		S_R	29%	57%	86%
Mean		T_S	6.13	6.25	**1.5**
		S_R	20%	25%	**77%**

TABLE III: MARS ablation with and without memory.

Task	Metric	MARS-NoMemory	MARS
Close Drawer (sim)	T_S	2	2
	S_R	20%	70%
Pick and Place Concave Box (real)	T_S	1	1
	S_R	57%	71%
Mean	T_S	1.5	**1.5**
	S_R	39%	**70.5%**

T_S denotes the number of trials until the first successful execution, where lower values indicate faster skill acquisition. S_R denotes the success rate across all trials, reported as a percentage, where higher values indicate more consistent reuse of successful skill configurations. MARS shows dominant performance in S_R against all baselines and T_S against baselines without monitor agent.

TABLE IV: MARS LH-MTLC results on CalvinC.

Method	1	2	3	4	5	Avg. Length
MARS	100%	80%	80%	60%	40%	3.6
MARS-NoMonitor	100%	87%	87%	57%	14%	3.4

Columns 1–5 report the percentage of task chains successfully completed up to each horizon length, and **Avg. Len** reports the average number of successfully completed instructions per five-step chain. Higher values indicate better long-horizon reliability.

In the CALVIN-style LH-MTLC evaluation shown in Table IV, both variants solve short task chains reliably- since all skills are pre-learned, but performance degrades as the horizon increases. Figure 2 shows sample runs, with the top figure shows the monitoring agent searching for the correct skill configuration by generating insightful feedback and skill suggestions. The bottom figure shows how an existing skill is used to infer the correct strategy for grasping a new object with the same topology. The primary sources of errors were: wrong grasp and placement poses, wrong force directions, pose detection errors, and vacuum gripper release.

V. Conclusion and Future Work

We presented MARS, a dual-agent framework for online manipulation skill acquisition and reuse. By combining symbolic planning with closed-loop monitoring, MARS explores skill configurations, stores successful ones, and reuses them in later tasks. The monitoring agent is the key contributor:

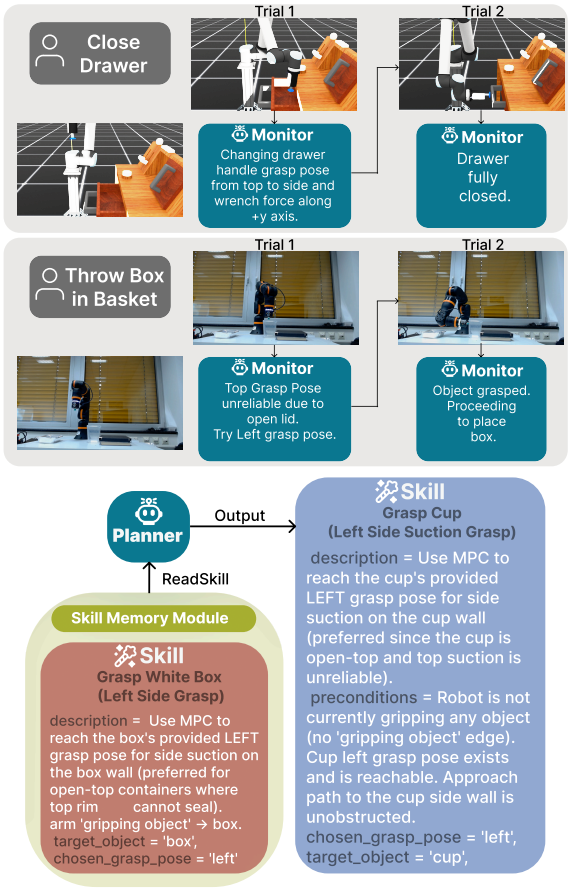

Fig. 2: Top: example runs with $T_S = 1$ showing monitor skill exploration. Bottom: real-world example showing zero-shot generalization to a new object (cup) based on the skill learned from interacting with a similar object (open-lid box).

without it, the planning-only system reaches only 25% mean success, while full MARS achieves 77% and finds a working configuration in an average of 1.5 trials. Skill memory further improves reliability, increasing success from 39% to 70.5% by reusing validated configurations instead of relearning them. CALVIN long-horizon tasks further show that previously learned skills can be reused across task sequences, though the monitor provides less additional benefit once the skills are already acquired. Limitations include the separation of skill acquisition and long-horizon execution, usage of the external scene graph with substantial time latency and static candidate grasp poses, and lack of safety mechanisms. An interesting interpretation is to view MARS through the lens of Hierarchical Reinforcement Learning (HRL) [25]. In this view, MARS presents a Semi-MDP [26] where skill preconditions define initiation sets, the controller defines the intra-option policy, and scene graph checks define termination, making skill generation analogous to an HRL option-level policy. We plan to address the limitations by investigating this connection and other directions such as skill planning in the latent space of world models [27]–[30] and Safe Exploration [31] .

