UNIVERSITY OF CALIFORNIA
SANTA CRUZ

BUILDING EMBODIED AGENTS FROM PRETRAINED FOUNDATION MODELS: FROM NEURO-SYMBOLIC COMPOSITION TO SELF-EVOLVING 3D GENERATION

A dissertation submitted in partial satisfaction
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

by

Kaizhi Zheng

June 2026

The Dissertation of Kaizhi Zheng
is approved:

Professor Xin Eric Wang, Chair

Professor Yi Zhang

Professor Cihang Xie

Peter Biehl
Vice Provost and Dean of Graduate Studies

Contents

Figures and Tables viii

Abstract xi

Dedication xiii

Acknowledgments xiv

1. Introduction 1
- 1.1. Why Compose Pretrained Models? . . . 2
- 1.2. Challenges . . . 4
- 1.3. Thesis Overview . . . 6
- 1.4. Contributions . . . 8

I. Embodied Reasoning and Interaction 10

2. VLMbench: A Compositional Benchmark for Vision-and-Language Manipulation 12
- 2.1. Introduction . . . 12
- 2.2. Related Work . . . 15
- 2.3. AMSolver: Automatic Manipulation Solver . . . 18
  - 2.3.1. Rule-based Unit Tasks . . . 18
  - 2.3.2. Object-centric Representation . . . 20
  - 2.3.3. Automatic Demonstration Generation . . . 21
- 2.4. VLMbench: Visual-and-Language Manipulation Benchmark . . . 21
  - 2.4.1. Problem Definition . . . 21
  - 2.4.2. Tasks and Dataset . . . 23
- 2.5. Vision-and-Language Manipulation Agent . . . 24
- 2.6. Experiments . . . 27
  - 2.6.1. Experimental Setup . . . 27
  - 2.6.2. Result Analysis . . . 29
- 2.7. Conclusion . . . 32

3. JARVIS: A Neuro-Symbolic Commonsense Reasoning Framework for Conversational Embodied Agents **33**
3.1. Introduction . . . 33
3.2. Related Work . . . 36
3.3. Neuro-Symbolic Conversational Embodied Agents . . . 37
3.3.1. Problem Formulation . . . 37
3.3.2. Proposed Methods . . . 39
3.4. Experiments . . . 43
3.4.1. Dataset and Tasks . . . 43
3.4.2. Experimental Setup . . . 44
3.4.3. Main Results and Analysis . . . 45
3.4.4. Few-Shot Learning . . . 47
3.4.5. Unit Test of Individual Module . . . 47
3.5. Conclusion . . . 48

II. Interleaved Multimodal Generation **49**

4. Interleaved Vision-and-Language Generation via Generative Voken **51**
4.1. Introduction . . . 51
4.2. Related Work . . . 54
4.3. Method . . . 56
4.3.1. Multimodal Understanding Module . . . 57
4.3.2. Multimodal Generation Module . . . 58
4.3.3. Training Strategy . . . 60
4.4. Experiments . . . 63
4.4.1. Implementation Details . . . 63
4.4.2. Experimental Setup . . . 64
4.4.3. Main Results . . . 67
4.4.4. Ablation Studies . . . 68
4.5. Conclusion . . . 72

III. Generative 3D World Construction **74**

5. Constructing A 3D Scene from a Single Image **76**
5.1. Introduction . . . 76
5.2. Related Work . . . 79

5.3. Method . . . 81
5.3.1. Structured Latent Representation . . . 82
5.3.2. Spatial Prior Initialization . . . 83
5.3.3. Region-based Generation . . . 84
5.3.4. Region Fusion . . . 87
5.4. Experiments . . . 88
5.4.1. Experimental Setup . . . 88
5.4.2. Main Results . . . 89
5.4.3. Ablation Study . . . 92
5.5. Conclusion . . . 93

6. Self-Evolving 3D Scene Generation from a Single Image 95
6.1. Introduction . . . 95
6.2. Related Work . . . 98
6.3. Method . . . 101
6.3.1. Overview . . . 101
6.3.2. Self-Evolving 3D Scene Generation Stages . . . 101
6.3.3. Implementation Details . . . 108
6.4. Experiments . . . 108
6.4.1. Experimental Setup . . . 108
6.4.2. Main Results . . . 109
6.4.3. Ablation Study . . . 112
6.5. Conclusion . . . 115

7. Conclusion and Future Work 116
7.1. Summary of Contributions . . . 116
7.2. Future Work . . . 118
7.2.1. Interactive 4D Dynamic World Simulators . . . 118
7.2.2. Training World Action Models via Generative World Models . . . 119
7.2.3. MLLM-driven Self-Correction and Evolution Loop . . . 120

Appendix 123

A. Appendix of VLMbench 123
A.1. Task Details . . . 123
A.1.1. Pick & Place Objects . . . 124
A.1.2. Stack Objects . . . 126
A.1.3. Drop Pencil . . . 127
A.1.4. Put Into Shape Sorter . . . 128

A.1.5. Pour Water . . . 129
A.1.6. Wipe Table . . . 131
A.1.7. Use Drawer . . . 133
A.1.8. Use Door . . . 134
A.2. Implementation Details . . . 135
A.2.1. AMSolver . . . 135
A.2.2. VLMbench . . . 136
A.2.3. 6D-CLIPort . . . 137
A.3. Dataset Details . . . 138
A.4. Additional Experiments . . . 140
A.4.1. Goal-Conditioned Success Rates . . . 141
A.4.2. Failure Cases . . . 142
A.4.3. Partially Modal Agents . . . 144

B. Appendix of JARVIS **147**
B.1. Implementation Details . . . 147
B.1.1. Learning Modules . . . 147
B.1.2. Task Completion Process of the JARVIS framework . . . 152
B.1.3. Experiment Details for Two Agent Task Completion (TATC) . . . 152
B.2. Notation Table . . . 153
B.3. Case Study . . . 155
B.4. Error Analysis . . . 157

C. Appendix of MiniGPT-5 **164**
C.1. Experimental Settings . . . 164
C.1.1. Datasets . . . 164
C.1.2. Data Format . . . 165
C.2. More Experiments . . . 167
C.2.1. Evaluation of Guidance Scale . . . 167
C.2.2. Evaluation of Voken Number . . . 168
C.2.3. Texture Preservation via FID . . . 169
C.3. More Qualitative Examples . . . 170

D. Appendix of SceneFuse-3D **176**
D.1. More Results . . . 176
D.1.1. Additional Qualitative Results . . . 176
D.1.2. Additional Baseline . . . 178
D.2. Method Details . . . 179
D.3. Experiment Settings . . . 182
D.4. Limitation . . . 184

E. Appendix of EvoScene **185**
E.1. Method Details 185
E.2. More Experiment Results 188

Figures and Tables

List of Figures

2.1. VLMbench Overview. 13
2.2. The Unit Task Templates of AMSolver. 17
2.3. VLMbench task types 22
2.4. The Structure of 6D-CLIPort. 26

3.1. JARVIS task example 35
3.2. JARVIS framework overview 38

4.1. Overview of MiniGPT-5 52
4.2. MiniGPT-5 pipeline overview 56
4.3. Qualitative Results of MiniGPT-5. 73

5.1. SceneFuse-3D teaser 77
5.2. SceneFuse-3D pipeline overview 81
5.3. SceneFuse-3D qualitative comparison 90
5.4. SceneFuse-3D qualitative ablation 92

6.1. EvoScene teaser 96
6.2. EvoScene pipeline overview 102
6.3. EvoScene qualitative comparison 110
6.4. EvoScene iterative refinement ablation 113
6.5. EvoScene depth guidance ablation 114

A.1. The Environment Used in VLMbench. 138
A.2. All Instance-level Tasks in VLMbench. 139
A.3. 6D-CLIPort failure cases 142
A.4. 6D-CLIPort with GT waypoints 143

B.1. JARVIS EDH example 154
B.2. JARVIS TfD example 157
B.3. JARVIS TATC example 162
B.4. JARVIS action failure examples 163
B.5. JARVIS sub-goal estimation failure 163

C.1. Human evaluation interface 167
C.2. CFG scale analysis on CC3M 168
C.3. Voken number analysis on CC3M 170
C.4. VIST multimodal image generation examples 172
C.5. VIST free multimodal generation examples 173
C.6. MMDialog multimodal dialog examples 174
C.7. CC3M text-to-image examples 175

D.1. Additional SceneFuse-3D comparisons 177
D.2. SceneFuse-3D real-image comparisons 178
D.3. WonderWorld comparison 179

E.1. Additional EvoScene comparisons 186
E.2. EvoScene iteration comparisons 189

List of Tables

2.1. Robotic manipulation benchmark comparison 16
2.2. VLMbench task categories 25
2.3. Success rates by task category 29
2.4. Success rates by task variation 30

3.1. JARVIS EDH and TfD results 43
3.2. JARVIS TATC results 44
3.3. JARVIS category success rates 44
3.4. JARVIS ablation studies 46

4.1. Image generation on VIST 66
4.2. Narration Generation on VIST 66
4.3. VIST human evaluation 67
4.4. MMDialog generation results 69
4.5. Method design ablation on CC3M 69
4.6. Prompt influence on VIST image generation 70
4.7. Model design ablation on CC3M 71
4.8. Backbone comparison on CC3M and VIST 71

5.1. SceneFuse-3D quantitative comparison 89
5.2. SceneFuse-3D rendered-view metrics 92
5.3. SceneFuse-3D ablation results 92

6.1. EvoScene quantitative comparison 107

6.2. EvoScene rendered-view metrics 109
6.3. EvoScene ablation results 109

A.1. VLMbench task variations 123
A.2. VLMbench object models 124
A.3. VLMbench episodes by task 145
A.4. Full task-category results 146
A.5. Full task-variation results 146

B.1. JARVIS symbolic predicates 153
B.2. JARVIS notation table 153
B.3. JARVIS action failure rates 157
B.4. JARVIS action failure categories 158

C.1. Texture preservation on VIST 169

D.1. SceneFuse-3D pipeline summary 183

Abstract

Building Embodied Agents from Pretrained Foundation Models: From Neuro-Symbolic Composition to Self-Evolving 3D Generation

by

Kaizhi Zheng

Artificial intelligence has produced a growing collection of powerful pretrained foundation models, including large language models, multimodal large language models, image and video diffusion models, image-to-3D generators, and depth estimators. The embodied AI community has, in parallel, begun training vision-language-action models on large-scale robot data. Yet generalist embodied competence spans more than any single pretrained model is designed to cover: grounding instructions in physical action, communicating through coherent multimodal content, and constructing 3D worlds.

This dissertation studies how to build embodied systems by composing pretrained foundation models, rather than training new large-scale embodied systems from scratch. The central claim is that, for each capability needed by an embodied agent, a strong pretrained model already exists, and the right interfaces between such models can turn independently trained components into capable embodied agents. The dissertation develops four such interfaces: symbolic, learnable token, spatial, and iterative.

Part I examines how pretrained perception and language models can be composed into embodied reasoning systems. I introduce VLMbench, a compositional benchmark for vision-and-language manipulation that exposes where current vision-language models fail to ground compositional instructions in physically constrained motion. I then present

JARVIS, a neuro-symbolic framework that treats pretrained vision and language models as sensors and binds their outputs through symbolic task-level and action-level reasoning, enabling agents to complete long-horizon household tasks from free-form dialog.

Part II studies the interface between pretrained multimodal language models and pretrained image generators. I present MiniGPT-5, which connects them through generative vokens, a small set of learnable tokens that project language hidden states directly into the diffusion conditioning space. Both backbones remain frozen, and only a lightweight interface is trained, enabling description-free interleaved text-and-image generation.

