

UCLA

UCLA Electronic Theses and Dissertations

Title

Embodiment-Aware Task Realization for Human—Robot Coexistence

Permalink

<https://escholarship.org/uc/item/98m050rb>

ISBN

9798190926316

Author

Wang, Weiqi

Publication Date

2026-09-10

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Embodiment-Aware Task Realization for
Human–Robot Coexistence

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mechanical Engineering

by

Weiqi Wang

2026

© Copyright by

Weiqi Wang

2026

ABSTRACT OF THE DISSERTATION

Embodiment-Aware Task Realization for Human–Robot Coexistence

by

Weiqi Wang

Doctor of Philosophy in Mechanical Engineering

University of California, Los Angeles, 2026

Professor Dennis W. Hong, Co-Chair

Professor Ying Nian Wu, Co-Chair

Robots in human environments face a physical gap between their embodiments and human-oriented spaces, and a knowledge gap between task descriptions and the information needed for execution. This dissertation examines these gaps through four complementary studies, with emphasis on human–robot co-activity rearrangement and Semantically Unified Programs (SUN).

The physical gap is approached by adapting robot motion to an existing environment or rearranging the environment itself. The Virtual Kinematic Chain (VKC) study includes the author’s sampling-based extension for coupled robot–object motion planning. Co-activity rearrangement treats human spatial organization and robot accessibility as joint design objectives. It combines semantic knowledge and learned spatial relations with robot-specific traversability, reachability, and interaction regions to optimize furniture layouts while retaining functional groups. Layouts account for robot footprint, reach, and intended activities. Across

154 SUNCG rooms sampled from a held-out partition, the method reports a 14 percent improvement in its robot-access measure and 30 percent more reachable objects.

The knowledge gap is examined at two scales. In tool use, the author’s initial trajectory-optimization formulation maps the desired physical effect of a single interaction to a robot-specific strategy. SUN addresses long-horizon, multistage manipulation by grounding language into executable, stage-wise geometric relations shared by transition predicates, model-predictive-control costs, and reinforcement-learning rewards. Model predictive control verifies programs before policy learning. Across five runs per task, repair increased program acceptance from 64.4 to 95.6 percent. The primary stage-conditioned policy with bounded residual learning reached 82.03 percent task-macro success, compared with 83.76 percent for the verification controller; three independent lineages averaged 79.43 percent.

The SUN controller produced 246.79 successful trajectory-minutes per GPU-hour, 10.57 times the human rate, and broke even near 513 demonstrations. With 500 trajectories per task, its DP3 policy reached 46.02 percent task-macro success. Zero-shot DP3 policies produced 36 successes in 106 physical trials (33.96 percent pooled success) and 34.72 percent task-macro success, with nonzero success on all tested tasks.

Together, these studies address physical compatibility and explicit task-knowledge representation as complementary requirements for robot operation: enabling action in shared environments and specifying the physical outcomes that action must achieve.

The dissertation of Weiqi Wang is approved.

Veronica J. Santos

Jacob Rosen

Ying Nian Wu, Committee Co-Chair

Dennis W. Hong, Committee Co-Chair

University of California, Los Angeles

2026

TABLE OF CONTENTS

1	Introduction	1
1.1	Robots in a Human World	1
1.2	Two Gaps in Embodiment-Aware Task Realization	2
1.2.1	The physical gap	2
1.2.2	The knowledge gap	3
1.3	Thesis and Research Questions	3
1.4	Contributions	4
1.4.1	Primary Contributions	4
1.4.2	Supporting Collaborative Foundations	5
1.5	Dissertation Organization	6
2	Background and Related Work	7
2.1	Part I: Human–Robot Co-Activity	8
2.1.1	Embodiment-aware motion planning as supporting context	8
2.1.2	Scene synthesis and activity-centered organization	9
2.1.3	Functional groups as a model of human preference	10
2.1.4	Robot accessibility in a fixed scene	11
2.1.5	Environment rearrangement for shared activity	12
2.2	Part II: SUN—From Language-Grounded Control to Learning	13
2.2.1	Executable task representations and temporal structure	13

2.2.2	Language grounding through skills and spatial constraints	13
2.2.3	Generated rewards and specification validity	14
2.2.4	Model-based validation and control-to-policy transfer	15
2.2.5	Autonomous data generation	16
2.2.6	Downstream policy learning and representation removal	17
2.2.7	Synthesis: what persists across the task lifecycle	17
3	Embodiment-Aware Motion Planning with Virtual Kinematic Chains . .	19
3.1	Why Base, Arm, and Object Motion Must Be Coupled	19
3.2	Kinematic Inversion and Chain Construction	20
3.3	Constrained Motion Planning	21
3.3.1	Motion-planning problem	21
3.3.2	Trajectory optimization	21
3.3.3	Sampling-based alternative	23
3.4	Evaluation	23
3.5	Boundary of the Representation	24
4	Scene Rearrangement for Human–Robot Co-Activity	26
4.1	When Planning Cannot Create Access	26
4.2	Related Work and the Co-Activity Gap	27
4.3	Co-Activity-Aware Problem Formulation	28
4.3.1	Weighted scene graph	28
4.3.2	Semantic and spatial terms	29

4.4	Robot Value: Reachable and Interaction Space	32
4.4.1	Reachable-space signed distance	32
4.4.2	Pseudo-interaction function	33
4.5	Human and Robot Objectives	35
4.6	Sampling and Hierarchical Optimization	37
4.7	Evaluation	38
4.7.1	Dataset split and quantitative measures	38
4.7.2	Qualitative and ablation results	38
4.7.3	Human evaluation	42
4.7.4	Embodiment sensitivity	42
4.8	Limitations	43
5	Physical-Effect-Centered Task Understanding for Tool Use	45
5.1	From Motion Reproduction to Effect Reproduction	45
5.2	Physical-Effect Reconstruction	46
5.3	Essential-Relation Learning	47
5.4	Robot-Specific Optimal Control	49
5.5	Evaluation	50
5.6	Scope and Limitations	51
6	Formation of Persistent SUN Programs	52
6.1	Task Formation as the Remaining Bottleneck	52
6.2	Positioning: What the Program Must Preserve	53

6.3	SUN Program Representation	54
6.4	Typed Constraint Vocabulary and Semantics	58
6.4.1	Common Frame and Cost Conventions	61
6.4.2	Translation Constraints	62
6.4.3	Orientation Constraints	63
6.4.4	Entry Invariants and Gripper Constraints	65
6.4.5	Condition Labels and Valid Term Combinations	67
6.4.6	Worked Drawer-Stage Compilation	68
6.5	Language and Scene Grounding	68
6.6	Physical Verification and Diagnostic Repair	70
6.7	Formation Reliability	72
6.8	Scope and Limitations	74
7	Optimal Control as Mentor for Program-Preserving Policy Learning . .	75
7.1	From Verified Control to Amortized Execution	75
7.2	MPC Mentor Demonstrations	76
7.3	Stage-Conditioned Behavioral Prior	76
7.4	Bounded Residual Reinforcement Learning	77
7.5	Experimental Protocol	79
7.6	Results	80
7.7	Discussion and Limits	82
8	Program-Guided Data Generation and Physical Deployment	84

8.1	From Task Policy to Data Engine	84
8.2	Outcome and Trajectory-Quality Definitions	85
8.3	Data Utility, Quality, and Throughput	86
8.4	Visual-Policy Distillation	87
8.5	Zero-Shot Physical Deployment	90
8.6	Discussion and Limitations	91
9	Discussion and Conclusion	92
9.1	Toward Fit-Centered Robot Autonomy	92
9.2	Cross-Chapter Synthesis	93
9.3	Limitations Across the Dissertation	94
9.4	Two Missing Links	95
9.5	Future Directions	95
9.6	Conclusion	96
A	Shared System Conventions	97
B	SUN Program Language and Task Library	98
B.1	Type System and Term Signatures	98
B.2	Compilation and Transition Invariants	98
B.3	Lifecycle Procedure	99
B.4	Canonical Task Catalog	100
B.5	Reported and Unreported Reproducibility Fields	101
B.6	SUN Prompt Specifications	101

B.6.1	Receiver-side tolerance assignment	104
B.6.2	Program-formation prompts	105
B.6.3	Learning and calibration prompts	123
C	Extended Experiments	132
C.1	Ablations, Robustness, and Compute	132
D	Real-Robot Systems	134
D.1	System Configuration, Calibration, and Trials	134
E	Scene-Rearrangement Supplement	135
E.1	Published and Defense Objectives	135
E.2	Task and Embodiment Sensitivity	135
E.3	Human-Study Reanalysis	136
E.4	Held-Out Quantitative Provenance	137
E.5	Reproducibility Boundary	137
	References	139

LIST OF FIGURES

3.1	VKC construction for articulated-object manipulation	22
3.2	Constrained drawer-opening evaluation	24
4.1	From a human-activity room to a co-activity room	27
4.2	Scene representation and co-activity optimization	30
4.3	Robot-access changes over the held-out SUNCG scenes	39
4.4	Qualitative scene-rearrangement results	40
4.5	Task-conditioned office rearrangement	41
4.6	Embodiment-dependent scene rearrangement	43
5.1	Tool bases and robot-tool VKC	47
5.2	From demonstrated effect to a robot-specific strategy	48
6.1	Persistent task knowledge through the KUAFU lifecycle	53
6.2	Unified SUN Program formation and policy learning	71
6.3	Nine-task SUN evaluation suite	73
7.1	Control-to-learning results and ablations	81
8.1	SUN data throughput and startup break-even	88

LIST OF TABLES

5.1	Walnut-cracking success in simulation	50
6.1	Positioning task representations	55
6.2	Prompt-visible SUN constraint operations	60
6.3	Compilation trace for the drawer-opening example	69
6.4	Canonical SUN Program size and formation cost	73
6.5	Repeated SUN formation and verification	74
7.1	Accepted-program control-to-learning performance	81
7.2	End-to-end controller compute	82
8.1	Fixed-budget downstream DP3 success	86
8.2	MPC-relative joint Q95 yield and successful-trajectory diagnostics	87
8.3	Visual-policy architecture results from the same KUAFU data	89
8.4	Zero-shot DP3 deployment on physical robots	90
9.1	Primary and supporting interventions for coexisting robot competence	93
B.1	Canonical nine-task SUN library	101
B.2	Reproducibility boundary of the active SUN manuscript	102
B.3	Prompt inventory for SUN formation and learning	103
B.4	System context for initial task decomposition	106
B.5	User template for initial task decomposition	108

B.6	System context for decomposition correction	108
B.7	User template for decomposition correction	110
B.8	System context for atomic-stage grounding	111
B.9	User template for atomic-stage grounding	114
B.10	System context for typed relation writing	115
B.11	User template for typed relation writing	115
B.12	System context for typed-relation correction	119
B.13	User template for typed-relation correction	119
B.14	System prompt for initial reward-weight selection	124
B.15	User template for initial reward-weight selection	125
B.16	System prompt for BC-prior-informed weight adjustment	126
B.17	User template for BC-prior-informed weight adjustment	127
B.18	System prompt for residual-authority calibration	128
B.19	User template for residual-authority calibration	131
C.1	Complete task-level ablation and baseline success	133
E.1	Paired reanalysis of the co-activity human study	136

PREFACE AND AUTHOR CONTRIBUTIONS

Four collaborative works underpin this dissertation. Its two primary bodies are the scene-rearrangement study in Chapter 4 and the SUN study in Chapters 6–8. The Virtual Kinematic Chain (VKC) material in Chapter 3 and the tool-use material in Chapter 5 provide supporting foundations and are limited to Weiqi Wang’s contributions. Chapters 1, 2, and 9 provide the framing, literature synthesis, and cross-chapter conclusions.

Figures reproduced or adapted from the IEEE publications listed below are identified individually in their captions and carry the corresponding IEEE copyright notice.

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of California, Los Angeles’s products or services. Internal or personal use of this material is permitted.

If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Chapter 3: Virtual Kinematic Chains. Chapter 3 is a modified and abbreviated version of material from:

Ziyuan Jiao, Zeyu Zhang, Weiqi Wang, David Han, Song-Chun Zhu, Yixin Zhu, and Hangxin Liu. “Efficient Task Planning for Mobile Manipulation: A Virtual Kinematic Chain Perspective.” In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8288–8294, 2021. doi:10.1109/IROS51168.2021.9636554. © 2021 IEEE.

Ziyuan Jiao and Zeyu Zhang, who were designated equal contributors in the publication, jointly designed the VKC representation and its fundamental functionality and implemented the trajectory-optimization version of the software. Ziyuan Jiao additionally designed most of the experiments. Weiqi Wang assisted with the design of the VKC and its experiments and implemented the sampling-based method for using the VKC, extending a system that

previously supported trajectory optimization. Chapter 3 excludes the symbolic-planning portion of the larger project and retains the VKC construction, constrained motion formulation, sampling-based extension, and motion-level evaluation relevant to Wang’s contribution. David Han, Song-Chun Zhu, Yixin Zhu, and Hangxin Liu provided technical consultation and project guidance; they did not participate in the software implementation or experimental execution reported in this chapter.

Chapter 4: Human–robot co-activity. Chapter 4 is an expanded version of material from:

WeiQi Wang, Zihang Zhao, Ziyuan Jiao, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. “Rearrange Indoor Scenes for Human–Robot Co-Activity.” In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11943–11949, 2023. doi:10.1109/ICRA48891.2023.10160303. © 2023 IEEE.

WeiQi Wang, Zihang Zhao, and Ziyuan Jiao were designated equal contributors in the publication. Wang led work across the full project: conceptualization and method development; complete implementation of the scene-rearrangement optimizer and experimental pipeline; experimental design and execution; analysis and visualization; and writing. Zihang Zhao contributed expertise in ConceptNet and supplied the ConceptNet data used to construct the semantic relations. Ziyuan Jiao mentored the project and contributed to its ideas and writing, but did not contribute to the implementation. The dissertation expands the published account with implementation details and an audit of the archived evaluation artifacts while labeling those additions separately from the published formulation. Yixin Zhu, Song-Chun Zhu, and Hangxin Liu provided technical consultation and project guidance; they did not participate in the software implementation or experimental execution.

Chapter 5: Physical-effect-centered tool use. Chapter 5 is a modified and abbreviated version of material from:

Zeyu Zhang, Ziyuan Jiao, Weiqi Wang, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. “Understanding Physical Effects for Effective Tool-Use.” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9469–9476, 2022. doi:10.1109/LRA.2022.3191793. © 2022 IEEE.

Zeyu Zhang and Ziyuan Jiao were designated equal contributors in the publication. Zhang led the project, formulated the overall tool-use problem, and developed the learning and finite-element components. Weiqi Wang formulated the initial trajectory-optimization problem and verified the method in MATLAB. Jiao subsequently incorporated that optimization into the VKC framework for a faster implementation and developed an associated real-robot demonstration. Chapter 5 retains the physical-effect interface and robot-specific optimization material needed to explain Wang’s initial method; the broader learning and finite-element developments are summarized rather than claimed as Wang’s individual contribution. Yixin Zhu, Song-Chun Zhu, and Hangxin Liu provided technical consultation and project guidance; they did not participate in the software implementation or experimental execution.

Chapters 6–8: Persistent SUN programs. Chapters 6–8 are based on:

Weiqi Wang, Zhi Li, Yudong Lei, David Martinez, Xiaofeng Gao, Yuxin Jiang, Chenfanfu Jiang, Yingnian Wu, Demetri Terzopoulos, and Ran Gong. “SUN: Persistent Programs for Language-Grounded Control-to-Learning-to-Real Policies.” Manuscript in preparation, 2026.

Weiqi Wang and Zhi Li are designated equal contributors in the manuscript. Wang led and carried out the overall project, including its conceptualization, method development, software implementation, experiments, analysis, visualization, and manuscript preparation. Zhi Li assisted with hardware acquisition, contributed ideas during development, and contributed to writing. Yudong Lei supported visual-policy training. David Martinez provided hardware support and machining for the physical system. Xiaofeng Gao, Yuxin Jiang, Chenfanfu Jiang, Yingnian Wu, Demetri Terzopoulos, and Ran Gong provided technical consultation and

project guidance; they did not participate in the software implementation or experimental execution. Chapters 6–8 reorganize this work around program formation, transfer from optimal control to policy learning, autonomous data generation, and physical deployment.

ACKNOWLEDGMENTS

I thank my committee members Dennis W. Hong, Ying Nian Wu, Jacob Rosen, and Veronica J. Santos for their guidance, close attention, and constructive feedback.

For the Virtual Kinematic Chain study, I thank Ziyuan Jiao and Zeyu Zhang for their close technical collaboration; David Han, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu for technical consultation and project guidance; and Jiang Xin for discussions on trajectory optimization.

For the tool-use study, I thank Zeyu Zhang and Ziyuan Jiao for their close technical collaboration; Yixin Zhu, Song-Chun Zhu, and Hangxin Liu for technical consultation and project guidance; and Minchen Li and Chenfanfu Jiang for the simulation setup.

For the human-robot co-activity study, I thank Zihang Zhao for his ConceptNet expertise and for providing the ConceptNet data, and Ziyuan Jiao for mentorship, ideas, and writing support. I thank Yixin Zhu, Song-Chun Zhu, and Hangxin Liu for technical consultation and project guidance.

For the SUN study, I thank Zhi Li for assistance with hardware acquisition, ideas during development, and writing; Yudong Lei for visual-policy training support; and David Martinez for hardware support and machining. I thank Xiaofeng Gao, Yuxin Jiang, Chenfanfu Jiang, Ying Nian Wu, Demetri Terzopoulos, and Ran Gong for technical consultation and project guidance.

I am grateful to my colleagues, friends, and family for sustaining me throughout the dissertation.

The work incorporated into Chapter 3 was supported by ONR N00014-19-1-2153, ONR MURI N00014-16-1-2007, and DARPA XAI N66001-17-2-4029. The work incorporated into Chapter 4 was supported in part by the National Key R&D Program of China,

grant 2022ZD0114900; the Beijing Municipal Science & Technology Commission, grant Z221100003422004; and the Beijing Nova Program.

VITA

May 2015	Bachelor of Science in Mechanical Engineering, Minor in Mathematics, University of Miami.
May 2017	Master of Science in Mechanical Engineering: Robotics, Johns Hopkins University.
2017	Research in the Laboratory for Computational Sensing and Robotics, Johns Hopkins University, Baltimore, Maryland (June–September).
2017–2019	Algorithm Engineer, CloudMinds/INNFOSS Technology Ltd., Beijing, China.
2020–2026	Graduate Student Researcher, Center for Vision, Cognition, Learning, and Autonomy, University of California, Los Angeles.
2020–2025	Instructor and Teaching Fellow, Mathematics for Life Scientists, University of California, Los Angeles (twelve quarters). Nominated for UCLA’s university-wide Distinguished Teaching Award for Teaching Assistants.
2021, 2023	Software Intern, OffWorld, Inc., Pasadena, California (June–September in each year).

PUBLICATIONS

Ehsan Zahedi, Fariba Khosravian, Weiqi Wang, Mehran Armand, Javad Dargahi, and Mehrdad Zadeh. “Towards Skill Transfer via Learning-Based Guidance in Human–Robot Interaction: An Application to Orthopaedic Surgical Drilling Skill.” *Journal of Intelligent & Robotic Systems*, vol. 98, no. 3–4, pp. 667–678, 2020.

Ziyuan Jiao, Zeyu Zhang, Weiqi Wang, David Han, Song-Chun Zhu, Yixin Zhu, and Hangxin Liu. “Efficient Task Planning for Mobile Manipulation: A Virtual Kinematic Chain Perspective.” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021.

Zeyu Zhang, Ziyuan Jiao, Weiqi Wang, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. “Understanding Physical Effects for Effective Tool-Use.” *IEEE Robotics and Automation Letters*, 2022.

Weiqi Wang, Zihang Zhao, Ziyuan Jiao, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. “Rearrange Indoor Scenes for Human–Robot Co-Activity.” *Proceedings of the IEEE International Conference on Robotics and Automation*, 2023.

Weiqi Wang, Zhi Li, Yudong Lei, David Martinez, Xiaofeng Gao, Yuxin Jiang, Chenfanfu Jiang, Yingnian Wu, Demetri Terzopoulos, and Ran Gong. “SUN: Persistent Programs for Language-Grounded Control-to-Learning-to-Real Policies.” *Manuscript in preparation*, 2026.

CHAPTER 1

Introduction

1.1 Robots in a Human World

Robots already perform work that is computationally complex, physically demanding, or highly repetitive. Their strongest deployments nevertheless tend to occur in environments arranged around the machine: objects arrive in known configurations, the workcell is structured, and the procedure is explicitly defined. A coexisting service robot faces a different setting. It must act in rooms arranged around people, among objects whose placement reflects human ergonomics and human habits (Merrell et al., 2011; Qi et al., 2018; Zhi et al., 2021). It must also interpret instructions that usually name an outcome rather than every action required to produce it (Ichter et al., 2023; Huang et al., 2023, 2025).

The instruction “put this in the drawer” illustrates the difference. The sentence does not specify a navigation plan, a collision-free approach, the constrained motion needed to open the drawer, the pick-and-place action, or the motion needed to close the drawer. A person supplies this missing structure from experience. For the robot, even a short household instruction can therefore hide a long sequence of geometric and physical decisions (Garrett et al., 2021; Toussaint, 2015).

Ordinary manipulation also hides an environmental infrastructure. Reaching a book, loading a dishwasher, inserting a paintbrush, and placing an object in a drawer appear simple because rooms and products are designed around human bodies (Qi et al., 2018). The

same tasks may require high, low, or deep reach; precise alignment; or several dependent manipulation stages. People do not consciously derive each constraint because they draw on extensive experience with their bodies, tools, and similar activities (Gallagher, 2005; Holmes and Spence, 2005). A coexisting robot can lack both the spatial compatibility and the accumulated task knowledge that make these routines ordinary for people.

1.2 Two Gaps in Embodiment-Aware Task Realization

This dissertation organizes the problem around two gaps.

1.2.1 The physical gap

Useful manipulation demands payload capacity, reach, and stability. Those requirements often produce a body that is larger than the small, agile bodies that move easily through crowded human environments. Better planning can use available clearance more efficiently, but planning cannot create clearance that does not exist, provide an absent degree of mobility, or extend a workspace beyond the robot’s physical reach. The *physical gap* is therefore the mismatch between a robot’s body and the human environment in which the task must be carried out (Zhi et al., 2021; Wang et al., 2023).

The physical gap can be approached from two directions. The robot can adapt its motion to the room by coordinating its entire body, or the room can be adapted to admit the robot while preserving the functions that matter to people. The first direction motivates the Virtual Kinematic Chain in chapter 3 (Shankar, 2016; Jiao et al., 2021). The second motivates the human–robot co-activity scene optimization in chapter 4 (Zhang et al., 2021; Wang et al., 2023).

1.2.2 The knowledge gap

Physical access is necessary but not sufficient. A robot that can reach every object must still know what physical effect the task requires and how progress through a multistage routine should be judged. Humans accumulate broad experience with their bodies and with physical interaction (Gallagher, 2005; Holmes and Spence, 2005). That experience connects words, objects, actions, and outcomes. Robots generally require demonstrations, carefully designed objectives, and substantial task-specific training in place of that experience (Mandlekar et al., 2023; Ma et al., 2024b; Carius et al., 2020).

The *knowledge gap* is the missing task knowledge between a brief instruction or demonstration and a complete, executable routine. This dissertation studies that gap at two scales. At the scale of a single dynamic interaction, chapter 5 reconstructs the physical effect of a human tool-use event rather than copying embodiment-specific motion (Liu et al., 2019; Zhang et al., 2022b). At the scale of a long-horizon routine, chapter 6–8 construct a Semantically Unified Program (SUN) whose meaning persists through control, learning, data generation, and deployment (Wang et al., 2026).

1.3 Thesis and Research Questions

The central claim is that a robot becomes a useful participant in human activities only when two conditions hold. First, its body and the environment must provide a physical opportunity to act. Second, the intended task must remain consistently defined as behavior moves from instruction to control, learning, and deployment. Physical compatibility without task meaning leaves the robot able to move but unable to complete the routine. Task meaning without physical compatibility leaves a well-specified objective that cannot be realized.

This claim leads to two primary research questions:

1. How can a furnished human environment be rearranged to improve robot access without destroying the semantic and spatial relations that support human activity?
2. How can a language-grounded task structure be verified for physical credibility and then retained through policy learning, autonomous data production, and physical deployment?

Two earlier collaborative studies provide narrower technical foundations for these questions. The VKC study asks how the mobile base, manipulator, grasp, and manipulated object can be represented as one coordinated system; within that project, the dissertation author contributed the sampling-based use of the VKC and assisted with design and evaluation. The tool-use study asks which physical effect should replace embodiment-specific demonstrated motion; within that project, the author formulated the initial robot-specific trajectory optimization and verified it in MATLAB. These supporting studies motivate the representational principles used by the two primary works, but they are not presented as equal dissertation contributions.

1.4 Contributions

The dissertation makes two primary contributions and documents the author’s work within two supporting collaborations. This hierarchy reflects both the dissertation’s emphasis and the scope of the author’s research contribution.

1.4.1 Primary Contributions

Human–robot co-activity environments. The scene-rearrangement method models human value through semantic relations, spatial relations, and functional object groups. It models robot value through reachable space and task-relevant interaction space. A combined

optimization then searches for a layout that improves robot access without treating the room as an empty geometric container or discarding its human organization (Wang et al., 2023).

Persistent task formation. SUN grounds a language instruction into a staged collection of geometric relations. Each relation has a shared residual and tolerance from which the system compiles a transition predicate, a model-predictive-control cost, and a reinforcement-learning term. Model predictive control verifies a candidate before learning; accepted behavior becomes a stage-conditioned prior; bounded residual reinforcement learning repairs execution errors; and the resulting controller generates data for visual policies and physical deployment. The task is not redefined at each handoff (Wang et al., 2026).

1.4.2 Supporting Collaborative Foundations

Sampling-based use of Virtual Kinematic Chains. The collaborative VKC project consolidates the mobile base, manipulator, grasp, and manipulated object into one kinematic chain so that base and arm motion can be coordinated. Within that project, the dissertation author assisted with the design of the VKC and its experiments and implemented the sampling-based method for using the representation, extending the trajectory-optimization-only implementation (Jiao et al., 2021). Chapter 3 retains the technical context and evaluation needed to explain that scoped contribution.

Initial optimization for effect-centered tool use. The collaborative tool-use project treats the intended physical effect, rather than the human trajectory, as the transferable task specification. The dissertation author’s contribution was the initial robot-specific trajectory optimization formulation and its verification in MATLAB. The broader project subsequently incorporated that optimization into VKC and combined it with finite-element reconstruction

and learning (Zhang et al., 2022b). Chapter 5 summarizes those surrounding components only to explain the interface and scope of the initial optimization method.

1.5 Dissertation Organization

Chapter 2 establishes the related representations and learning pipelines. Chapters 3 and 4 address the physical gap by adapting the robot representation and the environment, respectively. Chapter 5 begins the knowledge-gap discussion with a single dynamic tool interaction. Chapter 6 defines and verifies SUN programs; chapter 7 carries an accepted program into policy learning; and chapter 8 evaluates the resulting data engine, visual policies, and physical transfer. Chapter 9 returns to the two gaps, identifies the missing links of end-to-end integration and functional grounding, and summarizes the design principles that connect the two primary contributions with their supporting foundations.

CHAPTER 2

Background and Related Work

This chapter is divided into two parts because the dissertation’s two central works intervene at different levels of a robot system. Part I concerns the environment in which activity occurs. It reviews how indoor scenes encode human use, why a layout suitable for a person can exclude a mobile manipulator, and how scene organization can preserve human function while creating robot access. Part II concerns the task representation that directs activity. It follows SUN’s language-to-control-to-learning pipeline, asking how task meaning can be grounded, physically validated, transferred into a policy, and retained during data generation and deployment.

The earlier Virtual Kinematic Chain (VKC) and physical-effects studies appear only as supporting context in Part I. They establish two embodiment-related ideas used by the co-activity work: coordinated motion depends on a coupled representation of the robot and manipulated object, and successful interaction should be defined by the physical outcome rather than by copying a body-specific trajectory. The technical formulations of the two primary works and the relevant parts of the supporting studies remain in their respective chapters; the purpose here is to locate the two main works within the literature and make the gaps they address explicit.

2.1 Part I: Human–Robot Co-Activity

2.1.1 Embodiment-aware motion planning as supporting context

For mobile manipulators, whole-body planners address coupled base and arm motion in tasks such as reaching and opening doors (Shankar, 2016; Bodily et al., 2017; Chitta et al., 2010). These approaches show why navigation and manipulation cannot always be solved as independent episodes: a base pose that is reachable for navigation may leave the arm unable to complete the interaction. This limitation is especially visible when contact with a door or drawer constrains the end effector while the object moves through a large range.

The VKC work of Jiao et al. (2021) addresses this representation issue by placing the relevant base, arm, grasp, and object coordinates in one kinematic chain. Its virtual-mechanism lineage includes virtual model control for bipedal locomotion (Pratt et al., 1997, 2001), virtual chains for dual-arm manipulation (Likar et al., 2014), whole-body control of a dual-arm mobile robot (Wang et al., 2016), and Cartesian control of a wheeled-legged robot (Laurenzi et al., 2020). In chapter 3, the object chain is inverted and joined to the robot at the grasp, allowing a constrained-motion solver to coordinate locomotion and manipulation directly.

The physical-effects work supplies a second, narrower point about embodiment. A demonstrated tool trajectory contains the human demonstrator’s grasp, strength, reach, and ergonomics. Those properties need not be task requirements for a robot with a different arm or gripper. The formulation in chapter 5 instead treats the target object’s physical effect—through quantities such as force, pressure, energy, or fracture—as the invariant outcome and uses simulation to relate it to actionable variables such as impact velocity. Both supporting studies therefore separate task-relevant structure from an unnecessarily restrictive embodiment-specific decomposition.

2.1.2 Scene synthesis and activity-centered organization

Indoor scenes are organized for use, not merely for collision avoidance. Object selection, relative position, orientation, grouping, and the free space around objects all carry information about likely human activities. Early scene-synthesis systems encode these regularities through interior-design guidelines (Merrell et al., 2011) or positive examples of object arrangements (Fisher et al., 2012). Learning-based methods infer object-object relations (Yu et al., 2011; Kermani et al., 2016), connect human activities to object configurations (Qi et al., 2018; Jiang et al., 2018), mine topological relations among object groups (Keshavarzi et al., 2020), or learn latent layout distributions (Ritchie et al., 2019; Zhang et al., 2020; Nauata et al., 2021).

The representations range from explicit structure to learned distributions. PlanIT constructs a semantic relation graph and grounds its edges into object locations (Wang et al., 2019); ATISS instead generates an unordered set of furniture conditioned on the floor plan (Paschalidou et al., 2021). The human-centric grammar of Qi et al. (2018) makes activities, likely human poses, object groups, and affordance maps part of the scene description. Across these approaches, the training examples or design rules primarily describe rooms arranged for people. As the co-activity paper observes, their learned layout statistics therefore do not automatically supply the footprint, clearance, reach, or approach-direction statistics of a service robot.

This distinction matters because visual plausibility and robotic usability are different criteria. A generated bedroom can look natural and preserve familiar object relations while leaving a passage narrower than a mobile base or placing the usable side of a cabinet against an obstacle. The co-activity problem is thus not simply to synthesize another human-like room. It is to retain the human-centered organization learned by scene-synthesis methods while adding a second occupant whose morphology and interaction modes differ substantially.

2.1.3 Functional groups as a model of human preference

The co-activity paper models human preference through functional groups: collections such as a bed and nightstand or a table and chair whose relation supports an activity. Scene grammars and holistic scene models provide prior examples of representing functional or interaction structure explicitly (Zhao and Zhu, 2011; Chen et al., 2019). Other scene-synthesis methods recover such structure from spatial co-occurrence: object categories that repeatedly appear together and at characteristic distances are treated as related (Fisher et al., 2012; Yu et al., 2011; Wang et al., 2019; Zhang et al., 2022a).

Spatial evidence is useful but ambiguous in clutter. Two pieces of furniture can be close because both fit into a crowded room rather than because they support the same activity. Conversely, two semantically similar category names do not imply that their instances form a useful pair. The co-activity work therefore combines spatial statistics from indoor layouts with semantic association from ConceptNet (Speer et al., 2017). Semantic evidence helps distinguish a bed–nightstand relation from an incidental bed–chair proximity, while spatial evidence prevents category similarity alone from defining a group.

This literature also clarifies what should survive rearrangement. Preserving every original object pose would leave no useful design freedom, whereas discarding relations would allow an optimizer to produce open but dysfunctional rooms. Functional grouping supplies an intermediate representation: related objects should remain arranged together according to plausible relative-pose statistics, but the group and its members may move enough to create access. The weighted scene-graph construction and its equations are developed in chapter 4; here, the important related-work distinction is between preserving activity relations and preserving a single observed layout.

2.1.4 Robot accessibility in a fixed scene

Robot-planning research addresses several difficulties created by fixed indoor layouts. Mobile-manipulation methods plan object retrieval in confined or cluttered environments (Berenson et al., 2008; Cheong et al., 2020; Han et al., 2020), and whole-body methods coordinate the base and arm (Shankar, 2016; Bodily et al., 2017; Chitta et al., 2010). Such methods improve the use of space that already exists. They cannot, however, create clearance when a passage is narrower than the robot or when no valid base-and-arm configuration reaches the usable side of an object.