REFERENCES

[1] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, "Interactive language: Talking to robots in real time," 2022. [Online]. Available: https://arxiv.org/abs/2210.06407

[2] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. B. Amor, "Language-conditioned imitation learning for robot manipulation tasks," 2020. [Online]. Available: https://arxiv.org/abs/2010.12083

[3] S. Salimpour, L. Fu, K. Rachwał, P. Bertrand, K. O'Sullivan, R. Jakob, F. Keramat, L. Militano, G. Toffetti, H. Edelman, and J. P. Queralta, "Towards embodied agentic ai: Review and classification of llm- and vlm-driven robot autonomy and interaction," 2025. [Online]. Available: https://arxiv.org/abs/2508.05294

[4] A. Lykov, M. Konenkov, K. F. Gbagbe, M. Litvinov, R. Peter, D. Davletshin, A. Fedoseev, O. Kobzarev, A. Alabbas, O. Alyounes, M. A. Cabrera, and D. Tsetserukou, "Cognitiveos: Large multimodal model based system to endow any type of robot with generative ai," *arXiv.org*, 2024.

[5] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Sünderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable task planning," *Conference on Robot Learning*, 2023.

[6] K. Qu, G. Lan, R. Zurbrügg, C. Chen, C. E. Mower, H. Bou-Ammar, and M. Hutter, "A pragmatist robot: Learning to plan tasks by experiencing the real world," 2026. [Online]. Available: https://arxiv.org/abs/2507.16713

[7] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A survey of imitation learning: Algorithms, recent developments, and challenges," 2023. [Online]. Available: https://arxiv.org/abs/2309.02473

[8] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. M. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan, "Do as i can, not as i say: Grounding language in robotic affordances," *Conference on Robot Learning*, 2022.

[9] C. E. Mower, Y. Wan, H. Yu, A. Grosnit, J. Gonzalez-Billandon, M. Zimmer, J. Wang, X. Zhang, Y. Zhao, A. Zhai, P. Liu, D. Tateo, C. Cadena, M. Hutter, J. Peters, G. Tian, Y. Zhuang, K. Shao, X. Quan, J. Hao, J. Wang, and H. Bou-Ammar, "Ros-llm: A ros framework for embodied ai with task feedback and structured reasoning," *arXiv.org*, 2024.

[10] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, B. Ichter, T. Xiao, P. Xu, A. Zeng, T. Zhang, N. Heess, D. Sadigh, J. Tan, Y. Tassa, and F. Xia, "Language to rewards for robotic skill synthesis," 2023. [Online]. Available: https://arxiv.org/abs/2306.08647

[11] S. Patel, X. Yin, W. Huang, S. Garg, H. Nayyeri, F.-F. Li, S. Lazebnik, and Y. Li, "A real-to-sim-to-real approach to robotic manipulation with vlm-generated iterative keypoint rewards," *arXiv.org*, 2025.

[12] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *Conference on Robot Learning*, 2023.

[13] T. Wei, L. Ma, R. Chen, W. Zhao, and C. Liu, "Meta-control: Automatic model-based control synthesis for heterogeneous robot skills," *arXiv.org*, 2024.

[14] H. Kim, J. Kang, H. Kang, M. Cho, S. J. Kim, and Y. Lee, "Uniskill: Imitating human videos via cross-embodiment skill representations," 2025. [Online]. Available: https://arxiv.org/abs/2505.08787

[15] Z. Luo, Y. Yang, Y. Zhang, and F. Zheng, "Roboreflect: A robotic reflective reasoning framework for grasping ambiguous-condition objects," 2025. [Online]. Available: https://arxiv.org/abs/2501.09307

[16] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, B. Ichter, T. Xiao, P. Xu, A. Zeng, T. Zhang, N. Heess, D. Sadigh, J. Tan, Y. Tassa, and F. Xia, "Language to rewards for robotic skill synthesis," 2023. [Online]. Available: https://arxiv.org/abs/2306.08647

[17] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 3, pp. 7327–7334, 2022.