Part III asks the same composition question for 3D scene construction without any scene-level training data. SceneFuse-3D composes a pretrained image-to-3D object generator into a scene generator through region-wise decomposition and spatial-aware completion, using pretrained depth and detection priors as spatial anchors. EvoScene then couples pretrained 3D and video generators in a self-evolving 2D–3D loop, in which depth-conditioned video synthesis provides novel views that progressively refine the underlying mesh.

Together, these contributions outline a methodology for embodied AI grounded in the composition of pretrained foundation models through deliberately designed interfaces: symbolic glue in JARVIS, learnable token interfaces in MiniGPT-5, spatial priors in SceneFuse-3D, and iterative 2D–3D feedback in EvoScene. The dissertation argues that the next generation of embodied agents can be built by reusing, rather than rebuilding, the foundation models the field is producing.

Dedicated to everyone who has supported me along this journey.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Xin Eric Wang. His visionary guidance, unwavering support, and immense patience have been the cornerstone of my Ph.D. journey. Eric gave me the freedom to explore ambitious research directions, while also teaching me how to identify important problems, think clearly, and conduct rigorous research. His mentorship has profoundly shaped my growth as an independent researcher, and I could not have asked for a better advisor.

I am also sincerely grateful to my dissertation committee members, Prof. Yi Zhang and Prof. Cihang Xie. I truly appreciate the generous time and effort they invested in evaluating my work and serving on my committee. Their insightful feedback and constructive suggestions have significantly improved the quality of this dissertation.

My journey into robotics and embodied AI began at the University of Michigan, Ann Arbor. I owe a great debt of gratitude to my master's advisor, Prof. Odest Chadwicke Jenkins, for introducing me to this fascinating field and providing me with my earliest research opportunities. His guidance opened the door to a research path that has continued to inspire me throughout my Ph.D.

I have been incredibly fortunate to work alongside a group of brilliant and supportive labmates and collaborators. I would like to extend my heartfelt thanks to Xuehai He, Yue Fan, Jing Gu, Kaiwen Zhou, and all the members of the ERIC Lab. The late-night discussions, collaborative writing, and shared moments of joy throughout this journey are among my most cherished memories.

Furthermore, I am deeply thankful to my mentors and colleagues during my research internships at Microsoft, Samsung Research America, Adobe Research, ByteDance, and Utopai Studios. These industry experiences broadened my horizons and offered

perspectives that greatly enriched my academic research.

Finally, none of this would have been possible without the unconditional love, continuous encouragement, and unwavering belief of my family. To my parents, thank you for your endless sacrifices and for always supporting my dreams, no matter how far from home they took me. This dissertation would not have been possible without you.

Chapter 1.

Introduction

The past few years have produced a remarkable collection of pretrained foundation models. Large language models [1], multimodal large language models [2–4], text-to-image diffusion models [5], image-to-3D object generators [6–8], video diffusion models, and monocular depth estimators each encode substantial knowledge about language, perception, and the visual world. In parallel, the embodied AI community has begun training vision-language-action models directly on large-scale robot data. Yet generalist embodied competence spans more than any single pretrained model is designed to cover: interpreting human intent, reasoning about physical and semantic constraints, generating coherent multimodal content, and constructing new 3D worlds. Training a new monolithic embodied system for each capability is expensive, data-hungry, and rarely transfers across modalities and domains.

This dissertation studies how to build such embodied systems by composing pretrained foundation models, rather than training new large-scale embodied systems from scratch. The motivating observation is that embodied competence is multi-faceted, and for each facet the field has already produced a strong pretrained model. Vision-language-action

models complement these by providing low-level control, but they typically cover only one slice of embodied competence and require expensive embodied data to extend. Across the settings studied in this dissertation, the question is therefore not which embodied model to train next, but how to compose general pretrained models that already exist. The central question is:

> How can we compose pretrained foundation models, through the right interfaces, into embodied agents that ground language in action, generate coherent multimodal content, and construct editable 3D worlds from limited input?

The contributions develop four families of such interfaces. *Symbolic interfaces* (JARVIS) bind pretrained perception and language outputs through task-level and action-level reasoning. *Learnable token interfaces* (MiniGPT-5) project language hidden states into the conditioning space of a pretrained image generator while keeping both backbones frozen. *Spatial interfaces* (SceneFuse-3D) compose an object-level 3D generator into a scene generator through region decomposition and pretrained spatial priors. *Iterative interfaces* (EvoScene) close a loop between pretrained 3D and video generators so that geometry and appearance refine each other. Across these four interfaces, embodied capability can be unlocked from foundation models through deliberately designed composition, without retraining the backbones.

§ 1.1. Why Compose Pretrained Models?

The pretrained models listed above represent an unprecedented amount of knowledge about how the visual world looks, how language describes it, and how 3D structure

can be inferred from images [1–6]. For embodied AI, this is both an opportunity and a challenge.

The opportunity is that many sub-skills required for embodied competence are already implicit in pretrained models. A pretrained vision-language model can recognize objects and parse instructions. A pretrained image-to-3D generator can synthesize plausible geometry from a single view. A pretrained video diffusion model can imagine consistent appearance across viewpoints. If these capabilities can be combined, the cost of building embodied agents drops substantially compared to training new embodied models from scratch on robot-specific data, which remains scarce and expensive to collect at the scale modern pretraining demands.

The challenge is that pretrained models are not designed to be composed. Their outputs are not aligned with each other. Their training distributions differ. Object-centric 3D generators do not scale to scenes; image diffusion models do not maintain 3D consistency; language models do not natively emit conditioning features for diffusion. Composing them therefore requires explicit interface design: deciding what information flows between models, what representations they share, and what is trained on top to glue them together. The chapters of this dissertation can be read as a sequence of increasingly lightweight answers to this composition problem, moving from symbolic glue, to learnable token interfaces, to spatial priors, and finally to iterative 2D–3D feedback loops.

A second message is that composition exposes specific gaps in current pretrained models. Compositional manipulation benchmarks reveal that pretrained vision-language models still fail to ground language-specified object variants in physically constrained motion. Interleaved generation benchmarks show that connecting a multimodal language

model to a diffusion model via textual captions loses long-range context. Scene-level 3D generation shows that object-centric generators collapse without spatial decomposition. Each finding sharpens our understanding of which capabilities pretrained models already provide, and which capabilities the interface must supply.

§ 1.2. Challenges

Composing pretrained vision-language models for embodied manipulation. Language-guided manipulation requires an agent to map natural language instructions to physically executable trajectories. Existing manipulation benchmarks often contain fixed task templates, limited language variation, or task demonstrations that are difficult to scale to new objects and task compositions [9, 10]. As a result, it is hard to evaluate whether pretrained vision-language components actually ground compositional instructions in physically meaningful motion. A useful benchmark must scale across objects, task types, and language expressions while preserving physically realistic interactions, so that it can serve as a diagnostic for pretrained vision-language models in compositional manipulation.

Long-horizon reasoning over the outputs of pretrained perception and language models. Dialog-based embodied tasks require agents to extract goals from free-form human communication, build an internal representation of the environment from egocentric observations, and plan over long action sequences. End-to-end neural policies often struggle in this setting because the action space is large, training data is limited, and many failures arise from missing commonsense constraints rather than from low-level perception alone [11]. The challenge is therefore not to replace pretrained perception

and language modules, but to design an interface that turns their noisy outputs into a structured representation amenable to reliable reasoning.

Bridging pretrained language and image generators for interleaved generation. Large language models are strong at text generation, and diffusion models are strong at image generation, but combining them into a system that can generate interleaved text and images remains challenging. Naively chaining a comprehension model with a text-to-image model through a textual caption compresses dialogue context into a short prompt and loses long-range coherence. Full-model retraining is expensive and discards the value of the pretrained backbones. The challenge is to design a narrow, learnable interface that lets a pretrained language model directly drive a pretrained diffusion model without retraining either one.

Composing pretrained 3D and video generators for scene-level construction. Generating 3D worlds from a single image requires recovering information that is not directly observed. Neural rendering methods such as NeRF and 3D Gaussian Splatting can represent photorealistic appearance when sufficient views are available [12, 13], but single-image scene generation must infer geometry, layout, occluded content, and texture consistency from sparse evidence. Object-centric image-to-3D generators can produce impressive isolated assets, yet they degrade when applied to complex scenes because of resolution limits, layout misalignment, and training distribution shift [6–8]. Video diffusion models can imagine plausible appearance across viewpoints, but they lack explicit 3D consistency. The challenge is to design spatial and iterative interfaces that let object-level 3D priors, video priors, and geometric priors cooperate, without collecting scene-level 3D training data.

§ 1.3. Thesis Overview

This dissertation is organized around three parts, each developing a different family of composition strategies on top of pretrained foundation models.

Part I: Embodied Reasoning and Interaction. The first part studies how pretrained perception and language models can be composed into embodied reasoning systems. The VLMbench chapter presents a compositional benchmark for vision-and-language manipulation. By organizing manipulation tasks according to motion constraints and using object-centric representations to generate scalable, language-grounded demonstrations, VLMbench provides a diagnostic that exposes where pretrained vision-language components fail to ground compositional instructions in physically constrained motion.

The JARVIS chapter then presents a neuro-symbolic framework for conversational embodied agents that addresses these failures by symbolic composition. JARVIS uses a pretrained language model to extract sub-goals from free-form dialog and pretrained perception modules to maintain a semantic map of the environment. A symbolic reasoning module then applies task-level and action-level constraints over these structured outputs to produce executable actions. The chapter argues that pretrained models are most useful as sensors and planners when their outputs are bound through an explicit symbolic interface.

Part II: Interleaved Multimodal Generation. The second part introduces a much lighter interface between pretrained models. The MiniGPT-5 chapter presents a framework for interleaved vision-and-language generation built on top of a pretrained multimodal large language model and a pretrained diffusion model. The key idea is the introduction of *generative vokens*, a small set of learnable tokens that live in the language model's

output space and project into the diffusion model's conditional space. Combined with a description-free two-stage training scheme, this lets a comprehension-oriented multimodal model emit interleaved text-and-image sequences while keeping both backbones frozen. The chapter argues that generation capability can be added on top of pretrained components through a narrow, well-supervised token interface rather than through full-model retraining.

Part III: Generative 3D World Construction. The third part extends the composition idea to 3D, where multiple pretrained models must cooperate without any scene-level training data. The SceneFuse-3D chapter presents a training-free framework that composes a pretrained image-to-3D object generator into a scene generator. The scene is decomposed into overlapping regions, each region is delegated to the pretrained object generator, and a masked rectified flow procedure glues the regions into a globally coherent scene under pretrained depth and detection priors. This chapter demonstrates that object-level 3D priors can be reused for scene-level generation through a spatial interface.

The EvoScene chapter presents a self-evolving framework that closes a loop between pretrained 3D and video generators. Rather than generating a scene in one pass, EvoScene cycles between spatial prior initialization, visual-guided 3D mesh generation, and spatial-guided novel view synthesis. The mesh provides depth-based guidance to a pretrained video diffusion model, whose synthesized views become new geometric priors for the next iteration. Together with SceneFuse-3D, this chapter shows that high-fidelity 3D scene construction can be built by composing 2D and 3D pretrained priors under spatial and iterative interfaces.

§ 1.4. Contributions

The main contributions of this dissertation are as follows:

- **A composition-based methodology for embodied AI.** I argue that capable embodied agents can be built by composing pretrained foundation models through deliberately designed interfaces, rather than by training large monolithic embodied systems. The chapters develop four families of such interfaces: symbolic, learnable token, spatial, and iterative.