The co-activity paper consequently treats robot preference as accessible space with two components: the open region the robot can traverse and the objects it can reach from that region. The second component prevents free floor area from becoming an incomplete surrogate for activity. A cabinet may contribute no useful interaction when its door faces a wall even if the surrounding room is otherwise open. Interaction therefore depends on oriented approach regions as well as distance and collision clearance. This view is related to work that models unobserved interaction structure or the *dark matter* around agents and objects (Xie et al., 2018; Zhu et al., 2020).

Separating traversal from interaction also makes embodiment dependence explicit. The traversable region changes with the mobile-base footprint and its safety margin; the set of interactable objects changes with arm reach and with which object sides afford use. A layout is therefore not universally accessible. It is accessible relative to a particular robot body and activity model. Chapter 4 implements these quantities using signed distance fields and directional pseudo-interaction functions, but those technical choices instantiate the broader distinction established here.

2.1.5 Environment rearrangement for shared activity

Computational rearrangement has optimized environments for individual classes of users or objectives. Hospital-room layouts have been rearranged to reduce patient fall risk (Chaeibakhsh et al., 2021); functional workspace optimization has learned personal preferences from virtual experience (Liang et al., 2019); and robot-oriented design has considered navigation and access. Zhi et al. (2021) optimize furniture layout against user-defined human preferences together with nonholonomic wheelchair navigation and access constraints. Zhang et al. (2021) jointly optimize equipment placement and human workplans using completion time, congestion, walking and turning effort, and workload balance. More recently, Zhi and Lien (2025) combine architectural constraints with decentralized human-robot motion planning to evaluate layouts for collaborative activities.

These systems demonstrate that activity and layout should be designed together, but they also expose a modeling choice: whose activity defines a good room. Statistics from human-designed environments describe human preference but lack robot-specific access statistics; maximizing open area or navigation efficiency for a robot can in turn destroy human functional relations. The contribution positioned in chapter 4 represents those unlike values separately and optimizes their compromise. Human value is carried by functional relations supported jointly by semantic and spatial evidence. Robot value is carried by traversable and interaction space. This is the specific gap meant by *human-robot co-activity*: neither occupant’s accessibility is assumed to be a sufficient proxy for the other’s.

2.2 Part II: SUN—From Language-Grounded Control to Learning

2.2.1 Executable task representations and temporal structure

Long-horizon manipulation requires a representation that connects discrete task progression to continuous motion. Language-based program generation provides one interface: ProgPrompt produces situated robot task plans in program form, using executable structure to organize sequences of actions (Singh et al., 2023). In learning, reward machines express high-level task specifications as automata whose states provide reward structure and decomposition (Toro Icarte et al., 2018); signal temporal logic has likewise been incorporated into neural predictive control to express temporally extended constraints (Meng and Fan, 2023).

These representations solve different portions of the same problem. Program structure records what should happen next; geometric objectives make a step operational; automata or temporal constraints define progress and completion. These artifacts are usually designed for one stage of the pipeline—planning, control, or learning. A generated program step may still be physically infeasible, and a reward-state transition may not carry the same geometric tolerance used by a controller. The SUN work asks what additional contract is needed when one task representation must survive all of these interfaces.

2.2.2 Language grounding through skills and spatial constraints

Language-grounded systems connect an instruction to action at several levels. SayCan selects among pretrained skills by combining language-model relevance with learned affordance values that estimate whether a skill can succeed in the current state (Ichter et al., 2023). VoxPoser uses language and vision-language models to compose three-dimensional affordance and constraint value maps, which model-based control converts into motion (Huang et al., 2023). ReKep represents multistage manipulation as a sequence of relational keypoint

constraints optimized in closed loop (Huang et al., 2025), while OmniManip maps object-centric interaction primitives to spatial constraints and uses perception to track the resulting targets (Pan et al., 2025).

Together, these methods move from semantic proposals to physically situated objectives without requiring every task to be programmed from scratch. Their representations differ—skill scores, value maps, keypoint relations, and object-centric constraints—but they primarily support selection or optimization during online execution. When a learned policy later replaces the online procedure, the original grounded representation is not necessarily retained as the stage supervisor, completion monitor, or source of diagnostic terms. SUN’s concern is therefore not only whether language can produce an actionable objective, but whether the accepted objective remains identifiable after control is amortized.

2.2.3 Generated rewards and specification validity

Language models can instead enter through the learning objective. Language to Rewards maps task descriptions to reward parameters for robotic skill synthesis (Yu et al., 2023). Text2Reward generates executable dense-reward code from a description and an environment abstraction (Xie et al., 2024). Eureka searches over language-model-generated reward programs using policy-training feedback (Ma et al., 2024b), and DrEureka extends language-guided reward and domain-randomization design toward sim-to-real transfer (Ma et al., 2024a). For multistage behavior, DrS uses stage structure to learn reusable dense rewards (Mu et al., 2024).

This family reduces manual reward engineering, but it also couples specification search to policy optimization. Each reward revision may require another reinforcement-learning run before its consequences become visible. More fundamentally, high return under a proposed reward is not evidence that the reward describes the intended task: trial-and-error reward

design can overfit observed behavior or encode an invalid task specification (Booth et al., 2023). The distinction is central to SUN. A task proposal should first be checked for executable physical behavior and externally judged success; only an accepted proposal should become learning supervision.

2.2.4 Model-based validation and control-to-policy transfer

Model predictive control repeatedly solves a finite-horizon optimization under modeled dynamics, objectives, and constraints (Mayne et al., 2000). It therefore provides more than a trajectory generator in the SUN pipeline: closed-loop rollouts can expose incompatible spatial relations, unreachable stages, or premature completion before policy training. This use differs from evaluating a generated reward only through the eventual return of a learned agent. The controller serves as a comparatively direct physical acceptance test for the grounded task proposal.

Several lines of work transfer optimized control into faster policies. Guided Policy Search uses trajectory-optimization results as supervised policy targets (Levine and Koltun, 2013). PLATO adapts an MPC teacher to the state distribution induced by the learner (Kahn et al., 2017). MPC-Net trains a policy with a Hamiltonian loss that carries information from the original control objective and constraints (Carius et al., 2020). Residual reinforcement learning retains a nominal controller while learning corrective actions (Johannink et al., 2019); model-agnostic refinement methods such as Policy Decorator similarly add a learned correction to a prior policy (Yuan et al., 2025).

These approaches address distribution shift, execution speed, or corrective adaptation, but the transferred artifact is usually behavior, a control prior, or a stage-local objective. The task representation itself need not remain available to govern ordered transitions, apply the same satisfaction boundary, or attribute a failed rollout to a particular relation. The

SUN formulation in chapter 7 combines controller demonstrations with the accepted program: the behavior supplies a prior, while stage identity, progress terms, guards, and diagnostics continue to come from the persistent task representation.

2.2.5 Autonomous data generation

Robot-data generators reduce the number of new demonstrations required for each task. MimicGen retargets object-relative segments from a small set of successful human demonstrations across new scene configurations (Mandlekar et al., 2023). SkillGen decomposes demonstrations into reusable skills and connects their deployment through planned motions (Garrett et al., 2025). Both expand coverage efficiently, but the task’s semantics and success evidence originate in the seed demonstrations.

Generative systems move farther upstream. RoboGen uses generative simulation to propose tasks and scenes and to create training supervision through planning and reinforcement learning (Wang et al., 2024). GenSim2 uses multimodal and reasoning language models to scale simulation task and data generation (Hua et al., 2025). These systems show that foundation models and simulation can automate substantial portions of task construction and experience generation. The remaining question for SUN is how the generated experience is certified against a stable task definition. Its proposed answer is to generate data only after the task program passes controller-based and external-success checks, then reuse the same stage and relation identities to monitor collection.

This distinction separates demonstration amplification from task formation. The former begins with successful behavior and produces variants; the latter must turn an instruction into a physically meaningful object before any trusted behavior exists. Chapter 8 evaluates the accepted controller in both roles: as a source of successful trajectories and as a high-throughput data engine whose output can train policies that no longer receive privileged

controller state.

2.2.6 Downstream policy learning and representation removal

Once accepted behavior has been collected, multiple policy families can learn from it. ACT reduces effective temporal burden through action chunking (Zhao et al., 2023). Diffusion Policy represents visuomotor action generation as conditional denoising (Chi et al., 2025), while DP3 uses point-cloud observations in a three-dimensional diffusion-policy design (Ze et al., 2024). ManiFlow and FlowPolicy provide flow-based alternatives (Yan et al., 2025; Zhang et al., 2025b); SmolVLA and $\pi_{0.5}$ represent vision-language-action policies (Shukor et al., 2025; Black et al., 2025).

These learners differ in observation modality, temporal parameterization, and action-generation mechanism. For SUN, that diversity is useful methodologically: if data generated under one validated task representation benefit several policy families, the result is less likely to be an artifact of one learner’s inductive bias. It also creates a deliberately strict test of persistence. The downstream visual policies do not receive the controller’s privileged state or the complete task program, so their performance measures what useful behavior the generated data carry after the explicit scaffold is removed.

2.2.7 Synthesis: what persists across the task lifecycle

The SUN-related literature can be organized by what survives each interface. Language-conditioned planners retain an online objective. Reward generators retain executable scalar feedback. Formal task representations retain temporal progress. Optimal control retains modeled dynamics and constraints while solving the current horizon. Controller-to-policy methods retain demonstrations, losses, or priors, and data-generation systems retain trajectories for a downstream learner. Each artifact is valuable, but transferring between them can

change a relation’s identity, tolerance, stage membership, or success meaning.

SUN is designed around a persistent contract. A grounded relation has a scene binding, a residual, a satisfaction tolerance, and a relative learning weight. From that contract, the system derives a stage predicate, an MPC cost, and a reinforcement-learning term. The numerical forms may be operator-specific, but the relation identity, grounding, satisfaction boundary, and stage membership remain aligned. This continuity is the gap developed in chapters 6 to 8: language proposes task structure, model-based control tests it, learning amortizes accepted behavior, and monitoring continues to ask whether execution satisfies the same declared task.

Both parts preserve something under transformation but solve distinct problems: co-activity changes an environment so unlike embodiments can share it while retaining human scene relations, whereas SUN preserves executable task relations from language through control to learning.

CHAPTER 3

Embodiment-Aware Motion Planning with Virtual Kinematic Chains

Publication basis. This chapter draws on the Virtual Kinematic Chain study (Jiao et al., 2021). Its scope is limited to the VKC kinematic construction, coupled constrained-motion formulation, and motion-level evaluation that fall within the dissertation author’s contribution.

3.1 Why Base, Arm, and Object Motion Must Be Coupled

Opening a door or drawer should not be treated as manipulation layered on top of a completed navigation decision. If navigation selects and freezes a base pose first, that pose can block the end-effector approach or leave part of the articulated-object motion outside the stationary arm’s workspace. The task instead requires the base and arm to move together while the grasp constrains the robot to the object.

The Virtual Kinematic Chain (VKC) represents the mobile base, manipulator, grasp, and manipulated object as one coordinated kinematic system. A virtual joint connects the gripper to the grasped handle or tool frame, and the object’s kinematic chain is inverted. The resulting chain runs continuously from a virtual base fixed in the environment, through the robot and grasp, to the fixed frame of the articulated object. One constrained trajectory then determines both base and arm motion without preselecting a fixed manipulation base pose.

The remainder of the chapter follows this representation from construction to execution. It first derives the kinematic inversion and assembled chain, then formulates motion generation directly in the resulting configuration space. It next compares optimization- and sampling-based solvers, evaluates coordinated motion in a confined drawer-opening task, and closes by identifying the representation’s boundary that motivates environment rearrangement in the next chapter.

3.2 Kinematic Inversion and Chain Construction

Let the robot and object kinematic trees be $\mathcal{TG}^R$ and $\mathcal{TG}^O$. The branches of interest are the robot chain $\mathcal{C}^R$, ending at end-effector frame $\mathcal{F}_{ee}^R$, and the object chain $\mathcal{C}^O$, ending at an attachable frame $\mathcal{F}_{at}^O$. Write b_aT for the rigid transform that maps coordinates from frame a to frame b :

$${}^{i+1}_iT = \begin{bmatrix} {}^{i+1}_iR & {}^{i+1}_ip \\ \mathbf{0}^\top & 1 \end{bmatrix}. \quad (3.1)$$

Constructing a serial chain requires more than reversing the parent–child arrows of $\mathcal{C}^O$. A revolute or prismatic joint is expressed in its child frame, so its transform must also be inverted. For each link,

$${}^i_{i-1}T_{\text{inv}} = ({}^{i+1}_iT)^{-1} = \begin{bmatrix} ({}^{i+1}_iR)^\top & -({}^{i+1}_iR)^\top {}^{i+1}_ip \\ \mathbf{0}^\top & 1 \end{bmatrix}. \quad (3.2)$$

Applying equation (3.2) recursively makes the former parent of link i its child while preserving the same physical relative pose. After the inversion, a fixed, prismatic, or revolute virtual joint with transform ${}^{ee}_{at}T$ joins $\mathcal{F}_{ee}^R$ to $\mathcal{F}_{at}^O$. A virtual base $\mathcal{F}_b^V$ is then inserted ahead of the mobile base. Two perpendicular prismatic joints and one revolute joint represent planar

translation and yaw, keeping the assembled $\mathcal{C}^V$ serial while exposing locomotion to the same optimizer as the arm and object joints.

The direction of the resulting chain is important. For a door, its terminal frame is the link connected to the ground. Specifying the hinge angle therefore determines the task-space endpoint, while the base and arm configurations remain variables. The optimizer is therefore free to select locomotion and manipulation coordinates together throughout the object motion.

3.3 Constrained Motion Planning

3.3.1 Motion-planning problem

Let $\mathcal{Q}$ be the n -dimensional configuration space of the assembled chain, and let $\mathcal{Q}_{\text{free}} \subseteq \mathcal{Q}$ exclude self-collision and collision with the environment. Motion generation seeks

$$\mathbf{q}_{1:T} = (\mathbf{q}_1, \dots, \mathbf{q}_T), \quad \mathbf{q}_t \in \mathcal{Q}_{\text{free}}, \quad \mathbf{q}_1 = \mathbf{q}_{\text{start}}, \quad \mathbf{q}_T \in \mathcal{Q}_{\text{goal}}. \quad (3.3)$$

Here $\mathbf{q}_{\text{start}}$ is the measured configuration of the robot and object, while $\mathcal{Q}_{\text{goal}}$ is defined by the desired object state, such as a target drawer displacement or door angle. Because $\mathbf{q}_t$ contains the base, arm, grasp connection, and articulated-object coordinates, feasibility is evaluated on the complete coupled mechanism.

3.3.2 Trajectory optimization

Let $\mathbf{q}_t$ collect the VKC configuration at trajectory knot t , including the degrees of freedom exposed by the mobile base, arm, virtual joint, and manipulated object. With first and second finite differences $\delta\mathbf{q}_t$ and $\delta\dot{\mathbf{q}}_t$, the paper uses

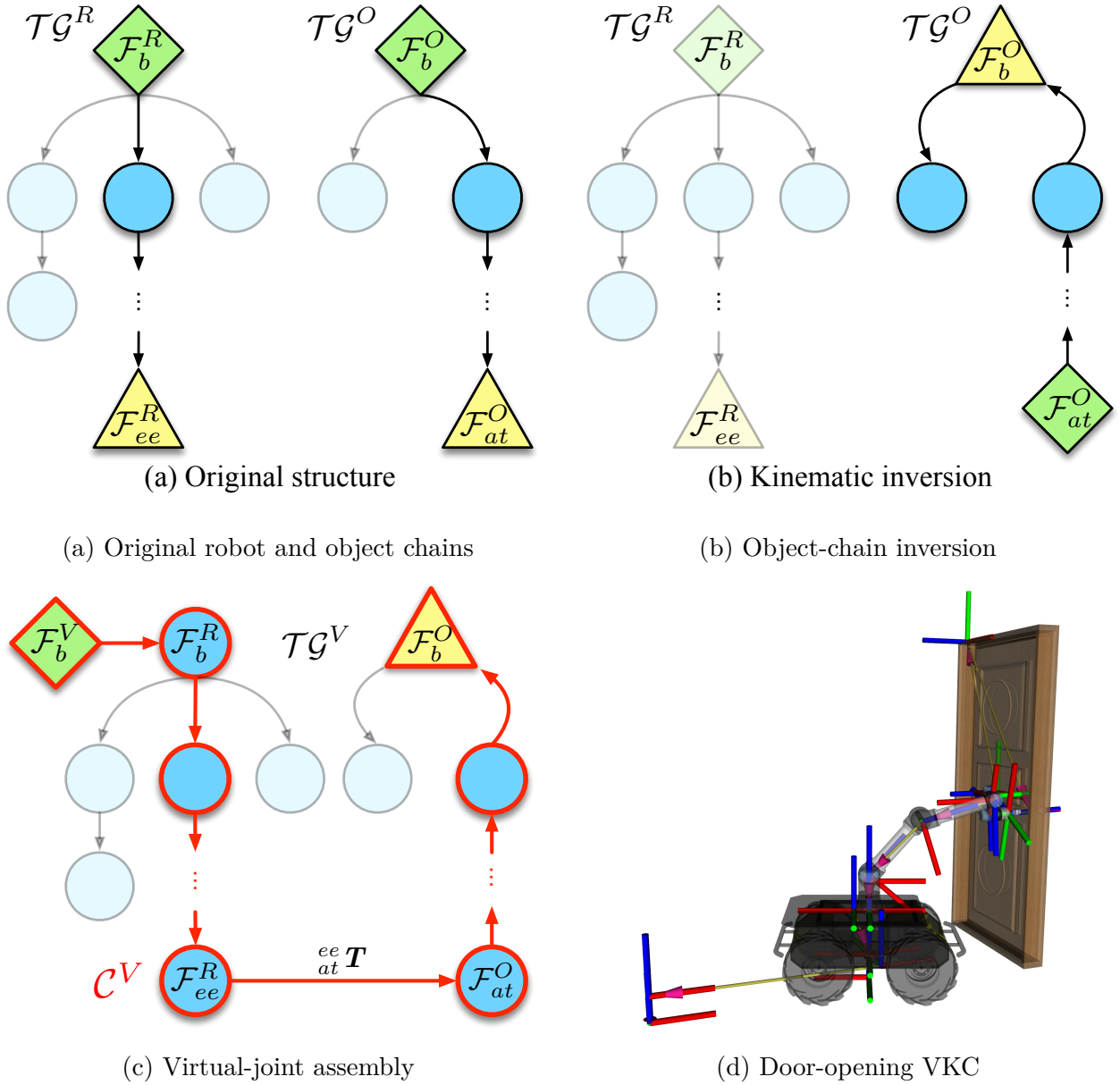


Figure 3.1: VKC construction using the original paper assets. Green diamonds are root frames, yellow triangles are terminal frames, blue circles are intermediate frames, and the red path is the assembled chain. In the door task, fixing the object’s ground-connected terminal frame lets the optimizer infer the base, arm, and door configurations jointly. Panels adapted from graphics in the published VKC study (Jiao et al., 2021). © 2021 IEEE.

$$\begin{aligned}
\min_{\mathbf{q}_{1:T}} \quad & \sum_{t=1}^{T-1} \|W_{\text{vel}}^{1/2} \delta \mathbf{q}_t\|_2^2 + \sum_{t=2}^{T-1} \|W_{\text{acc}}^{1/2} \delta \dot{\mathbf{q}}_t\|_2^2 \\
\text{s.t.} \quad & h_{\text{chain}}(\mathbf{q}_t) = \mathbf{0}, \quad t = 1, \dots, T, \\
& \|f_{\text{task}}(\mathbf{q}_T) - \mathbf{g}\|_2^2 \leq \xi_{\text{goal}}.
\end{aligned} \tag{3.4}$$

The diagonal matrices W_{vel} and W_{acc} assign costs to motion of individual joints. The equality $h_{\text{chain}}(\mathbf{q}_t) = \mathbf{0}$ contains forward-kinematic and closed-chain constraints. The task map $f_{\text{task}} : \mathcal{Q} \rightarrow \mathbb{R}^k$, with $k \leq n$, projects the terminal chain state into the task’s goal space. Consequently, the tolerance in equation (3.4) constrains only the k task-relevant coordinates; the remaining $n - k$ coordinates can be chosen by the objective. Joint limits, velocity and acceleration bounds, and link–link and link–object collision constraints are enforced in the same trajectory problem.

3.3.3 Sampling-based alternative

The same motion problem can instead be solved by a bidirectional RRT-Connect search in $\mathcal{Q}_{\text{free}}$, growing trees from the initial and goal configurations. This establishes that VKC is a representation rather than a commitment to one optimizer. The distinction matters in constrained interactions: sampled waypoints are interpolated only after search, whereas equation (3.4) incorporates smoothness and chain constraints throughout optimization.

3.4 Evaluation

The motion-level evaluation uses a confined drawer-opening task. With VKC, the mobile base and arm remain variables in one coupled configuration throughout the drawer motion. The comparison fixes the base before commanding the arm, which can leave the end of the drawer trajectory outside the stationary arm’s workspace. Both trajectory optimization and

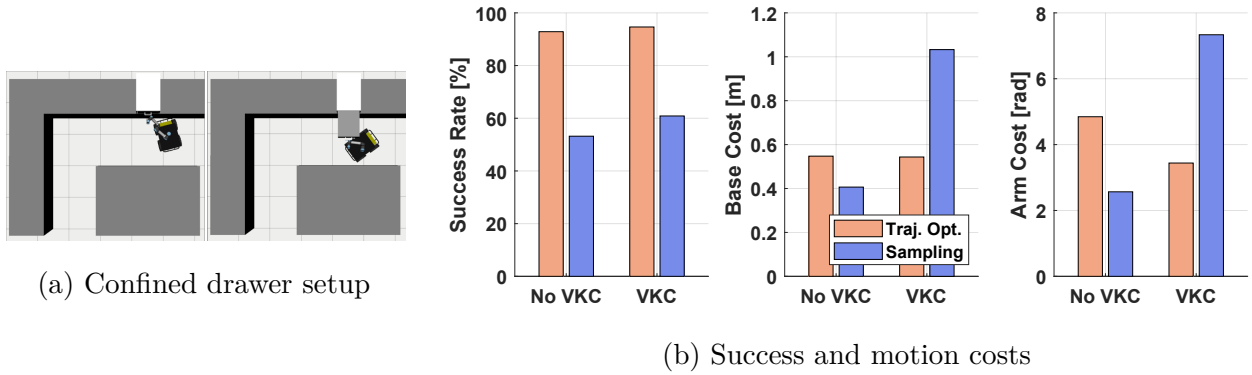


Figure 3.2: The original drawer setup and vector result plot from the VKC paper. Trajectory optimization maintains high success while coordinating base and arm motion. Sampling produces larger execution costs, particularly arm motion in the VKC configuration, because its constrained-space waypoints are not optimized for smoothness. Panels adapted from graphics in the published VKC study (Jiao et al., 2021). © 2021 IEEE.

sampling-based motion generation are evaluated in this scene.

The result separates representation from solver behavior. VKC lets the base reposition late in the drawer motion rather than depend on the stationary arm’s initial workspace. Trajectory optimization uses that freedom with shorter base and arm travel. Sampling remains more successful with VKC than without it but has larger costs and can violate safety limits; reported failures include reach and open-motion timeouts.

These results isolate the representation’s contribution: unified base, arm, and object coordinates enable whole-body motion with either solver, while optimization distributes that motion more effectively.

3.5 Boundary of the Representation

VKC changes how the robot uses space, not the room’s geometry. A gap narrower than its footprint, or an interaction region with no reachable approach, remains infeasible. The next

chapter therefore changes the design variable from robot adaptation within a human-centered room to the room itself.

CHAPTER 4

Scene Rearrangement for Human–Robot Co-Activity

Publication basis. This chapter is based on “Rearrange Indoor Scenes for Human–Robot Co-Activity” (Wang et al., 2023). The published ICRA formulation is the primary source. A later defense formulation is identified explicitly as an extension rather than silently replacing the published objective. Details recovered from the released reference implementation are likewise identified explicitly and do not replace the published formulation. Human-study results are re-audited from the supplied response data.

4.1 When Planning Cannot Create Access

The VKC representation uses available space more effectively, but a sufficiently narrow passage remains impossible. A room can provide ample clearance for a person while excluding a mobile manipulator from the nightstand, shelf, or interaction side of a piece of furniture. A coexisting robot therefore requires more than a controller designed for a fixed human environment. In some cases it requires a co-activity environment.

The analogy is a new roommate. People reorganize a shared room around both the new occupant’s preferences and those of the existing occupants. A robot may likewise require wider traversable regions and usable approach space, but moving furniture only to maximize open area would destroy the spatial relationships that make the room useful. A bed, desk, chair, nightstand, and television have different interaction sides and belong to functional



Figure 4.1: A human-activity layout (left) can exclude a mobile manipulator at the passages marked A and B even though people can use the room. The published rearranged layout (right) creates traversal and approach space while preserving human functional relations. Reproduced from the published co-activity study (Wang et al., 2023). © 2023 IEEE.

groups. The problem is to enlarge traversable and reachable space while preserving those relations.

The chapter develops human and robot preference models and their joint optimization, then evaluates accessibility, relation preservation, embodiment sensitivity, and human ratings.

4.2 Related Work and the Co-Activity Gap

Scene synthesis has long encoded human organization through interior-design rules, example arrangements, learned object relations, activity models, and generative scene priors (Merrell et al., 2011; Fisher et al., 2012; Yu et al., 2011; Kermani et al., 2016; Qi et al., 2018; Ritchie et al., 2019; Zhang et al., 2020; Nauata et al., 2021). These methods explain why collision-

free furniture placement alone is not a sufficient model of a useful room: relative position, orientation, grouping, and human activity jointly determine function.

Computational rearrangement has also optimized individual objectives such as fall-risk reduction, robot navigation, personalized workspace utility, or joint workspace and workplan design (Chaeibakhsh et al., 2021; Zhi et al., 2021; Liang et al., 2019; Zhang et al., 2021). The gap addressed here is the simultaneous treatment of two unlike occupants. Statistics from human-designed rooms describe human preference but do not encode the clearance, footprint, reach, or approach-direction requirements of a mobile manipulator. Conversely, maximizing robot free space can destroy the object relations that make the room usable by a person. The proposed method therefore models human and robot value separately and optimizes their compromise.

4.3 Co-Activity-Aware Problem Formulation

The method assigns separate representations to human and robot value. Human value is carried by semantic relations, spatial relations, and the functional groups induced by strong object relationships. Robot value is carried by footprint-cleared open space and by task-relevant interaction regions around furniture. The optimization variable is the layout Ψ , the collection of planar furniture poses. The objective is not to make the room empty; it is to make relevant furniture usable by the robot without undoing the human organization already present.

4.3.1 Weighted scene graph

Let the scene be a weighted relation graph

$$\mathcal{G} = (V, S). \quad (4.1)$$

Each object is a node

$$v_i = (o_i, B_i, p_i) \in V, \quad \Psi = \{p_i \mid i = 1, \dots, |V|\}, \quad (4.2)$$

where index i distinguishes instances, o_i is the semantic object label, B_i is an oriented three-dimensional bounding box, and p_i is the planar position and orientation. Changing Ψ rearranges the furniture while retaining object identity and geometry.

An edge $s_{i,j} = (v_i, v_j, w_{ij}) \in S$ records the strength of the relationship between two objects. The weight combines semantic and spatial evidence:

$$w_{ij} = P(v_i, v_j \mid \mathcal{G}) = \frac{1}{Z} P_{\text{sem}}(o_i, o_j \mid \mathcal{G}) P_{\text{spa}}(o_i, o_j \mid \mathcal{G}), \quad (4.3)$$

with scene normalization

$$Z = \sum_{\substack{o_i, o_j \\ i \neq j}} P_{\text{sem}}(o_i, o_j \mid \mathcal{G}) P_{\text{spa}}(o_i, o_j \mid \mathcal{G}). \quad (4.4)$$

The product in equation (4.3) follows the independence assumption in the published formulation. A thicker edge indicates a stronger relation and hence a greater preference to preserve that pair during rearrangement.

4.3.2 Semantic and spatial terms

ConceptNet provides semantic relations such as *RelatedTo* and *UsedFor* together with a relatedness score h_{ij} (Speer et al., 2017). Within the scene graph, the semantic contribution is normalized across edges:

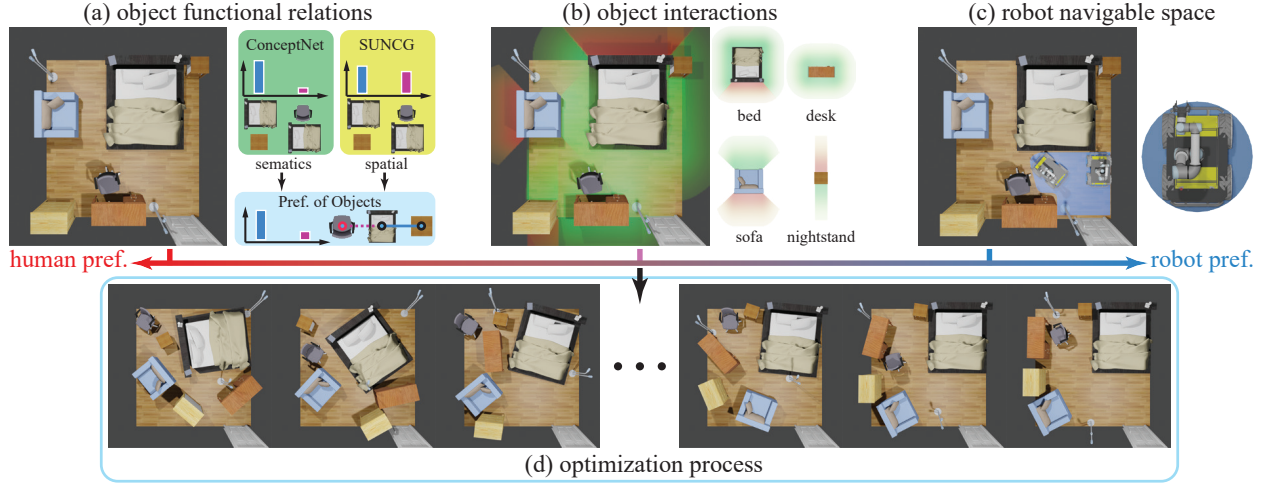


Figure 4.2: The published source figure connects object functional relations, object interaction regions, and robot navigable space to the iterative layout search. Human and robot preferences remain separate terms in the joint objective. Reproduced from the published co-activity study (Wang et al., 2023). © 2023 IEEE.

$$P_{\text{sem}}(o_i, o_j \mid \mathcal{G}) = \frac{h_{ij}^*}{\sum_{s_{m,n} \in S} h_{mn}^*}. \quad (4.5)$$

ConceptNet can assign a spuriously large relatedness score when two category labels are synonyms or stand in an *IsA* relation. Such an edge should not become the dominant structural relation merely because the ontology uses two near-equivalent names. The published formulation detects those pairs and replaces their raw score by the mean score of the remaining, non-synonymous category pairs in the scene:

$$h_{ij}^* = \begin{cases} h_{ij}, & \neg \text{IsA}(o_i, o_j), \\ \text{mean}(\{h_{mn} : \neg \text{IsA}(o_m, o_n)\}), & \text{IsA}(o_i, o_j). \end{cases} \quad (4.6)$$

The corrected value h_{ij}^* is used in equation (4.5). This neutral correction preserves semantic evidence in the published formulation while preventing taxonomy artifacts from defining

a functional group. The released reference implementation¹ takes a stricter operational approach: it removes ConceptNet edges labeled *IsA* or *Synonym* before constructing the relation graph. Starting from the scene’s object categories, it expands a local ConceptNet neighborhood to depth two and retains at most ten candidate edges at each expansion (breadth ten). Thus, equation (4.6) records the paper’s analytical treatment, whereas filtering and the depth–breadth limits record the executable graph construction.

The spatial term is estimated from pairwise object-category relations in SUNCG (Song et al., 2017). A relative pose is represented in polar coordinates by $d(v_i, v_j)$. Category co-occurrence is

$$P_{co}(o_i, o_j) = \frac{N_{ij}}{\min(\sum_{k \in \mathcal{O}} N_{jk}, \sum_{k \in \mathcal{O}} N_{ik})}, \quad (4.7)$$

where N_{ij} counts co-occurrences and $\mathcal{O}$ is the set of semantic object labels. The spatial contribution is proportional to the conditional pose density after this co-occurrence correction:

$$P_{spa}(o_i, o_j \mid \mathcal{G}) \propto P_d(d \mid o_i, o_j) P_{co}(o_i, o_j). \quad (4.8)$$

The paper summarizes P_d as a distribution over relative poses. The released implementation instantiates it with five directed descriptors for each ordered category pair: **FRONTOF**, the center-line angle in the reference object’s front frame; **FACING**, the angle between an object’s forward direction and the direction toward its partner; **PARALLEL**, the angle between the two forward directions; **CLOSETOBOUND**, the closest distance between an object boundary and its partner’s center proxy; and **FACINGBOUND**, the angular miss between the forward direction and the angular interval subtended by the partner’s bounding box. The last quantity is zero when that direction intersects the box’s angular interval. After removing undefined

¹Reference implementation: https://github.com/Rayckey/scene_coactivity, inspected at commit d9f0df8.

observations, the code fits a separate log-normal distribution to each descriptor for every ordered category pair represented by at least five observations. When forming candidate relation edges, it retains **FRONTOF**, **PARALLEL**, and **CLOSETOBOUND**, together with whichever of **FACING** and **FACINGBOUND** has the larger fitted likelihood.

Together, equations (4.5) and (4.8) determine which relationships are expensive to change when robot space is needed. Preserving every pairwise edge is unnecessary and can overconstrain the layout. A Gaussian mixture model partitions the edge weights; low-weight clusters are pruned to form a filtered graph $\mathcal{G}^* = (V, S^*)$. Each connected component of the remaining graph becomes a functional group. The hierarchy later uses these groups as sub-scenes.