[18] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control*, 04 2011, pp. 43–66.

[19] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[20] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[21] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, 2nd ed. Cambridge University Press, 2022.

[22] M. Mittal, P. Roth, J. Tigue, A. Richard, O. Zhang, P. Du, A. Serrano-Muñoz, X. Yao, R. Zurbrügg, N. Rudin, L. Wawrzyniak, M. Rakhsha, A. Denzler, E. Heiden, A. Borovicka, O. Ahmed, I. Akinola, A. Anwar, M. T. Carlson, J. Y. Feng, A. Garg, R. Gasoto, L. Gulich, Y. Guo, M. Gussert, A. Hansen, M. Kulkarni, C. Li, W. Liu, V. Makoviychuk, G. Malczyk, H. Mazhar, M. Moghani, A. Murali, M. Noseworthy, A. Poddubny, N. Ratliff, W. Rehberg, C. Schwarke, R. Singh, J. L. Smith, B. Tang, R. Thaker, M. Trepte, K. V. Wyk, F. Yu, A. Millane, V. Ramasamy, R. Steiner, S. Subramanian, C. Volk, C. Chen, N. Jawale, A. V. Kuruttukulam, M. A. Lin, A. Mandlekar, K. Patzwaldt, J. Welsh, H. Zhao, F. Anes, J.-F. Lafleche, N. Moënne-Loccoz, S. Park, R. Stepinski, D. V. Gelder, C. Amevor, J. Carius, J. Chang, A. H. Chen, P. de Heras Ciechomski, G. Daviet, M. Mohajerani, J. von Muralt, V. Reutskyy, M. Sauter, S. Schirm, E. L. Shi, P. Terdiman, K. Vilella, T. Widmer, G. Yeoman, T. Chen, S. Grizan, C. Li, L. Li, C. Smith, R. Wiltz, K. Alexis, Y. Chang, D. Chu, L. J. Fan, F. Farshidian, A. Handa, S. Huang, M. Hutter, Y. Narang, S. Pouya, S. Sheng, Y. Zhu, M. Macklin, A. Moravanszky, P. Reist, Y. Guo, D. Hoeller, and G. State, "Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning," *arXiv preprint arXiv:2511.04831*, 2025. [Online]. Available: https://arxiv.org/abs/2511.04831

[23] Commonplace Robotics GmbH. (2025) Cri-python-lib. [Online]. Available: https://github.com/CommonplaceRobotics/CRI-Python-Lib

[24] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. D. Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull, "Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning," *IEEE International Conference on Robotics and Automation*, 2023.

[25] M. Klissarov, A. Bagaria, Z. Luo, G. Konidaris, D. Precup, and M. C. Machado, "Discovering temporal structure: An overview of hierarchical reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2506.14045

[26] R. Sutton and D. Precup, "Between mdps and semi-mdps: Learning, planning, and representing knowledge at multiple temporal scales," 08 1998.

[27] D. Ha and J. Schmidhuber, "World models," 2018. [Online]. Available: https://zenodo.org/record/1207631

[28] S. Kobayashi, Y. Schimpf, M. Schlegel, A. Steger, M. Wolczyk, J. von Oswald, N. Scherrer, K. Maile, G. Lajoie, B. A. Richards, R. A. Saurous, J. Manyika, B. A. y Arcas, A. Meulemans, and J. Sacramento, "Emergent temporal abstractions in autoregressive models enable hierarchical reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2512.20605

[29] J. Schmidhuber, "On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models," 2015. [Online]. Available: https://arxiv.org/abs/1511.09249

[30] W. Zhang, B. Terver, A. Zholus, S. Chitnis, H. Sutaria, M. Assran, R. Balestriero, A. Bar, A. Bardes, Y. LeCun, and N. Ballas, "Hierarchical planning with latent world models," 2026. [Online]. Available: https://arxiv.org/abs/2604.03208

[31] M. Xu, Z. Liu, P. Huang, W. Ding, Z. Cen, B. Li, and D. Zhao, "Trustworthy reinforcement learning against intrinsic vulnerabilities: Robustness, safety, and generalizability," 2022. [Online]. Available: https://arxiv.org/abs/2209.08025