- **A compositional benchmark for embodied manipulation.** I introduce VLM-bench, a scalable benchmark for vision-and-language manipulation that organizes tasks by motion constraints, supports compositional language, and exposes where pretrained vision-language components fail to ground compositional instructions in physically constrained motion.

- **Symbolic composition of pretrained perception and language models.** I develop JARVIS, a modular neuro-symbolic framework that treats pretrained language and vision modules as sensors and binds their outputs through task-level and action-level symbolic reasoning, enabling robust dialog-based long-horizon embodied task completion.

- **Learnable-token composition of pretrained language and diffusion models.** I propose MiniGPT-5, which introduces generative vokens as a small, learnable interface that projects multimodal language model hidden states directly into the conditioning space of a pretrained diffusion model, enabling description-free interleaved text-and-image generation with both backbones frozen.

- **Spatial composition of pretrained 3D priors for scene construction.** I present SceneFuse-3D, a training-free framework that composes a pretrained image-to-3D object generator into a scene generator through overlapping region decomposition, pretrained depth and detection priors, and spatial-aware completion under masked rectified flow.

- **Iterative composition of pretrained 3D and video generators.** I introduce EvoScene, a self-evolving framework that couples pretrained 3D and video generators in a 2D–3D feedback loop, where depth-conditioned video synthesis provides novel views that progressively refine the underlying mesh.

Together, these contributions outline how the foundation models that the field is producing can be reused, rather than rebuilt, as the substrate for embodied intelligence. The systems studied here gain interpretability, generalization, and data efficiency precisely because they decline to collapse perception, generation, and 3D construction into a single monolithic model, and instead design explicit interfaces for the pretrained models that already encode much of the needed knowledge.

Part I.

Embodied Reasoning and Interaction

Embodied agents are evaluated not by what they recognize but by what they can do in response to language. The two chapters in this part approach this problem from the perspective of *composing pretrained models*. VLMbench (Chapter 2) builds a diagnostic that exposes where pretrained vision-language models, when applied directly, fail to ground compositional instructions in physically constrained motion: it organizes manipulation tasks around motion constraints rather than fixed task templates, and uses object-centric representations to scale language-grounded demonstrations across object types and task compositions. JARVIS (Chapter 3) then asks how a pretrained language model and pretrained perception modules can be combined into an embodied agent that operates under free-form dialog, long horizons, and partial observability, by treating them as sensors and binding their outputs through symbolic task- and action-level reasoning.

Together these chapters argue that the gap between pretrained vision-language understanding and embodied competence is closed by structural composition. Compositional benchmarks expose where direct use of pretrained models collapses under language and physical variation; neuro-symbolic execution shows how explicit symbolic interfaces over pretrained perception and language modules recover what end-to-end approaches cannot. The interfaces developed here form the first family of pretrained-model composition explored in this dissertation, on which the lighter token interfaces of Part II and the spatial and iterative interfaces of Part III subsequently build.

Chapter 2.

VLMbench: A Compositional Benchmark for Vision-and-Language Manipulation

§ 2.1. Introduction

“Can you help me to clean the disks in the sink?" — humans communicate with each other using language to issue tasks and specify the requirements. Although recent progress in embodied AI pushes intelligent robotic systems to reality closer than at any other time before, it is still an open question how the agent learns to manipulate objects following language instructions. Therefore, we introduce the Vision-and-Language Manipulation (VLM) task, where the agent must follow language instructions to do robotic manipulation. Recent benchmarks were developed to evaluate robotic manipulation tasks with language guidance and visual input [9, 14, 15]. However, the collected task demonstrations are not modular and can hardly scale because they lack (1) adaptation to novel objects and (2) categorization for modular and flexible composition to complex tasks. Additionally, the lack of variations in language also leads to biases in visual reasoning learning. To deal with these problems, we expect an inclusive, modular, and scalable benchmark to

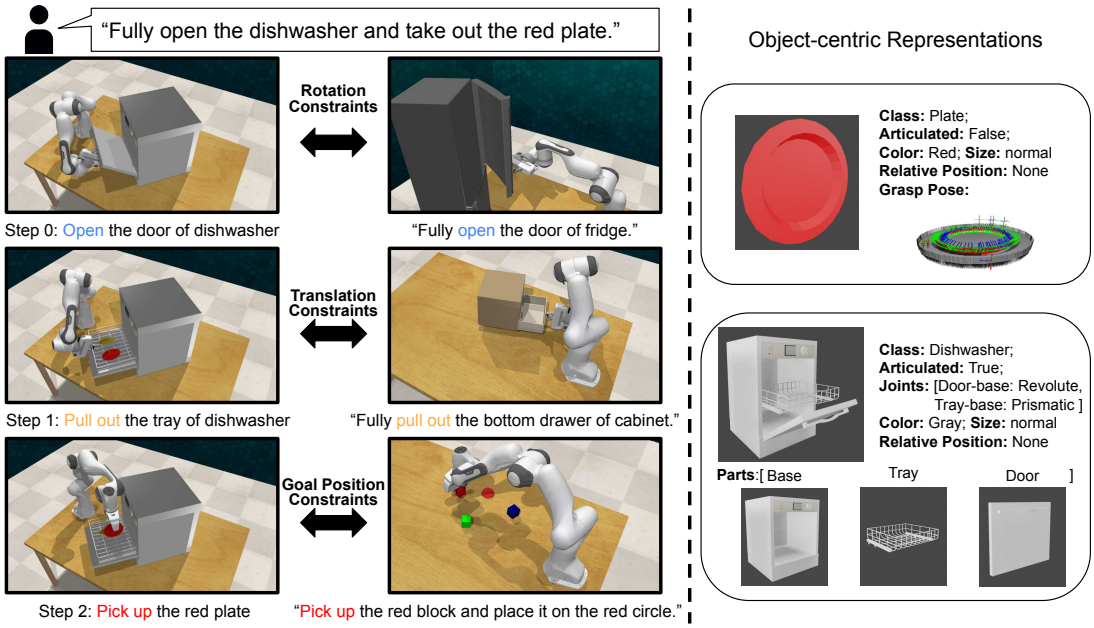


Figure 2.1.: Given the language instructions and observations, the VLMbench requires the agent to generate an executable manipulation trajectory for specific task goals. On the left, we show that the complex tasks can be divided into unit tasks according to the constraints of the end-effector, like "Open the door of the dishwasher" and "Open the door of the fridge" should both follow the rotation constraints of the revolute joint. On the right, we show examples of object-centric representations, where all graspable objects or parts generate local grasping poses as their attributes. Depending on the modular design, we can generate reasonable VLM data automatically.

evaluate embodied agents for various language-guided manipulation tasks.

An ideal VLM benchmark should have at least three characteristics: The first one is scalability. Such a benchmark should automatically generate various physics-realistic 6 degrees of freedom (DoF) interactions with affordable objects and expand new tasks effortlessly. The second is task categorization, which exploits commonality concerning robot motion between semantic tasks and is almost ignored in existing works. The third is reasonable language generation, which requires the benchmark to generate language instructions for testing diverse visual reasoning abilities without biases. However, existing

benchmarks [9, 14–16] lack at least one characteristic for VLM tasks. Motivated by these attributes, we present VLMbench, a highly categorical robotic manipulation benchmark with compositional language for visual reasoning. To build and scale VLMbench, we propose AMSolver, an automatic unit task builder that can compose unit tasks to create complex multi-step tasks and seamlessly adapt to novel objects. Compared to previous benchmarks, VLMbench categorizes manipulation tasks into various meta-manipulation actions according to the constraints of robot trajectories for the first time. Meanwhile, the combinations of compositional language templates and object-centric representations provide numerous variations for visual reasoning in VLMbench, as shown in Figure 2.1.

To investigate the difficulty of the benchmark, we test them with several partially modal methods and a keypoint-based method, 6D-CLIPort, modified from the state-of-the-art language-guided manipulation method CLIPort [10]. The results show that there is still a massive room for improvement in the robust manipulation action generations and accurate language-guided visual understanding. To sum up, our contributions to this work include the following:

- AMSolver, an automatic demonstration generator for various task semantics, motion constraints, object types, and states defined in a novel task template formulation.
- VLMbench, a robot manipulation benchmark on 3D tasks with visual observation and compositional language instructions, where we categorize the manipulation tasks by constraints and provide variations with minimal biases in the first time.
- 6D-CLIPort, a general vision-and-language manipulation baseline model evaluated on all kinds of VLMbench tasks.

§ 2.2. Related Work

Robotic Manipulation Benchmarks There are plenty of benchmarks proposed related to visual-language robotic tasks. ALFRED [17] was proposed to do virtual object rearrangement tasks guided by visual observation and language instruction between different room-scale locations. ManipulaTHOR [18] introduced realistic interaction with objects using a 6 DoF configurable mobile manipulator. Habitat 2.0 [16] and BEHAVIOR [19] incorporated real robot navigation with simple object interaction implementation for more comprehensive mobile manipulation tasks. CALVIN [20] collects 24 hours of playing data with natural language instructions for long-horizon manipulation tasks. Regarding static manipulation benchmarks, MetaWorld [15] collects a series of translation-only tasks with demonstrations for reinforcement learning. CausalWorld [14] focused on causal inference within manipulation and showed examples of several simple object rearrangement tasks. RLBench [9] collected 100 different tasks by specifying robot arm end-effector waypoints for each of them for robot learning. Besides, crowd-sourcing platforms collect human demonstrations of a variety of common manipulation tasks through VR/AR devices, such as Robosuite [21] and Robomimic [22]. Compared to these works, our VLMbench includes high-level task descriptions and low-level robot action translations and realizes automatic task builders for easier generation of complex tasks.

Vision and Language Tasks for Embodied AI Vision and language tasks, such as Vision Question Answering (VQA) [23, 24], Video Captioning [25–27], Image-Text Retrieval [28, 29], connect vision and NLP research to produce semantics from combined embedding features. Recent works such as referring object spatial relations [30] and

Table 2.1.: Comparison of existing robotic manipulation benchmarks to VLMbench.

Benchmark	**novel object adaptation**	**automatic trajectory generation**	**variant object property**	**automatic 6-DoF grasping**	**constraint-based task formulation**
CausalWorld [14]	✗	✗	✓	✗	✗
MetaWorld [15]	✗	✓	✗	✗	✗
ManipulaTHOR [18]	✗	✗	✓	✗	✗
Habitat 2.0 [16]	✓	✓	✓	✗	✗
Robomimic [22]	✗	✗	✗	✗	✗
BEHAVIOR [19]	✗	✓	✓	✗	✗
RLBench [9]	✗	✓	✓	✗	✗
CALVIN [20]	✗	✗	✗	✗	✗
VLMbench (ours)	✓	✓	✓	✓	✓

dynamic events [31] provide valuable reasoning results for the agent to act. Vision-and-Language Navigation (VLN) [32–38] proceeds another step to use visual observations and language instructions for robotic navigation directly. Recent benchmarks [16, 17, 19] combined VLN with abstracted object interaction to include more object rearrangement tasks. Compared to the tasks mentioned above, our benchmark is designed for more complex robotic manipulation that simulates physics-realistic 6DoF grasping and requires reasoning ability from the visual-language space to the executable action space.