The same reference implementation makes this grouping procedure more specific. It first sorts the candidate relation edges by weight. For each candidate component count from 2 through 10, K-means clustering supplies labels whose silhouette score is evaluated; the best-scoring count then sets the number of components in a Gaussian mixture model. After mixture prediction, the implementation retains the component containing the largest sorted edge weight and obtains functional groups as connected components of the resulting undirected graph. Finally, the groups are corrected using the scene’s physical support hierarchy, so an object supported by another object remains associated with its supporting root. These choices operationalize the paper’s shorter GMM description; they are implementation details rather than additional learned relations.

4.4 Robot Value: Reachable and Interaction Space

4.4.1 Reachable-space signed distance

Let $f_B : \mathbb{R}^2 \rightarrow \mathbb{R}$ be the signed distance from a query point q to the scene’s object bounding boxes, positive in collision-free space, and let r_b be the radius of the robot’s inflated circular

footprint. The published accessible-space field is

$$f_{\mathcal{R}}(q) = f_{\mathcal{B}}(q) - r_b. \quad (4.9)$$

Positive values represent feasible base-center clearance after footprint inflation; negative values represent collision. The paper denotes the entire clearance field by $\mathcal{R}$ but does not specify how disconnected free-space components are treated. The released implementation rasterizes this field at 0.1 m resolution and retains only the component connected to a robot start pose q_0 constructed just inside the room from the detected door and its front direction. Its operational reachable set is therefore

$$\mathcal{R}_{\text{conn}} = \{q \mid f_{\mathcal{B}}(q) > r_b \wedge q \rightsquigarrow q_0 \text{ through } \{q' : f_{\mathcal{B}}(q') > r_b\}\}. \quad (4.10)$$

This connected-component step prevents an isolated pocket of geometric free space from being counted as robot-accessible. Within the retained component, the signed-distance magnitude still distinguishes narrow clearance from a broader approach region.

4.4.2 Pseudo-interaction function

Reachable space alone does not encode how an agent uses a piece of furniture. A bed may preferably remain against a wall while retaining space along the side from which a person enters and a robot cleans. Each object therefore receives a pseudo-interaction field $\mathcal{I}_i$ around its bounding box B_i . The field uses the published hinge function:

$$f_{\mathcal{I}_i}(q) = \text{sgn}_i(q) \max\left(0, 1 - \frac{f_{B_i}^+(q)}{d^{\max}}\right), \quad (4.11)$$

where $f_{B_i}^+$ returns the nonnegative distance from q to object i 's bounding box. The function converges to zero beyond the prescribed extent $d^{\max}$, set to the maximum reach of

the robot arm. The sign is positive in an interaction region and negative in an anti-interaction region. The latter discourages layouts that expose a side of the furniture that is not normally used while blocking the side that is.

In the released implementation, sgn_i is explicitly category-dependent and is expressed in each object’s local frame before being rotated using its annotated front direction. Tables, coffee tables, and ottomans receive positive approach regions on all four sides. Beds receive positive regions along the front and two lateral sides and a negative region at the rear. Armchairs receive a positive front region, while sofas receive a positive front region and a negative rear region. Cabinets and televisions are also front-positive and rear-negative. Categories without a defined interaction template contribute no interaction field. Thus, arm reach determines the radial extent in equation (4.11), whereas the category template determines which directions are useful or undesirable.

The implementation consequently uses a stricter operational definition of an object being reachable than membership of its center in free space. For object i , let

$$\mathcal{A}_i = \{q \in \mathcal{R}_{\text{conn}} \mid f_{\mathcal{I}_i}(q) > 0\}. \quad (4.12)$$

The object is counted as reachable only if it has a category interaction template and $\mathcal{A}_i \neq \emptyset$. Among those feasible raster cells, the implementation chooses the interaction pose nearest the object’s center,

$$q_i^* = \arg \min_{q \in \mathcal{A}_i} \|q - c_i\|_2^2, \quad (4.13)$$

where c_i is the object center. An eight-connected A* search then connects the door-derived start pose to these interaction poses in task order. If no interaction pose or path exists for a segment, the code assigns the finite penalty $2(W + D)$, where W and D are the room’s

planar extents. This is the operational procedure behind both the reachable-object count and the planning term used by the implementation.

4.5 Human and Robot Objectives

For a spatial relation $s_{i,j}$, the human cost compares the current relative configuration d with the highest-probability configuration under the learned category distribution:

$$H_{s_{i,j}} = 1 - \frac{P_d(d \mid o_i, o_j; s_{i,j} \in S^*)}{\sup \|P_d(\cdot \mid o_i, o_j; s_{i,j} \in S^*)\|}. \quad (4.14)$$

Translation and rotation are evaluated separately using the same normalized form. A common configuration has low cost; an unusual relation has high cost.

In the published formulation, the robot cost combines interaction affordance and reachable-space clearance over their overlap:

$$I_i = - \int_{q \in \mathcal{I}_i \cap \mathcal{R}} [f_{\mathcal{I}_i}(q) + \alpha f_{\mathcal{R}}(q)] dq, \quad (4.15)$$

where α balances object interaction against open-space clearance. Lower I_i favors a larger usable overlap and wider navigable regions.

The published combined layout problem is

$$\begin{aligned} \min_{\Psi} \quad & \sum_{s \in S^*} H_s + \beta \sum_{v_i \in V} I_i \\ \text{s.t.} \quad & d(v_i, v_j) > 0, \quad \forall v_i, v_j \in V, i \neq j, \end{aligned} \quad (4.16)$$

where β is an empirical balance and the distance constraint prevents object collisions. Only relations retained in S^* contribute to the human term, so incidental proximity does not overconstrain the room.

The defense introduced an alternative overlap product,

$$V_i = \int_{q \in \mathcal{I}_i \cap \mathcal{R}} f_{\mathcal{R}}(q) f_{\mathcal{I}_i}(q) dq, \quad J_{\text{def}}(\Psi) = \frac{1}{|S^*|} \sum_{s \in S^*} H_s + \exp\left(-\sigma \sum_i V_i\right). \quad (4.17)$$

Unlike the additive published cost, this extension gives the largest reward to interaction regions that coincide with high-clearance reachable space. The two objectives are not treated as interchangeable: the published objective defines the paper result, while equation (4.17) records the later dissertation/defense variant.

The released implementation evaluates a still more explicit five-term cost:

$$J_{\text{impl}}(\Psi) = 10C_{\text{col}} + 5C_{\text{int}} + 15C_{\text{anti}} + C_{\text{plan}} + 10C_{\text{rel}}. \quad (4.18)$$

Here C_{col} converts PyBullet contact penetration and a 0.01 m clearance margin into a collision penalty. C_{int} applies a saturating transform to the sampled overlap between the positive, category-dependent interaction field and the start-connected reachable-space SDF. C_{anti} favors placing an object’s negative interaction side against the wall region rather than consuming useful interior approach space. C_{plan} is a saturating A* path-length cost from the door-derived start through the requested interaction sequence, with a finite penalty when a segment cannot be found. Finally, C_{rel} penalizes deviations from the retained log-normal spatial-relation models. All five terms are enabled by the implementation’s default layered solver. Consequently, equation (4.18) documents executable behavior, while equations (4.16) and (4.17) retain the two analytical formulations and their provenance.

4.6 Sampling and Hierarchical Optimization

The objective is nonconvex and is evaluated through geometric and relational fields, so the method uses derivative-free sampling. Adaptive Simulated Annealing (ASA) provides the global-search phase (Ingber, 1996). It samples candidates and accepts them according to a temperature-dependent Metropolis criterion. Early in the search, this permits escape from local minima; as temperature decreases, acceptance becomes more selective. Translation and rotation candidates are sampled uniformly within their current ranges, and ASA adapts the corresponding step sizes so that accepted and rejected proposals for each variable remain approximately balanced. ASA can converge slowly near a good solution, so its result initializes the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 2001). CMA-ES samples $\Psi' \sim \mathcal{N}(m, \sigma^2 C)$ and updates its mean, global scale, and covariance toward better candidates, supplying the local-refinement phase.

The paper specifies this global-to-local sequence but does not report the room boundary parameterization, numerical ASA-to-CMA switch, population sizes, or stopping tolerances. Those settings cannot be reconstructed from the paper text alone. The released implementation supplies a particular normalized pose parameterization and concrete ASA/CMA settings, while leaving the CMA population at the pinned library’s dimension-dependent default. These values are therefore code-snapshot provenance rather than parameters reported in the paper.

Large scenes further increase the sampling dimension. The optimization is therefore hierarchical. First, each functional group identified from the weighted relation graph is treated as a sub-scene and optimized independently. Second, the optimized group is treated as a single composite object, and the whole layout is optimized over group poses. The hierarchy uses the same human and robot objectives while reducing the number of variables considered at once.

4.7 Evaluation

4.7.1 Dataset split and quantitative measures

Pairwise category statistics are estimated from 90% of SUNCG, and the remaining 10% is held out for testing. The published quantitative study uses 154 rooms randomly selected from that held-out partition; 154 is not asserted to be the size of the complete partition. It reports two changes before and after rearrangement: a simple count of reachable objects and the robot term I_i from equation (4.15), which combines the interaction and open-space fields. Averaged over the reported set, the layouts provide 30% more reachable objects and a 14% improvement in this robot-access measure. The paper and plotting artifacts use inconsistent shorthand for the second quantity, so it is not recast here as a geometrically pure free-space-area statistic; see section E.4.

The two measures need not improve monotonically together. A layout can expose additional object sides while reducing the integrated interaction/clearance value, or improve that value without increasing the object count. The later defense extension encodes a stronger coupling through the product integral in equation (4.17), rather than free-space area alone.

4.7.2 Qualitative and ablation results

The published qualitative evaluation uses SUNCG living spaces. Blue shading shows robot-accessible space, while red objects are out of reach. Compared with the original layouts, the redesigned scenes enlarge accessible regions and make more objects reachable.

The human-cost ablation clarifies why accessibility is not a sufficient objective. Layouts optimized without human cost can obtain similar robot access by pushing non-task furniture aside. Chairs cease to support tables, and seating no longer faces the television. When the human term is included, those functional relations are preserved while robot access improves.

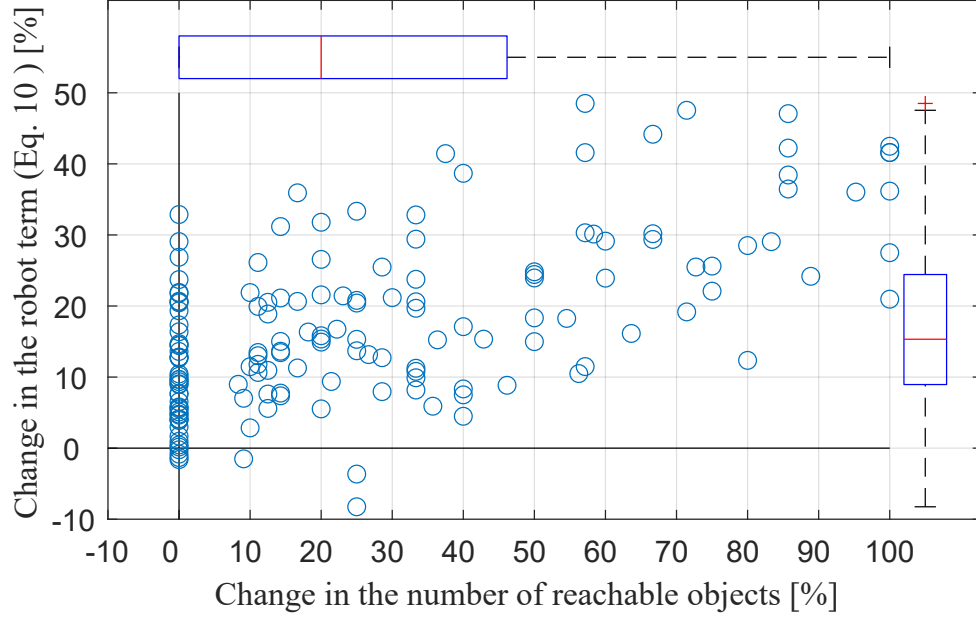


Figure 4.3: Published quantitative results. Each point is a rearranged room; marginal boxplots show the distributions of change in reachable-object count and in the paper’s robot term. The aggregate 30% and 14% improvements do not imply that both quantities improve in every room. Reproduced from the published co-activity study (Wang et al., 2023). © 2023 IEEE.

The result supports the intended interpretation of H_s : it protects the organization of the room rather than merely penalizing displacement.

The larger office example shows a second property: there is no single universally optimal co-activity layout. Starting from the same scene, assigning different activities produces substantially different arrangements and robot routes. Co-activity design must therefore be conditioned on both human function and the particular robot activities the space is intended to support.

The office activities make that conditioning explicit. A cleaning route visits the working functional group; restocking visits cabinets and shelves; and distribution visits all three

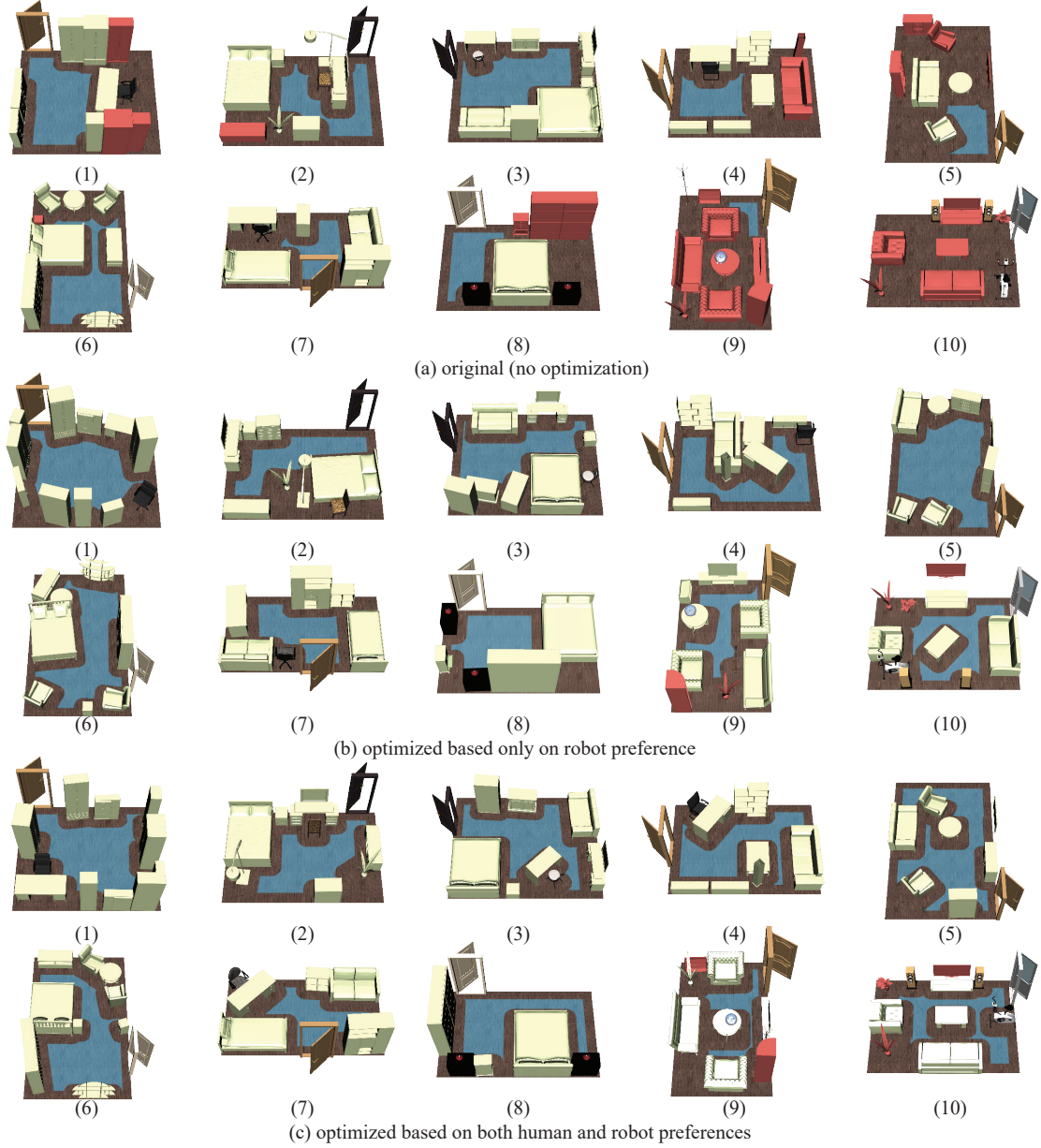


Figure 4.4: Ten published room examples in three groups: original rooms (top), robot-only optimization (middle), and joint human-robot optimization (bottom). Each group has two rows of five scenes; matching scene numbers identify the same room across groups. Blue denotes robot-accessible space. Reproduced from the published co-activity study (Wang et al., 2023). © 2023 IEEE.

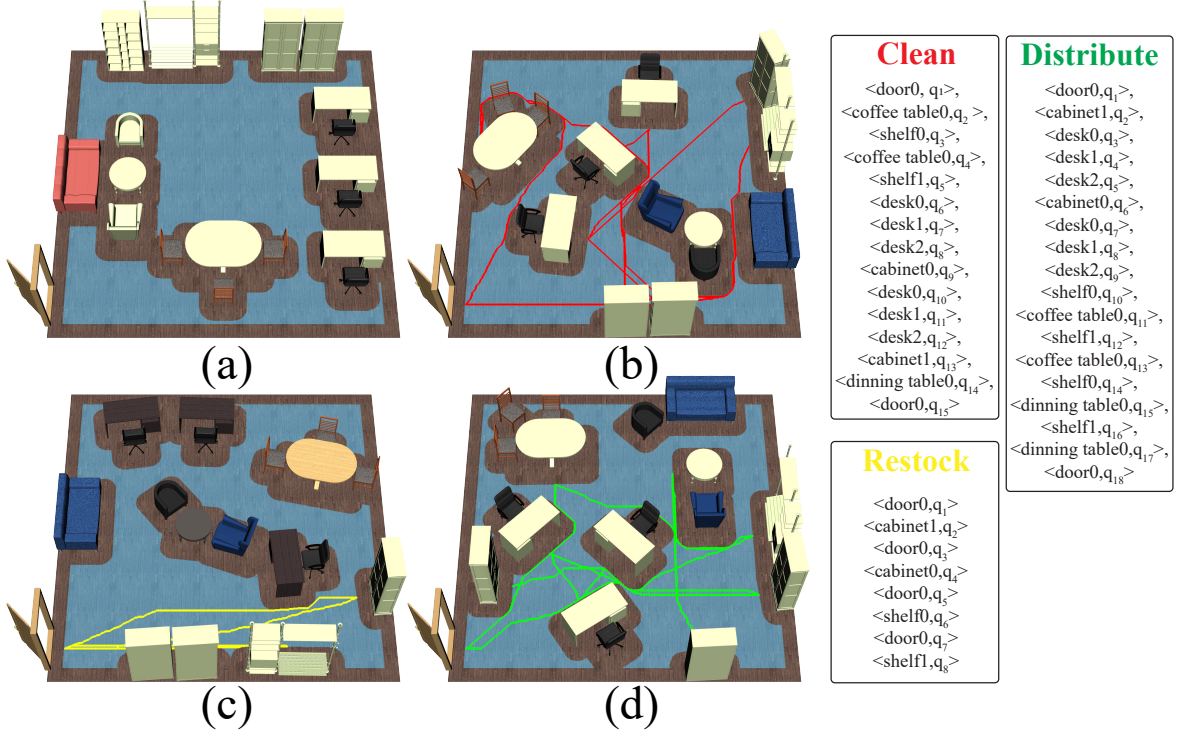


Figure 4.5: The same office optimized for cleaning, restocking, and distribution. Different activity routes induce different layouts, demonstrating that co-activity accessibility is task-conditioned. Reproduced from the published co-activity study (Wang et al., 2023). © 2023 IEEE.

functional groups. For a task T consisting of navigation activities, the published task-motion cost is

$$M_T(\Psi) = \sum_{t \in T} \mathcal{L}_\Psi(p(q_i^t, q_j^t)), \quad (4.19)$$

where $p(q_i^t, q_j^t)$ is a planned path between consecutive configurations for activity t , and $\mathcal{L}_\Psi$ is its length in layout Ψ . Adding M_T to the layout objective produces distinct task-specific arrangements from the same initial office.

4.7.3 Human evaluation

Eleven retained respondents evaluated seven original/rearranged scene pairs on five-point naturalness and functionality scales. Because each respondent rated both layouts, the dissertation audit treats the observations as paired rather than independent. It also controls the family of 14 comparisons with Holm’s procedure. Under that audit, none of the first five scenes differs on either measure after adjustment; scene 6 differs in functionality, and scene 7 differs in both naturalness and functionality. The full means and p -values are given in section E.3.

This result supports a narrower conclusion than the original paper wording. The study identifies conspicuous failures—an unusual bed orientation and a chair whose use is impeded by a desk—but a nonsignificant difference does not establish equivalence or prove that human preference was preserved. The failures nevertheless expose what category-level statistics omit: resident-specific conventions and the fine geometry of embodied human use.

4.7.4 Embodiment sensitivity

The same room was optimized for a compact Dingo mobile base with a UR3 arm and for a larger Husky base with a UR5 arm. The resulting layouts differ because the footprint changes accessible space and the arm reach changes $d^{\max}$. A layout optimized for the smaller system transfers only partially to the larger system; the paper notes that a sufficiently large footprint may leave no layout that accommodates the robot’s activities. The reported office task layouts use the Dingo embodiment. This experiment reinforces the central claim of the dissertation: accessibility is a relation among body, environment, and task, not an embodiment-independent property of a room.

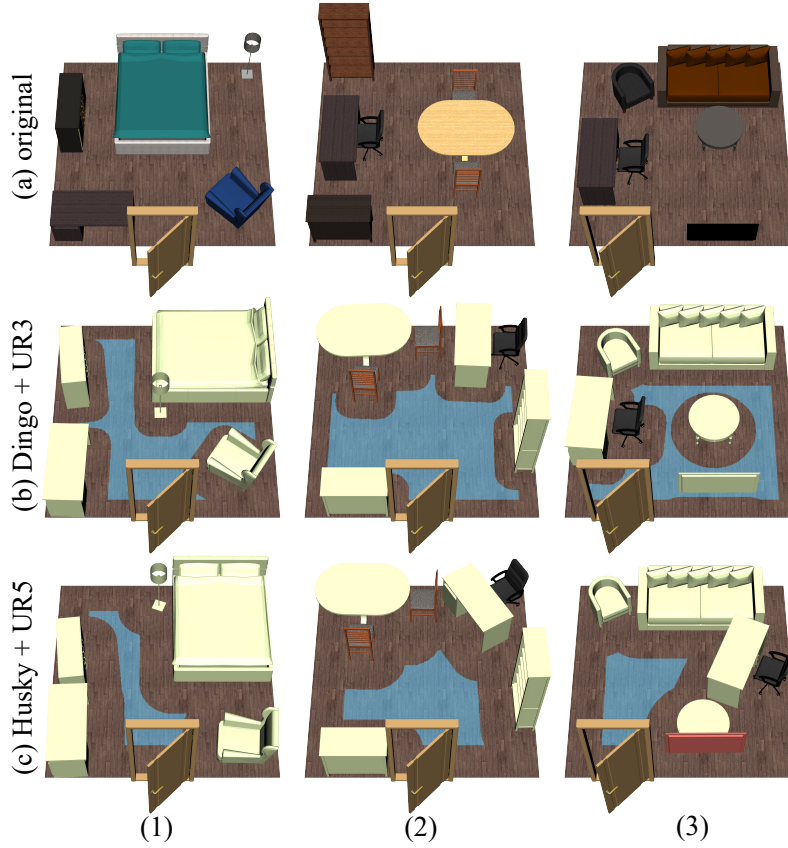


Figure 4.6: Three rooms optimized for a compact Dingo/UR3 and a larger Husky/UR5. Footprint and reach alter the preferred layout and can determine whether co-activity remains feasible. Reproduced from the published co-activity study (Wang et al., 2023). © 2023 IEEE.

4.8 Limitations

The formulation assumes movable furniture, known interaction regions, and no moving people; its planar scene also omits pose uncertainty. The SUNCG split tests unseen layouts but not cultural, household, or taxonomy shifts, and category statistics cannot track changing resident preferences. Without a matched joint human–robot baseline or large human study, the evidence supports objective improvement—not broad superiority or perceptual equivalence. The exact 154-room subset also remains ambiguous, as documented in section E.4. These

gaps motivate uncertainty-aware perception, preference learning, and dynamic-human safety.

CHAPTER 5

Physical-Effect-Centered Task Understanding for Tool Use

Publication basis. This chapter is based on *Understanding Physical Effects for Effective Tool-Use* (Zhang et al., 2022b). It is intentionally concise because the dissertation’s principal technical emphasis is the scene-rearrangement and SUN work. The presentation is limited to material already introduced in the original dissertation.

5.1 From Motion Reproduction to Effect Reproduction

A demonstrated tool trajectory is tied to the human arm, grasp, and tool geometry that produced it. Direct reproduction can therefore be inefficient or infeasible for a robot with different kinematics and dynamics. The transferable requirement is instead the physical effect: walnut cracking requires sufficient impact, whereas cutting additionally depends on blade orientation and contact geometry. The method recovers the conditions that cause the effect and then searches for a robot-specific way to realize them.

The chapter follows that transfer in four steps. It defines the tool strategy, summarizes how demonstrations yield actionable physical requirements, develops the robot-specific optimal-control problem, and closes with the minimum evaluation needed to establish the approach and its limitations.

A tool-use strategy is

$$\mathcal{S} = (\mathcal{B}_a, \mathcal{B}_f, \mathcal{Q}), \quad (5.1)$$

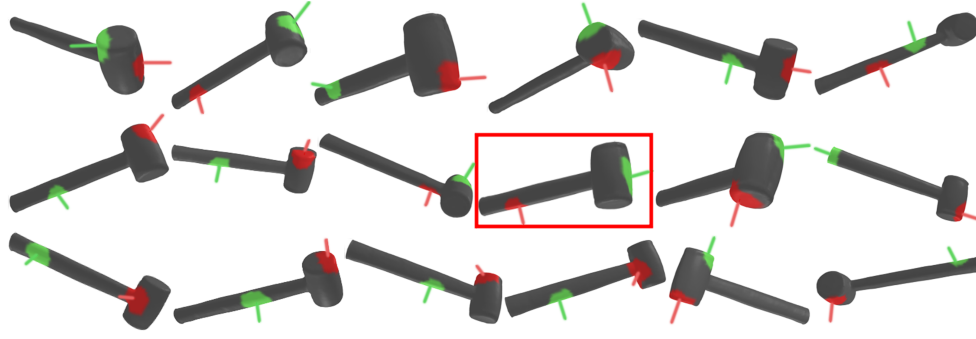
where the affordance basis $\mathcal{B}_a$ is the region grasped by the robot, the functional basis $\mathcal{B}_f$ contacts the target, and $\mathcal{Q} = \mathbf{q}_{1:T}$ is the robot–tool trajectory. The tool mesh is partitioned into candidate submeshes, with local surface normals defining grasp and contact directions. Searching over both bases and $\mathcal{Q}$ allows the robot to change how it holds an object, which part it uses, and how it moves.

Attaching the selected tool extends the robot’s body model. The virtual joint, tool inertia, torque limits, and functional-basis direction become part of the same Virtual Kinematic Chain (VKC) as the robot. Motion of $\mathcal{B}_f$ can then be planned for the actual embodiment rather than treating the demonstrated tool pose as a trajectory that must be copied.

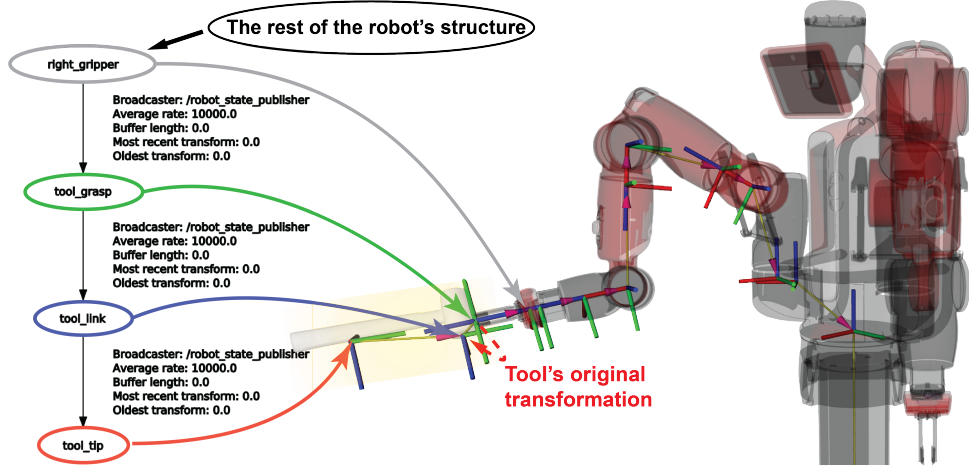
5.2 Physical-Effect Reconstruction

The demonstrations provide tool motion and pre-scanned object geometry. The paper reconstructs each event with watertight meshes in a finite-element simulator, using Incremental Potential Contact for non-smooth contact and a strain threshold for fracture. Young’s modulus and the fracture threshold are calibrated so the simulated outcome agrees qualitatively with the observation.

The reconstruction exposes variables that video alone does not reliably provide, including contact force, pressure, area, impulse, energy, momentum, and fragment count. These signals are used only to identify the physical requirements passed to planning; the robot does not reproduce the simulated human trajectory.



(a) Candidate affordance and functional bases



(b) Robot-tool virtual kinematic chain

Figure 5.1: Original paper assets for strategy construction. A virtual joint connects the gripper to $\mathcal{B}_a$, and $\mathcal{B}_f$ becomes the terminal frame of the robot-tool chain. Panels adapted from graphics in the published tool-use study (Zhang et al., 2022b). © 2022 IEEE.

5.3 Essential-Relation Learning

The available variables are organized into action, simulation, and effect levels,

$$\mathcal{V} = \mathcal{V}_A \cup \mathcal{V}_S \cup \mathcal{V}_E. \quad (5.2)$$

Iterative Deepening Symbolic Regression (IDSR) searches this hierarchy from a desired effect toward a compact set of explanatory variables. It retains a coarse variable when that

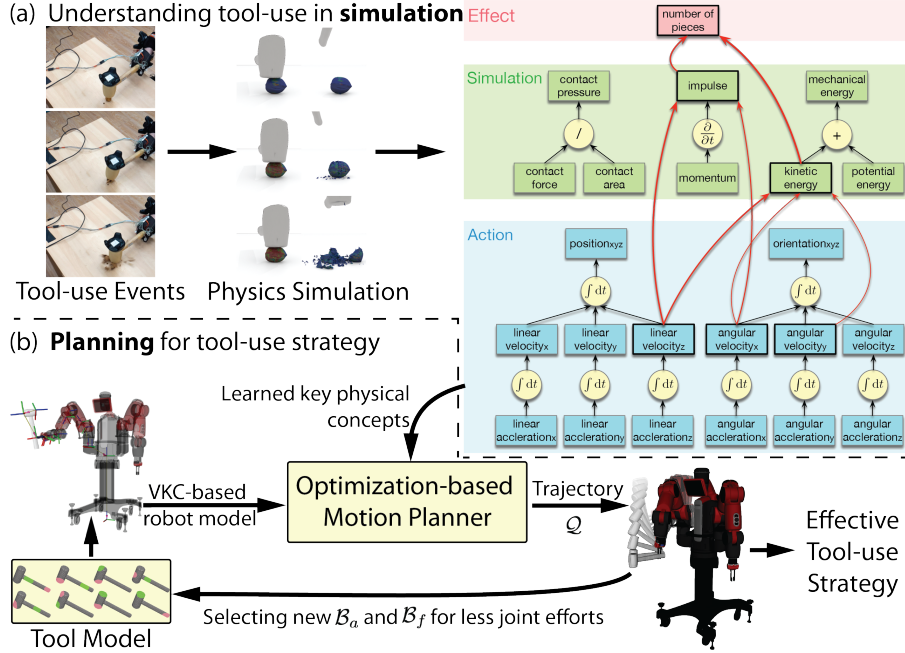


Figure 5.2: The original paper framework summarizes the retained interface: simulation exposes physical variables, learned relations specify actionable conditions, and VKC optimization selects a strategy for the robot embodiment. Reproduced from the published tool-use study (Zhang et al., 2022b). © 2022 IEEE.

variable explains the outcome and expands to its children only when a finer description is needed. Gaussian-mixture conditionals then convert a selected relation into numerical goals; for example,

$$P(F \mid E = e^*) = \frac{P(E = e^*, F)}{P(E = e^*)}, \quad v_z \sim P(v_z \mid F), \quad (5.3)$$

where E is the desired effect, F is contact force, and v_z is impact velocity. The resulting actionable requirement is primarily impact velocity for walnut cracking. For carrot cutting, the retained contact-area relation also produces a tool-orientation requirement. These inferred quantities, not the full relation-search procedure, form the interface to the planner.