Language-Instructed Manipulation Various manipulation tasks have been recently researched with language input, either describing the entire task or serving interactive input for task specifications. HULC [39] proposes a hierarchical network for long-horizon manipulation tasks, which contains multi-modal transformers for task planning and a policy network for action generation. Structformer [40] proposes an object selection network from language and visual encodings and a language-conditioned pose generator for semantic object rearrangement. Stepputtis et al. [41] proposed a closed-loop control model for pouring tasks. CLIPort [10] proposed a two-stream framework to learn a spatial attention map for 2D object manipulations. Lynch et al. [42] fed natural language instructions to a goal-conditioned policy pretrained using imitation learning for various

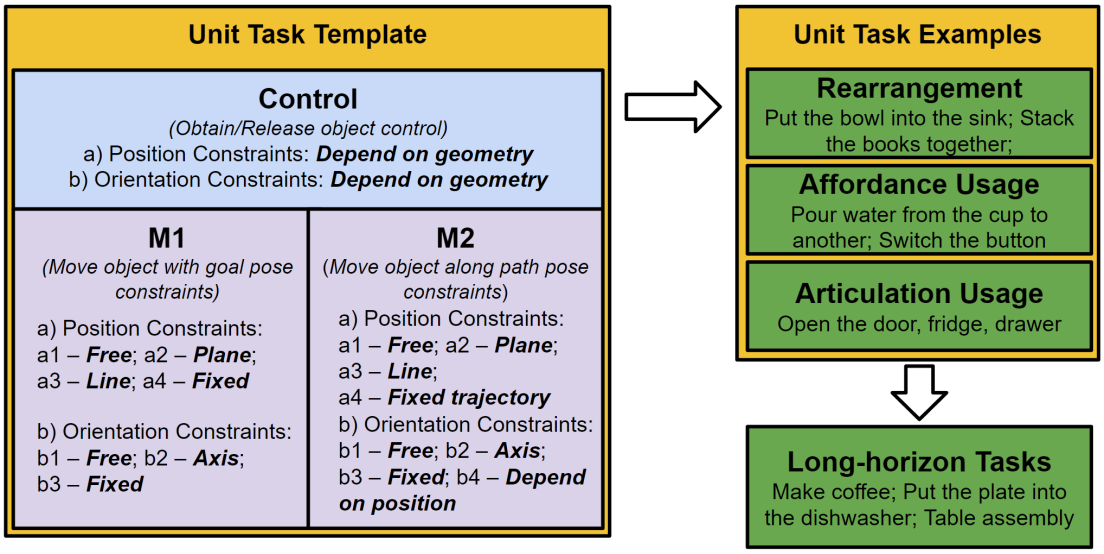


Figure 2.2.: The unit task templates of AMSolver. On the left, we show three unit task templates parameterized by position and orientation constraints over the robot end-effector on either the goal pose or the entire path. By combining these unit task templates, various task examples can be generated. For example, on the right, we show three main common task types household tasks and long-horizon tasks composed of unit tasks.

tasks. INVIGORATE [43] proposed an interactive system that takes language input to correct false estimations and re-plan for object grasping in clutter. Shao et al. [44] trained a joint language-vision model on 7-DoF goal trajectory estimation with another task classifier for multi-task training. Goyal et al. [45] learned zero-shot task adaptation to accomplish novel tasks through differences described in natural language. In this paper, our 6D-CLIPort model learns full 6D grasping and conducts variate-length robot manipulation tasks from language instructions.

§ 2.3. AMSolver: Automatic Manipulation Solver

We consider a rule-based task categorization that supports most daily manipulation tasks, which follows a unified formulation. To this extent, we propose Automatic Manipulation Solver (AMSolver)*. We focus on simple unit tasks and introduce a unit task template that categorizes motion constraints and generates a wide range of household tasks (see examples in Figure 2.2). The formulation treats objects as a representation that could have variations in appearance and enables automatic demonstration generation using off-the-shelf tasks and motion planners.

2.3.1. Rule-based Unit Tasks

Since the tasks have notable variations, we assume that each complex task can be decomposed into combinations of several unit tasks from the aspects of end-effector trajectories. We define a *unit task* as the semantic step of completing a sub-goal of the entire task. Specifically, a unit task is defined in a formula of 'take action on an object under certain constraints' where two constraints can parameterize a unit task: (a) **position constraints** and (b) **orientation constraints**, which describe the valid spatial space or orientation range, respectively, of the end-effector for a specific task. We propose three unit task templates detailed below that can compose the aforementioned complex tasks.

(1) Control is a preparation or ending step of a task, which models the transition of the object state, where the state indicates whether the robot can move the object or not. In this work, we specify one way of transition: to obtain control by grasping it and

*AMSolver is implemented in CoppeliaSim [46] (Free Educational License) and codes are based on RLbench [9] (RLBench Software License) and PyRep [47] (MIT License).

release control by opening the gripper. There are other ways to obtain control, like pushing, hitting, etc. However, these transitions will naturally lead to the constraints in the following sub-tasks, so they cannot be considered a general component for any complex task. In this unit task, the position and orientation constraints depend on the geometry of object instances.

(2) M1 denotes moving the target object with goal pose constraints, which can be modeled as a 6 DoF transform in the robot's workspace. The position constraints define a bounded goal space in $\mathbb{R}^3$, while the orientation constraints define a valid 3D orientation $SO(3)$. We consider four types of position constraints: (a1) *Free*, (a2) *Plane*, (a3) *Line*, and (a4) *Fixed*, which means the goal position is any point inside a 3D space (a1), constrained in a 2D plane (a2), constrained in a line-shape area (a3), or fixed to a certain point (a4). There are three types of orientation constraints: (b1) *Free*, (b2) *Axis*, and (b3) *Fixed*, which means the goal orientation is unlimited in 3D rotation space (b1), only rotated along an axis in space (b2), or fixed at a given orientation (b3). For example, M1[a2, b1] can represent placing the object on a tabletop (with plane position constraint), while M1[a3, b2] can represent moving the object to a position on one line and ending with a constrained orientation of one axis, like dropping a stick into the hole.

(3) M2 denotes moving the target object along a trajectory while satisfying the motion constraints during the entire path, which implies a more strict condition than **M1**. The constraints are mostly from object articulation, like revolute joints on doors, or task-specific requirements, like keeping the opening upward for a full-filled mug. Therefore, there are four kinds of position constraints: (a1) *Free*, (a2) *Plane*, (a3) *Line*, and (a4) *Fixed trajectory*, which means any position inside a space (a1), a plane (a2), a line (a3) or a predefined path (a4) should be feasible for the trajectory, and four kinds of orientation

constraints: (b1) *Free*, (b2) *Axis*, (b3) *Fixed*, and (b4) *Depend on position*, which means every pose in the trajectory should be unlimited orientations (b1), at most rotated along an axis (b2), fixed to a certain orientation (b3), or depended on the corresponding positions. For example, M2[a2, b2] means moving the object inside a 2D plane while maintaining the orientation of one axis, like wiping the table. In contrast, M2[a4, b2] means moving the object along a fixed trajectory with maintaining the orientation of one axis, like using a screwdriver to tighten a screw. It is worth mentioning that M2[a1,b1] will degenerate to M1[a1,b1]. According to our unit task definition, we have covered every feasible 6 DoF pose of the end-effector. Therefore, we have reason to believe that these unit tasks can represent any complex task in the action space.

2.3.2. Object-centric Representation

Some recent works [48, 49] have used object-centric representations for manipulation. Since the properties are defined on objects, these representations can easily cross the variations of environments, agents, and tasks. Our benchmark assumes that objects used in the tasks are rigid and their fundamental properties will not change during the tasks. Therefore, we can parameterize the objects as a set of configurations, including class, color, size, and geometry shape. If the object is articulated, its whole configuration will contain the configuration of each part and the physical constraint of each connection. For example, a door consists of three parts: the door base, plank, and handle, so its configuration will contain each part's configuration and record the positions and ranges of two revolute joints.

2.3.3. Automatic Demonstration Generation

To create a task example, the user could compose a task template from the unit library of formulation, such as a unit task of object rearrangement as **Control**-**M1** or a more complex task of object stacking as **Control**-**M1**-**Control**-**M1**-**Control**-**M1**, etc., and then select the objects from a given set to be included in the scene as the object to be manipulated and distractor objects. The object placement could be randomized to the customer's specifications in the simulation. The corresponding language descriptions could also be generated from templates.

To implement **Control** as grasping, we create an object-wise grasp pose dictionary. Specifically, a point cloud-based grasping pose detection algorithm [50] is implemented to search all feasible grasping poses, given the object's shape and robot gripper parameters. The grasp poses are saved and transformed to world space for a particular task by simply multiplying with the object's pose. To implement **M1** and **M2**, we integrate customized motion constraints in the OMPL motion planner library so that the calculated trajectory satisfies the constraints automatically. Please refer to Appendix A.2 for more details and task examples.

§ 2.4. VLMbench: Visual-and-Language Manipulation Benchmark

2.4.1. Problem Definition

Given language instructions, the Vision-and-Language Manipulation (VLM) task requires an embodied agent to follow the instructions to complete tabletop manipulation tasks.

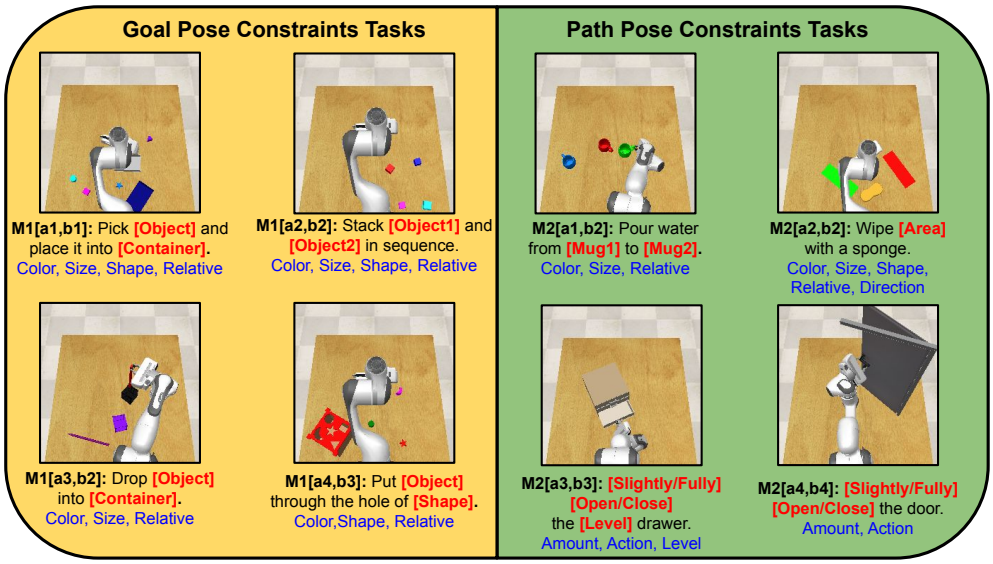


Figure 2.3.: We show four task categories with goal-pose constraints on the left side, and on the right side, we show four task categories with path-pose constraints. For each task category, we visualize the observations of the overhead view and list the main unit task, variations, and instruction templates. The red words with brackets indicate the blank in which the variation descriptions can fill. The blue words indicate the variations in the tasks. The combinations of variations will lead to various instance-level tasks.

Formally, at the beginning of the task, the agent receives a set of language instructions $L = \{L_1, L_2, ..., L_n\}$, where L_i denotes one sentence of arbitrary length. The initial state s_0 contains multi-view RGB images, depth images, segmentation information, and robot states, including joint angles, velocities and torques, and end-effector pose. Given the observations and language instructions, the agent needs to estimate an executable action command a_0, directly working on the end-effector or joints. Then, at each step t, the agent receives new observations o_t and generates the action $a_t = f(s_t|s_0, s_1, ..., s_{t-1}, L)$ for the next step. The step loop will repeat until the agent sends a stop action or should be terminated, e.g., achieve the success conditions or the limitation steps. The agent should obey the constraints provided by language instructions during the run.

2.4.2. Tasks and Dataset

In previous works, the researchers manually designed manipulation tasks by implicit prior knowledge without categories. Instead, we are trying to build tasks from the perspectives of elementary manipulation abilities. In other words, since different tasks have various semantic meanings, we consider the task with the same unit tasks combination should be in the same category from the aspect of the action space. For example, “Open the door of the fridge," and “Open the door of the microwave" require the same action ability except for semantic meanings and grasping poses which depend on the object geometry. Therefore, we define eight general task categories, represented by one typical task in each category, shown in Table 2.2 and Fig. 2.3. We use the definitions in the unit task templates to represent the main constraints of each task category. The task details can be found in Appendix A.1, and dataset statistics can be found in Appendix A.3.

Task Variations The manipulated object’s properties can randomly change for each task category, and every combination leads to a task instance. In the VLMbench, we use eight variations: color, size, relative position, shape, direction, level, amount, and action type. The variations are from two perspectives: object and motion. Object variations include color, size, shape, relative position, and direction. Here, the **color** is chosen from 20 colors in the seen settings. The **size** contains the relative descriptions between two objects, “smaller" and “larger", and descriptions between three objects, which are “large",“medium", and “small". The **shape** contains five types of objects for the seen and unseen settings. The **relative position** describes the spatial relationship between two objects, like the top, front, rear, left, and right. The **direction** contains two descriptions for rectangular prism in a plane:“horizontal" and “vertical". For the objects that have the vertical structure, the **level** includes “top", “middle," and “bottom." From the motion

view, the variations are amount and action type. The **amount** means how far the task needs to be done, consisting of “fully" and “slightly." The **action type** includes “open" and “close", especially for the articulated objects. The table of these variations and models used can be found in Appendix A.1.

Unseen Settings All tasks in the unseen settings are unseen <color, shape> combinations from an unseen color collection and an unseen shape collection (where the shapes include all object classes and variants). The unseen color collection has five new colors that do not appear during training, including brown, gold, pink, chocolate, and coral. As for the unseen shape collection, it has some overlap with the seen library for the tasks with color variations (but the <color, shape> combinations are always unseen), and is exclusive for the tasks without color variations (e.g., we introduce a new door model with a rotatable handle for the unseen setting of Door tasks). So there are mainly three kinds of unseen combinations: <unseen color,unseen shape>, <unseen shape>, and <unseen color,seen shape>. The exact object models used for each task can be found in Appendix A.1.

§ 2.5. Vision-and-Language Manipulation Agent

To provide a baseline method to solve VLM tasks, we propose a neural-network-based agent 6D-CLIPort that takes in the input of multi-view RGB-D observations and task language descriptions and outputs 6 DoF pose keypoints along the path that can accomplish a specific task. For instance, for the pick-and-place task, the agent will iterate twice and output the object’s 6D pose for pick, and place, separately.

The overall flow of 6D-CLIPort* is shown in Figure 2.4. The input data passes

*6D-CLIPort is implemented based on CLIPort [10] (Apache-2.0 License)

Table 2.2.: The table contains the category-level tasks in our dataset, with their main constraints, variations and instruction samples.

Task Categories	Main Constraints	Variations	Instructions Samples
Pick & Place objects	M1 – Position: Free, Orientation: Free	Color, Size, Shape, Relative Position	"Pick the red cube and place it into the green container."
Stack objects	M1 – Position: Plane, Orientation: Axis	Color, Size, Shape, Relative Position	"Stack the small star and the medium star in sequence."
Drop pencil	M1 – Position: Line, Orientation: Axis	Color, Size, Relative Position	"Drop the left pencil into the right container."
Put into shape sorter	M1 – Position: Fixed, Orientation: Fixed	Color, Shape, Relative Position	"Put the triangular prism through the hole of triangle."
Pour water	M2 – Position: Free, Orientation: Axis	Color, Size, Relative Position	"Pour the water from the green mug to the red mug."
Wipe table	M2 – Position: Plane, Orientation: Axis	Color, Size, Directions, Relative Position	"Wipe the horizontial area with a sponge."
Use drawer	M2 – Position: Line, Orientation: Depend on position	Amount, Action Type, Level	"Fully close the top drawer."
Use door	M2 – Position: Fixed trajectory, Orientation: Depend on position	Amount, Action Type	"Slightly open the door."

through visual and language encoders. The embedded features are fused together to obtain a pixel-wise feature map that shows the probability of the object of manipulation interest. The encoding module is reused in 3 models (Attention, Value, and Key Module) as follows: The Attention module takes the maximum of the feature map to get 2D image sample crops. The Key module takes the input of the rotated crops and output feature map crops. Then, the feature map crops are used as convolution kernels and pass through the feature map of the entire image from the Value module to get a 3D pose heatmap (x-y 2D position and yaw rotation, and their estimations are obtained by maximizing the probability). Three fully connected neural network modules will regress one of the remaining three dimensions each (z position and roll, pitch rotation) from this heatmap. Finally, a full 6 DoF pose is composed. Compared to the state-of-the-art method CLIPort [10], we resolved its two constraints: 1. increase 3 DoF only motions to full 6 DoF; 2. output an arbitrary number of poses than the fixed two outputs for pick

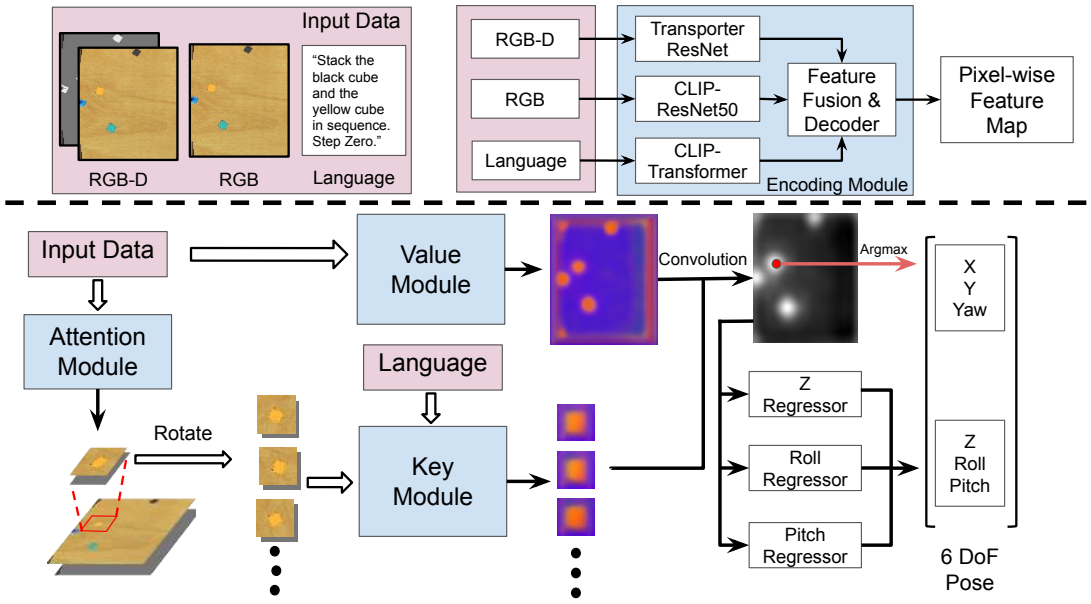


Figure 2.4.: The structure of 6D-CLIPort. Please refer to Sec. 2.5 for details.

and place actions. Besides, we introduce some details below.

Multi-view Vision Fusion To get a better RGB-D input, we first fuse RGB-D input from several cameras with known poses into a 3D colored point cloud, and then project it along the vertical direction facing towards the table plane to get a top-view RGB-D image.

Encoding Module The agent has two feature extraction streams: semantic and spatial streams. The agent uses the pretrained CLIP's ResNet50 and Transformer model [51] to encode the RGB and languages in the semantic stream. An untrained Transporter ResNet [52], which has 43 layers and 8 strides, is used in the spatial stream to encode the RGB-D image. Then, the decoder fuses these latent space features by concatenation, fully convolution, and up-sampling layers and predicts a dense pixel-wise feature map. More details can be found in [10].

Implementation Details We separately train the agent for each task category. For

example, the agent for pick and place tasks will jointly train on the data of all variations mentioned in Table 2.2. In details, since the VLMbench is built by the unit task templates, the input demonstrations $D = \{L, \zeta_1, \zeta_2, ..., \zeta_i\}$ can be divided into different sub demonstrations by the waypoints generated from unit task templates, where each step $\zeta_i = (L_i, o_i, g_i)$ consists of language instruction L_i, observation o_i, and the 6 DoF sub-goal waypoint pose g_i for the current step. For step i, we use the observations in the first frame of this step as o_i and prompt high-level instruction L with "Step i" as L_i. The input RGB-D image has a resolution of 160×128. The feature heatmap is dimension 16 in x-y 2D plane, and the crops from the Attention Module are rotated 36 times before feeding into the Key Module.

§ 2.6. Experiments

2.6.1. Experimental Setup

Evaluation Settings Before testing each baseline, we preprocess the tasks to help the agent eliminate trivial steps. 1) we divided tasks into the sub-goal sequences by the ground truth waypoints generated by AMSolver, so that the agent only needs to estimate the actions of the predefined unit task sequence for each task. 2) Since we have the ground truth gripper state for each sub-goal, we also provide whether the gripper should open or close for each action estimation. 3) To increase the grasping stability, we use the pre-generated grasping pose instead of estimations if these two poses are close enough (distance is less than 5 cm, and the rotation is less than 10 degrees). 4) To reduce the failure grasping cases due to the motion planning, we use a predefined pre-grasping offset, which is 8cm backward along the z-axis of the target grasping pose, and a post-grasping

offset, which is 8cm upward along the z-axis of the world frame.

In each episode of one task variation, the simulator imports a test configuration from the pre-collected test dataset, which includes the initialization poses of objects, the success conditions of the task, and a set of waypoints that can finish the task for reference. The agent should solve the task in the online simulator within a limited number of steps. The success rate is used as the primary evaluation metric, calculated by dividing the number of success conditions satisfied by the number of tests. We use the average success rate of all variations for each task category. The success conditions are mainly determined by an object or joint detector. The object detector returns true if particular objects have moved inside the predefined space, and the joint detector returns true when the joint angle reaches the predefined range. The success conditions of each task can be found in Appendix A.1.

Baselines In addition to the 6D-CLIPort model, we provide two kinds of baseline models for comparison, one with partial input modalities and the other with partial ground-truth predictions.

To test the influence of different input modalities, we train two other agents with partial modalities: a *Language-Only* agent and a *Vision-Only* agent. These agents use the same model architecture as 6D-CLIPort but have different input modalities. The Language-Only agent uses the CLIP transformer for language encoding, and the visual inputs are all zeros. The Vision-Only agent uses the same RGBD inputs as in the 6D-CLIPort agent, but its language input will only include the prompt for steps indication like "Step zero" without any high-level language instructions.

To measure the capabilities and limitations of 6D-CLIPort, we individually test its position or orientation estimation abilities by giving the ground-truth values of the other.

Table 2.3.: Success rates of all task categories, including both seen and unseen settings.