5.4 Robot-Specific Optimal Control

For each sampled pair $(\mathcal{B}_a, \mathcal{B}_f)$, the robot and tool form a VKC. Let x collect their positions and velocities, let joint torques u be the controls, and let T be the terminal time. Forward dynamics are computed with an articulated-body algorithm, and the planner solves

$$\begin{aligned} \min_{x,u,T} \quad & \int_0^T \left(\dot{q}(t)^\top W_{\dot{q}} \dot{q}(t) + u(t)^\top W_u u(t) \right) dt + T \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t)), \quad g(x(t), u(t)) = \mathbf{0}, \\ & x_{\text{lb}} \leq x(t) \leq x_{\text{ub}}, \quad u_{\text{lb}} \leq u(t) \leq u_{\text{ub}}, \quad t \in [0, T]. \end{aligned} \quad (5.4)$$

The running cost penalizes joint velocity and torque effort, the terminal term penalizes elapsed time, and the bounds enforce workspace and actuator limits. The constraint g is constructed from the physical requirement learned above. Thus every candidate grasp and functional basis must achieve the same effect, while the objective determines which feasible strategy best suits the robot.

Let $\mathbf{p}_g$ be the desired contact position, $p_f(q)$ the position of $\mathcal{B}_f$, and $f_z(q)$ its surface normal. The inferred impact velocity $\mathbf{v}_{\text{tool}}$ and required angle θ_{tool} define the terminal conditions

$$p_f(q_g) = \mathbf{p}_g, \quad \frac{f_z(q_g)^\top \mathbf{v}_{\text{tool}}}{\|f_z(q_g)\|_2 \|\mathbf{v}_{\text{tool}}\|_2} = \cos \theta_{\text{tool}}, \quad (5.5)$$

$$\dot{q}_g = J_{\text{VKC}}^\top \left(J_{\text{VKC}} J_{\text{VKC}}^\top \right)^{-1} \mathbf{v}_{\text{tool}}, \quad q(T) = q_g, \quad \dot{q}(T) = \dot{q}_g. \quad (5.6)$$

Here J_{VKC} is the geometric Jacobian from the robot base to $\mathcal{B}_f$; the pseudoinverse expression assumes the relevant rows have full rank. The equations make the transfer explicit: the robot need not match the demonstrator's joints or grasp, but its functional basis must

Table 5.1: Walnut-cracking success in simulation. Values reported in the published tool-use study (Zhang et al., 2022b). © 2022 IEEE.

Robot	Kinematic baseline	Effect-centered planner
Baxter	14%	62%
UR5	16%	52%

reach the required contact pose with the required velocity. Different embodiments can therefore produce different minimum-effort strategies for the same effect.

5.5 Evaluation

The evaluation covers walnut cracking and carrot cutting with a human arm model, Baxter, and UR5. The objective recovers the familiar hammer strategy for the human model but selects different bases, trajectories, and sometimes tools for the two robots. In cutting, velocity alone does not align the blade; adding the inferred orientation condition produces the intended contact.

For walnut cracking, fifty trials per robot and method are simulated under 10% variation in fracture parameters. Ten online participants classify all 200 outcomes as uncracked, correctly cracked, or smashed, with majority-class “cracked” counted as success.

Because the baseline already targets the walnut center and aligns its contact normal with gravity, the improvement supports the value of effect constraints beyond geometric contact. These experiments are evidence for the representation and robot-specific optimization, rather than a claim of broad task coverage.

5.6 Scope and Limitations

This simulation-only work assumes watertight geometry, homogeneous calibrated materials, and a valid grasp. Costly FEM reconstruction and two exemplars do not establish coverage of fluids, non-fracturing deformables, or compound tools. The scoped contribution separates task meaning from demonstrated motion and re-instantiates it as robot-specific constraints.

CHAPTER 6

Formation of Persistent SUN Programs

Publication basis. This chapter is based on the task-formation and representation portions of “SUN: Persistent Programs for Language-Grounded Control-to-Learning-to-Real Policies” (Wang et al., 2026). Equations, terminology, and reported values follow the SUN manuscript. Operator-level equations additionally describe the accompanying implementation; implementation-specific details and dissertation-only interpretation are labeled explicitly.

6.1 Task Formation as the Remaining Bottleneck

Language models can propose action sequences, objectives, or rewards, but a robot still needs one physical meaning for the task across control, learning, monitoring, and execution. Small inconsistencies are consequential. A controller may optimize distance to a drawer handle while a monitor declares the stage complete from gripper motion; a learned reward may favor an easy proxy while an external evaluator requires the drawer to move. Re-generating each interface independently allows these definitions to drift.

SUN addresses this gap by forming one typed executable program, verifying it in closed-loop physics, and preserving the accepted artifact through later handoffs. The larger system, KUAFU, uses optimal control as the first physical critic. In the manuscript’s reported protocol, once a program passes that critic, the grounded relations, thresholds, logic, and stage order are frozen. Later learning may amortize the controller and calibrate scalar learning

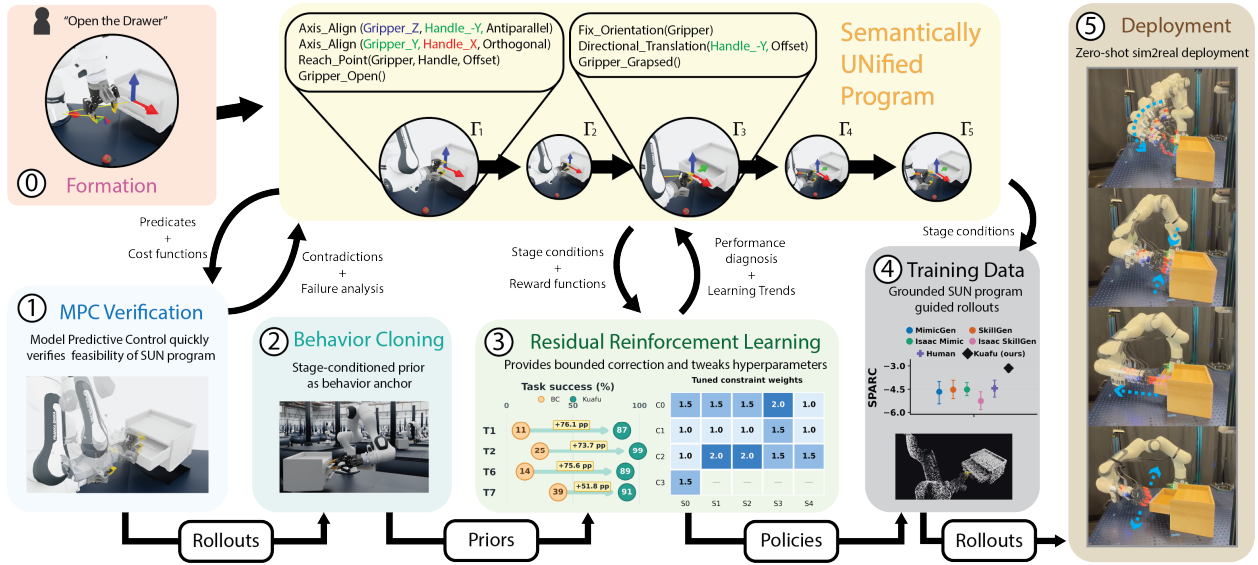


Figure 6.1: One accepted SUN Program survives task formation, MPC verification, program-preserving policy learning, autonomous data generation, visual-policy supervision, and physical deployment. Updated lifecycle overview supplied by the author.

parameters, but it does not silently rewrite the accepted task. The updated implementation’s separate MPC and learning tolerance profiles are distinguished from this reported protocol in section B.6.1.

6.2 Positioning: What the Program Must Preserve

Long-horizon manipulation needs discrete task progression and continuous geometric execution to agree. Logic-geometric programming connects symbolic actions to motion optimization (Toussaint, 2015); reward machines and signal-temporal-logic control make temporal progress explicit (Toro Icarte et al., 2018; Meng and Fan, 2023). Language-conditioned systems expose other useful structures: ProgPrompt generates programs over robot APIs (Singh et al., 2023), VoxPoser composes three-dimensional value maps (Huang et al., 2023), ReKep optimizes relational keypoint constraints (Huang et al., 2025), and OmniManip grounds object-centric

interaction primitives as spatial constraints (Pan et al., 2025).

A second line turns task descriptions into learning supervision. Language to Rewards, Text2Reward, Eureka, and DrEureka generate rewards or randomization programs (Yu et al., 2023; Xie et al., 2024; Ma et al., 2024b,a); DrS learns reusable dense rewards for multi-stage tasks (Mu et al., 2024). Optimized control can instead provide behavior directly: PLATO adapts an MPC teacher to the learner’s state distribution, MPC-Net retains control structure in a policy loss, and residual RL learns corrections around a nominal controller (Kahn et al., 2017; Carius et al., 2020; Johannink et al., 2019).

Finally, data-generation systems such as MimicGen and SkillGen expand seed demonstrations, while RoboGen and GenSim2 automate broader portions of task, scene, and rollout production (Mandlekar et al., 2023; Garrett et al., 2025; Wang et al., 2024; Hua et al., 2025). These systems solve important pieces of task formation, execution, learning, or data scaling. The specific SUN claim is narrower and auditable: one grounded relation identity, frame binding, satisfaction boundary, and stage membership survives MPC verification, policy acquisition, learned execution, monitoring, and data validation. This persistence—not the unsupported claim that every comparator lacks a broad capability—is the distinction evaluated in the following chapters.

The capability-by-capability comparison from the manuscript source is retained in table 6.1. A check mark records explicit and experimentally demonstrated support; a cross means only that the property is not demonstrated in the cited system’s stated pipeline role, not that the method is fundamentally incompatible with it.

6.3 SUN Program Representation

Let a task be represented by an ordered program

Table 6.1: Positioning task representations across online control and policy learning. The method rows, criteria, and assignments reproduce the SUN manuscript’s positioning matrix.

Representative systems	Explicit spec.	Stage aware	Stage guards	Structured geometry	Direct ctrl.	Pre-policy validation	Spec. to policy	One task policy	Spec. to execute	Policy diag.
<i>A. Language-generated task representations used for online execution</i>										
SayCan (Ichter et al., 2023)	✓	✓	✓	×	✓	×	×	×	✓	×
Code as Policies (Liang et al., 2022)	✓	✓	✓	×	✓	×	×	×	×	×
ProgPrompt (Singh et al., 2023)	✓	✓	✓	×	✓	×	×	×	×	×
Language to Rewards (Yu et al., 2023)	✓	×	×	✓	✓	×	×	×	×	×
VoxPoser (Huang et al., 2023)	✓	✓	×	✓	✓	×	×	×	×	×
ReKep (Huang et al., 2025)	✓	✓	✓	✓	✓	×	×	×	×	×
OmniManip (Pan et al., 2025)	✓	✓	×	✓	✓	×	✓	✓	×	×
CoRAL (Cicek et al., 2026)	✓	✓	✓	✓	✓	×	×	×	×	×
STeP (Torshizi et al., 2026)	✓	✓	✓	✓	✓	×	×	×	✓	✓
<i>B. Generated or learned task supervision used for policy acquisition</i>										
Text2Reward (Xie et al., 2024)	✓	✓	✓	×	×	×	✓	×	×	×
Eureka / DrEureka (Ma et al., 2024b.a)	✓	×	×	×	×	×	✓	×	×	×
DrS (Mu et al., 2024)	×	✓	✓	×	×	×	✓	✓	×	×
DEMO3 (López Escoriza et al., 2025)	×	✓	✓	×	✓	×	✓	✓	✓	×
IKER (Patel et al., 2025)	✓	✓	✓	✓	×	×	✓	×	✓	×
ReWiND (Zhang et al., 2025a)	×	×	×	×	×	×	✓	×	×	×
SARM2 / SPIRAL (Chen et al., 2026)	×	✓	✓	×	×	×	✓	✓	×	×
<i>C. Automated task-to-policy formation</i>										
AnyTask (Gong et al., 2025)	✓	✓	×	×	✓	✓	✓	✓	×	×
Ours: Kuafu	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Criteria. **Explicit spec.:** a separately represented, inspectable task artifact; **Stage aware:** explicitly decomposes, sequences, labels, or conditions supervision on distinct task stages; **Stage guards:** explicit state-dependent logic gates advancement between ordered substeps; **Structured geometry:** the task artifact contains residuals, frame or axis relations, 3-D keypoint constraints, spatial value maps, contact geometry, clearances, or formal metric predicates; **Direct ctrl.:** the representation is consumed directly by a planner or controller; **Pre-policy validation:** the generated task or reward artifact is validated before policy training; **Spec. to policy:** the representation directly supervises policy acquisition; **One task policy:** one learned controller spans the multi-stage task; **Spec. to execute:** the representation remains consulted during learned execution; and **Policy diag.:** term-level failures are localized during execution.

$$\Gamma = \{\Gamma_k\}_{k=1}^K, \quad \Gamma_k = \{(e_{k,j}(x, \xi_k), \epsilon_{k,j}, w_{k,j})\}_{j=1}^{m_k}. \quad (6.1)$$

Here K is the number of stages; $e_{k,j}$ is a typed grounded residual; $\epsilon_{k,j}$ is its satisfaction tolerance; $w_{k,j} \geq 0$ is its relative learning weight; x is the physical state; and ξ_k stores stage-entry references such as the drawer pose when pulling begins. Terms are selected from a finite typed operator library $\mathcal{L}$ over registered scene frames $\mathcal{F}$. The RAL manuscript exemplifies axis alignment, directional movement, fixed orientation, point reaching, gripper-open, and gripper-grasped relations. The accompanying implementation exposes the complete prompt-visible catalog given in section 6.4. Type and frame binding occur before control.

The monitor augments the state with the active stage and its entry references,

$$\tilde{x}_t = (x_t, z_t, \xi_{z_t}), \quad (6.2)$$

so the same physical relation may be evaluated relative to a pose captured at stage entry rather than an unrelated global frame. Each term compiles into aligned interfaces:

$$\text{Compile}(\Gamma_k) = \left\{ \left(\phi_{k,j}^{\text{mpc}}, P_{k,j}, c_{k,j}^{\text{rl}} \right) \right\}_{j=1}^{m_k}. \quad (6.3)$$

The MPC term contributes to the stage objective, $P_{k,j}$ is evaluated by the monitor, and $c_{k,j}^{\text{rl}}$ contributes to learning. Their exact numerical forms are left operator-specific in the active manuscript and are instantiated from the accompanying implementation in section 6.4. What is shared is the relation identity, frame grounding, stage membership, and declared satisfaction boundary. For a nonnegative residual, the satisfaction margin and predicate are

$$\gamma_{k,j} = \epsilon_{k,j} - e_{k,j}, \quad P_{k,j} = \mathbf{1}[\gamma_{k,j} \geq 0]. \quad (6.4)$$

Thus γ is a term-level margin, not the stage guard. Compilation also retains diagnostic metadata such as the source stage, frames, axis, tolerance, measured residual, and margin.

Stage completion is conjunctive:

$$D_k(\tilde{x}_t) = \bigwedge_{j=1}^{m_k} P_{k,j}(\tilde{x}_t). \quad (6.5)$$

The stage index advances only when D_k is true and never decreases. This irreversibility makes execution and diagnosis simple, but it also creates a clear limitation: the current program cannot roll back to repair an earlier stage after the scene changes.

The runtime ordering is consequential. After executing u_t , the system observes x_{t+1} ; evaluates current-stage predicates and the external outcome; assigns progress, transition, and terminal signals; and only then updates z_{t+1} . When a transition occurs, the next stage’s entry-scoped references are captured from x_{t+1} . This prevents a predicted future action or a stale snapshot from advancing the program.

The program’s own terminal predicate is not used as ground truth. Each task has an independently authored external success evaluator g_{ext} that checks the intended physical outcome. False completion is

$$FC(\tau) = D_K(\tau) \wedge \neg g_{\text{ext}}(\tau). \quad (6.6)$$

For the drawer task, for example, the program checks grounded position, orientation, grasp, and motion relations, while the external evaluator requires at least 12 cm of drawer displacement. Separating these definitions exposes a plausible but semantically wrong program instead of rewarding it.

6.4 Typed Constraint Vocabulary and Semantics

The language interface is deliberately narrower than free-form reward or controller generation. It first decomposes an instruction into [Approach], [Manipulate], and [Release] subtasks. Each subtask contains one controller-stage-sized conceptual change—one spatial change or one gripper-state change—together with any simultaneous pose or grasp requirements. A second pass grounds exactly one such subtask into frame-, axis-, distance-, and feedback-specific language. A final pass selects terms from $\mathcal{L}$ and emits a flat program for one Γ_k ; it does not write new controller code. This three-pass boundary is what turns a long instruction into atomic residual programs without hiding a sequence such as approach–grasp or align–insert inside one stage. For auditability, section B.6 reproduces the active system and user templates, while table B.3 maps each call to the run-time evidence inserted into it.

The implementation names these operators in lower-case snake case. `directional_move` corresponds to the manuscript label *Directional Translation*; `freeze_orientation` to *Fix Orientation*; `open_gripper` to *Gripper Open*; and `object_grasped` to *Gripper Grasped*. The `axis_align` and `reach_point` names differ from their displayed manuscript labels only in typography. The equations below retain the paper’s symbols for programs and compiled views while using the exact current operator names at the language–implementation boundary.

Grounding uses only names in $\mathcal{F}$. Every translation and orientation names its reference frame and a signed axis. The tool $+z$ axis is the approach axis and tool $+y$ is the finger axis. Thus a top-down approach makes tool $+z$ antiparallel to world $+z$; a leveled approach makes those axes orthogonal while pointing tool $+z$ at the target; a direct approach uses the pointing relation; and an axis approach makes tool $+z$ antiparallel to the selected signed object axis. For grasp setup, tool $+y$ is orthogonal to an elongated object’s long axis or to a selected clamping axis of a box, while rotational symmetry is left unconstrained. These

rules remove geometric choices that natural-language phrases such as *approach the handle* otherwise leave ambiguous.

The atomic-grounding templates in tables B.8 and B.9 also separate clearance from contact. If l_c and l_r are the moving and reference objects’ relevant extents—their longest extents unless an axis-specific extent is supplied—and m is a non-contact margin, the requested center separation is

$$d_{\text{sep}} = \frac{l_c}{2} + \frac{l_r}{2} + m, \quad m \geq 0.10 \text{ m}, \quad (6.7)$$

with $m \geq 0.15 \text{ m}$ for a *large* clearance unless the instruction specifies a tighter value. Contact and grasp setup instead target the named frame origin or object center and state the intended contact feedback. The updated prompts specify target geometry, not completion tolerances; the receiving application assigns the latter before compilation, as detailed in section B.6.1. Clearance remains a target separation, not an acceptance-band parameter.

Table 6.2 gives the ten operations exposed by the current constraint-writing prompt, reproduced with its system context in tables B.10 and B.11. The first seven are geometric; the last three act on or test the gripper. The `controlled_entity` is either a registered `robot_frame` or the `gripped_object` propagated through a fixed grasp transform. Basis and origin are specified separately. The `reference_basis` is world, object, or robot frame. The `reference_origin` identifies the pose supplying its origin and axes. Distances are in metres and angles are supplied in radians. The prompt omits tolerance fields for all seven geometric operators. The receiver supplies ϵ_p or ϵ_θ according to the operator and selected implementation profile; these remain parameters of the compiled predicates below, not values requested from the LLM.

Table 6.2: The current finite TermAPI vocabulary. Parameters shown here are semantic rather than a literal JSON schema; every name is bound to the registered scene interface before compilation. Numerical predicate tolerances are assigned by the receiver and are not part of the model’s output schema.

Operation	Principal parameters	Physical meaning
<code>reach_point</code>	controlled entity/target, reference basis/origin, offset	Reach one frame-relative point; the intervening path is unconstrained.
<code>directional_move</code>	controlled entity/target, reference axis, $D > 0$	Translate by D along one signed axis while suppressing lateral drift.
<code>axis_align</code>	controlled and reference signed axes, relation	Enforce codirectional, antiparallel, bipolar, or orthogonal axes.
<code>point_toward</code>	controlled signed axis, target point	Aim an axis at a frame-relative point.
<code>directional_rotate</code>	reference signed axis, angle θ	Rotate from the stage-entry orientation by a specified amount.
<code>freeze_position</code>	controlled entity/target	Preserve the position captured on stage entry.
<code>freeze_orientation</code>	controlled entity/target	Preserve the orientation captured on stage entry.
<code>open_gripper</code>	none	Require the released/open mechanical state.
<code>close_gripper</code>	none	Require mechanical finger closure, without claiming a grasp.
<code>object_grasped</code>	none	Require physical holding and retain it during later motion.

6.4.1 Common Frame and Cost Conventions

The notation below translates the implementation into the paper’s compiled views in equation (6.3). Let ${}^W p_A$ and ${}^W q_A$ denote the world position and scalar-first unit quaternion of frame A , let $R(q) \in SO(3)$ be its rotation matrix, and let superscript 0 denote the value stored in ξ_k on entry to the current stage. For a local offset δ , reference origin O , and reference robot frame F , the target point used by `reach_point` is

$${}^W p^\star = \begin{cases} {}^W p_O(t) + \delta, & \text{world basis,} \\ {}^W p_O(t) + R({}^W q_O(t))\delta, & \text{object basis,} \\ {}^W p_F^0 + R({}^W q_F^0)\delta, & \text{robot-frame basis.} \end{cases} \quad (6.8)$$

In the world branch, δ is a world-coordinate offset from the queried `reference_origin`; it is an absolute world point only when that origin is the reserved `world` frame. When the controlled entity is an object held by end effector E , the entry snapshot supplies the fixed grasp transform

$$q_{O/E} = ({}^W q_E^0)^* \otimes {}^W q_O^0, \quad r_{O/E} = R({}^W q_E^0)^\top ({}^W p_O^0 - {}^W p_E^0), \quad (6.9)$$

and the optimized end-effector state implies

$${}^W q_O = {}^W q_E \otimes q_{O/E}, \quad {}^W p_O = {}^W p_E + R({}^W q_E)r_{O/E}. \quad (6.10)$$

Thus the same position and orientation operations can control either E or a rigidly attached object without adding object pose to the MPC decision state. For compactness, define the quadratic MPC penalty and the bounded learning cost

$$Q(v) = \frac{1}{2}v^\top v, \quad B_a(r) = \frac{r^2}{r^2 + a^2}, \quad 0 \leq B_a(r) < 1. \quad (6.11)$$

The function B_a is named a sigmoid in the source, although it is the rational map in equation (6.11). A lower c^{rl} therefore means better constraint satisfaction; the learned environment subsequently converts the normalized stage cost into a progress score. In each derivation, the scalar distance or angular error tested by P is the concrete realization of the paper-level relation error $e_{k,j}$; vector coordinate errors are intermediate arguments to Q .

6.4.2 Translation Constraints

For `reach_point`, let p_c be the controlled robot-frame position or the composed object position from equation (6.10). Its error, MPC view, predicate, and learning view are

$$\begin{aligned} e_{\text{reach}} &= p_c - {}^W p^\star, & d_{\text{reach}} &= \|e_{\text{reach}}\|_2, \\ \phi_{\text{reach}}^{\text{mpc}} &= Q(e_{\text{reach}}), & P_{\text{reach}} &= \mathbf{1}[d_{\text{reach}} \leq \epsilon_p], \\ c_{\text{reach}}^{\text{rl}} &= B_{0.5}(d_{\text{reach}}). \end{aligned} \quad (6.12)$$

The tolerance changes the stage predicate, not the quadratic or bounded cost, and the operation constrains the endpoint rather than the path.

For `directional_move`, resolve the requested signed basis axis as the world unit vector $\hat{a}$, let $D > 0$ be the requested travel, and measure motion from the controlled entity's entry position $p_c^0 \in \xi_k$:

$$\begin{aligned} \Delta p &= p_c - p_c^0, & d_{\parallel} &= \hat{a}^\top \Delta p, & e_{\parallel} &= d_{\parallel} - D, \\ d_{\perp} &= \Delta p - \hat{a} d_{\parallel}, & m_{\text{move}} &= \sqrt{e_{\parallel}^2 + \frac{1}{2}\|d_{\perp}\|_2^2}. \end{aligned} \quad (6.13)$$

The compiled views are

$$\phi_{\text{move}}^{\text{mpc}} = \frac{1}{2} \left(e_{\parallel}^2 + \|d_{\perp}\|_2^2 \right), \quad P_{\text{move}} = \mathbf{1}[m_{\text{move}} \leq \epsilon_p], \quad c_{\text{move}}^{\text{rl}} = B_{D/2}(m_{\text{move}}). \quad (6.14)$$

The predicate and learning path therefore weight squared lateral deviation by one half inside m_{move}^2 , whereas the MPC penalty uses its full value. This is an implementation-specific numerical asymmetry, not a change in the grounded directional relation.

6.4.3 Orientation Constraints

Let $\hat{e}_i$ and $\hat{e}_j$ be selected signed canonical axes. Once resolved in world coordinates,

$$\hat{a}_c = R(\widehat{q_c^W})\hat{e}_i, \quad \hat{a}_r = R(\widehat{q_r^W})\hat{e}_j, \quad s = \hat{a}_c^\top \hat{a}_r, \quad (6.15)$$

where the reference axis may instead be a fixed world direction. The `axis_align` operation selects one of four errors and predicate thresholds:

relation	$\eta(s)$	$\bar{\epsilon}(\epsilon_\theta)$
codirectional	$1 - s$	$1 - \cos \epsilon_\theta$
antiparallel	$1 + s$	$1 - \cos \epsilon_\theta$
bipolar	$1 - s $	$1 - \cos \epsilon_\theta$
orthogonal	$\sqrt{s^2 + 10^{-18}}$	$\sin \epsilon_\theta$

(6.16)

For all four modes,

$$\phi_{\text{align}}^{\text{mpc}} = Q(\eta), \quad P_{\text{align}} = \mathbf{1}[\eta \leq \bar{\epsilon}], \quad c_{\text{align}}^{\text{rl}} = \begin{cases} B_{0.25}(\eta/2), & \text{co-, anti-, or bipolar,} \\ B_{0.25}(|\eta|), & \text{orthogonal.} \end{cases} \quad (6.17)$$

For `point_toward`, the controlled signed axis aims from the end effector toward a target point:

$$\hat{d} = \widehat{{}^W p^\star - {}^W p_E}, \quad \eta_{\text{point}} = 1 - \hat{a}_c^\top \hat{d}, \quad (6.18)$$

$$\phi_{\text{point}}^{\text{mpc}} = Q(\eta_{\text{point}}), \quad P_{\text{point}} = \mathbf{1}[\eta_{\text{point}} \leq 1 - \cos \epsilon_\theta], \quad c_{\text{point}}^{\text{rl}} = B_{0.25}(\eta_{\text{point}}/2). \quad (6.19)$$

The current implementation is narrower than the prompt signature here: the MPC residual accepts only a robot frame, and its target is ${}^W p_O(t) + \delta$ regardless of the supplied basis. The batched predicate can use a gripped object's orientation for a_c , but $\hat{d}$ still begins at the configured end effector. These restrictions are retained explicitly rather than presenting the schema's intended generality as implemented fact.

For `directional_rotate`, a positive angle θ about the requested signed axis $\hat{a}$ defines

$$q_\Delta = \left(\cos \frac{\theta}{2}, \hat{a} \sin \frac{\theta}{2} \right), \quad q_\Delta^W = \begin{cases} q_R^0 \otimes q_\Delta \otimes (q_R^0)^*, & \text{object or robot-frame basis,} \\ q_\Delta, & \text{world basis,} \end{cases} \quad (6.20)$$

and, for controlled entry orientation q_c^0 , the implemented target and views are

$$\begin{aligned} q^\star &= q_\Delta^W \otimes q_c^0, & e_q &= q_c - q^\star, \\ \phi_{\text{rotate}}^{\text{mpc}} &= Q(e_q), & P_{\text{rotate}} &= \mathbf{1}[\|e_q\|_2 < \epsilon_\theta], \\ c_{\text{rotate}}^{\text{rl}} &= B_{0.25}(\|e_q\|_2). \end{aligned} \quad (6.21)$$

Although the field is named an angular tolerance, the active predicate compares it directly with quaternion-coordinate distance; this metric is neither a geodesic angle nor invariant to the equivalent signs q and $-q$. The prompt also forbids combining this finite rotation with an

`axis_align` that controls or references the rotating frame. Alignment belongs in an earlier approach stage and rotation in the subsequent manipulation stage.

6.4.4 Entry Invariants and Gripper Constraints

The two freeze operations encode invariants, not new goals. Position freezing uses

$$e_f^p = p_c(t) - p_c^0, \quad \phi_{f,p}^{\text{mpc}} = Q(e_f^p), \quad P_{f,p} = \mathbf{1}[\|e_f^p\|_2 \leq \epsilon_p], \quad c_{f,p}^{\text{rl}} = B_{0.25}(\|e_f^p\|_2). \quad (6.22)$$

For orientation, the predicate and learning path use the sign-invariant geodesic distance between normalized live and entry quaternions,

$$\theta_f = 2 \arccos\left(\text{clip}\left(|\hat{q}_c(t)^\top \hat{q}_c^0|, 0, 1\right)\right), \quad P_{f,q} = \mathbf{1}[\theta_f \leq \epsilon_\theta], \quad c_{f,q}^{\text{rl}} = B_{0.25}(\theta_f). \quad (6.23)$$

The current MPC path instead uses

$$\phi_{f,q}^{\text{mpc}} = Q(q_E - q_E^0), \quad (6.24)$$

and always freezes the configured end-effector state frame, presently ignoring the operation's controlled entity and target. Consequently, `freeze_orientation` should be described as the entry invariant it is intended to express while preserving this backend-specific limitation.

For normalized optimizer gripper state $g = (g_L, g_R)$,

$$g_{\text{closed}} = (0, 0), \quad g_{\text{open}} = (1, 1), \quad \phi_z^{\text{mpc}} = Q(g - g_z), \quad c_z^{\text{rl}} = B_1(\|g - g_z\|_2), \quad (6.25)$$

for $z \in \{\text{closed}, \text{open}\}$. The measured openness is normalized from calibrated finger-joint limits,

$$o = \frac{1}{n_g} \sum_{j=1}^{n_g} \frac{q_j - q_j^{\text{closed}}}{q_j^{\text{open}} - q_j^{\text{closed}}}, \quad P_{\text{close}} = \mathbf{1}[o \leq 0.1], \quad P_{\text{open}} = \mathbf{1}[o \geq 0.9]. \quad (6.26)$$

The distinction between `close_gripper` and `object_grasped` is semantic and measurable: the former tests only mechanical closure, whereas the latter requires a held object. Let F_L, F_R be the two filtered finger-contact magnitudes, let F_{max} be the configured maximum finger-force scale, let o_{max} be the maximum grasp openness, and define

$$\begin{aligned} o_c &= \text{clip}(o, 0.03, o_{\text{max}}), & \alpha_o &= \frac{o_c - 0.03}{o_{\text{max}} - 0.03}, \\ F_{\text{req}}(o) &= F_{\text{max}}(0.1 + 0.6\alpha_o), & C &= \mathbf{1}[0.1 < o \leq o_{\text{max}}], \\ B &= \mathbf{1}[F_L \geq F_{\text{req}}] \wedge \mathbf{1}[F_R \geq F_{\text{req}}], \\ A &= (\mathbf{1}[F_L \geq 10] \oplus \mathbf{1}[F_R \geq 10]) \wedge \mathbf{1}[o < 0.3] \wedge C. \end{aligned} \quad (6.27)$$

The robot-specific grasp predicate is

$$P_{\text{grasp}} = \begin{cases} (B \wedge C) \vee A, & \text{Boston Franka or Kinova Robotiq,} \\ \mathbf{1}[F_L \geq 5] \wedge \mathbf{1}[F_R \geq 5] \wedge C, & \text{Slichter Franka Panda.} \end{cases} \quad (6.28)$$

Slichter deliberately omits the generic one-finger exception A . In both cases, the MPC and learning views of `object_grasped` reuse the close-gripper costs in equation (6.25); only its predicate tests physical holding. When this operation appears, the complete stage predicate must remain true for five consecutive frames by default before the stage advances. It must therefore remain present in later pulling, carrying, and reorientation stages that assume the object is still held.

6.4.5 Condition Labels and Valid Term Combinations

Condition labels attach satisfaction or feedback semantics to an operation. `geometry` selects its geometric predicate; `gripper_opened` and `gripper_closed` describe intended finger state; and `manipulator_high_effort` and `manipulator_low_effort` describe arm/TCP interaction rather than finger squeeze. For exponentially filtered arm effort and a configured high-effort threshold E_{high} ,

$$\bar{\tau}_t = 0.2\tau_t + 0.8\bar{\tau}_{t-1}, \quad E_t = \sum_{j=1}^7 |\bar{\tau}_{t,j}|, \quad P_{\text{low}} = \mathbf{1}[E_t \leq 10], \quad (6.29)$$

sustained high effort is

$$h_t^E = \begin{cases} \min(h_{t-1}^E + 1, 5), & E_t > E_{\text{high}}, \\ 0, & E_t \leq E_{\text{high}}, \end{cases} \quad P_{\text{high}} = \mathbf{1}[h_t^E \geq 5]. \quad (6.30)$$

Effort labels have no standalone MPC residual or bounded learning cost. They can instead provide a local stop alternative inside a relevant term. For example, an into-contact term can evaluate

$$T_{k,j} = P_{k,j}^{\text{geometry}} \vee P_{\text{high}}, \quad D_k = \bigwedge_{j=1}^{m_k} T_{k,j}. \quad (6.31)$$

Thus alternatives are internal to one required term; the outer stage remains conjunctive as in equation (6.5). Normal terms use `logic=and`. The prompt and compiler additionally enforce the following composition rules: a stage contains at most one position operation among `reach_point`, `directional_move`, and `freeze_position`; a freeze operation cannot substitute for a new pose goal; and `close_gripper` and `object_grasped` cannot appear together. `directional_move` is restricted to manipulation and release stages and is selected only when the grounded requirement explicitly supplies a direction and positive distance;

an approach or otherwise concrete target uses `reach_point`. Every term must be justified by the current atomic stage. If a required clause is outside $\mathcal{L}$, it is omitted from executable terms and recorded explicitly in the grounding rationale rather than approximated by an unrelated constraint.

The runtime retains `reach_distance` and `free_move` for compatibility with older saved programs, but the current constraint-writing prompt does not expose them. They are therefore not members of the active prompt-visible $\mathcal{L}$ enumerated in table 6.2.

6.4.6 Worked Drawer-Stage Compilation

The drawer example makes the shared contract concrete without inventing unreported thresholds. The scene registry names the drawer, its handle, the gripper, the handle axis, and the drawer’s operating direction. During approach, axis-alignment and reach-point relations place the gripper at the handle while the gripper-open predicate prevents premature closure. During pull, a directional-move relation measures drawer displacement relative to the stage-entry snapshot, a fix-orientation relation limits rotation relative to that snapshot, and a grasp predicate requires continued attachment.

The independent evaluator remains deliberately different: it declares the physical drawer task successful only after at least 12 cm of drawer-joint displacement. The table therefore explains how a program is unified without making its internal completion predicate the ground truth.

6.5 Language and Scene Grounding

Formation receives a natural-language instruction $\mathcal{I}$, registered scene semantics $\mathcal{F}$ containing named objects and frames, robot capabilities, and the typed operator library $\mathcal{L}$. An LLM

Table 6.3: Every consumer is generated from the same grounded relation. Symbols are shown because the active paper does not publish the accepted numerical tolerances or weights.

Grounded relation	MPC interface	Monitor interface	Learning/diagnostic interface
Axis alignment: gripper–handle	Penalize angular misalignment	Test against ϵ_{axis}	Alignment progress; frames and axes
Reach handle point	Penalize gripper–handle offset	Test against ϵ_{reach}	Reach progress; measured distance
Directional drawer move	Penalize remaining signed displacement	Test displacement from entry snapshot	Pull progress; direction and margin
Fix gripper orientation	Penalize entry-relative rotation	Test against ϵ_{rot}	Orientation progress; snapshot and margin
Gripper grasped	Enforce/penalize grasp violation	Boolean attachment predicate	Grasp status and failure attribution

agent decomposes the instruction into stages, chooses grounded relations and axes, binds all arguments to registered frames, and emits geometry and contact conditions. In the updated implementation, the receiver inserts tolerance fields after JSON parsing and before schema compilation; the LLM does not propose them. The current constraint prompt has no per-term weight field: compilation initializes weights to one unless a separate scalar interface supplies them, and later calibration may change those weights without changing the accepted relation. The output is schema checked before simulation. This constrains language to select and parameterize known physical operators; it does not permit arbitrary code to redefine the robot or evaluator. The exact formation and correction interfaces are preserved in section B.6, rather than reconstructed from accepted programs or model responses. Figure 6.2 connects the three grounding transformations, MPC verification, and diagnostic repair to the mentor-data and policy-learning handoff developed in chapter 7. The accepted program

supplies both successful MPC behavior for Stage-BC and persistent reward and monitor interfaces for bounded residual learning. Formation repair and collapse-triggered learning calibration are separate feedback loops: only the former may revise the task semantics. Their combined procedural form appears as algorithm B.1 in chapter B.

Grounding is the main point at which apparently correct language can fail. A direction term may be expressed in the world frame instead of the drawer frame; an orientation constraint may refer to the object rather than the grasp frame; or a completion tolerance may be too loose. Diagnostic metadata retains the source stage, term, frames, axis, and threshold so that a failed rollout can be mapped back to the smallest editable program element.

Here, *verification* denotes empirical testing with MPC in closed-loop physics rather than formal proof: grounding produces typed relations, compilation creates their aligned interfaces, and verification tests the resulting candidate. Acceptance is therefore evidence over the sampled initial conditions, not a guarantee of correctness for every state.

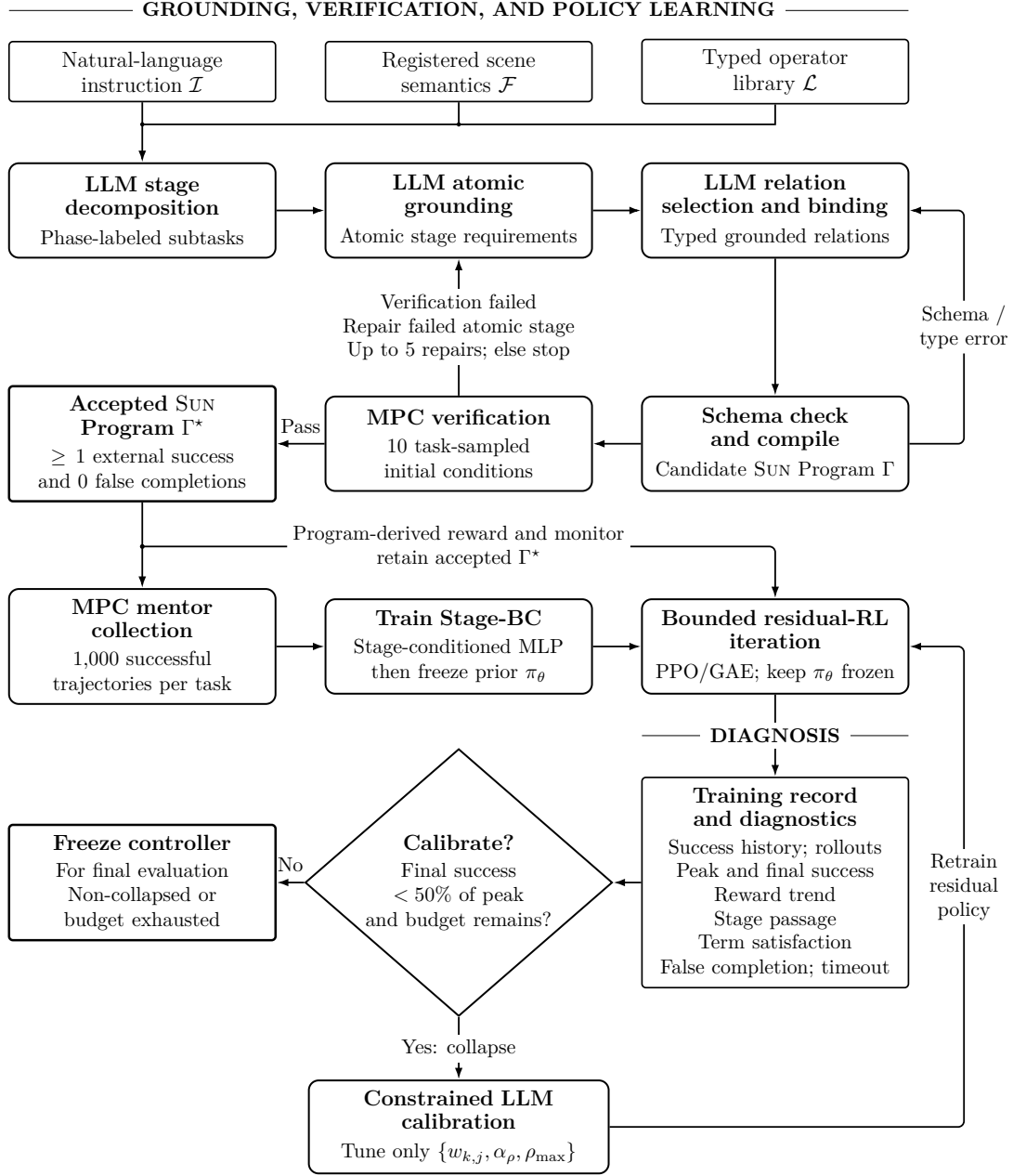
6.6 Physical Verification and Diagnostic Repair

For stage k , the compiled MPC terms form

$$\ell_k(\tilde{x}) = \sum_{j=1}^{m_k} w_{k,j} \phi_{k,j}^{\text{mpc}}(\tilde{x}). \quad (6.32)$$

At state x_t , direct multiple shooting solves the active-stage finite-horizon problem reported in the manuscript,

$$\begin{aligned} \min_{x_{1:H}, u_{0:H-1}} \quad & \sum_{h=0}^{H-1} \ell_{z_t}(\tilde{x}_h) \\ \text{s.t.} \quad & x_0 = x_t, \quad x_{h+1} = f(x_h, u_h). \end{aligned} \quad (6.33)$$



Fixed after acceptance: predicate tolerances, grounded relations, Boolean logic, stage order, and term definitions.

Learning budget: at most 5 residual-RL iterations.

Figure 6.2: Reported formation-to-learning workflow. Verification diagnostics repair failed stages; schema/type errors return to grounded relations. Acceptance supplies MPC mentor behavior and the persistent program. Stage-BC becomes the frozen behavioral anchor. Only late success collapse with budget remaining triggers scalar calibration; accepted task semantics remain fixed.

The active stage and its entry snapshot remain fixed inside a solve. Only the first optimized action is executed before the state is observed, predicates are evaluated, and the problem is solved again. The manuscript leaves the specific state/action parameterization, horizon, control rate, admissible constraints, and solver tolerances unreported.

Each candidate is tested from ten task-sampled initial conditions. A program is accepted only if at least one rollout satisfies g_{ext} and no rollout exhibits false completion. A failed verification report identifies unsatisfied terms and their measured violations. Those diagnostics return to the failed atomic stage, where the repair agent revises the affected requirement and its grounded relations. The decomposition- and typed-relation-repair interfaces are reproduced in tables B.6, B.7, B.12 and B.13. Up to five repair attempts are allowed. In the reported protocol, once accepted, relations, tolerances, logic, and order are frozen and the program becomes the canonical semantic artifact for the task.

6.7 Formation Reliability

The nine-task Isaac Lab suite spans E-stop pushing and twisting (T0, T3), two- and three-cube stacking (T1, T6), drawer opening and drawer-open-plus-place (T2, T7), bottle righting (T4), marker insertion into a cup (T5), and hanging a cup by its handle (T8). It therefore varies contact mode, orientation precision, object count, and horizon rather than merely changing object appearance.

Five independent formation-and-verification runs were performed per task. Of 45 candidates, 29 passed directly and 16 entered diagnostic repair; 14 repairs succeeded, raising formation yield from 64.4% to 95.6% (43/45). T7 alone had unrecovered runs (3/5 final acceptance) and accounted for 11 of 30 recovery cycles. The 15-stage T6 program passed all five runs without repair, showing that phase heterogeneity, not stage count alone, governs

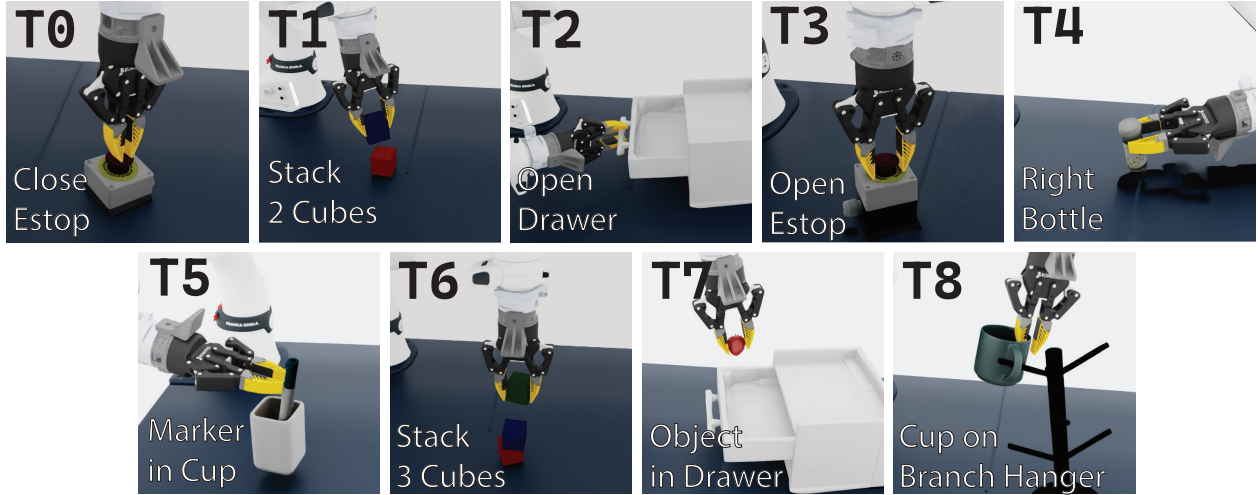


Figure 6.3: The nine Isaac Lab 5.1 tasks span constrained translation and rotation, stacking, drawer manipulation, reorientation, insertion, and dexterous hanging. T0/T3, T1/T6, and T2/T7 provide paired increases in interaction or compositional horizon. Reproduced from the active SUN paper’s original vector source.

Table 6.4: Programs contain 3–15 stages and 12–53 constraints. Values follow the active SUN manuscript’s accepted formation runs.

Metric	T0	T1	T2	T3	T4	T5	T6	T7	T8	Mean
Stages K	3	8	6	6	7	9	15	14	10	8.67
Constraints	12	25	19	19	27	31	53	44	31	29.00
Formation iterations	1	1	1	1	2	2	2	1	1	1.33
Refinement iterations	0	0	0	0	1	1	0	0	2	0.44
LLM calls/task	11.2	19.8	15.2	16.2	24.6	29.2	37.0	32.8	28.8	23.87

Table 6.5: Each entry is out of five independent runs. Recovery cycles count term-level diagnostic-repair cycles.

Metric	T0	T1	T2	T3	T4	T5	T6	T7	T8	Total
Success without recovery	5	5	5	4	0	1	5	2	2	29/45
Recovery entry	0	0	0	1	5	4	0	3	3	16/45
Final success	5	5	5	5	5	5	5	3	5	43/45
Recovery cycles	0	0	0	2	5	9	0	11	3	30

formation difficulty.

Language cost is paid once per task. KUAFU averages 110.50k tokens per accepted task, versus 234.40k for Eureka’s one-time search. VoxPoser spends 16.99 calls and 31.60k tokens per rollout, exceeding KUAFU after two executions by calls and four by tokens. Persistence thus amortizes language interaction across executions.

6.8 Scope and Limitations

Formation assumes registered frames, known relation types, and a simulator for proposed contacts and motions. Its typed operator library improves reliability but needs new operators and physics for deformables, fluids, and unknown tools. The evaluation does not separately test unseen instructions, operators, semantic compositions, or scene registries. Because the stage machine is irreversible and the external evaluator manually authored, MPC verification establishes physical consistency only within the available model and vocabulary—not open-world task correctness.

CHAPTER 7

Optimal Control as Mentor for Program-Preserving Policy Learning

Publication basis. This chapter is based on the control-to-learning and ablation portions of “SUN: Persistent Programs for Language-Grounded Control-to-Learning-to-Real Policies” (Wang et al., 2026). Results follow the active RAL manuscript snapshot.

7.1 From Verified Control to Amortized Execution

An accepted MPC controller demonstrates that a SUN Program is executable, but it is not yet the desired deployment policy. MPC uses privileged state and a model, incurs online optimization cost, and can become brittle when the physical system differs from the simulator. A learned policy amortizes this computation, but a direct handoff risks losing precisely the task structure that made the controller valid.

KUAFU therefore transfers two mentor artifacts: successful MPC behavior and the accepted executable program. The trajectories provide a behavioral prior; the program continues to provide the active stage, progress costs, guards, internal completion, and diagnostic attribution. The separately authored g_{ext} supplies physical-success and false-completion outcomes. Policy learning changes how actions are produced, not what the task means.

7.2 MPC Mentor Demonstrations

For each task, the accepted MPC system is rolled out over randomized initial conditions until 1,000 successful trajectories are collected. A trajectory retains the observation o_t , executed action u_t , active stage z_t , and monitor outcome. Failed and false-complete runs are excluded from the mentor set. Because z_t is generated by the accepted program rather than inferred from trajectory time, demonstrations of different duration remain aligned by physical progress.

The mentor set is intentionally generated after program acceptance. Collecting from an unvalidated objective could train a high-accuracy policy for an invalid proxy; the independent criterion g_{ext} and false-completion check prevent that circularity.

7.3 Stage-Conditioned Behavioral Prior

The Stage-BC policy π_θ predicts a ten-step chunk of absolute actions from (o_t, z_t) . It is a simple deterministic feed-forward multilayer perceptron. Each action uses the implementation’s 11-dimensional absolute end-effector representation,

$$a_t = \begin{bmatrix} p_t^\top & (R_t^{6D})^\top & g_t^\top \end{bmatrix}^\top \in \mathbb{R}^{11}, \quad (7.1)$$

where $p_t \in \mathbb{R}^3$ is the Cartesian position, $R_t^{6D} \in \mathbb{R}^6$ is the continuous six-dimensional rotation representation, and $g_t \in \mathbb{R}^2$ contains the two gripper targets. Chunks are not allowed to cross a stage boundary. When fewer than ten actions remain in the stage, the final within-stage action is padded so that supervision never asks the learner to anticipate a transition that the monitor has not yet declared. Training minimizes normalized mean-squared error over the mentor chunks. A compact notation for that reported loss is

$$\mathcal{L}_{\text{BC}}(\theta) = \frac{1}{N} \sum_{n=1}^N \left\| D_u^{-1} (\hat{U}_n - U_n) \right\|_2^2, \quad (7.2)$$

where D_u denotes action-normalization scales; their values and the precise normalization convention are not given in the active manuscript. At runtime only the first action of the predicted chunk is executed. The program then reevaluates the stage, making the learned policy responsive to the actual transition rather than an open-loop schedule.

Stage-BC is a useful anchor but not a complete controller. Covariate shift and model mismatch compound over 3–15 stages; in the nine-task evaluation its task-macro success is 24.75%, compared with 83.76% for online MPC.

7.4 Bounded Residual Reinforcement Learning

The behavioral prior is frozen and an additive residual π_ψ is trained with PPO and generalized advantage estimation. The residual actor–critic is also a simple feed-forward multilayer perceptron; PPO changes the learning objective rather than introducing a more elaborate model family. The combined policy is

$$u_t = \text{clip}\left(\pi_\theta^{(1)}(o_t, z_t) + \alpha_\rho \pi_\psi(o_t, z_t), u_{\min}, u_{\max}\right), \quad \|\pi_\psi(o_t, z_t)\|_\infty \leq \rho_{\max}. \quad (7.3)$$

The prior supplies the mode of behavior accepted by MPC. The residual repairs state-distribution and model errors but is coordinate-wise bounded so it cannot arbitrarily replace that behavior. This is a semantic constraint as well as a regularizer: removing or broadly relaxing the residual bound collapsed all nine tasks to 0% in the component study.

For stage k , program costs compile into a normalized stage score

$$S_k(\tilde{x}) = 1 - \frac{\sum_{j=1}^{m_k} w_{k,j} c_{k,j}^{\text{rl}}(\tilde{x})}{\sum_{j=1}^{m_k} w_{k,j}}, \quad (7.4)$$

where a larger value indicates greater satisfaction of the active stage. The residual reward is

$$\begin{aligned} r_t = & \lambda_{\text{prog}} \left[S_{z_t}(\hat{x}_{t+1}) - S_{z_t, t-1}^{\text{best}} \right]_+ + \lambda_{\text{stage}} \mathbf{1}[z_{t+1} > z_t] \\ & + \lambda_{\text{true}} \mathbf{1}[TS_t] - \lambda_{\text{false}} \mathbf{1}[FC_t] - \lambda_{\text{act}} \|u_t\|_2^2. \end{aligned} \quad (7.5)$$

S^{best} is the best score reached since entering the active stage, initialized to zero and reset after a transition. The positive-part progress term rewards new progress rather than oscillation; the stage bonus rewards a guarded transition; and true- and false-completion terms keep program completion separate from physical task success.

Before the learning lineage is frozen, the current initial weight-selection system and user templates in tables B.14 and B.15 expose the stage semantics, neighboring stages, geometry-only operator arguments, and current weights. They select only relative reward weights. The BC-prior-informed variant in tables B.16 and B.17 adds diagnostic rollout evidence. Tolerance values are absent from these requests; the updated code assigns them through the versioned policy in section B.6.1. That policy’s pre-learning broadening is an implementation distinction from the fixed-boundary protocol reported below, not an LLM calibration action or a newly evaluated result.

Calibration is triggered by a specific training failure rather than invoked after every run. A residual-RL iteration is classified as *collapsed* only when its final training-success rate falls below 50% of the peak reached during that iteration. This test detects the failure mode in which the residual policy reaches useful behavior and then loses more than half of that

success by the end of training, indicating drift away from the frozen Stage-BC anchor; a smaller late dip does not trigger calibration.

When collapse occurs and fewer than five residual-RL iterations have been used, learning and reward trends, stage-passage and term-satisfaction statistics, and outcome counts are supplied to an LLM through a constrained schema. The external evaluator itself remains hidden; only its binary outcomes contribute to the reported statistics. The LLM may tune the relative reward weights $w_{k,j}$, residual scale α_ρ , and residual bound $\rho_{\max}$ before another complete residual-RL iteration. Predicate tolerances, term definitions, grounded frames, Boolean logic, and stage order remain fixed at their MPC-accepted values in this reported protocol. Refinement stops at the first non-collapsed configuration or when the five-iteration budget is exhausted, after which the controller is frozen for evaluation. The current implementation’s BC-informed weight and residual-authority pairs are reproduced in tables B.16 to B.19; their complete inventory and version provenance appear in section B.6. The learning portion of figure 6.2 connects the MPC mentor and Stage-BC handoff to this trigger, constrained update, and termination logic. Its calibration loop returns only to residual learning, not to program formation or Stage-BC training. The corresponding procedural form appears as algorithm B.1 in chapter B.

7.5 Experimental Protocol

The evaluation uses the nine tasks defined in chapter 6. All LLM calls use GPT-5.5; physical simulation uses Isaac Lab 5.1; state-policy evaluation uses 8,192 parallel environments; and rollouts are capped at 180 seconds. Every method receives the same task-specific initial-condition sampler. Performance is the unweighted mean of the nine external-success rates, so a short task cannot dominate a long one through additional rollouts.

The active manuscript does not expose the numerical MPC horizon and control rate, Stage-BC layer sizes and optimizer schedule, PPO/GAE hyperparameters, task-specific residual bounds, or complete domain-randomization ranges. The equations in this chapter define the evaluated interfaces, but those omitted settings are an artifact-level reproducibility requirement and are not inferred here.

The principal controls separate accepted behavior, program supervision, and bounded correction: online MPC; Stage-BC; bounded residual learning with sparse success reward; the complete KUAFU handoff; stage-free cloning; pure RL; stage-free BC plus residual RL; TD-MPC2 with and without a behavioral prior; Eureka reward search; and direct VoxPoser execution.

7.6 Results

The complete L42 handoff reaches 82.03% task-macro success, nearly matching MPC’s 83.76% while no longer optimizing online. Across three independently formed and trained lineages, the mean is 79.43%. False completion remains low: 0.58% for L42 and 0.60% across three lineages. This check reduces the concern that apparent success comes from exploiting the program’s completion predicate.

The components are complementary. Stage-free BC reaches 9.44%; pure RL with a naive reward reaches 0.07%; pure RL with the SUN reward reaches 0%; and stage-free BC plus residual RL reaches 18.56%. TD-MPC2 improves from 12.63% to 31.95% when given the prior, but remains concentrated on a few tasks. Eureka’s best candidate within five search rounds reaches 18.24%, while its final policy falls to 12.85%, revealing non-monotonic reward-search behavior. Direct VoxPoser execution reaches 10.00%.

Horizon-paired tasks quantify retention as sequences lengthen. L42 loses 8.53 points from

Table 7.1: External task success in percent. L42 is the primary lineage used for the fixed downstream comparisons; the final row reports the taskwise mean over three independently formed and trained lineages.

Method	T0	T1	T2	T3	T4	T5	T6	T7	T8	Mean
MPC	100.00	96.15	100.00	89.29	62.50	58.14	94.34	70.92	82.50	83.76
Stage-BC	36.01	10.79	25.11	29.80	22.47	31.82	13.81	38.80	14.16	24.75
Bounded RRL + sparse reward	58.67	87.20	49.56	58.67	58.96	0	0	7.95	0	35.67
KUAFU (L42)	98.48	95.78	98.01	84.62	58.80	69.70	87.25	95.21	50.43	82.03
KUAFU (three-lineage mean)	98.37	94.75	97.58	82.09	56.24	62.04	84.72	94.06	44.99	79.43

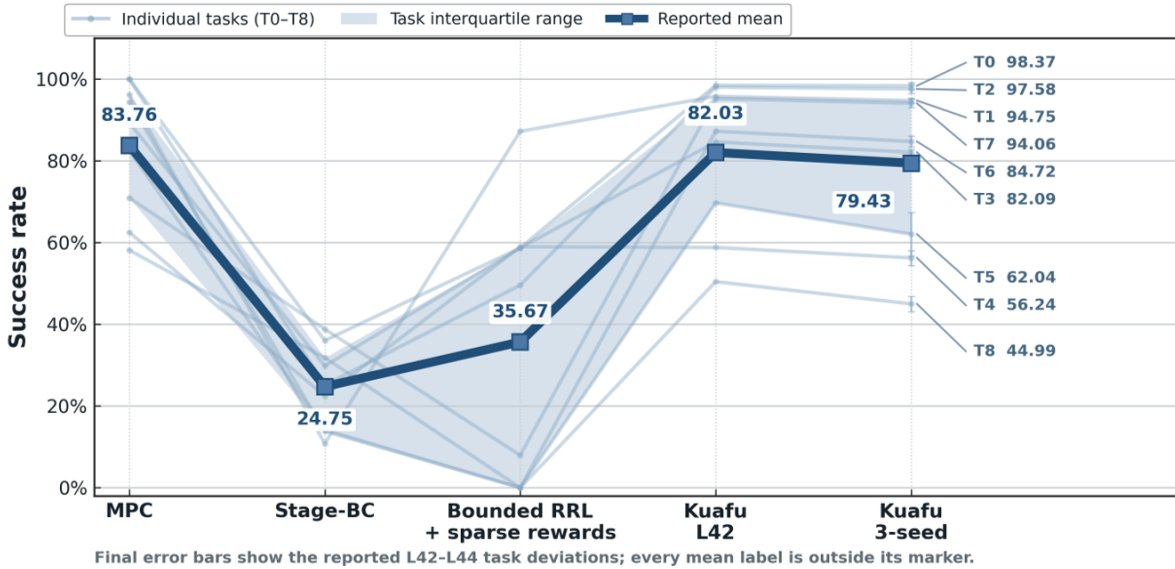


Figure 7.1: The complete handoff approaches MPC performance. The plot compares MPC, the Stage-BC mentor prior, bounded residual learning with only sparse reward, and the complete L42 and three-lineage handoffs. It is a supplied SUN project plot whose values are cross-checked against the active paper table.

Table 7.2: One-GPU worker-hours, including mentor collection. Formation is inexpensive relative to physical simulation and policy learning.

Method	Formation/search	MPC	BC	RRL/final	Total
KUAFU	1.55	69.03	3.21	48.86	122.67
Eureka	412.24	—	—	67.36	479.60

two-cube stacking T1 to three-cube stacking T6, 2.80 points from drawer opening T2 to open-and-place T7, and 13.86 points from close-E-stop T0 to open-E-stop T3. It retains 91.4% of the shorter-task performance across the three pairs; the matched sparse-reward handoff falls from 65.14% to 22.21%. No non-KUAFU learned handoff or baseline reaches 50% on T6 or T7.

End-to-end compute is 122.67 GPU-hours for KUAFU versus 479.60 for Eureka, a $3.91\times$ difference. MPC verification and mentor collection dominate KUAFU at 69.03 hours; formation itself is 1.55 hours. The advantage is therefore not explained by giving the proposed method a larger search budget.

7.7 Discussion and Limits

Optimal control is a mentor in two senses: it supplies accepted behavior and it physically verifies the task structure that later supervises learning. Neither artifact alone is sufficient. The policy results support a constrained division of labor in which imitation provides the behavioral mode, residual RL adapts locally, and the persistent program retains semantics and diagnosis.

These conclusions remain bounded by state-based simulation and the nine-task operator library. The 180-second cap, task samplers, and independent evaluators define success

operationally; different dynamics or sensing can change the ranking. The strong failure under an unbounded residual is an empirical result, not a universal theorem about residual policies. Finally, irreversible stage logic cannot recover when a later collision invalidates an earlier grasp. A future program should support guarded rollback without reintroducing semantic drift.

CHAPTER 8

Program-Guided Data Generation and Physical Deployment

Publication basis. This chapter is based on the data-generation, visual-policy, and sim-to-real portions of “SUN: Persistent Programs for Language-Grounded Control-to-Learning-to-Real Policies” (Wang et al., 2026). Results and architecture coverage follow the active RAL manuscript snapshot.

8.1 From Task Policy to Data Engine

After residual learning, the state controller replaces online MPC and becomes a parallel data generator. The accepted SUN Program remains active: it supplies the stage z_t , evaluates guards, detects terminal success and false completion, and attributes failures to program terms. Randomized initial object and robot states are sampled from the same task distributions used for policy evaluation.

This persistence is confined to production. Retained trajectories contain the downstream observation and action streams; the visual policy receives neither z_t nor any other monitor output during training or deployment. Consequently, the program improves the provenance and selection of supervision without being a privileged input to the final visual policy.

8.2 Outcome and Trajectory-Quality Definitions

Every generator is evaluated with the same independently authored physical criterion g_{ext} . For a successful trajectory of length T , the diagnostics include completion time, end-effector path length, action path length, and action second difference:

$$L_{\text{eef}} = \sum_{t=1}^T \|p_t - p_{t-1}\|_2, \quad (8.1)$$

$$L_u = \sum_{t=1}^T \|u_t - u_{t-1}\|_2, \quad (8.2)$$

$$J_{\Delta^2 u} = \sum_{t=1}^{T-1} \|u_{t+1} - 2u_t + u_{t-1}\|_2. \quad (8.3)$$

For reporting, each metric is normalized by the task-specific median among successful MPC trajectories (MPC50). A rollout is Q95-qualified only if it is successful and all four diagnostics are no greater than the corresponding task-specific 95th percentile among successful MPC trajectories. Q95 yield is the fraction of evaluated rollouts satisfying this joint criterion. It measures mentor-aligned efficiency and smoothness, not universal trajectory quality; the fixed-learner test below is therefore the primary data-utility comparison.

The human reference is collected through simulation teleoperation with an AgileX PIKA gripper. Measured hand poses set simulated end-effector targets, while the simulated robot remains subject to dynamics, collision, and task constraints. The active manuscript does not report a retained trajectory count or participant demographics, so earlier draft-specific values are not asserted here.

Table 8.1: Success in percent from 500 successful trajectories per task. The KUAFU row follows the active SUN table; its displayed task values average to 46.02%.

Source	T0	T1	T2	T3	T4	T5	T6	T7	T8	Mean
MimicGen	94.52	4.95	36.12	33.33	8.27	0	0	0	0	19.69
SkillGen	97.07	3.91	30.65	62.00	8.05	0	0	0	0.26	22.44
Isaac Lab Mimic	90.78	14.21	40.92	36.54	9.35	0	0	0	0	21.31
Isaac SkillGen	72.82	1.04	22.54	15.25	5.47	0	0	0	0.26	13.04
KUAFU	95.70	58.70	55.50	85.30	0.78	15.80	12.70	69.00	20.70	46.02

8.3 Data Utility, Quality, and Throughput

The primary utility experiment gives each source the same budget of 500 successful trajectories per task and trains the same DP3 learner (Ze et al., 2024). This separates the value of a trajectory from how many trajectories a generator can produce. The seed-42 KUAFU source is fixed in advance and used for all nine tasks; lineages L43 and L44 are not selected task-by-task.

KUAFU produces nonzero downstream success on all nine tasks and reaches 46.02% task-macro success, $2.05\times$ the 22.44% of the strongest alternative source, SkillGen (Garrett et al., 2025). On T5–T7 all four alternatives are 0%, whereas KUAFU reaches 15.80%, 12.70%, and 69.00%. The advantage is therefore not limited to high-yield short tasks.

KUAFU attains 57.3% Q95 yield, compared with 20.4% for MimicGen. Among successful rollouts it also has the shortest normalized end-effector and action paths and the smallest action second difference. These gains are consistent with the control lineage: the learned generator is anchored to smooth, constraint-aware MPC behavior instead of only being filtered after rollout.

Table 8.2: Lower diagnostic ratios are better. Human Q95 is zero because all four thresholds must be satisfied jointly; this does not imply that human motion is intrinsically poor.

Source	Q95 (%)	Time	EEF path	Action path	Action Δ^2
MimicGen	20.4	1.11	1.14	1.94	1.27
SkillGen	6.9	1.66	1.15	3.01	1.28
Isaac Lab Mimic	12.1	1.12	1.38	1.96	1.25
Isaac SkillGen	6.2	1.61	1.46	2.89	1.21
KUAFU	57.3	1.04	0.67	0.52	0.28
Human reference	0.0	3.06	1.59	2.81	1.25

Steady-state production is measured as minutes of successful trajectory per active production hour. Task-macro throughput is 246.79 minutes/hour for KUAFU, 23.34 for human teleoperation, 29.36 for MimicGen, 26.48 for SkillGen, 32.19 for Isaac Lab Mimic, and 24.79 for Isaac SkillGen. The KUAFU-to-human ratio is $10.57\times$. This metric permits more than 60 trajectory-minutes per wall-clock hour because 8,192 environments run in parallel.

Controller readiness requires 13.63 GPU-hours on average. At that time a human would have produced 494 demonstrations; KUAFU catches up at 513 and needs a further 29.10 minutes of active collection to break even. The across-task readiness range is 4.84–25.52 GPU-hours and the break-even range is 225–854 demonstrations, so one-time cost should be considered when only a small dataset is needed.