	Pick&Place		**Stack**		**Drop**		**Shape Sorter**	
Agent	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen
Language-Only	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Vision-Only	6.31	9.85	6.89	1.79	0.00	0.00	0.00	0.33
6D-CLIPort	**28.28**	**27.53**	**22.19**	**18.37**	**6.42**	**6.42**	**17.33**	**12.33**
6D-CLIPort (GT Ori)	28.03	26.26	26.53	26.02	17.91	16.22	24.00	15.67
6D-CLIPort (GT Pos)	83.84	75.25	58.93	50.51	16.89	11.82	18.00	17.33
	Pour		**Wipe**		**Door**		**Drawer**	
	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen
Language-Only	0.00	0.00	0.00	0.00	0.00	0.00	4.17	1.04
Vision-Only	0.00	0.00	19.80	20.80	0.00	0.00	14.58	7.29
6D-CLIPort	**1.00**	**1.00**	**22.40**	**21.00**	**6.00**	**5.00**	**22.92**	**15.63**
6D-CLIPort (GT Ori)	3.67	3.67	25.80	25.20	6.00	5.00	23.96	17.71
6D-CLIPort (GT Pos)	0.33	0.67	60.20	53.40	27.00	27.00	43.75	52.08

Although the trajectories of finishing the tasks are various, and we cannot obtain the optimal position and rotation information, we can regard the poses of waypoints as sub-optimal solutions. *GT Pos* means given ground truth *x/y/z* positions while the other three parameters for 3D orientation are estimated, and *GT Ori* suggests the contrary (known orientation, using estimated 3D position).

2.6.2. Result Analysis

Main Results on Different Task Categories and Variations The main results on different task categories are shown in Table 2.3. We observe that 6D-CLIPort performs better on the tasks that have lower rotation variances, including “Pick&Place," “Stack,"“Shape Sorter," “Wipe," and "Drawer." It indicates that 6D-CLIPort can better estimate the positions than orientations in the 3D spaces. Moreover, 6D-CLIPort performs poorly on the “Pour" tasks since the task needs to adjust the pouring poses following the

Table 2.4.: Success rates of all variations, including both seen and unseen settings.

	Color		**Shape**		**Size**		**Relative Position**	
Agent	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen
Language-Only	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Vision-Only	6.00	5.00	6.00	8.50	6.05	5.10	6.80	5.10
6D-CLIPort	**15.17**	**13.00**	**23.00**	**19.50**	**18.35**	**14.92**	**15.31**	**15.82**
6D-CLIPort (GT Ori)	18.67	17.67	28.00	27.25	20.97	17.74	21.43	18.37
6D-CLIPort (GT Pos)	40.17	35.00	56.25	51.25	44.96	38.51	38.61	34.86
	Direction		**Level**		**Action Type**		**Amount**	
	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen
Language-Only	0.00	0.00	4.17	1.04	2.04	0.51	2.04	0.51
Vision-Only	21.00	24.00	14.58	7.29	7.14	3.57	7.14	3.57
6D-CLIPort	**22.00**	**26.00**	**22.92**	**15.63**	**14.29**	**10.20**	**14.29**	**10.20**
6D-CLIPort (GT Ori)	26.00	27.00	23.96	17.71	14.80	11.22	14.80	11.22
6D-CLIPort (GT Pos)	53.00	41.00	43.75	52.08	35.20	39.29	35.20	39.29

grasping pose, which introduces additional difficulties. Moreover, although the success rates in the unseen settings are generally lower than those in the seen settings, the performance drop is reasonable and not dramatic, showing that 6D-CLIPort can transfer the learned manipulation knowledge from seen objects to unseen objects, benefiting from the powerful transfer ability of the pre-trained CLIP model [51].

We also show the results from the perspective of variations across task categories in Table 2.4. The results show that the agent is more sensitive to the novel compositions and thus has a larger seen-unseen performance drop on variations such as "Shape", "Level", "Action Type", and "Amount".

Impact of Different Input Modalities Table 2.3 also compares 6D-CLIPort with Language-Only and Vision-Only agents, demonstrating the impact of different input modalities. (1) The baseline vision-and-language manipulation model 6D-CLIPort performs the best on all tasks, showing the importance of both visual observations and language guidance in VLMbench tasks. (2) The Language-Only agent nearly fails at all

the tasks as it is visually blind and thus unable to localize the objects in the 3D space. For “Drawer" tasks, since language instructions can provide the level information (e.g., top, middle, and bottom) and action directions (e.g., open and close), the Language-Only agent has some chance to close the drawer by collisions. (3) Without language guidance, the Vision-Only agent has a significant performance degradation on all tasks. It fails completely on tasks that require more strict pose constraints, including “Drop”, “Shape Sorter”, “Pour”, and “Door”. For other tasks such as “Pick&Place", “Stack," “Wipe," and “Drawer”, the Vision-Only agent can succeed in few cases (though with pretty low success rates) by randomly grasping an object in the scene for manipulation because those tasks have a lower variance of the grasping actions and following movements.

Ablation Study on Position and Orientation Estimation We provide partial ground-truth predictions and do a unit test of the agent’s position and orientation estimation abilities. The results are shown in Table 2.3 and Table 2.4. From the results, we can see the position estimation ability significantly limits the performance of 6D-CLIPort in those tasks which need correct object localization, such as “Pick & Place”, “Stack”, and “Shape Sorter” tasks. One primary reason why the GT position brings more improvement is that it eliminates the difficulties of localizing the target object, one of the main challenges in compositional reasoning. For example, the instruction “place the red cube into the green container” requires the model to localize the correct cube and container and providing GT position makes the task much easier. Besides, from the perspective of pose variances, when we divide the task into steps, the orientation of each step has fewer variances than the position in many tasks, such as picking, stacking, and wiping. Furthermore, for the tasks requiring the cooperation of position and orientation, such as “Pour” and “Door” tasks, we observe that giving partially ground truth may still not guarantee better task

completion results. More results and analysis can be found in Appendix A.4.

§ 2.7. Conclusion

Vision-and-Language Manipulation (VLM) tasks are essential since they are inevitable for embodied AI. For further research in this area, we propose VLMbench, which includes various VLM tasks, and AMSolver, used for automatic VLM task generation. In addition, we test the 6D-CLIPort agent, a keypoint-based 6 DoF agent, on the benchmark. The results show that the current models can finish VLM tasks, but it is still a new area needed to be explored. We hope the VLMbench can push the research to find general language-guided manipulation agents.

Chapter 3.

JARVIS: A Neuro-Symbolic Commonsense Reasoning Framework for Conversational Embodied Agents

§ 3.1. Introduction

A long-term goal of embodied AI research is to build an intelligent agent capable of communicating with humans in natural language, perceiving the environment, and completing real-life tasks. Such an agent can autonomously execute tasks such as household chores, or follow a human commander to work in dangerous environments. Figure 3.1 demonstrates an example of dialog-based embodied tasks: the agent communicates with the human commander and completes a complicated task "making a sandwich", which requires reasoning about dialog and visual environment, and procedural planning of a series of sub-goals.

Although end-to-end deep learning models have extensively shown their effectiveness in various tasks such as image recognition [53, 54] and natural language understanding

and generation [55, 56], they achieved little success on dialog-based embodied navigation and task completion with high task complexity and scarce training data due to an enormous action space [11]. In particular, they often fail to reason about the entailed logistics when connecting natural language guidance with visual observations, and plan efficiently in the huge action space, leading to ill-advised behaviors under unseen environments. Conventionally, symbolic systems equipped with commonsense knowledge are more conducive to emulating humanlike decision-makings that are more credible and interpretable. Both connectionism and symbolism have their advantages, and connecting both worlds would cultivate the development of conversational embodied agents.

To this end, we propose JARVIS, a neuro-symbolic commonsense reasoning framework towards modular, generalizable, and interpretable embodied agents that can execute dialog-based embodied tasks such as household chores. First, to understand free-form dialogs, a large language model (LLM) is applied to extract task-relevant information from human guidance and produce actionable sub-goals for completing the task (e.g., the sub-goals 1.1, 1.2,...,3.6 in Figure 3.1). During the process, a semantic world representation of the house environment and object states is actively built from raw visual observations as the agent walks in the house. Given the initial sub-goal sequence and the semantic world representation being built, we design a symbolic reasoning module to generate executable actions based on task-level and action-level common sense.

We evaluate our JARVIS framework on three different levels of dialog-based embodied task execution on the TEACh dataset [11], including Execution from Dialogue History (EDH), Trajectory from Dialogue (TfD), and Two-Agent Task Completion (TATC). Our framework achieves state-of-the-art (SOTA) results across all three settings. In a

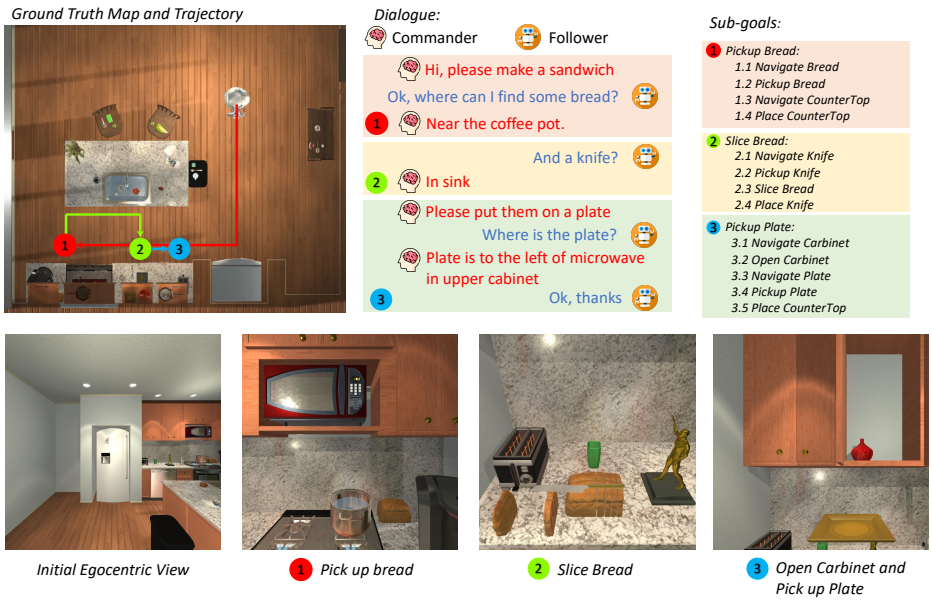


Figure 3.1.: **Dialogue-based embodied navigation and task completion.** The Commander (often a human) issues a task such as making a sandwich, and the Follower agent completes the task while communicating with the Commander. Unlike the agent in fine-grained instruction following tasks, the Follower agent needs to extract sub-goals from the free-form dialogue and execute actions in the visual environment. Note that the Follower agent can only navigate and interact with objects in an egocentric view and has no access to the map or other oracle information.

more realistic few-shot setting where available expert demonstrations are limited for training, we show our framework can learn and adapt well to unseen environments. Meanwhile, we also systematically analyze the modular structure of JARVIS in a variety of comparative studies. Our contributions are as follows:

- We propose a neuro-symbolic commonsense reasoning framework blending the two worlds of connectionism and symbolism for conversational embodied agents. The neural modules convert dialog and visual observations into symbolic information, and the symbolic modules integrate sub-goals and semantic maps into actions based on task-level and action-level common sense.

- Our framework is modular and can be adapted to different levels of conversational embodied tasks, including EDH, TfD, and TATC. It consistently achieves the SOTA performance across all three tasks, with great generalization ability in new environments.
- We systematically study the essential factors that affect the task performance, and demonstrate that our framework can be generalized to few-shot learning scenarios when available training instances are limited.