8.4 Visual-Policy Distillation

The visual-policy study compares single-task RGB, single-task point-cloud, and one-policy multi-task vision-language-action (VLA) learners. All models see only their declared visual observations and actions. The active manuscript does not state a training-trajectory budget for every architecture; its explicitly reported 1,000-trajectory budget applies to the physical-deployment policies.

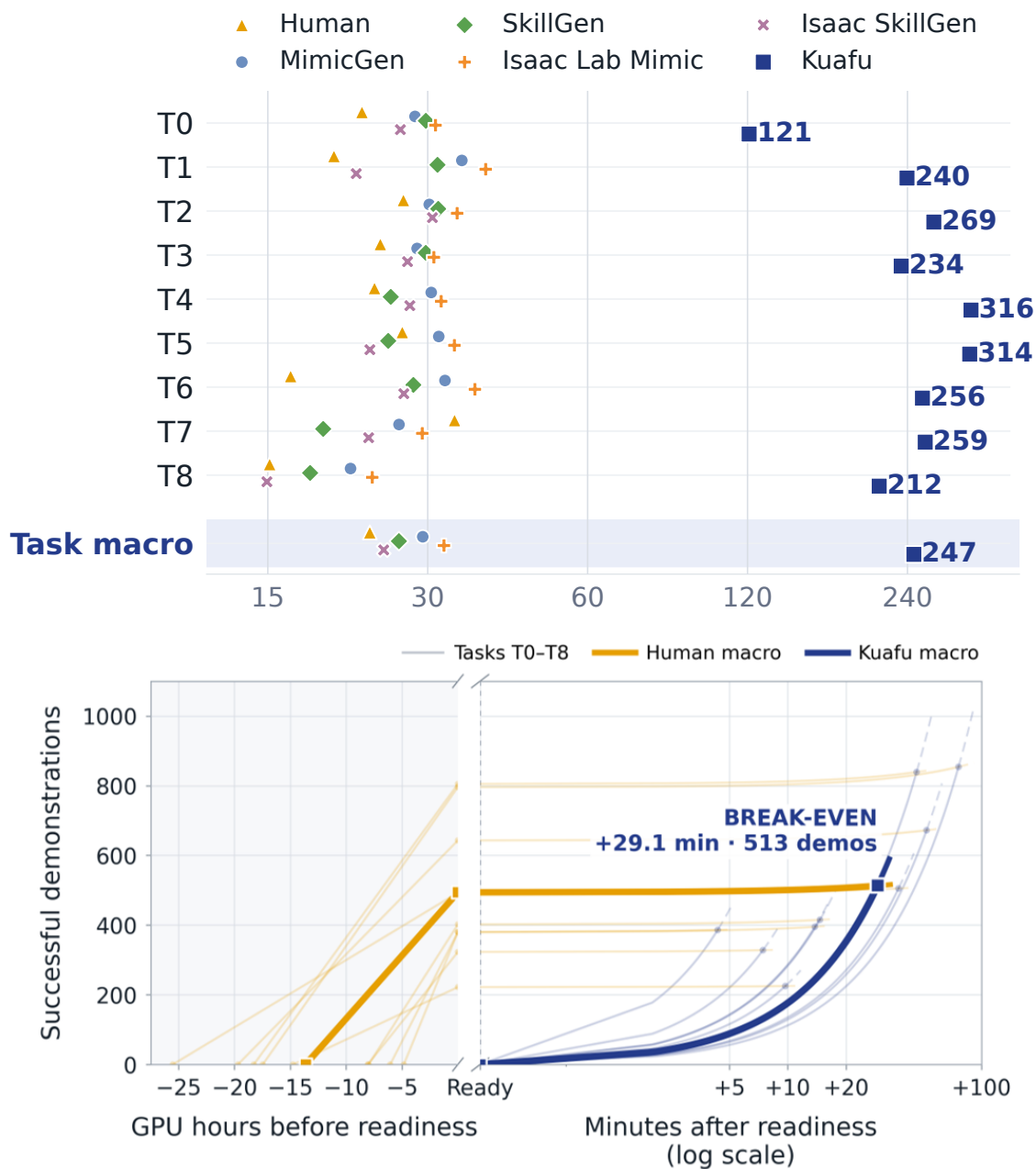


Figure 8.1: Top: successful trajectory-time throughput by task and task-macro average, measured in successful trajectory-minutes per active production hour. Bottom: controller readiness and the post-readiness collection time needed to overtake human demonstration production. Both panels are supplied SUN project plots.

Table 8.3: Task-macro success in percent. Each single-task family trains a separate policy per task; VLA methods train one policy across T0–T8.

Modality	Architecture	Mean success (%)
RGB	Diffusion Policy / DP (Chi et al., 2025)	38.52
RGB	ACT (Zhao et al., 2023)	35.40
RGB	ManiFlow-RGB (Yan et al., 2025)	34.09
RGB	SmolVLA (single-task) (Shukor et al., 2025)	26.73
Point cloud	ManiFlow-PC (Yan et al., 2025)	46.46
Point cloud	DP3 (Ze et al., 2024)	46.02
Point cloud	FlowPolicy (Zhang et al., 2025b)	21.01
Point cloud	MP1 (Sheng et al., 2026)	12.94
VLA	$\pi_{0.5}$ / WALL-X / SmolVLA (Black et al., 2025; Zhai et al., 2025; Shukor et al., 2025)	20.67 / 11.79 / 2.35

The strongest point-cloud architecture, ManiFlow-PC, reaches 46.46%, followed closely by DP3 at 46.02%; the strongest RGB method reaches 38.52%. RGB and point-cloud families obtain nonzero per-task maxima on all nine tasks, while the union of the VLA policies has nonzero coverage on seven. The per-task modality maxima are 43.72% for RGB and 51.53% for point clouds, compared with 82.03% for the privileged state controller. That gap measures losses from partial observability, architecture, and visual conditioning rather than a change in task definition.

Table 8.4: Policies are trained on 1,000 simulated trajectories per task and receive no real demonstrations or fine-tuning.

Robot	Gripper	Metric	T1	T2	T4	T5	T7	T8
Franka FR3	Franka Hand	Success/trials	3/20	11/20	—	—	7/20	9/30
		Rate (%)	15.0	55.0	—	—	35.0	30.0
Franka FR3	Robotiq 2F-85	Success/trials	—	—	4/10	—	—	—
		Rate (%)	—	—	40.0	—	—	—
Kinova Gen3	Robotiq 2F-85	Success/trials	—	—	—	2/6	—	—
		Rate (%)	—	—	—	33.3	—	—

8.5 Zero-Shot Physical Deployment

Simulation reproduces each workspace with two or three calibrated RGB-D cameras and constrained pose randomization. The DP3 policies are deployed without real-world fine-tuning. The active manuscript does not report the camera models, calibration parameters, or point-cloud preprocessing convention. Two arms and three robot–gripper configurations test whether the generated supervision transfers beyond a single setup.

Across six tasks, the policies achieve 36 successes in 106 trials. The unweighted task-macro rate is 34.72%; the pooled rate is 33.96%, with a 95% Wilson interval of 25.6–43.4%. Every tested task has at least one success. Precise grasp acquisition is the dominant observed failure: T1 has the lowest rate at 15%, and grasping is the most frequent failure mode on T7. This result is an initial zero-shot feasibility demonstration, not evidence of solved physical robustness; the number of trials is small and the setups retain calibrated camera configurations and constrained simulation ranges.

8.6 Discussion and Limitations

The data result completes the artifact lifecycle. Formation produces a task program; MPC makes it executable; Stage-BC and bounded residual learning amortize execution; the learned controller produces high-yield supervision; and a visual policy discards privileged program input before zero-shot deployment. The important continuity is semantic provenance, not that one representation is used by every model.

Q95 is normalized to successful MPC and can penalize safe, slower human behavior. The stronger fixed-budget DP3 result covers one learner and simulator; throughput measures GPU-parallel production after controller readiness, not single-robot speed. The 106 calibrated physical trials omit long-term drift, unregistered objects, moving people, and unmodeled contacts, bounding the deployment claim.

CHAPTER 9

Discussion and Conclusion

9.1 Toward Fit-Centered Robot Autonomy

This dissertation treats competence as relational rather than as a property of the policy or robot in isolation. A capable controller cannot pass through a physically excluded room; an accessible object cannot be used correctly if the task’s physical meaning is missing; and a correct task formulation has limited value if learning replaces it with a proxy. Coexistence therefore requires both a place in the room and a place in the routine.

The two primary contributions intervene at the environment and task-program levels. Scene rearrangement changes the environment when planning cannot create clearance, while SUN preserves grounded task meaning across formation, control, learning, data generation, and physical deployment. Two collaborative studies provide supporting foundations: VKC coordinates the robot, grasp, and object within one motion representation, and effect-centered tool use changes the task specification from a human trajectory to the physical result it causes. The dissertation claims only the author’s scoped contributions to those two supporting studies.

Table 9.1: The two primary contributions and two supporting collaborative studies adapt different components while retaining explicit artifacts that subsequent reasoning can inspect.

Role / gap	Adapted component	Representation	Validation	Retained artifact
Supporting: body/space	Robot motion	Virtual kinematic chain	Constrained whole-body planning	Unified chain and trajectory
Primary: room/access	Environment	Relation graph plus accessible-space and interaction fields	Held-out layouts, human ratings, robot task access	Co-activity layout
Supporting: single effect	Task goal	Physical relation graph	FEM outcome and robot-specific execution	Essential effect relation
Primary: long routine	Task program and policy	Typed staged SUN Program	Independent outcome, MPC verification, false-completion monitor	Accepted program plus learned controller

9.2 Cross-Chapter Synthesis

Several design principles recur across the primary contributions and supporting studies. First, representation determines what an optimizer can coordinate. The collaborative VKC study exposes previously separated motion variables as one chain. The scene graph likewise keeps human relations explicit when free-space optimization would otherwise erase them.

Second, task meaning should be separated from demonstrated motion. The collaborative tool-use study recovers force, velocity, and orientation relations that a different body can instantiate. SUN generalizes the same principle over a longer horizon: stage relations and guards define the routine independently of whether MPC, Stage-BC, a residual policy, or a visual policy produces the actions.

Third, validation should produce a reusable artifact. A successful MPC rollout is evidence, but the accepted program is what lets the evidence survive. Its lineage supports later monitoring, false-completion detection, and diagnostic repair. Learning is then an amortization

of physically validated structure rather than a semantic reset.

Finally, the environment is part of the autonomy system. This is literal in scene rearrangement, but it also appears in SUN through registered frames, task samplers, cameras, and external evaluators. Results must therefore be read with their environment assumptions, rather than as isolated policy scores.

9.3 Limitations Across the Dissertation

The body work optimizes motion for a supplied morphology; it does not co-design the robot. The scene work assumes static occupants, movable furniture, planar poses, known interaction regions, and category statistics that may not represent an individual household. The physical-effect work depends on calibrated FEM, watertight meshes, and a small set of fracture and cutting processes.

SUN assumes registered frames and a finite typed operator library. It does not yet ground deformable or fluid interactions without adding corresponding operators and physics. Its experiments do not separately test held-out instructions, terms, compositions, and semantic registries. Stage progression is irreversible, external success evaluators are manually authored, and all state controllers are learned in simulation.

The deployment evidence is initial rather than longitudinal. The 36/106 physical result uses calibrated RGB-D setups and constrained randomization; it does not cover camera drift, changing illumination, moving people, or arbitrary objects. Q95 is MPC-relative, throughput assumes parallel GPU simulation after controller readiness, and downstream conclusions are conditioned on the chosen architectures. These qualifications bound the claims without erasing the main result: a physically verified persistent representation can produce useful learned behavior and real successes.

9.4 Two Missing Links

The defense identifies two coupled gaps. The first is the full routine. An integrated coexisting robot should accept language or demonstration, infer the intended effect, navigate, select a tool, manipulate it, verify the effect, and return. This dissertation develops elements of that chain in separate systems; connecting them requires one objective and monitor to survive across mobile planning, tool selection, contact, recovery, and completion.

The second gap is functional grounding from perception. The present systems assume prior knowledge of grasp and contact regions, tool orientation, handling mode, and supported physical interactions. An unfamiliar household object does not arrive with those annotations. Larger interaction datasets and methods that transfer functional evidence across object geometry, task, and embodiment are needed to supply those arguments to the persistent program.

The gaps depend on each other. A program can preserve an objective only if perception supplies the correct functional entities; functional grounding is useful at scale only if the resulting knowledge remains coherent throughout the full routine.

9.5 Future Directions

One direction is a reversible, uncertainty-aware task machine. Guards would carry confidence, failed later stages could trigger safe rollback, and external outcome checks could revise the program without discarding its lineage. Physical verification would then reject not only deterministic violations but specifications whose success is too sensitive to perception or dynamics uncertainty.

A second direction is open-vocabulary operator discovery. Interaction data could propose

new residual types for deformables, fluids, and articulated tools, while simulation and physical trials test whether each proposed relation is causal, measurable, and compilable. The finite registry would grow through validated operators rather than unrestricted code generation.

The physical and environmental components can also be optimized jointly. Robot morphology, storage location, movable furniture, approach region, and task program affect one another. A unified design problem could choose which burden belongs to body, scene, or behavior while enforcing human preference and safety.

Finally, deployment should become continual. The real robot can compare monitor predictions with observed outcomes, detect systematic grasp or perception failures, and request targeted data. The accepted program supplies a structured place to attach those revisions, preserving what was learned while making the limits of each update visible.

9.6 Conclusion

Bringing robots into human life requires more than placing a capable machine in a house. The robot must be given a place in the room and a place in the routine. The dissertation’s two primary contributions construct these forms of fit: co-activity optimization adapts shared environments around embodied capability, and SUN preserves grounded task knowledge from formation through deployment. The supporting VKC and tool-use studies establish narrower motion and physical-effect foundations for that argument. Physical compatibility creates the opportunity to act. Persistent task meaning ensures that the activity remains the one people intended.

APPENDIX A

Shared System Conventions

Coordinate frames and state conventions. Rigid transforms use b_aT for the pose of frame b expressed in frame a . Position residuals are Euclidean, orientation residuals use the shortest unit-quaternion rotation, and scene poses are planar. SUN leaves x_t , o_t , and u_t vector parameterization unspecified. The stage-entry reference ξ_k is captured only when its guard advances.

Shared control interface. The SUN MPC uses $x_{h+1} = f(x_h, u_h)$; dimensions and backend parameterization are unreported. The monitor reads post-action state and never advances from a command alone. The visual-policy handoff retains observations and actions but removes privileged state, stage, and monitor channels.

Common evaluation definitions. Task success uses independently authored g_{ext} , and task-macro success averages across tasks. A false completion has true terminal predicates but false g_{ext} . Data-quality normalization uses task-specific successful-MPC medians and 95th percentiles from chapter 8, so values are not comparable as raw units.

Reported simulation settings. SUN uses Isaac Lab 5.1, GPT-5.5, 8,192 parallel environments, and 180-second rollouts. Scene statistics use 90% of SUNCG; co-activity tests 154 held-out rooms. FEM applies Incremental Potential Contact under 10% material variation.

APPENDIX B

SUN Program Language and Task Library

B.1 Type System and Term Signatures

Every relation has typed arguments bound to registered scene entities. The active manuscript illustrates axis-alignment, directional-move, fixed-orientation, reach-point, gripper-open, and gripper-grasped relations. It describes a finite typed operator library but does not publish an exhaustive schema catalog. The constraint-writing prompt specifies that catalog; section 6.4 presents its ten operations, grounded parameters, composition rules, and compiled equations. A term stores its grounding and tolerance together with a compiler-managed learning weight; typed binding precedes simulation. In the updated prompt interface the receiver, not the LLM, assigns the tolerance fields; see section B.6.1.

B.2 Compilation and Transition Invariants

For every accepted term, equation (6.3) generates an MPC interface, completion predicate, and RL interface. The active SUN manuscript permits their numerical forms to be operator-specific; semantic unification requires them to retain the same relation identity, grounding, stage membership, and satisfaction boundary. The term margin is $\gamma = \epsilon - e$, while the stage guard is the conjunction $D_k = \bigwedge_j P_{k,j}$. These objects must not be conflated.

At each control step the implementation (1) executes one action, (2) observes the resulting

state, (3) evaluates the current-stage predicates and independent external evaluator, (4) computes learning and diagnostic signals, and (5) updates the stage index. Stage indices are monotone and references are captured on entry. In the manuscript’s reported protocol, after MPC acceptance, term definitions, grounded arguments, tolerances, Boolean logic, and stage order are immutable during policy learning; only reward weights and residual-policy scales may be calibrated. The newer code’s distinct initialization profiles are documented separately in section B.6.1; the unified procedure below restates the reported protocol rather than asserting numerical equality of those profiles.

B.3 Lifecycle Procedure

The procedure below summarizes formation, policy learning, and data production using the SUN paper’s unified algorithm (Wang et al., 2026).

Algorithm B.1 Kuafu. The right column names the compiled view of Γ^* that each step

consumes; all four derive from the same grounded relations and are never re-authored.

Input: $\mathcal{I}$, scene frames $\mathcal{F}$, operator library $\mathcal{L}$, evaluator g_{ext}

Grounding and verification

- 1: LLM stages $\mathcal{I}$ into atomic requirements
- 2: **for** repair = 1, ..., 5 **do**
- 3: LLM binds $\mathcal{L} \rightarrow \mathcal{F}$, giving $\{(e_{k,j}, \epsilon_{k,j}, w_{k,j})\}$
- 4: Compile each relation once $\triangleright \phi^{\text{mpc}}, P, c^{\text{rl}}, \chi$
- 5: MPC rollouts from 10 initial states $\triangleright \phi^{\text{mpc}}$
- 6: Test completion against g_{ext} $\triangleright P, D_K$
- 7: **if** a success and no $D_K \wedge \neg g_{\text{ext}}$ **then** fix Γ^* , **break**
- 8: Repair the earliest stage with $e_{k,j} > \epsilon_{k,j}$ $\triangleright \chi$
- 9: **end for**

Policy amortization

- 10: Clone 1,000 mentor rollouts into π_θ ; freeze $\triangleright P$ for z_t
- 11: **for** iteration = 1, ..., 5 **do**
- 12: PPO on bounded residual π_ψ over π_θ $\triangleright c^{\text{rl}}, P$
- 13: **if** final $\geq \frac{1}{2}$ peak success **then break**
- 14: LLM rescales $\{w_{k,j}, \alpha_\rho, \rho_{\text{max}}\}$ only $\triangleright \chi$
- 15: **end for**

Data production

- 16: Roll out frozen π ; drop false completions $\triangleright P, D_K$
 - 17: Train the visual policy on (o_t, u_t) alone $\triangleright \text{none}$
-

B.4 Canonical Task Catalog

This is a task catalog, not a machine-readable program. Canonical programs also include scene-specific registered frame names, axes, tolerances, and runtime compiler weights. Reconstructing those fields from prose would create an unvalidated artifact, so they are not invented here.

Table B.1: Program sizes are those reported for the active SUN evaluation.

ID	Family	Physical routine	Stages	Constraints
T0	E-stop	Push/close E-stop	3	12
T1	Stacking	Stack two cubes	8	25
T2	Drawer	Pull drawer open	6	19
T3	E-stop	Twist/open E-stop	6	19
T4	Reorientation	Right a bottle	7	27
T5	Insertion	Insert marker into cup	9	31
T6	Stacking	Stack three cubes	15	53
T7	Drawer + placement	Open drawer and place object	14	44
T8	Dexterous placement	Hang cup by handle	10	31

B.5 Reported and Unreported Reproducibility Fields

This distinction is manuscript-specific. The dissertation supplies prompt text and operator equations from implementation artifacts that the RAL paper omits; the table still separates paper-reported fields from source-derived details.

To preserve traceability, reported outcomes are incorporated exactly; unreported parameters are not considered reproduced before the experiment artifact is deposited.

B.6 SUN Prompt Specifications

This appendix presents the prompt templates for SUN Program formation and policy calibration. The prompts and accompanying source code will be made publicly available on GitHub. Braced fields are replaced at run time by scene data, task state, diagnostics, or learning statistics. The templates are reproduced verbatim, without run-specific conversations or

Table B.2: A field marked reported appears in the active paper snapshot; unresolved fields require the authoritative experiment artifact rather than reconstruction from prose.

Component	Reported	Not reported in the active manuscript
Formation	Inputs, typed binding, ten verification starts, five repair attempts	Prompt text, model snapshot settings, accepted programs and scalar fields
MPC	Generic state/action notation, stage cost, direct multiple shooting, receding-horizon use	State/action parameterization and dimensions, horizon, control rate, solver tolerances, full constraints
Stage-BC	Stage conditioning, ten-action chunks, boundary padding, first-action execution	Network sizes, optimizer, epochs, normalization constants
Residual RL	PPO/GAE, frozen prior, bounded additive action, reward structure	PPO/GAE settings, taskwise bounds and reward weights, randomization ranges
Visual policy	Fixed-DP3 and deployment data budgets, policy families, camera count, zero-shot protocol	Per-architecture training budgets, camera models and calibration parameters, point-cloud preprocessing, training schedules

Table B.3: Each row names the verbatim templates reproduced below. “System” denotes persistent behavioral instructions; “user” denotes the run-time template populated with the listed evidence.

Pipeline step	Prompt pair	Run-time evidence inserted
Task decomposition	Initial system and user templates	Instruction, completed subtasks, pending subtasks
Decomposition repair	Correction system and user templates	Previous decomposition and execution error
Atomic grounding	Atomic-stage system and user templates	Current subtask, robot state/control descriptions, operator API
Typed relation writing	Constraint system and user templates	Atomic action, registered frames, completed/pending context
Relation repair	Constraint-correction system and user templates	Previous grounding, type or execution error, predicate diagnostics
Initial weight selection	Initial-sweep system and user templates	Stage semantics, neighboring stages, geometry-only operator arguments, weights
BC-prior weight adjustment	BC-diagnostic system and stage user templates	BC-prior outcomes, satisfaction times, timeout/stuck component rates, reward values
Residual-authority calibration	Residual system and user templates	Action dimensions, outcome counts, stage passage, training traces, collapse plot

model responses. They describe the current implementation; differences from the reported experimental protocol are identified in section B.6.1. Each prompt table maps a source-line number to its verbatim text and identifies the system or user role in the column header. Numbers count blank source lines, which appear as spacing between rows; wrapped display lines retain the same source-line number. Line numbers, captions, and continuation headers are presentation aids, not prompt content.

The prompt tables distinguish the two roles described in chapters 6 and 7. Formation uses a system context and a user template at each transformation or correction step. Learning uses a system contract defining the editable fields and a user template carrying the measured evidence. All formation calls request a JSON object. The current implementation uses

GPT-5.5 for these calls, with sampling controls left at the provider defaults.

B.6.1 Receiver-side tolerance assignment

The model specifies operators, grounded targets, frames, distances, offsets, rotation amounts, and contact conditions, but no `tolerance_m` or `tolerance_rad` fields. After parsing a new or corrected constraint response, the receiving application assigns $\epsilon_{k,j}$ from the operator, stage semantics, effort conditions, and scene dimensions before compilation. This deterministic policy ignores any tolerance value supplied by the model and records its own profile and per-component decision. Thus the compiled program still has numerical satisfaction boundaries even though the prompt-visible geometry schema does not expose them.

The implementation distinguishes two deterministic profiles. Fresh MPC formulation caps positional tolerances at 0.01 m, or 0.02 m for a nonzero-offset reach target, and assigns the nominal angular field 0.20 rad. Learning preparation instead derives positional fields in [0.01, 0.05] m and angular fields in [0.30, 0.50] rad according to geometry and interaction role. Saved MPC programs retain their recorded values on replay. Learning values are set before weight selection and held fixed during subsequent scalar tuning. These profiles have different MPC and learning boundaries and are distinct from the fixed-boundary experimental protocol restated in algorithm B.1. The reported results apply to that experimental protocol, not to these differing profiles.

Before a request is sent, an outgoing-context filter removes receiver-owned setting fields and recognized tolerance annotations from prompt copies, including embedded JSON, correction diagnostics, and text history. Geometric targets and their numeric values remain intact. Unrecognized prose is preserved rather than blocking the request; numerical tolerance assignment is enforced by the receiving application. The filter does not alter stored programs or image attachments. Initial and BC-informed weight calls request only **Diagnosis**, **Reasoning**,

and **Weights**; any unsolicited **Parameters** field is ignored.

B.6.2 Program-formation prompts

The five prompt pairs below implement the transformations and diagnostic return paths in figure 6.2 and algorithm B.1. Their wording and placeholder names match those used by the implementation.

B.6.2.1 Initial task decomposition

Table B.4: System context for initial task decomposition

Line	System prompt text (verbatim)
1	You write pre-task manipulation plans as phase-labeled planner subtasks. Each subtask starts with "[Approach]", "[Manipulate]", or "[Release]" and describes one controller-stage-sized conceptual action.
3	Inputs include the top-level task, observations, visible object names, robot frames/joints, previously seen objects, and an Important rule limiting references to listed objects/frames/locations.
5	Priority rules:
6	1. Output exactly one JSON object with keys "Reasoning" and "Subtasks". No extra text, code fences, or backticks.
7	2. "Subtasks" is an ordered string array. Each subtask must be executable as one later controller stage: one spatial change OR one gripper-state change, plus any required fixed-position/fixed-orientation condition. Split sequential combinations such as approach+grasp, grasp+pull, lift+carry, carry+lower, release+retract, align+insert, and pull+release.
8	3. Use only listed objects, frames, and locations. Use exact names; listed '_frame' names are allowed task targets. Prefer specific observed targets over vague references like "inside", "there", or "drawer".
9	4. Do not mention coordinates, axes, coordinate-frame mechanics, numeric distances/speeds/angles, or low-level geometric terms such as codirectional, antiparallel, bipolar, orthogonal, collinear, or coaxial.
10	5. For grasp-then-move/pull interactions, follow the GRASP pattern. For contact-without-grasp interactions, follow the CONTACT pattern. If rules conflict, priority rules win, then patterns, then other guidance.
12	Scope:
13	- Plan conceptually: describe WHAT the robot does, not controller details or trajectories.
14	- If manipulating through the scene, include a clearance-creating [Manipulate] subtask before carrying/repositioning when surfaces, obstacles, or receptacles matter.
15	- One subtask must not hide a sequence.
17	Approach label selection:
18	Every [Approach] subtask contains exactly one of: "top-down approach", "leveled approach", "direct approach toward the object", or "axis approach". Choose by the immediate interaction target/feature, not only the object category. Use the first matching rule:
19	1. "axis approach": direction-constrained interactions such as buttons, switches, knobs, levers, doors, drawers, handles, insertions, threading, hooking, or openings. Do not use for a generic grasp unless the local feature is direction-constrained.
20	2. "top-down approach": grasping a small object from above while it rests on a surface, or a subtask that explicitly approaches from above. Do not use for side grasps of upright objects/features.
21	3. "leveled approach": side interaction with a tall, upright, or standing object/body.

Continued on next page

Table B.4: System context for initial task decomposition (continued)

Line	System prompt text (verbatim)
22	4. "direct approach toward the object": fallback when no stronger cue exists; briefly state finger orientation relative to the target without coordinates/frames.
24	GRASP pattern for picking, moving, carrying, placing, or pulling by grasping <obj>:
25	1. "[Approach] Use an appropriate <approach label> to move the open gripper <relation> <obj> without touching it." Use "above" for top-down and "near" otherwise.
26	2. "[Approach] Move the open gripper into contact with <obj> while keeping the gripper orientation fixed." This step must not include standoff.
27	3. "[Manipulate] Close the gripper to grasp <obj> while keeping both the gripper position and orientation fixed."
28	4. Move the grasped object with the gripper closed. Split multi-stage motion into clearance, carry/align, and optional lower/place subtasks. Do not combine lift, carry, align, and lower. If engagement is needed, split into clearance when transporting, alignment-only, then engagement-only; for insertion/threading/sliding through an opening, the alignment-only subtask must explicitly say "<obj> aligned with the opening of <target> for insertion" and must not use vague words like suitable/appropriate.
29	5. "[Release] Open the gripper to release <obj> at the desired state or location while keeping both the gripper position and orientation fixed."
30	6. "[Release] Retract the gripper away from <obj> to a non-contact configuration with the gripper open, preserving continuity with the immediately preceding interaction."
31	Keep milestone order. Omit release/retract only if the task does not require releasing the object.
33	CONTACT pattern for pressing, pushing, tapping, depressing, or brief contact without grasping <obj>:
34	1. "[Approach] Use an appropriate <approach label> to move the end-effector near <obj> without touching it, with the gripper closed." The gripper stays closed unless the task explicitly requires otherwise.
35	2. "[Approach] Move the end-effector into contact with <obj> while keeping the gripper orientation fixed."
36	3. "[Manipulate] Perform the required contact motion on <obj> to achieve the desired state, stating whether orientation, position, or both stay fixed." Do not force push/press verbs unless required.
37	4. "[Release] Retract the end-effector away from <obj> to a non-contact configuration, preserving continuity with the immediately preceding interaction."
38	Keep milestone order and required fixed-orientation phrase.
40	Reasoning: 2-6 sentences stating the strategy, interactions, phase decomposition, approach-label justifications, required fixed orientation/pose points, and any clearance expansion.
42	Output shape:
43	{
44	"Reasoning": "...",
45	"Subtasks": ["[Approach] ...", "[Manipulate] ..."]
46	}

Table B.5: User template for initial task decomposition

Line	User prompt text (verbatim)
1	Top-level task: {task_description}
3	Inputs:
4	- Optional scene image including the robot
5	- Robot frames / joints: {frames}
6	- Observed objects: {history}
7	- Important: reference only objects, frames, and locations listed in the inputs.
9	Decompose the top-level task into phase-labeled subtasks.
11	Requirements:
12	- Each subtask starts with exactly one phase tag: "[Approach]", "[Manipulate]", or "[Release]".
13	- Each subtask describes one controller-stage-feasible conceptual action: one spatial change OR one gripper-state change, plus any required fixed-position/fixed-orientation condition.
14	- Split sequential combinations such as approach+grasp, grasp+pull, lift+carry, carry+lower, release+retract, align+insert, and pull+release.
15	- Follow the system instructions for approach labels and GRASP / CONTACT patterns.
16	- Prefer exact observed target names, including listed '_frame' names, over vague references like "inside", "there", or "drawer".
18	Return exactly one JSON object:
19	- "Reasoning": 2-6 sentences describing the strategy and decomposition.
20	- "Subtasks": ordered array of phase-labeled subtasks.
22	Reply with JSON only: double-quoted keys/strings, no extra text, no code fences, no backticks.

B.6.2.2 Task-decomposition correction

Table B.6: System context for decomposition correction

Line	System prompt text (verbatim)
1	You correct pre-task manipulation plans as phase-labeled planner subtasks. Each subtask starts with "[Approach]", "[Manipulate]", or "[Release]" and describes one controller-stage-sized conceptual action.
3	Inputs include the top-level task, observations, visible object names, robot frames/joints, previously seen objects, execution log/completed subtasks, an untrusted previous plan, pending/completed subtasks from that plan, an error/issue, and an Important rule limiting references to listed objects/frames/locations.
5	Priority rules:
6	1. Output exactly one JSON object with keys "Diagnosis", "Reasoning", and "Subtasks". No extra text, code fences, or backticks.
7	2. Treat the execution log as authoritative. Do not repeat completed actions; continue from the current state.

Continued on next page

Table B.6: System context for decomposition correction (continued)

Line	System prompt text (verbatim)
8	3. "Subtasks" is an ordered string array. Each subtask must be executable as one later controller stage: one spatial change OR one gripper-state change, plus any required fixed-position/fixed-orientation condition. Split sequential combinations such as approach+grasp, grasp+pull, lift+carry, carry+lower, release+retract, align+insert, and pull+release.
9	4. Use only listed objects, frames, and locations. Use exact names; listed '_frame' names are allowed task targets. Prefer specific observed targets over vague references like "inside", "there", or "drawer".
10	5. Do not mention coordinates, axes, coordinate-frame mechanics, numeric distances/speeds/angles/counts, or low-level geometric terms such as codirectional, antiparallel, bipolar, orthogonal, collinear, or coaxial.
11	6. For grasp-then-move/pull interactions, follow the GRASP pattern. For contact-without-grasp interactions, follow the CONTACT pattern. If rules conflict, priority rules win, then patterns, then other guidance.
13	Correction scope:
14	- Keep compliant subtasks verbatim when possible.
15	- Make the smallest set of changes needed to fix invalid references, missing pattern steps/phrases, missing clearance before manipulation-through-scene, wrong labels/order, execution-log conflicts, and the stated error.
16	- Plan conceptually: describe WHAT the robot does, not controller details or trajectories.
18	Approach label selection:
19	Every [Approach] subtask contains exactly one of: "top-down approach", "leveled approach", "direct approach toward the object", or "axis approach". Choose by the immediate interaction target/feature. Use the first matching rule:
20	1. "axis approach": direction-constrained interactions such as buttons, switches, knobs, levers, doors, drawers, handles, insertions, threading, hooking, or openings.
21	2. "top-down approach": grasping a small object resting on a surface or a subtask that explicitly approaches from above; not for side grasps of upright objects/features.
22	3. "leveled approach": side interaction with a tall, upright, or standing object/body.
23	4. "direct approach toward the object": fallback; briefly state finger orientation relative to the target without coordinates/frames.
25	GRASP pattern for picking, moving, carrying, placing, or pulling by grasping <obj>:
26	1. "[Approach] Use an appropriate <approach label> to move the open gripper <relation> <obj> without touching it." Use "above" for top-down and "near" otherwise.
27	2. "[Approach] Move the open gripper into contact with <obj> while keeping the gripper orientation fixed." This step must not include standoff.
28	3. "[Manipulate] Close the gripper to grasp <obj> while keeping both the gripper position and orientation fixed."