§ 3.2. Related Work

Embodied AI Tasks Embodied agents capable of navigation and interaction have been long studied in AI, with early work focusing on goal-directed navigation in indoor and outdoor environments [57–59]. More recently, research has shifted toward language-grounded agents. Vision-and-Language Navigation [60–67] explores how agents follow natural language instructions to reach target locations, while Vision-and-Dialog Navigation [68–71] incorporates real-time dialogue for guidance. Beyond navigation, recent efforts [72–74] introduce interactive task completion involving both navigation and object manipulation. Dialog-based embodied tasks [11, 75] further align with real-world settings, enabling agents to collaborate with humans through conversation. Our work builds on this line, aiming to develop a conversational agent for complex household tasks.

Vision-and-Language Navigation and Task Completion Prior work in embodied AI has explored vision-and-language navigation [67, 76–81], vision-and-dialog navigation [82–84], and task-oriented object navigation [85]. Vision-and-language task

completion has also gained attention [86–89], with early methods like [72] using LSTMs for multi-modal fusion, and [86] introducing transformers for improved temporal modeling. Modular approaches further decompose tasks: [88] use semantic voxel grids with hierarchical controllers, and [87] integrate semantic policies for fine-grained object search. However, these systems are designed for structured, instruction-following tasks and often struggle with the ambiguity and complexity of dialog-based scenarios. In contrast, our approach extracts task-relevant information directly from free-form dialogue and performs neuro-symbolic reasoning to generalize across diverse dialog-based embodied tasks.

Neuro-Symbolic Reasoning While neural models have achieved great success across vision and language tasks [53, 54, 90, 91], they often lack interpretability and reasoning ability in complex, multi-modal settings. Neuro-symbolic methods address this by combining neural perception with symbolic reasoning [92–96]. For example, [92] introduce a symbolic program generator for visual question answering, and [93] extend it to video reasoning. In dialog-based household tasks, neural models struggle to integrate diverse inputs and infer correct actions [11]. To address this, we propose a modular neuro-symbolic framework that converts multi-modal observations into symbolic representations and applies commonsense reasoning for robust task execution.

§ 3.3. Neuro-Symbolic Conversational Embodied Agents

3.3.1. Problem Formulation

We study dialogue-based embodied agents, where a *Follower* agent must interpret natural language instructions from a *Commander* (a human or another agent) to complete long-

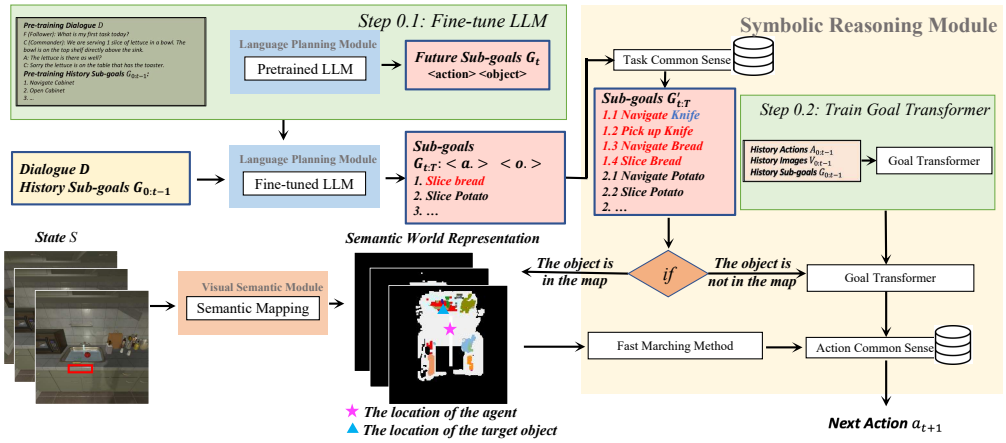


Figure 3.2.: **An overview of our JARVIS framework.** The fine-tuned language planning model takes dialogue and previous sub-goals $G_{0:t-1}$ as input and produces the future sub-goals $G_{t:T}$ (Section 3.3.2). $G_{t:T}$ will be further examined by our Task-Level Common Sense model and converted to more reasonable and detailed future sub-goals $G'_{t:T}$. Meanwhile, the Visual Semantic module actively updates the semantic world representations (Section 3.3.2). If the object is found in the world representation, the next action is determined by the Fast Marching method. If not, the Goal Transformer will generate the next action. The next action $a_{t+1} \in \mathcal{A}$ will be post-processed by the Action-Level Common Sense model.

horizon tasks in a visual environment. Each task begins from an initial state $s_i \in \mathcal{S}$ and proceeds through multi-turn dialogue $D = \{(p_i, u_i)\}$, where $p_i \in \{\text{Commander}, \text{Follower}\}$ indicates the speaker and u_i is the utterance. At each timestep t, the *Follower* receives a visual observation v_t and executes an action $a_t \in \mathcal{A} : \mathcal{S}_t \rightarrow \mathcal{S}_{t+1}$, aiming to reach a goal state $s_f \in \mathcal{S}$.

Tasks often involve intermediate sub-goals (e.g., "Find bread", "Cook egg") under a high-level goal (e.g., "Make breakfast"). The *Commander* has oracle access to task and environment details, while the *Follower* has only egocentric perception and can request clarification during the dialogue.

Following [11], we consider three settings: in Execution from Dialog History (EDH),

the agent completes an unfinished task given past dialogue and partial trajectory; in Trajectory from Dialog (TfD), the agent reconstructs a full action sequence from complete dialogue; and in Two-Agent Task Completion (TATC), the agent collaborates interactively with a *Commander* to complete the task. Our proposed JARVIS framework is designed to handle all three scenarios.

3.3.2. Proposed Methods

Our JARVIS framework (as shown in Figure 3.2) consists of a Language Planning module, a Visual Semantic module, and a Symbolic Reasoning module. Specifically, the Language Planning module utilizes a pre-trained large language model to process free-form language input and produces procedural sub-goals of the task. In the Visual Semantic module, we use a semantic segmentation model, a depth prediction model, and the SLAM algorithm to transform raw visual observations into a more logistic form — semantic maps containing spatial relationships and states of the objects. Finally, in the Symbolic Reasoning module, we utilize task-level and action-level commonsense knowledge to reason about transferred symbolic vision-and-language information and generate actions, and a Goal Transformer is trained to deal with uncertainty by directly producing actions when no relevant information can be retrieved from the visual symbolic representations. Below we introduce our methods in detail; please refer to Appendix B.1 for more implementation details and Appendix B.2 for our notation table.

Language Understanding and Planning The language commands in dialog-based embodied tasks are free-form and high-level, and do not contain low-level procedural instructions. Therefore, it is essential to break a command like "can you make breakfast

for me?" into sub-goals such as "Find bread", "Find knife", "Slice bread", "Cook egg", and then generate actions a_i to complete the sub-goals sequentially. Large language models (LLMs) are shown to be capable of extracting actionable knowledge from learned world knowledge [97]. In order to understand free-form instructions and generate the sub-goal sequence for action planning, we leverage an LLM, the pre-trained BART [55] model, to process dialog and action, and predict future sub-goal sequence $G_{t:T}$ for completing the whole task:

$$\widehat{G}_{t:T} = \text{LLM}(D, G_{0:t-1}) \tag{3.3.1}$$

where $D = \{(p_i, u_i)\}$ is the collected set of user-utterance pairs and the previous sub-goal sequence $G_{0:t-1} = \{g_0, g_1, g_2, ..., g_{t-1}\}$. For sequential inputs $G_{0:t-1}$, we encode the sub-goals into tokens and concatenate them as the input of the LLM model. For the ground-truth sub-goal sequence $G_{0:T}$, we acquire it by a rule-based transformation from the action sequence $A_{0:T} = \{a_0, a_1, a_2, ..., a_t, ..., a_T\}$. Concretely, we note that the actions can be categorized as navigations and interactions. For interactions, we coalesce the action and targeted object as the sub-goal. For example, if the embodied agent executes the action of picking up a cup, we will record the sub-goal "PickUp Cup". For navigation, we coalesce "Navigate" with the target object of the next interaction.

Note that in cases like the Trajectory from History (TfD) task, where only dialog information D is given and other history information is missing, Equation 3.3.1 reduces to

$$\widehat{G}_{t:T} = \text{LLM}(D) \tag{3.3.2}$$

The BART model is trained with a reconstruction loss by computing the cross entropy

between the generated future sub-goal sequence $\widehat{G}_{t:T}$ and the ground truth future sub-goal sequence $G_{t:T}$.

Semantic World Representation The visual input into a embodied agent is usually a series of RGB images $V = \{v_0, v_1, v_2, ..., v_T\}$. One conventional way in deep learning is to fuse the visual information with other modalities into a neural network (e.g., Transformers) for decision making, which, however, is uninterpretable and often suffers from poor generalization under data scarcity or high task complexity. Thus, we choose to transform visual input into a semantic representation similar to [88] and [87], which can be used for generalized symbolic reasoning later.

At each step t, we first use a pre-trained Mask-RCNN model [54] for semantic segmentation, which we fine-tune on in-domain training data of the TEACh dataset, to get object types $O_t = \{o_0, o_1, o_2, ..., o_k\}$ and semantic mask $M_t = \{m_0, m_1, m_2, ..., m_k\}$ from egocentric RGB input v_t at time stamp t. Then we use a Unet [98] based depth prediction model as [88] to predict the depth of each pixel of the current egocentric image frame. Then, by combining the agent location and camera parameters, we transform the visual information into symbolic information: if a certain object exists in a 3D space, which we store in a 3D voxel. Then we further project the 3D voxel along the height dimension into a 2D map and obtain more concise 2D symbolic information. So far, we have transformed the egocentric RGB image into a series of 2D semantic maps, among which each map records a certain object's spatial location in the projected 2D plane. This symbolic environment information will be maintained and updated during the task completion process as in Figure. 3.2.

Action Execution via Symbolic Commonsense Reasoning Once the sub-goal sequence and semantic world representation are obtained, the agent must generate executable actions. However, these inputs may be noisy—sub-goals can be misordered or infeasible, and the semantic map may be incomplete. To address this, we introduce a Symbolic Reasoning module that leverages task-level and action-level commonsense logic to validate and refine execution plans. Sample logic predicates are listed in Table B.1 (Appendix B.1.1).

Task-level reasoning enforces logical preconditions between sub-goals. For instance, the predicate $Pick(agent, x)$ holds only if x is movable and the agent is free-handed. Sub-goals violating such constraints are revised—e.g., inserting "PickUp knife" before "Slice bread." This process assumes ideal execution and updates the agent's internal state accordingly.

Given validated sub-goals and the semantic map, we employ two action generation methods. If the target is visible, the Fast Marching Method (FMM) [99] plans a path to the nearest empty space near the object. Otherwise, we use a Goal Transformer (GT), adapted from the Episodic Transformer [86], trained on TEACh data. GT predicts the next action from past observations $V_{0:t-1}$, actions $A_{0:t-1}$, and sub-goals $G_{0:t-1}$:

$$\hat{a}_t = \text{GT}([V_{0:t-1}, A_{0:t-1}, G_{0:t-1}]). \tag{3.3.3}$$

Action-level reasoning ensures that planned actions respect environmental constraints. For example, $Move(x, y)$ is valid only if a path exists from y to x. If constraints are violated, the agent adapts—e.g., executing a fallback policy like random exploration or navigating to the nearest feasible location.

Table 3.1.: Results in percentages on the EDH and TfD validation sets, where trajectory length weighted (TLW) metrics are included in [brackets]. For all metrics, higher is better.