Continued on next page

Table B.6: System context for decomposition correction (continued)

Line	System prompt text (verbatim)
29	4. Move the grasped object with the gripper closed. Split multi-stage motion into clearance, carry/align, and optional lower/place subtasks. Do not combine lift, carry, align, and lower. If engagement is needed, split into alignment-only then engagement-only; do not combine alignment and insert/hook/thread/slide engagement.
30	5. "[Release] Open the gripper to release <obj> at the desired state or location while keeping both the gripper position and orientation fixed."
31	6. "[Release] Retract the gripper away from <obj> to a non-contact configuration with the gripper open, preserving continuity with the immediately preceding interaction."
32	Keep milestone order. Omit release/retract only if not needed by the remaining task.
34	CONTACT pattern for brief contact without grasping <obj>:
35	1. "[Approach] Use an appropriate <approach label> to move the end-effector near <obj> without touching it in a chosen gripper state." Explicitly choose open/closed as contact posture; this is not a grasp unless the robot must hold the object.
36	2. "[Approach] Move the end-effector into contact with <obj> while keeping the gripper orientation fixed."
37	3. "[Manipulate] Perform the contact action on <obj> to achieve the desired state, stating whether orientation, position, or both stay fixed."
38	4. "[Release] Retract the end-effector away from <obj> to a non-contact configuration, preserving continuity with the immediately preceding interaction."
39	Keep milestone order and required fixed-orientation phrase.
41	Diagnosis: exactly "Evidence: ...; Cause: ...; Fix: ..." and cite concrete violations/mismatches.
42	Reasoning: 2-6 sentences stating the updated strategy, interactions, phase decomposition, approach-label justifications, fixed orientation/pose points, and any clearance split.
44	Output shape:
45	{
46	"Diagnosis": "Evidence: ...; Cause: ...; Fix: ...",
47	"Reasoning": "...",
48	"Subtasks": ["[Approach] ...", "[Manipulate] ..."]
49	}

Table B.7: User template for decomposition correction

Line	User prompt text (verbatim)
1	Top-level task: {task_description}
3	Observations and execution state:
4	* Optional scene image(s): {images}
5	* Robot frames / joints: {frames}
6	* Observed objects: {history}
7	* Execution log / completed subtasks: {completed}
9	Previous pre-task plan, untrusted:

Continued on next page

Table B.7: User template for decomposition correction (continued)

Line	User prompt text (verbatim)
10	* Reasoning: {planner_reasoning}
11	* Subtasks: {planner_subtasks}
12	* Pending subtasks: {pending}
13	* Completed subtasks from original plan: {completed_plan}
15	Error / Issue:
16	* Description: {error}
18	Using the system instructions, return a corrected phase-labeled plan.
20	Requirements:
21	- Diagnose the issue.
22	- Do not repeat completed actions.
23	- Keep valid previous subtasks when possible.
24	- Each corrected subtask must be one controller-stage-feasible conceptual action: one spatial change OR one gripper-state change, plus any required fixed-position/fixed-orientation condition.
25	- Split sequential combinations such as approach+grasp, grasp+pull, lift+carry, carry+lower, release+retract, align+insert, and pull+release.
27	Return exactly one JSON object:
28	- "Diagnosis": "Evidence: ...; Cause: ...; Fix: ..."
29	- "Reasoning": explanation of the correction.
30	- "Subtasks": ordered array of corrected phase-labeled subtasks.
32	Reply with JSON only: double-quoted keys/strings, no extra text, no code fences, no backticks.

B.6.2.3 Atomic-stage grounding

Table B.8: System context for atomic-stage grounding

Line	System prompt text (verbatim)
1	You ground ONE current phase-labeled planner subtask into exactly ONE natural-language grounded stage requirement for a downstream constraint writer. You never write code or TermAPI JSON.
3	Inputs include the top-level task, current subtask, completed/pending subtasks for continuity, observations, robot frames, observed objects/poses/sizes, robot state, control interface, and available API/residual context.
5	Output JSON:
6	- Exactly keys "Reasoning" and "AtomicActions"; no extra text, comments, code fences, or trailing commas.
7	- "Reasoning": brief interpretation and chosen frames/axes/clearances.
8	- "AtomicActions": an array with exactly one string. First line starts with the current phase tag; second line is "Constraints:"; following lines are numbered constraint sentences.
10	Stage rule:
11	- The single string is one controller stage. It may list simultaneous constraints, but not sequential motions or sequential gripper changes.

Continued on next page

Table B.8: System context for atomic-stage grounding (continued)

Line	System prompt text (verbatim)
12	- If the current subtask is compound, encode only the first controller-feasible component and explain the split issue in Reasoning.
14	Global rules:
15	1. Frames/entities
16	- Use only observed objects and frames; do not invent entities or surfaces.
17	- Every pose, orientation, and translation names its frame: agent/robot base, world, object, or tool frame as appropriate.
18	- Do not assert frame/world alignment unless explicitly stated in observations.
20	2. Axes/orientation
21	- Always name signed axes, e.g. tool +z, tool +y, object +x, world +z.
22	- In [Approach], approach axis is tool +z. Do not reinterpret [Manipulate] translations as tool +z unless the current subtask says so.
23	- Finger axis is tool +y; use it for all finger-axis relations.
24	- Axis relation words are only: antiparallel, codirectional, bipolar, orthogonal, or "tool +z points toward the center of <object>".
25	- Use axis relation words for axes, not motion direction.
27	3. Distances/translations
28	- Directional translation phrasing, "translate along <signed axis> in <frame> by <positive distance>", is allowed only in [Manipulate] and [Release], not [Approach].
29	- Distances are positive; flip the axis sign instead of using negative distances.
30	- Non-contact clearance margin must be ≥ 0.10 m unless tighter is explicit; "small clearance" ≥ 0.10 m, "large clearance" ≥ 0.15 m.
31	- If size is available, non-contact separation = $0.5 \times \text{moving clearance-extent} + 0.5 \times \text{target clearance-extent} + \text{margin}$. Clearance-extent is the longest extent unless an axis-specific extent is given.
32	- For contact/into-contact, aim for the target frame origin or object center and state contact feedback. Do not define contact using object surfaces/extents.
34	4. Approach positional goals
35	- Non-contact axis approach: reach "<object/frame> origin translated along the chosen/inferred SIGNED object axis by D in object frame"; $D \geq 0.10$ m unless tighter is explicit. Do not express this only as scalar standoff and do not flip the sign.
36	- Other non-contact approaches: use a positive standoff from an object reference point.
37	- Contact approaches: reach the target frame origin or object center.
38	- Never express [Approach] standoff as along a tool axis.
40	5. Shape/finger rules for grasps
41	- Elongated object: tool +y orthogonal to long axis.
42	- Box-like object: choose a clamping edge/axis and make tool +y orthogonal to it.
43	- Spherical/rotationally symmetric: tool +y is free within symmetry; mention it without over-constraining.
44	- Top-down grips: tool +y must not align with tool +z.
46	Approach label mapping:

Continued on next page

Table B.8: System context for atomic-stage grounding (continued)

Line	System prompt text (verbatim)
47	- If no approach label appears, assume direct approach: tool +z points toward the object center; add no top-down/leveled/axis orientation constraints.
48	- If the subtask says keep orientation fixed, do not add approach or shape orientation constraints; state only fixed orientation.
49	- top-down approach: tool +z antiparallel to world +z.
50	- leveled approach: tool +z orthogonal to world +z AND points toward the object center.
51	- direct approach toward the object: tool +z points toward the object center.
52	- axis approach: tool +z antiparallel to the chosen signed object axis. If the subtask says "appropriate axis approach" and observations give an interaction direction, infer and state that signed axis; if sign is ambiguous, say bipolar.
54	Phase rules:
55	[Approach]
56	- Keep the phase tag.
57	- Unless orientation is fixed, state tool +z orientation from the approach mapping; if setting up a grasp, also state tool +y shape/finger relation.
58	- Use the positional-goal rules above.
59	- State stop conditions using the geometric target and monitored feedback. Use contact/effort feedback only when touching is intended. Do not say hold position/pose unless the subtask explicitly says fixed.
61	[Manipulate]
62	- Decide whether the stage changes gripper state with pose fixed, or moves a tool/object while in contact/grasp.
63	- For gripper-state changes, state which pose is fixed and define firm grasp/release using contact/effort. Later conversion should use object_grasped when success requires a real held object; close_gripper is only mechanical closure.
64	- For moving while in contact/grasp, use named axes/positive distances or rotations; lift/raise = world +z, lower = world -z, independent of tool orientation. Mention clearances and stop conditions such as geometric completion or effort at a stop.
65	- For "Align <frame_A> with <frame_B>" without engagement/contact: do not model only center distance; include an axis relation if axes are given, use bipolar if sign is ambiguous, laterally align while maintaining positive axial standoff, and if size is known use the separation formula.
66	- Never use the same scalar for both "within d" and "minimum standoff >= d" on the same distance; they must be different components or values.
67	- If reorienting a grasped object, use object-axis relations and preserve grasp/no-slip; do not add absolute tool +z constraints unless explicitly requested.
68	- Only mention tool +z in [Manipulate] if the subtask says fixed tool orientation or names a tool-axis relation.
69	- Always state monitored feedback.
71	[Release]
72	- For opening/loosening, hold a pose fixed and indicate contact/effort drops toward free-space.

Continued on next page

Table B.8: System context for atomic-stage grounding (continued)

Line	System prompt text (verbatim)
73	- For retract, use the explicit direction if given; otherwise mirror the most recent approach for the same object/frame: top-down -> world +z; leveled/direct -> tool -z; axis approach -> same signed object axis offset point, not flipped. Use D >= 0.10 m unless tighter is explicit.
74	- State whether tool +z and tool +y preserve previous orientation during retract when relevant.
76	Self-check before output:
77	1. Exactly one AtomicActions string and correct phase tag.
78	2. All poses/orientations/translations name frames and signed axes.
79	3. [Approach] position goal uses the correct form: axis-offset, positive standoff, or contact target.
80	4. Size-based standoffs use the extent+margin rule.
81	5. Output is valid JSON with only "Reasoning" and "AtomicActions".

Table B.9: User template for atomic-stage grounding

Line	User prompt text (verbatim)
1	Top-level task: {task_description}
3	Current high-level subtask: {current_subtask}
4	Completed high-level subtasks: {completed}
5	Pending high-level subtasks: {pending}
7	Inputs:
8	* Optional scene image including the robot
9	* Robot frames / joints: {frames}
10	* Observed objects: {history}
12	Technical context:
13	* robot type: {robot}
14	* current state: {state}
15	* control interface: {control}
17	Ground ONLY the current subtask into exactly ONE detailed natural-language grounded stage requirement.
19	Rules:
20	- Preserve the current phase tag and intent.
21	- Keep the natural-language grounding layer; do not emit TermAPI JSON here.
22	- The downstream constraint stage treats the entire AtomicActions array as one controller stage, so do not use the array to describe a sequence.
23	- The single grounded stage may describe simultaneous constraints, but not multiple sequential motions or sequential gripper changes.
24	- If the current subtask is compound, ground only the first controller-feasible component and explain the split issue in "Reasoning".
25	- Use the system instructions for frames, tool +z / tool +y, approach labels, translations, clearances, object extents, finger-vs-shape rules, stop conditions, and effort/contact feedback.
26	- For [Approach] position goals: non-contact axis approach uses an axis-offset point; other non-contact approaches use positive standoff; contact approaches reach the target frame origin or object center.

Continued on next page

Table B.9: User template for atomic-stage grounding (continued)

Line	User prompt text (verbatim)
27	- Specify target geometry and contact/effort conditions. State target distances, standoffs, clearances, and rotation amounts when needed.
29	Return exactly one JSON object:
30	- "Reasoning": how you interpreted the current subtask and applied frame/geometric rules.
31	- "AtomicActions": an array containing exactly one string formatted as:
32	first line = phase + summary;
33	second line = "Constraints:";
34	following lines = numbered constraint sentences specifying frames, axes, motions, clearances, stop conditions, gripper state, and feedback signals.

B.6.2.4 Typed relation writing

Table B.10: System context for typed relation writing

Line	System prompt text (verbatim)
1	You translate the CURRENT grounded natural-language action into one flat JSON list of Constraint terms. You do not write code.
2	The list represents one controller stage. Treat constraints as conjunctive stage requirements: use 'logic="and"' for normal terms and do not rely on top-level OR unless the runtime/error message explicitly supports it. Encode explicit geometry/effort stop alternatives within the same relevant term's 'condition' labels when representable.

Table B.11: User template for typed relation writing

Line	User prompt text (verbatim)
1	### SCOPE & INPUTS
3	Task
4	* Current grounded stage requirements: {task_description}
5	* Completed subtasks, context only: {completed}
6	* Pending subtasks, context only: {pending}
8	Observations
9	* Robot Frames: {frames}
10	* Observed Objects, visible + history: {history}
11	* Only refer to names in Robot Frames and Observed Objects; there are no others. "world" is a reserved keyword.
13	Translate the current grounded natural-language action into a flat JSON list of Constraint terms. Do not write code.
14	Return only the fields in the schema below. Preserve target distances, offsets,
15	rotation angles, frames, and contact conditions.
17	### STAGE SEMANTICS

Continued on next page

Table B.11: User template for typed relation writing (continued)

Line	User prompt text (verbatim)
19	The Constraints list represents ONE controller stage.
20	* Cover all required components of the stage; do not drop a component because another is easier to encode.
21	* Stage requirements are conjunctive unless a runtime/error message explicitly says top-level OR is supported.
22	* For normal terms, set 'logic' to '"and"'.
23	* Do not rely on top-level 'logic="or"' for alternatives. If the text gives a supported stop alternative such as geometric progress OR manipulator_high_effort, encode it in the same relevant term by including both condition labels, e.g. '["geometry", "manipulator_high_effort"]'. If the alternative cannot be represented, keep the required representable term and note the omitted alternative in Reasoning.
25	Condition labels describe satisfaction/feedback categories:
26	* 'geometry': pose, motion, orientation, or freeze constraints.
27	* 'gripper_opened' / 'gripper_closed': intended gripper state metadata; not arm contact effort. 'gripper_closed' alone does not prove a grasp.
28	* 'manipulator_high_effort' / 'manipulator_low_effort': arm/TCP interaction effort for pushing, pressing, pulling, or sliding; not finger squeeze.
30	### TERM API
32	Use ONLY these ops and EXACTLY these args keys:
34	```json
35	{
36	"reach_point": {
37	"args": {
38	"controlled_entity": "robot_frame gripped_object",
39	"controlled_target": "NAME",
40	"reference_basis": "world object robot_frame",
41	"reference_origin": "NAME world",
42	"offset_from_origin_xyz": [float, float, float]
43	}
44	},
45	"directional_move": {
46	"args": {
47	"controlled_entity": "robot_frame gripped_object",
48	"controlled_target": "NAME",
49	"reference_basis": "world object robot_frame",
50	"reference_origin": "NAME world",
51	"axis": "+x -x +y -y +z -z",
52	"travel_distance_m": float
53	}
54	},
55	"axis_align": {
56	"args": {
57	"controlled_entity": "robot_frame gripped_object",
58	"controlled_target": "NAME",
59	"reference_basis": "world object robot_frame",
60	"reference_origin": "NAME world",

Continued on next page

Table B.11: User template for typed relation writing (continued)

Line	User prompt text (verbatim)
61	"controlled_axis": "+x -x +y -y +z -z",
62	"reference_axis": "+x -x +y -y +z -z",
63	"alignment": "codirectional antiparallel bipolar orthogonal"
64	}
65	},
66	"point_toward": {
67	"args": {
68	"controlled_entity": "robot_frame gripped_object",
69	"controlled_target": "NAME",
70	"controlled_axis": "+x -x +y -y +z -z",
71	"reference_basis": "world object robot_frame",
72	"reference_origin": "NAME world",
73	"offset_from_origin_xyz": [float, float, float]
74	}
75	},
76	"directional_rotate": {
77	"args": {
78	"controlled_entity": "robot_frame gripped_object",
79	"controlled_target": "NAME",
80	"reference_basis": "world object robot_frame",
81	"reference_origin": "NAME world",
82	"rotation_axis": "+x -x +y -y +z -z",
83	"rotation_angle_rad": float
84	}
85	},
86	"freeze_position": {
87	"args": {
88	"controlled_entity": "robot_frame gripped_object",
89	"controlled_target": "NAME"
90	}
91	},
92	"freeze_orientation": {
93	"args": {
94	"controlled_entity": "robot_frame gripped_object",
95	"controlled_target": "NAME"
96	}
97	},
98	"open_gripper": {"args": {}},
99	"close_gripper": {"args": {}},
100	"object_grasped": {"args": {}}
101	}
102	'''
104	NAME must be a valid robot frame or object/object-attached frame from Observations. Do not add or omit args keys.
106	### FIELD MEANINGS
108	* 'controlled_entity="robot_frame"' requires 'controlled_target' to be a robot frame.

Continued on next page

Table B.11: User template for typed relation writing (continued)

Line	User prompt text (verbatim)
109	* 'controlled_entity="gripped_object"' requires 'controlled_target' to be the object/object-attached frame currently in the gripper.
110	* 'reference_basis' is the coordinate system used for vectors/axes; 'reference_origin' is the pose defining that basis origin/axes.
111	* 'offset_from_origin_xyz' is applied at 'reference_origin' in 'reference_basis'.
112	* '_m' values are meters; '_rad' values are radians.
114	### OPERATOR RULES
116	* 'reach_point': reach the specified target point; path is unconstrained. Use for concrete targets, pre-grasps, standoffs, and approach-side/axis offsets.
117	* 'directional_move': move strictly along one signed axis by positive 'travel_distance_m'. Use only when the text explicitly specifies axis/direction plus distance; otherwise use 'reach_point'.
118	* 'axis_align': align axes with codirectional, antiparallel, bipolar, or orthogonal relation. Use 'bipolar' when either direction is acceptable.
119	* 'point_toward': make a controlled signed axis aim at a target point.
120	* 'directional_rotate': rotate about a signed axis by an explicit angle.
121	* Do not combine 'directional_rotate' with 'axis_align' in the same stage when the 'axis_align' constrains or references the frame being rotated. Put approach/pre-grasp axis alignment in an earlier stage; the rotation stage should keep the finite 'directional_rotate' goal.
122	* 'freeze_position': keep current position fixed; do not use as a target-reaching goal.
123	* 'freeze_orientation': keep current orientation fixed; do not use as a goal-alignment operator. If the stage establishes a new axis relation, emit 'axis_align' or 'directional_rotate'; 'freeze_orientation' alone is invalid for that.
124	* 'open_gripper': gripper open/released at success.
125	* 'close_gripper': mechanical finger closure only when a real held object is not required.
126	* 'object_grasped': object must be physically held; use for grasp, hold, keep holding, and any later moving/pulling/carrying/aligning stage that assumes the object remains in the gripper. Do not emit both 'close_gripper' and 'object_grasped' in one list.
128	General constraints:
129	1. At most one position op in the whole list: 'reach_point' OR 'directional_move' OR 'freeze_position'.
130	2. Every constraint must be justified by the current stage requirements.
131	3. Describe the intended contact/fit precisely through geometry and conditions.
132	4. If a required clause cannot be represented, omit that clause from Constraints and state it in Reasoning.
134	### REQUIRED OUTPUT
136	Return exactly one JSON object:
138	```json
139	{
140	"Reasoning": "string",
141	"Constraints": [
142	{

Continued on next page

Table B.11: User template for typed relation writing (continued)

Line	User prompt text (verbatim)
143	"logic": "and or",
144	"condition": ["geometry", "gripper_opened", "gripper_closed", "manipulator_
145	high_effort", "manipulator_low_effort"],
146	"op": "reach_point directional_move axis_align point_toward
147	directional_rotate freeze_position freeze_orientation open_gripper close_
148	gripper object_grasped",
149	"args": {}
150	}
151	]
152	}
153	'''
154	Reasoning briefly explains success, each constraint, any condition-label
155	alternative, and any unrepresented clause.

B.6.2.5 Typed-relation correction

Table B.12: System context for typed-relation correction

Line	System prompt text (verbatim)
1	You correct the PREVIOUS grounded output for the CURRENT stage after a failure.
2	You do not write code.
3	Use the error message as the primary diagnostic signal. Make the smallest change
4	needed, keeping correct constraints verbatim.
5	The corrected Constraints list is one controller stage: prefer 'logic="and"'; do
6	not rely on top-level OR unless the runtime/error message explicitly supports it.
7	Encode representable stop alternatives inside the same term's 'condition' labels.

Table B.13: User template for typed-relation correction

Line	User prompt text (verbatim)
1	### SCOPE & INPUTS
2	Task
3	* Current grounded stage requirements: {task_description}
4	* Completed subtasks, context only: {completed}
5	* Pending subtasks, context only: {pending}
6	Observations
7	* Robot Frames: {frames}
8	* Observed Objects, visible + history: {history}
9	* Only refer to names in Robot Frames and Observed Objects; there are no others.
10	"world" is a reserved keyword.
11	Previous grounded output to correct:
12	* {previous_grounding}

Continued on next page

Table B.13: User template for typed-relation correction (continued)

Line	User prompt text (verbatim)
16	Current error message, primary signal:
17	* {error_message}
19	### YOUR JOB
21	Return only the fields in the schema below. Correct geometry/contact conditions
22	while preserving the intended target and required physical result.
24	Diagnose the failure, then return corrected Constraint JSON that:
25	1. Uses only TermAPI ops and exactly their allowed args keys.
26	2. References only valid observed names.
27	3. Preserves all intended success criteria for the current stage.
28	4. Changes as little as possible: edit only fields implicated by the error, schema/API violations, missing coverage, unsupported top-level OR, or the one-position-op rule. Keep correct fields verbatim.
30	### STAGE SEMANTICS AND REPAIR RULES
32	The Constraints list represents ONE controller stage.
33	* Cover all required components; do not drop a component because another is easier to encode.
34	* Stage requirements are conjunctive unless a runtime/error message explicitly says top-level OR is supported.
35	* For normal terms, set 'logic' to "and".
36	* Do not rely on top-level 'logic="or"' for alternatives. If the text gives a supported stop alternative such as geometric progress OR manipulator_high_effort, encode it in the same relevant term by including both condition labels, e.g. ["geometry", "manipulator_high_effort"]. If the alternative cannot be represented, keep the required representable term and note the omitted alternative in Diagnosis or Reasoning.
37	* Across the whole list, keep at most one position op: 'reach_point' OR 'directional_move' OR 'freeze_position'. If violated, keep the one that best matches the explicit positional requirement and mention the dropped op in Diagnosis.
38	* Remove only exact duplicates: same 'op' and identical 'args'; merge 'condition' labels by unique union. Do not collapse merely similar constraints.
39	* 'gripper_opened' / 'gripper_closed' are gripper-state metadata, not arm contact effort. 'manipulator_high_effort' / 'manipulator_low_effort' are arm/TCP contact effort, not finger squeeze. Use 'object_grasped' when a real held object is required; do not represent contact/effort with gripper ops.
41	### TERM API
43	Use ONLY these ops and EXACTLY these args keys:
45	```json
46	{
47	"reach_point": {
48	"args": {
49	"controlled_entity": "robot_frame gripped_object",
50	"controlled_target": "NAME",
51	"reference_basis": "world object robot_frame",
52	"reference_origin": "NAME world",
53	"offset_from_origin_xyz": [float, float, float]
54	}

Continued on next page

Table B.13: User template for typed-relation correction (continued)

Line	User prompt text (verbatim)
55	},
56	"directional_move": {
57	"args": {
58	"controlled_entity": "robot_frame gripped_object",
59	"controlled_target": "NAME",
60	"reference_basis": "world object robot_frame",
61	"reference_origin": "NAME world",
62	"axis": "+x -x +y -y +z -z",
63	"travel_distance_m": float
64	}
65	},
66	"axis_align": {
67	"args": {
68	"controlled_entity": "robot_frame gripped_object",
69	"controlled_target": "NAME",
70	"reference_basis": "world object robot_frame",
71	"reference_origin": "NAME world",
72	"controlled_axis": "+x -x +y -y +z -z",
73	"reference_axis": "+x -x +y -y +z -z",
74	"alignment": "codirectional antiparallel bipolar orthogonal"
75	}
76	},
77	"point_toward": {
78	"args": {
79	"controlled_entity": "robot_frame gripped_object",
80	"controlled_target": "NAME",
81	"controlled_axis": "+x -x +y -y +z -z",
82	"reference_basis": "world object robot_frame",
83	"reference_origin": "NAME world",
84	"offset_from_origin_xyz": [float, float, float]
85	}
86	},
87	"directional_rotate": {
88	"args": {
89	"controlled_entity": "robot_frame gripped_object",
90	"controlled_target": "NAME",
91	"reference_basis": "world object robot_frame",
92	"reference_origin": "NAME world",
93	"rotation_axis": "+x -x +y -y +z -z",
94	"rotation_angle_rad": float
95	}
96	},
97	"freeze_position": {
98	"args": {
99	"controlled_entity": "robot_frame gripped_object",
100	"controlled_target": "NAME"
101	}

Continued on next page

Table B.13: User template for typed-relation correction (continued)

Line	User prompt text (verbatim)
102	},
103	"freeze_orientation": {
104	"args": {
105	"controlled_entity": "robot_frame gripped_object",
106	"controlled_target": "NAME"
107	}
108	},
109	"open_gripper": {"args": {}},
110	"close_gripper": {"args": {}},
111	"object_grasped": {"args": {}}
112	}
113	'''
115	NAME must be a valid robot frame or object/object-attached frame from Observations. Do not add or omit args keys.
117	### FIELD AND OPERATOR RULES
119	* 'controlled_entity="robot_frame"' requires a robot-frame target; 'controlled_ entity="gripped_object"' requires an object/object-attached frame currently in the gripper.
120	* 'reference_basis' is the coordinate system for vectors/axes; 'reference_origin' defines the pose. '_m' = meters; '_rad' = radians.
121	* 'reach_point': concrete target point, pre-grasp, standoff, or approach offset; path unconstrained.
122	* 'directional_move': strictly along one signed axis by positive distance; only when axis/direction plus distance is explicit. Otherwise use 'reach_point'.
123	* 'axis_align': axis relation; use 'bipolar' when either direction is acceptable.
124	* 'point_toward': make a controlled axis aim at a target point.
125	* 'directional_rotate': explicit rotation about an axis by an angle.
126	* Do not combine 'directional_rotate' with 'axis_align' in the same stage when the 'axis_align' constrains or references the frame being rotated. Put approach/ pre-grasp axis alignment in an earlier stage; the rotation stage should keep the finite 'directional_rotate' goal.
127	* 'freeze_position': keep current position fixed; not a target-reaching goal.
128	* 'freeze_orientation': keep current orientation fixed; not goal alignment. If the stage establishes a new axis relation, corrected output must contain 'axis_ align' or 'directional_rotate'.
129	* 'open_gripper': open/released. 'close_gripper': mechanical closure only. 'object_grasped': real held object for grasp/hold/keep holding and later carrying/pulling/manipulating. Do not emit both 'close_gripper' and 'object_ grasped'.
131	### REQUIRED OUTPUT
133	Return exactly one JSON object:
135	'''json
136	{
137	"Diagnosis": "string",
138	"Reasoning": "string",
139	"Constraints": [
140	{

Continued on next page

Table B.13: User template for typed-relation correction (continued)

Line	User prompt text (verbatim)
141	"logic": "and or",
142	"condition": ["geometry", "gripper_opened", "gripper_closed", "manipulator_
143	high_effort", "manipulator_low_effort"],
144	"op": "reach_point directional_move axis_align point_toward
145	directional_rotate freeze_position freeze_orientation open_gripper close_
146	gripper object_grasped",
147	"args": {}
148	}
149	]
150	}
151	'''
152	Diagnosis ties the failure to the error message. Reasoning explains success, each
153	constraint, any condition-label alternative, and any unrepresented clause.
154	Constraints are the corrected ordered list.

B.6.3 Learning and calibration prompts

These templates expose scalar choices and measured evidence without permitting the model to invent a new operator, stage, reward implementation, or task evaluator. Initial selection adjusts weights without rollout evidence. The BC-prior diagnostic variant then uses a deterministic prior rollout, not residual-RL training statistics, to identify components blocking progress. Its system prompt includes the source’s BC-specific note in addition to the shared supervision and weight-adjustment instructions. The separate residual-authority pair uses learning traces. None of these prompts receives or edits the code-owned tolerance fields described in section B.6.1.

The optional video summarizer is an evidence-producing auxiliary model rather than a program-writing or scalar-editing agent. Its prose suggestions do not have write authority: the governing calibration system prompts below impose the allowed fields, numeric ranges, and output schema. The appendix therefore reproduces the formation and active initialization/calibration templates, while excluding the deprecated stage/global-runner variants, raw video-captioning instructions, and all run-specific transcripts.

B.6.3.1 Initial reward-weight selection

Table B.14: System prompt for initial reward-weight selection

Line	System prompt text (verbatim)
2	You are a supervisor for a reinforcement learning agent. The agent is training on a multi-stage task where each stage has its own set of reward components.
4	These reward components are based on cost terms that have been effective in prior optimal control settings. Do not add, remove, or redefine reward components.
6	Each reward component <i>*i*</i> defines:
8	1. a goal requirement,
9	2. a shaping reward function that increases as the agent gets closer to meeting that requirement, and
10	3. a constraint check that measures whether the goal requirement has been met.
12	A stage is considered complete when all component constraints are satisfied.
14	Task names, object labels, stage numbers, and component indices identify the
15	provided data; they do not determine which component deserves more weight.
16	Base each decision on the current physical requirement and, when provided,
17	measured evidence. Do not infer a stored solution from a familiar name.
19	Short rule examples, not task solutions:
20	- Weights [1, 2] and [2, 4] express the same relative preference after normalization.
21	- A pose hold during release and a pose requirement that validates the final
22	object state can have different roles even when both use the same operator.
24	Physical roles used by the weight rules:
25	- Broad body: roughly solid/boxlike/round grasp cross-section ≥ 0.05 m,
26	without a narrow axis or receiver-fit requirement.
27	- Small solid body: cross-section above 0.03 m and below 0.05 m; size alone
28	does not make orientation mission-critical.
29	- Small/slender: cross-section ≤ 0.03 m, or a long/thin/axis-critical shape.
30	- Release guard: robot/tool pose held while opening after a supported result;
31	distinguish this from a component checking the object's final validity.
32	- Precision progress: the traveled amount itself validates current contact,
33	fit, or actuation; free-space waypoints are not precision progress.
35	You initialize reward-component Weights for ONE stage before RL.
36	Adjust ONLY reward-component Weights.
38	Do NOT change reward/constraint definitions, indices, ordering, logic
39	structure, targets, distances, or any other configuration.
41	Use only the provided task/stage semantics, observed objects, adjacent-stage
42	context, component explanations, and current weights.
43	Do NOT assume rollout statistics, success rates, reward traces, or hidden task
44	facts.
46	Weights:
47	- Set weights by semantic role, not performance. Reward is normalized by $\text{sum}(\text{weights})$, so tune relative ratios.
48	- Default init-sweep weight is 1.0. Use simple ratios.
49	- Broad free-space progress, broad-body ordinary pose holds, broad-body top-down/side orientation maintenance, and non-critical maintenance terms usually stay in [1.0, 1.5].

Continued on next page

Table B.14: System prompt for initial reward-weight selection (continued)

Line	System prompt text (verbatim)
50	- For broad_body tasks with no small/slender, receiver/opening, constrained-feature, or final-object-validity requirement, avoid weights above 2.0. Grasp/object-state may be 2.0 if losing the state invalidates all progress, but do not inflate it to 2.5-3.0 on init sweep.
51	- Direct fit/contact, small-cross-section grasp-lock, constrained-feature manipulation, precision directional progress, mission-critical angular constraints, and final object validity may use [1.5, 3.0] depending on whether losing that component invalidates the stage.
52	- Open-gripper setup/contact orientation terms and broad sliding-pull orientation terms are usually [1.0, 1.5] and should not be weighted as mission-critical angular constraints unless the current stage truly requires angular fit, insertion, final object validity, or a force/torque axis.
53	- Gripper/object state components are important when they define contact, release, or grasp validity; usually use 1.5, use 2.0 for broad-body state validity when losing state invalidates progress, and reserve 2.5-3.0 for small/slender, receiver/opening, constrained-feature, or final-object-validity cases.
54	- Release_guard robot/tool pose holds usually stay in [1.0, 1.5]; do not give them narrow-feature/manipulation-level weights unless the stage explicitly says the held tool pose itself is the contact-validity condition.
55	- Avoid weight inflation: do not set every core component above 2.0 on the init sweep, and do not make broad-body orientation/maintenance terms higher than the primary progress/contact term.
56	- weight must be finite and in [0.1, 10.0]. Correct any out-of-bound value.
58	Return ONLY one valid JSON object with exactly:
59	"Diagnosis", "Reasoning", "Weights"
61	"Diagnosis": briefly list the semantic role of each tunable component.
62	"Reasoning": briefly justify the labels and relative weight ratios.
63	"Weights" MUST contain EXACTLY the same indices as current weights.

Table B.15: User template for initial reward-weight selection

Line	User prompt text (verbatim)
2	Overall task description: {task_description}
4	Detectable objects: {detectable_objects}
6	Stage index/total number of stages: {stage_id}/{stage_count}
8	Stage description: {atomic_action}
10	Current subtask constraint ops/args:
11	{current_constraint_ops}
13	Adjacent subtask context (for transition semantics only; tune ONLY the current stage):
14	Previous subtask:
15	{previous_stage_context}
17	Next subtask:
18	{next_stage_context}
20	Index-ordered reward component explanations:
21	{constraints_explanations}

Continued on next page

Table B.15: User template for initial reward-weight selection (continued)

Line	User prompt text (verbatim)
23	Current weights:
24	{current_weights}

B.6.3.2 BC-prior-informed reward-weight adjustment

Table B.16: System prompt for BC-prior-informed weight adjustment

Line	System prompt text (verbatim)
2	You are a supervisor for a reinforcement learning agent. The agent is training on a multi-stage task where each stage has its own set of reward components.
4	These reward components are based on cost terms that have been effective in prior optimal control settings. Do not add, remove, or redefine reward components.
6	Each reward component <i>*i*</i> defines:
8	1. a goal requirement,
9	2. a shaping reward function that increases as the agent gets closer to meeting that requirement, and
10	3. a constraint check that measures whether the goal requirement has been met.
12	A stage is considered complete when all component constraints are satisfied.
14	Task names, object labels, stage numbers, and component indices identify the
15	provided data; they do not determine which component deserves more weight.
16	Base each decision on the current physical requirement and, when provided,
17	measured evidence. Do not infer a stored solution from a familiar name.
19	Short rule examples, not task solutions:
20	- Weights [1, 2] and [2, 4] express the same relative preference after normalization.
21	- A pose hold during release and a pose requirement that validates the final
22	object state can have different roles even when both use the same operator.
24	Physical roles used by the weight rules:
25	- Broad body: roughly solid/boxlike/round grasp cross-section ≥ 0.05 m,
26	without a narrow axis or receiver-fit requirement.
27	- Small solid body: cross-section above 0.03 m and below 0.05 m; size alone
28	does not make orientation mission-critical.
29	- Small/slender: cross-section ≤ 0.03 m, or a long/thin/axis-critical shape.
30	- Release guard: robot/tool pose held while opening after a supported result;
31	distinguish this from a component checking the object's final validity.
32	- Precision progress: the traveled amount itself validates current contact,
33	fit, or actuation; free-space waypoints are not precision progress.
35	You are doing an initial weight acquisition pass from a deterministic BC-prior rollout.
36	The rollout statistics are not RL-training results; they show where the BC prior times out.
37	The timeout/stuck component rates are the key evidence: high not_satisfied means that
38	component blocked the BC prior for envs that reached and timed out in that stage.

Continued on next page

Table B.16: System prompt for BC-prior-informed weight adjustment (continued)

Line	System prompt text (verbatim)
40	Adjust ONLY reward-component Weights for the current stage.
42	You tune ONE stage by adjusting ONLY reward-component Weights.
44	Do NOT change reward/constraint definitions, indices, ordering, logic structure, or any other configuration.
46	Use stage semantics and population stats.
47	Reward is normalized by sum(weights), so tune relative ratios.
49	For each component, identify its semantic role and main blocker:
50	- poor progress / control
51	- weight tradeoff
52	- required state not maintained
54	Judge meaningful completion by whether the resulting state is actually suitable for the intended interaction, not merely whether predicates became true.
55	Do NOT use weighted reward magnitude alone as evidence of geometric closeness.
57	Return ONLY one valid JSON object (no markdown, no code fences, no extra text).
58	Top-level keys MUST be EXACTLY:
59	"Diagnosis", "Reasoning", "Weights"
61	- "Weights" MUST contain EXACTLY the same indices as the provided current weights.
62	- Do NOT add/remove/rename indices.
64	Weights:
65	- finite, final value in [0.1, 10.0]
66	- per-index step limit: new_w[i] in [0.5*old_w[i], 2.0*old_w[i]]
67	- if a current weight lies outside the hard bounds, you MUST correct it into the hard-bound interval on this pass
69	Optimization priority:
70	1) Increase meaningful stage completion.
71	2) Preserve semantic validity of the interaction.
72	3) Improve precision only if (1) and (2) do not worsen.
73	4) Improve efficiency only if (1), (2), and (3) do not worsen.

Table B.17: User template for BC-prior-informed weight adjustment

Line	User prompt text (verbatim)
2	Overall task description: {task_description}
4	Detectable objects: {detectable_objects}
6	Stage index/total number of stages: {stage_id}/{stage_count}
8	Stage description: {atomic_action}
10	Index-ordered reward component explanations:
11	{constraints_explanations}
13	Current weights:
14	{current_weights}
16	=====
17	END-TO-END PERFORMANCE (ALL ENVS)
18	=====
19	{end_to_end_feedback}
21	Stage satisfaction times:
22	{stage_satisfaction_times}

Continued on next page

Table B.17: User template for BC-prior-informed weight adjustment (continued)

Line	User prompt text (verbatim)
24	Per-component constraint satisfaction rates:
25	{constraint_satisfaction_rates}
27	Per-component reward values (multiplied by weights):
28	{reward_values}

B.6.3.3 Residual-authority calibration

Table B.18: System prompt for residual-authority calibration

Line	System prompt text (verbatim)
2	You are a supervisor for a reinforcement learning agent. The agent is training on
	a multi-stage task where each stage has its own set of reward components.
4	These reward components are based on cost terms that have been effective in prior
	optimal control settings. Do not add, remove, or redefine reward components.
6	Each reward component <i>*i*</i> defines:
8	1. a goal requirement,
9	2. a shaping reward function that increases as the agent gets closer to meeting
	that requirement, and
10	3. a constraint check that measures whether the goal requirement has been met.
12	A stage is considered complete when all component constraints are satisfied.
14	Task names, object labels, stage numbers, and component indices identify the
15	provided data; they do not determine which component deserves more weight.
16	Base each decision on the current physical requirement and, when provided,
17	measured evidence. Do not infer a stored solution from a familiar name.
19	Short rule examples, not task solutions:
20	- Weights [1, 2] and [2, 4] express the same relative preference after
	normalization.
21	- A pose hold during release and a pose requirement that validates the final
22	object state can have different roles even when both use the same operator.
24	Physical roles used by the weight rules:
25	- Broad body: roughly solid/boxlike/round grasp cross-section ≥ 0.05 m,
26	without a narrow axis or receiver-fit requirement.
27	- Small solid body: cross-section above 0.03 m and below 0.05 m; size alone
28	does not make orientation mission-critical.
29	- Small/slender: cross-section ≤ 0.03 m, or a long/thin/axis-critical shape.
30	- Release guard: robot/tool pose held while opening after a supported result;
31	distinguish this from a component checking the object's final validity.
32	- Precision progress: the traveled amount itself validates current contact,
33	fit, or actuation; free-space waypoints are not precision progress.
35	You tune residual-RL agent hyperparameters for a manipulation task.
37	Allowed edits only: params.env.clip_actions, params.config.residual_rl.residual_
	bounds,
38	params.config.residual_rl.clip_composed_actions, params.config.residual_rl.alpha_
	final,

Continued on next page

Table B.18: System prompt for residual-authority calibration (continued)

Line	System prompt text (verbatim)
39	and params.config.residual_rl.alpha_warmup_epochs. All reward, constraint, task,
40	stage,
41	robot, and BC-prior settings are fixed.
42	The prompt gives full detectable_objects JSON, short stage labels,
43	action-dimension labels,
44	outcome/stage-satisfaction stats, compact TensorBoard scalar summaries, and when
45	available
46	a 240x240 image of Episode/Hariti/success_rate_window.
47	Collapse definition:
48	- A collapse requires the latest post-warmup success_rate_window to be at least
49	50% lower
50	than the previous post-warmup peak: peak_to_latest_drop_fraction >= 0.5.
51	- Do not classify smaller late dips as collapse. If collapse_after_peak is false,
52	explicitly
53	mention that the curve may be a normal run unless other evidence strongly
54	indicates instability.
55	- Inspect the attached success_rate_window plot, when present, and use the visual
56	trend to
57	sanity-check the numeric summary before deciding.
58	The scalar summary includes:
59	- Episode/Hariti/success_rate_window after critic_warmup_epochs only, including
60	first valid value,
61	stable peak, collapse onset, latest value, peak-to-latest drop, and 9 or fewer
62	trend samples
63	- Episode/Hariti/false_complete_rate
64	- Episode/Hariti/timeout_rate
65	- Hariti/rollout_step_reward_mean/window/iter
66	- Hariti/rollout_epoch_step_reward_mean/iter
67	Use the action-dimension labels when editing residual_bounds. residual_bounds[i]
68	controls the
69	residual range for action dimension i.
70	Decision policy:
71	- First read the success_rate_window trend samples and inspect the attached plot,
72	if present.
73	- Use collapse_onset_step, stable_peak_step, latest_post_warmup, and peak_to_
74	latest_drop_fraction
75	to decide whether the 50% collapse rule is actually met.
76	- If collapse_after_peak is true under the 50% rule, slow down residual authority
77	by increasing
78	alpha_warmup_epochs, up to 1000.
79	- If collapse happens near or after the current warmup limit, also reduce alpha_
80	final or tighten
81	the action dimensions most likely causing drift.
82	- If collapse_after_peak is false, do not reduce aggression solely because the
83	latest value is
84	slightly below the peak; say it may be a normal run and use outcome stats, plot
85	shape, false

Continued on next page

Table B.18: System prompt for residual-authority calibration (continued)

Line	System prompt text (verbatim)
75	completions, timeouts, and rewards to choose minimal changes.
76	- Treat enabling clip_composed_actions, reducing clip_actions, reducing alpha_final, and tightening
77	residual_bounds as restrictive safety moves. Use them only when there is clear collapse,
78	instability, reward-proxy drift, high false-complete rate, or direct action-clipping evidence.
79	- If there is no clear collapse/instability, do not make action clipping more restrictive. Preserve
80	the current clip_composed_actions value, or set it false/unlock clipping if stable success appears
81	limited by composed-action clipping.
82	- If success is low without a peak/drop pattern and rewards are flat, allow a small amount more
83	correction: consider increasing clip_actions up to its hard limit, loosening the relevant
84	residual_bounds within bounds, or keeping alpha_final high rather than reducing it.
85	- If reward improves but success_rate_window drops sharply, treat that as reward-proxy drift and make the residual safer.
86	- Change the fewest hyperparameters needed; prefer alpha_warmup_epochs/alpha_final changes before broad residual_bounds changes when the signature is alpha-collapse.
88	Hard numeric bounds:
89	- clip_actions must be finite and in [0.5, 2.0].
90	- alpha_final must be finite and in [0.05, 1.5].
91	- alpha_warmup_epochs must be an integer in [0, 1000].
92	- residual_bounds must be the same length as the current list.
93	- each residual_bounds value must be finite and in [0.02, 1.0].
94	- Per-iteration step limit: each numeric value should stay within [0.5 * current, 1.5 * current],
95	unless the current value is outside the hard numeric bounds and must be corrected.
97	Return ONLY one valid JSON object with exactly:
98	"Diagnosis", "Reasoning", "Hyperparameters"
100	"Diagnosis": summarize whether the run appears under-corrected, over-corrected, unstable, or stable.
101	"Reasoning": justify each changed hyperparameter and why unchanged values should remain unchanged.
102	"Hyperparameters" MUST contain exactly these keys:
103	"clip_actions", "residual_bounds", "clip_composed_actions", "alpha_final", "alpha_warmup_epochs"
104	Do not add, remove, or rename keys.

Table B.19: User template for residual-authority calibration

Line	User prompt text (verbatim)
2	Task:
3	{task_description}
5	Detectable objects:
6	{detectable_objects}
8	Stages:
9	{stage_labels}
11	Action mode:
12	{action_mode}
14	Action dimension labels:
15	{action_dimension_labels}
17	Current residual-RL hyperparameters:
18	{current_hyperparameters}
20	Outcome stats:
21	{outcome_stats}
23	Stage satisfaction summary:
24	{stage_satisfaction_summary}
26	Exact TensorBoard scalar summaries:
27	{reward_success_event_summary}
29	Success-rate plot:
30	{success_rate_plot_summary}

APPENDIX C

Extended Experiments

C.1 Ablations, Robustness, and Compute

Component and baseline ablations. Table C.1 gives the complete taskwise values behind the aggregate baseline and ablation comparisons in chapter 7.

False completion and robustness. The L42 controller’s task-level false-completion rates are 0.37, 0, 0.05, 3.26, 0.45, 0, 0, 0, and 1.05% for T0–T8, respectively (0.58% macro). Across lineages L42–L44, the corresponding macro mean is 0.60%. These values are evaluated against g_{ext} , not against the program’s own terminal guard.

Across the horizon pairs T1/T6, T2/T7, and T0/T3, L42 retains 91.4% of the shorter-task performance. The matched sparse-reward handoff falls from a 65.14% average on the shorter members to 22.21% on the longer members.

Compute accounting. KUAUFU consumes 1.55 GPU-hours for formation, 69.03 for MPC verification and mentor collection, 3.21 for Stage-BC, and 48.86 for residual learning, totaling 122.67. Eureka consumes 412.24 for search and 67.36 for final training, totaling 479.60. These are one-GPU worker-hours, not elapsed multi-GPU wall-clock time.

Table C.1: External success in percent. This table expands the aggregate discussion in chapter 7.

Method	T0	T1	T2	T3	T4	T5	T6	T7	T8	Mean
Stage-free BC	52.20	12.16	3.38	11.17	5.29	0	0	0	0.77	9.44
Pure RL, naive reward	0.04	0	0.13	0	0.43	0	0	0	0	0.07
Pure RL, SUN reward	0	0	0	0	0	0	0	0	0	0
Stage-free BC + RRL	1.20	67.30	0.03	0	60.31	17.49	3.83	0.16	16.69	18.56
TD-MPC2, naive reward	79.64	27.00	0	5.99	1.07	0	0	0	0	12.63
TD-MPC2 + prior	77.58	13.00	97.47	99.27	0.20	0	0	0	0	31.95
Eureka: Train@1	45.14	0.02	20.25	0.19	1.03	0.01	0	0	0.01	7.41
Eureka: Train@5	0.21	0.01	78.96	8.53	5.29	0	0	0	0	10.33
Eureka: Best ≤ 5	60.53	0.02	78.96	18.17	6.47	0.01	0	0.01	0.01	18.24
Eureka: final	25.85	0.01	71.75	15.16	2.83	0.01	0	0	0.01	12.85
VoxPoser: direct	50	10	0	20	0	0	10	0	0	10.00

APPENDIX D

Real-Robot Systems

D.1 System Configuration, Calibration, and Trials

Robot and gripper configurations. Configurations are Franka FR3 with Franka Hand on T1, T2, T7, and T8; FR3/Robotiq 2F-85 on T4; and Kinova Gen3/Robotiq 2F-85 on T5. Together they cover two arms, two grippers, and six tasks.

Perception and calibration. Simulated workspaces reproduce the physical setups using two or three calibrated RGB-D cameras and constrained pose randomization. Camera models, calibration, point-cloud fusion, frame conventions, and ranges are unreported; transfer is zero-shot without real-world fine-tuning. Robustness beyond this envelope is untested: extrinsic drift, missing depth, novel clutter, and severe occlusion remain risks before operation around people.

Trial accounting and failures. The 106 trials comprise 20 each for T1, T2, and T7; 30 for T8; 10 for T4; and 6 for T5, yielding 3, 11, 7, 9, 4, and 2 successes. Pooled success is 36/106 (33.96%) and task-macro success 34.72%. Grasp acquisition is the leading failure; T1 is lowest at 15%, with grasp failures prominent in T7. Unreported control rates and safety limits are not fabricated.

APPENDIX E

Scene-Rearrangement Supplement

E.1 Published and Defense Objectives

The ICRA paper defines the additive robot cost in equation (4.15) and balances it against human-relation costs on the filtered graph in equation (4.16). The later defense formulation instead averages the retained human edges and uses the product integral $\int f_{\mathcal{R}}(q)f_{\mathcal{I}_i}(q)dq$, transformed by $\exp(-\sigma \sum_i V_i)$; see equation (4.17). The main chapter preserves both versions and identifies which one supports each claim. The published 14% and 30% aggregates are attributed to the published formulation, not retrospectively to the defense extension.

E.2 Task and Embodiment Sensitivity

Office cleaning visits the working group, restocking visits cabinets and shelves, and distribution visits all three functional groups. The additional path-length term in equation (4.19) produces different layouts from the same initial office. Separate optimization for Dingo/UR3 and Husky/UR5 shows that footprint and arm reach change the optimum and can change feasibility; the task-specific office layouts use Dingo. The source figures are reproduced in figures 4.5 and 4.6.

Table E.1: Ratings use a five-point scale and $n = 11$ retained respondents. Bold adjusted values remain below 0.05 after Holm correction over all 14 comparisons.

Scene	Naturalness				Functionality			
	Original	Rearranged	p	p_H	Original	Rearranged	p	p_H
1	2.82	2.00	0.0816	0.7340	3.00	2.73	0.3911	1.0000
2	3.09	3.73	0.0455	0.4554	3.18	3.64	0.2111	1.0000
3	3.55	3.27	0.5884	1.0000	3.91	3.36	0.1669	1.0000
4	3.36	3.00	0.4592	1.0000	3.82	3.09	0.1669	1.0000
5	2.82	2.36	0.1377	1.0000	3.00	3.09	0.7961	1.0000
6	3.64	1.91	0.0170	0.1872	3.64	2.18	0.0016	0.0195
7	3.18	1.91	0.0003	0.0044	3.18	1.82	0.0013	0.0172

E.3 Human-Study Reanalysis

The published study reports 11 participants and seven original/rearranged scene pairs. Its archived CSV contains 21 respondent columns per question, while the archived plotting script sets `skip=10` and retains the final 11 columns. No rationale for excluding the first ten columns is present in the paper artifact. To remain reproducible without inventing that rationale, the audit below uses the same retained subset and the same displayed scene order (3, 4, 5, 7, 8, 6, 10) from the source script.

Each retained respondent rated both members of a scene pair, so two-sided paired t -tests ($df = 10$) replace the independent-samples tests used in the archived script. Holm adjustment controls family-wise error over the 14 scene–measure comparisons. Table entries report original and rearranged means, unadjusted p , and Holm-adjusted p_H .

The archived script also forms purported 95% error bars as $1.96s/n$ rather than $1.96s/\sqrt{n}$.

For $n = 11$, those bars are narrower by a factor of $\sqrt{11} \simeq 3.32$ than conventional normal-approximation standard-error bars. The dissertation therefore does not reproduce the paper’s human-rating plot. More importantly, absence of a significant difference is not an equivalence test: the first five pairs provide no detected adjusted difference, but they do not prove preservation of human preference.

E.4 Held-Out Quantitative Provenance

The paper says pairwise statistics use 90% of SUNCG and its quantitative plot samples 154 held-out rooms. The archived script instead reads a 155-row `final_results2.csv`, reproducing 29.19% more reachable objects and 13.52% on its second measure, which round to the reported 30% and 14%; another 154-row file yields different means. The script calls the second metric interaction area, the source PDF calls it the robot term from Eq. 10, and the prose calls it accessible or open space. With subset and name unresolved, this dissertation calls it the published robot-access measure and does not claim an independent sample reconstruction.

The aggregate gains do not imply that both measures improve in every room: some layouts exchange open area for access to more functional object sides. The source scatterplot in figure 4.3 exposes this distribution.

E.5 Reproducibility Boundary

The paper supplies relation models, objectives, hierarchical ASA/CMA-ES, its data split, evaluation definitions, and result files, but omits optimizer bounds, switching and stopping rules, initial robot-pose construction, and the exact subset. This code snapshot specifies the first three through normalized poses, explicit ASA/CMA settings, and a door-derived start;

these remain source-derived. The subset is unresolved; no settings are inferred from figures or defense prose.

REFERENCES

- Dmitry Berenson, James Kuffner, and Howie Choset. An optimization approach to planning for mobile manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008.
- Kevin Black et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40, 2025.
- Daniel M. Bodily, Thomas F. Allen, and Marc D. Killpack. Motion planning for mobile robots using inverse kinematics branching. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017.
- Serena Booth, W. Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, and Alessandro Allievi. The perils of trial-and-error reward design: Misdesign through overfitting and invalid task specifications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5920–5929, 2023. doi: 10.1609/aaai.v37i5.25733.
- Jan Carius, Farbod Farshidian, and Marco Hutter. MPC-Net: A first principles guided policy search. *IEEE Robotics and Automation Letters*, 5(2):2897–2904, 2020.
- Sarvenaz Chaeibakhsh, Roya Sabbagh Novin, Tucker Hermans, Andrew Merryweather, and Alan Kuntz. Optimizing hospital room layout to reduce the risk of patient falls. In *International Conference on Operations Research and Enterprise Systems*, 2021.
- Qianzhong Chen, Hau Zheng, Justin Yu, Suning Huang, Jiankai Sun, Ken Goldberg, Chuan Wen, Pieter Abbeel, Yide Shentu, Philipp Wu, and Mac Schwager. SARM2: Multi-task stage aware reward modeling for self improving robotic manipulation, 2026.
- Yixin Chen, Siyuan Huang, Tao Yuan, Siyuan Qi, Yixin Zhu, and Song-Chun Zhu. Holistic++ scene understanding: Single-view 3d holistic scene parsing and human pose estimation with human-object interaction and physical commonsense. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.
- Sang Hun Cheong, Brian Y. Cho, Jinhwi Lee, ChangHwan Kim, and Changjoo Nam. Where to relocate?: Object rearrangement inside cluttered and confined environments for robotic manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020.
- Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ

- Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668.
- Sachin Chitta, Benjamin Cohen, and Maxim Likhachev. Planning for autonomous door opening with a mobile manipulator. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010.
- Berk Cicek, Mert K. Er, and Ozgur S. Oguz. CoRAL: Contact-rich adaptive llm-based control for robotic manipulation, 2026.
- Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM Transactions on Graphics*, 31(6):1–11, 2012.
- Shaun Gallagher. *How the Body Shapes the Mind*. Clarendon Press, 2005. doi: 10.1093/0199271941.001.0001.
- Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:265–293, 2021. doi: 10.1146/annurev-control-091420-084139.
- Caelan Reed Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. SkillMimicGen: Automated demonstration generation for efficient skill learning and deployment. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2750–2790, 2025.
- Ran Gong, Xiaohan Zhang, Jinghuan Shang, Maria Vittoria Minniti, Jigarkumar Patel, Valerio Pepe, Riedana Yan, Ahmet Gundogdu, Ivan Kapelyukh, Ali Abbas, Xiaoqiang Yan, Harsh Patel, Laura Herlant, and Karl Schmeckpeper. AnyTask: An automated task and data generation framework for advancing sim-to-real policy learning, 2025.
- Zhao Han, Jordan Allspaw, Gregory LeMasurier, Jenna Parrillo, Daniel Giger, S. Reza Ahmadzadeh, and Holly A. Yanco. Towards mobile multi-task manipulation in a confined and integrated environment with irregular objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020.
- Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
- Nicholas P. Holmes and Charles Spence. Beyond the body schema: Visual, prosthetic, and

- technological contributions to bodily perception and awareness. In Gunther Knoblich, Ian M. Thornton, Marc Grosjean, and Maggie Shiffrar, editors, *Human Body Perception from the Inside Out*, pages 15–64. Oxford University Press, 2005. doi: 10.1093/oso/9780195178371.003.0003.
- Pu Hua, Minghuan Liu, Annabella Macaluso, Yunfeng Lin, Weinan Zhang, Huazhe Xu, and Lirui Wang. GenSim2: Scaling robot data generation with multi-modal and reasoning llms. In *Proceedings of the 8th Conference on Robot Learning*, volume 270, pages 5030–5066, 2025.
- Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3d value maps for robotic manipulation with language models. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 540–562, 2023.
- Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 4573–4602, 2025.
- Brian Ichter et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318, 2023.
- Lester Ingber. Adaptive simulated annealing (ASA): Lessons learned. *Control and Cybernetics*, 25(1):33–54, 1996.
- Chenfanfu Jiang, Siyuan Qi, Yixin Zhu, Siyuan Huang, Jenny Lin, Lap-Fai Yu, Demetri Terzopoulos, and Song-Chun Zhu. Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars. *International Journal of Computer Vision*, 126(9):920–941, 2018.
- Ziyuan Jiao, Zeyu Zhang, Weiqi Wang, David Han, Song-Chun Zhu, Yixin Zhu, and Hangxin Liu. Efficient task planning for mobile manipulation: A virtual kinematic chain perspective. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8288–8294, 2021. doi: 10.1109/IROS51168.2021.9636554.
- Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, et al. Residual reinforcement learning for robot control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6023–6029, 2019.
- Gregory Kahn, Tianhao Zhang, Sergey Levine, and Pieter Abbeel. PLATO: Policy learning

- using adaptive trajectory optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3342–3349, 2017.
- Z. Sadeghipour Kermani, Zicheng Liao, Ping Tan, and H. Zhang. Learning 3d scene synthesis from annotated rgb-d images. *Computer Graphics Forum*, 35(5):197–206, 2016.
- Mohammad Keshavarzi, Aakash Parikh, Xiyu Zhai, Melody Mao, Luisa Caldas, and Allen Y. Yang. SceneGen: Generative contextual scene augmentation using scene graph priors. *arXiv preprint arXiv:2009.12395*, 2020.
- Arturo Laurenzi, Enrico Mingo Hoffman, Matteo Parigi Polverini, and Nikos Tsagarakis. An augmented kinematic model for the cartesian control of the hybrid wheeled-legged quadrupedal robot CENTAURO. *IEEE Robotics and Automation Letters*, 5(2):508–515, 2020. doi: 10.1109/LRA.2019.2961846.
- Sergey Levine and Vladlen Koltun. Guided policy search. In *Proceedings of the International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1–9, 2013.
- Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control, 2022.
- Wei Liang, Jingjing Liu, Yining Lang, Bing Ning, and Lap-Fai Yu. Functional workspace optimization via learning personal preferences from virtual experiences. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):1836–1845, 2019.
- Nejc Likar, Bojan Nemec, and Leon Žlajpah. Virtual mechanism approach for dual-arm manipulation. *Robotica*, 32(6), 2014. doi: 10.1017/S0263574713000763.
- Hangxin Liu, Chi Zhang, Yixin Zhu, Chenfanfu Jiang, and Song-Chun Zhu. Mirroring without overimitation: Learning functionally equivalent manipulation actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.
- Adrià López Escoriza, Nicklas Hansen, Stone Tao, Tongzhou Mu, and Hao Su. Multi-stage manipulation with demonstration-augmented reward, policy, and world model learning. In *Proceedings of the International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 15542–15563, 2025. URL <https://proceedings.mlr.press/v267/escoriza25a.html>.
- Yecheng Jason Ma, William Liang, Hung-Ju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. DrEureka: Language model guided sim-to-real transfer. In *Robotics: Science and Systems*, 2024a.

- Yecheng Jason Ma et al. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024b.
- Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Ireteyio Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1820–1864, 2023. URL <https://proceedings.mlr.press/v229/mandlekar23a.html>.
- David Q. Mayne, James B. Rawlings, Christopher V. Rao, and Pierre O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.
- Yue Meng and Chuchu Fan. Signal temporal logic neural predictive control. *IEEE Robotics and Automation Letters*, 8(11):7719–7726, 2023.
- Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM Transactions on Graphics*, 30(4): 1–10, 2011.
- Tongzhou Mu, Minghua Liu, and Hao Su. DrS: Learning reusable dense rewards for multi-stage tasks. In *International Conference on Learning Representations*, 2024.
- Nelson Nauata, Sepidehsadat Hosseini, Kai-Hung Chang, Hang Chu, Chin-Yi Cheng, and Yasutaka Furukawa. House-GAN++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.
- Mingjie Pan, Jiyao Zhang, Tianshu Wu, Yinghao Zhao, Wenlong Gao, and Hao Dong. OmniManip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17359–17369, 2025.
- Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. ATISS: Autoregressive transformers for indoor scene synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 12013–12026, 2021.
- Shivansh Patel, Xinchun Yin, Wenlong Huang, Shubham Garg, Hooshang Nayyeri, Li Fei-Fei, Svetlana Lazebnik, and Yunzhu Li. A real-to-sim-to-real approach to robotic manipulation with VLM-generated iterative keypoint rewards. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 8258–8266, 2025. doi: 10.1109/ICRA55743.2025.11127585.

- Jerry Pratt, Peter Dilworth, and Gill Pratt. Virtual model control of a bipedal walking robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.
- Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *International Journal of Robotics Research*, 20(2):129–143, 2001.
- Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5899–5908, 2018.
- Daniel Ritchie, Kai Wang, and Yu-An Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.
- Krishna Shankar. *Kinematics and Local Motion Planning for Quasi-Static Whole-Body Mobile Manipulation*. PhD thesis, California Institute of Technology, 2016.
- Juyi Sheng, Ziyi Wang, Peiming Li, and Mengyuan Liu. MP1: MeanFlow tames policy learning in 1-step for robotic manipulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18532–18539, 2026. doi: 10.1609/aaai.v40i22.38919.
- Mustafa Shukor et al. SmolVLA: A vision-language-action model for affordable and efficient robotics. arXiv:2506.01844, 2025.
- Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating situated robot task plans using large language models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 11523–11530, 2023.
- Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017.
- Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4444–4451, 2017.
- Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, volume 80, pages 2107–2116, 2018.

- Kasra Torshizi, Anukriti Singh, Sidharth Mathur, Khuzema Habib, Leo Du, and Pratap Tokekar. STeP: Signal temporal logic for precise specifications for action generation with vision language models, 2026.
- Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1930–1936, 2015.
- Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. PlanIT: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics*, 38(4):132:1–132:15, 2019.
- WeiQi Wang, Zihang Zhao, Ziyuan Jiao, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu. Rearrange indoor scenes for human–robot co-activity. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 11943–11949, 2023. doi: 10.1109/ICRA48891.2023.10160303.
- WeiQi Wang, Zhi Li, Yudong Lei, David Martinez, Xiaofeng Gao, Yuxin Jiang, Chenfanfu Jiang, Yingnian Wu, Demetri Terzopoulos, and Ran Gong. SUN: Persistent programs for language-grounded control-to-learning-to-real policies, 2026. Manuscript in preparation.
- Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, et al. RoboGen: Towards unleashing infinite data for automated robot learning via generative simulation. In *Proceedings of the International Conference on Machine Learning*, volume 235, pages 51936–51983, 2024.
- Yuquan Wang, Christian Smith, Yiannis Karayiannidis, and Petter Ögren. Whole body control of a dual-arm mobile robot using a virtual kinematic chain. *International Journal of Humanoid Robotics*, 13(01), 2016.
- Dan Xie, Tianmin Shu, Sinisa Todorovic, and Song-Chun Zhu. Learning and inferring Dark Matter and predicting human intents and trajectories in videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(7):1639–1652, 2018. doi: 10.1109/TPAMI.2017.2728788.
- Tianbao Xie et al. Text2Reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations*, 2024.
- Ge Yan et al. ManiFlow: A general robot manipulation policy via consistency flow training. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 2268–2293, 2025.

- Lap Fai Yu, Sai Kit Yeung, Chi Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: Automatic optimization of furniture arrangement. *ACM Transactions on Graphics*, 30(4), 2011.
- Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, et al. Language to rewards for robotic skill synthesis. In *Proceedings of the 7th Conference on Robot Learning*, volume 229, pages 374–404, 2023.
- Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=e5jGTEiJMT>.
- Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Robotics: Science and Systems*, 2024.
- Andy Zhai, Brae Liu, Bruno Fang, Chalse Cai, Ellie Ma, Ethan Yin, Hao Wang, et al. Igniting VLMs toward the embodied space. arXiv:2509.11766, 2025.
- Jiahui Zhang, Yusen Luo, Abrar Anwar, Sumedh Anand Sontakke, Joseph J. Lim, Jesse Thomason, Erdem Biyik, and Jesse Zhang. ReWiND: Language-guided rewards teach robot policies without new demonstrations, 2025a.
- Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. FlowPolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(14):14754–14762, 2025b. doi: 10.1609/aaai.v39i14.33617.
- Song-Hai Zhang, Shao-Kui Zhang, Wei-Yu Xie, Cheng-Yang Luo, Yong-Liang Yang, and Hongbo Fu. Fast 3d indoor scene synthesis by learning spatial relation priors of objects. *IEEE Transactions on Visualization and Computer Graphics*, 28(9):3082–3092, 2022a. doi: 10.1109/TVCG.2021.3050143.
- Yongqi Zhang, Haikun Huang, Erion Plaku, and Lap-Fai Yu. Joint computational design of workspaces and workplans. *ACM Transactions on Graphics*, 40(6):228:1–228:16, 2021.
- Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *ACM Transactions on Graphics*, 39(2):1–21, 2020.
- Zeyu Zhang, Ziyuan Jiao, Weiqi Wang, Yixin Zhu, Song-Chun Zhu, and Hangxin Liu.

- Understanding physical effects for effective tool-use. *IEEE Robotics and Automation Letters*, 7(4):9469–9476, 2022b. doi: 10.1109/LRA.2022.3191793.
- Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.
- Yibiao Zhao and Song-Chun Zhu. Image parsing with stochastic scene grammar. In *Advances in Neural Information Processing Systems*, 2011.
- Jixuan Zhi and Jyh-Ming Lien. Improving human-robot collaboration via computational design. *IEEE Robotics and Automation Letters*, 10(2):1074–1081, 2025. doi: 10.1109/LRA.2024.3519863.
- Jixuan Zhi, Lap-Fai Yu, and Jyh-Ming Lien. Designing human-robot coexistence space. *IEEE Robotics and Automation Letters*, 6(4):7161–7168, 2021.
- Yixin Zhu, Tao Gao, Lifeng Fan, Siyuan Huang, Mark Edmonds, Hangxin Liu, Feng Gao, Chi Zhang, Siyuan Qi, Ying Nian Wu, Josh Tenenbaum, and Song-Chun Zhu. Dark, beyond deep: A paradigm shift to cognitive ai with humanlike common sense. *Engineering*, 6(3): 310–345, 2020.