	EDH				**TfD**			
	Seen		*Unseen*		*Seen*		*Unseen*	
Model	SR [TLW]	GC [TLW]	SR [TLW]	GC [TLW]	SR [TLW]	GC [TLW]	SR [TLW]	GC [TLW]
E.T. [11]	8.4 [0.8]	14.9 [3.0]	6.1 [0.9]	6.4 [1.1]	1.0 [0.2]	1.4 [4.8]	0.5 [0.1]	0.4 [0.6]
Ours	**15.1 [3.3]**	**22.6 [8.7]**	**15.8 [2.6]**	**16.6 [8.2]**	**1.7 [0.2]**	**5.4** [4.5]	**1.8 [0.3]**	**3.1 [1.6]**
E.T. (few-shot)[2]	6.1 [1.0]	4.7 [2.8]	6.0 [0.9]	4.8 [3.6]	0.0 [0.0]	0.0 [0.0]	0.0 [0.0]	0.0 [0.0]
Ours (few-shot)	**10.7 [1.5]**	**15.3 [5.3]**	**13.7 [1.7]**	**12.7 [5.4]**	**0.6 [0.0]**	**3.6 [3.0]**	**0.3 [0.0]**	**0.6 [0.1]**

§ 3.4. Experiments

3.4.1. Dataset and Tasks

We evaluate JARVIS on the TEACh dataset [11], which contains over 3,000 human-human interaction sessions involving household tasks. Each session begins from an initial state s_i, proceeds through a multi-turn dialogue D between a *Commander* and a *Follower*, and ends in a final state s_f after executing a reference action sequence $A = \{a_0, a_1, \dots\}$. These sessions form the basis for three evaluation settings:

Execution from Dialog History (EDH) requires the agent to complete part of a task by predicting future actions $A_{t:T}$, given the current state s_i, dialogue history D, and previous actions $A_{0:t-1}$. Success is determined by whether the final simulated state $\hat{s}$ matches the reference s_f.

Trajectory from Dialog (TfD) tasks the agent with reconstructing the entire action sequence $A_{0:T}$ from the complete dialogue D and initial state s_i, aiming to reach the target final state s_f.

Two-Agent Task Completion (TATC) models both *Commander* and *Follower* roles. Given an initial instruction, the *Follower* communicates with the *Commander* to complete

Table 3.2.: Success Rate results on the TATC task with different assumptions of the Commander agent. Our JARVIS establishes the best performance among the current implemented methods.

	w/ full info.		w/o GT seg.		w/o GT & goal loc.	
Model	*Seen*	*Unseen*	*Seen*	*Unseen*	*Seen*	*Unseen*
E.T.[3]	0.0 [0.0]	0.0 [0.0]	0.0 [0.0]	0.0 [0.0]	0.0 [0.0]	0.0 [0.0]
Ours	**22.1 [5.8]**	**16.4 [4.2]**	**9.4 [2.1]**	**5.6 [1.9]**	**3.9 [0.5]**	**1.2 [0.1]**

Table 3.3.: Success Rates of 12 different task categories in the validation set. For EDH, the results are based on relatively shorter EDH instances, while the results for TdD and TATC tasks are from instances with full trajectories.

	Plant	Coffee	Clean All X	Y	Boil	Toast N	Slices X	One Y	Cooked	Sndwch	Salad	Bfast
EDH	21.0	21.3	14.5	15.2	15.5	12.8	22.1	19.6	15.8	15.5	10.3	14.0
TfD	13.5	6.7	6.3	4.2	1.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TATC	70.1	29.3	40.0	18.1	5.6	4.9	0.0	25.0	0.0	0.0	0.0	0.0

the task. The *Commander* has oracle access to environment metadata via three APIs: *ProgressCheck*, which lists task-relevant state differences; *SelectOid*, for querying object identifiers; and *SearchObject*, for locating objects. Following TEACh conventions, we implement JARVIS as two interacting agents: the *Commander* uses a Task-Level Commonsense Module to issue instructions, while the *Follower* executes actions and queries for help via an Action Execution Module.

3.4.2. Experimental Setup

Evaluation Metrics We adopt Success Rate (SR), Goal-Condition Success (GC), and Trajectory Weighted Metrics (TLW) as evaluation metrics. Task success is a binary

**Our reimplementation version of [11].

†From the official TATC Challenge: https://github.com/GLAMOR-USC/teach_tatc. At the time of submission, the E.T. baseline fails on all the TATC instances.

value, defined as 1 when all the expected state changes s_f are presented in $\hat{s}$ otherwise 0. SR is the ratio of success cases among all the instances. GC Success is a fraction of expected state changes in s_f present in $\hat{s}$ which is in $(0, 1)$ and the GC success of the dataset is the average of all the trajectories. Trajectory weighted SR (TLW-SR) and GC (TLW-GC) are calculated based on a reference trajectory A_R and a predicted action sequence $\widehat{A}$. The trajectory length weighted metrics for metric value m can be calculated as

$$\text{TLW-}m = \frac{m * |A_R|}{\max\left(|A_R|, \left|\widehat{A}\right|\right)} \tag{3.4.1}$$

In our evaluation, we use $m = 1$, and calculate the weighted average for each split by using $\frac{|A_R|}{\sum_{i=1}^{N} |A_R^i|}$ as the weight of each instance. Following evaluation rules in TEACh [11], the maximum number of action steps is 1,000 and the failure limit is 30.

Baseline We select Episodic Transformer (E.T.) [11] as the baseline method. E.T. achieved SOTA performance on the TEACh dataset. It learns the action prediction based on the TEACh dataset using a ResNet-50 [91] backbone to encode visual observations, two multi-modal transformer layers to fuse the embedded language, visual and action information, and output the executable action directly.

3.4.3. Main Results and Analysis

EDH We establish the state-of-the-art performance with a large margin over the baseline E.T.model. Our model achieves a higher relative performance than baselines on trajectory-weighted metrics, as shown in Table 3.1, suggesting that adding symbolism can also reduce navigation trajectory length. Besides, our framework has less performance gap between seen and unseen environments, showing better generalizability in new

Table 3.4.: Ablation studies on the EDH and TfD validation sets. **Language**: the agent directly teleports to the target goals generated by the language language planning module, thus trajectory length weighted metrics do not make sense here. **Executor**: the agent will use the ground truth sub-goals. **Reasoning**: the agent use both ground truth sub-goals and visual information. **No Goal Transformer**: the agent explore random place when it did not find the target object.

	EDH				**TfD**			
	Seen		*Unseen*		*Seen*		*Unseen*	
Model	SR [TLW]	GC [TLW]	SR [TLW]	GC [TLW]	SR [TLW]	GC [TLW]	SR [TLW]	GC [TLW]
JARVIS	15.1 [3.3]	22.6 [8.7]	15.8 [2.6]	16.6 [8.2]	1.7 [0.2]	5.4 [4.5]	1.8 [0.3]	3.1 [1.6]
Language (w/ gt executor)	19.9 [—]	32.9 [—]	18.8 [—]	36.3 [—]	8.4 [—]	24.7 [—]	10.2 [—]	21.5 [—]
Executor (w/ gt subgoals)	38.7 [12.5]	34.7 [17.9]	30.5 [8.5]	27.3 [12.1]	5.1 [1.3]	8.0 [4.3]	1.8 [0.2]	4.0 [0.9]
Reasoning (w/ gt subgoals & visual)	62.7 [27.9]	67.9 [36.7]	60.9 [26.2]	63.6 [35.4]	13.8 [4.0]	24.5 [15.5]	12.6 [4.2]	20.7 [14.6]
No Goal Transformer	17.3 [3.2]	21.7 [9.0]	15.7 [1.6]	16.9 [6.3]	0.6 [0.0]	3.1 [1.1]	1.8 [0.2]	1.4 [0.6]

environments. We find that E.T. usually gets stuck in a corner or keeps repeatedly circling, while our model barely suffers from those issues, benefiting from neuro-symbolic commonsense reasoning.

TfD Our model achieves state-of-the-art performance across all metrics in TfD tasks, as shown in Table 3.1, which shows that JARVIS has better capability for task execution based on offline human-human conversation. Compared with EDH tasks, TfD tasks provide only the entire dialog history for the agent, which increases the difficulty and causes performance decreases in both models, while our JARVIS still outperforms the E.T. by a large margin, showing that the end-to-end model can not learn an effective and generalized strategy for long tasks completion providing only dialog information.

TATC We evaluate our framework on the TATC task with a *Commander* under three distinct constraint levels, simulating varied human assistance. The *Commander* is provided with: 1) only the current sub-goal; 2) the sub-goal and target object location; or 3) the sub-goal, location, and segmentation mask (the original TATC setting [11]). As shown in Table 3.2, JARVIS greatly outperforms the only open-source baseline,

which fails on all TATC instances, highlighting the task's complexity for end-to-end methods. Furthermore, the Success Rate (SR) improves as the *Commander* is given more information, demonstrating that JARVIS effectively adapts its instructions to leverage all available knowledge.

3.4.4. Few-Shot Learning

Data scarcity has been known as a severe issue for deep neural methods. Especially, it is even more severe in language-involved embodied agent tasks since collecting training data is more expensive and time-consuming. Here we also conduct experiments in the few-shot setting, shown in Table 3.1. We randomly sample ten instances from each of the 12 types of household tasks in the TEACh dataset and train the language understanding and planning module and Goal Transfomer in the same way as the whole dataset setting. For E.T., we notice a significant performance drop on both EDH and TfD (e.g., 0 success rate in TfD tasks), since it overfits and can not learn effective and robust strategy. Since our framework breaks down the whole problem into smaller sub-problems and incorporates a solid symbolic commonsense reasoning module, it still has the ability to complete some complex tasks. This also indicates the importance of connecting connectionism and symbolism.

3.4.5. Unit Test of Individual Module

To analyze the performance gain of JARVIS, we conduct ablation studies on EDH and TfD tasks (Table 3.4). With ground truth perception and sub-goals, the Symbolic Commonsense Reasoning module achieves over 60% success in EDH, showing its

effectiveness in action inference when provided accurate inputs. Replacing our language planner with ground truth yields greater improvement than replacing the executor, confirming the importance of symbolic reasoning in short-horizon tasks.

In longer TfD tasks, where a single sub-task failure leads to overall task failure, executor quality becomes the bottleneck. Here, replacing our executor with a teleport agent boosts performance more than using ground truth sub-goals. We also observe that our Goal Transformer improves trajectory-weighted success, indicating more efficient action planning.

§ 3.5. Conclusion

This work studies how to teach embodied agents to execute dialog-based household tasks. We propose JARVIS, a neuro-symbolic framework that can incorporate the perceived neural information from multi-modalities and make decisions by symbolic commonsense reasoning. Our framework outperforms baselines by a large margin on three benchmarks of the TEACh [11] dataset. We hope the methods and findings in this work shed light on the future development of neuro-symbolic embodied agents.

Part II.

Interleaved Multimodal Generation

Having established how pretrained perception and language modules can be composed for embodied reasoning, this part turns to a second family of composition: how can two pretrained generators, one of language and one of images, be composed so that the resulting system *produces*, rather than merely interprets, coordinated text and visual content? Large language models excel at text, diffusion models excel at images, but the interface between them has been the bottleneck. Monolithic generators require expensive full-model retraining; naively chaining a comprehension model with a text-to-image model through a textual caption discards long-range context whenever more than a short prompt is needed.

MiniGPT-5 (Chapter 4) addresses this interface directly through *generative vokens*, a small set of learnable tokens that live in the language model's output space and project into the diffusion model's conditional space. Combined with a description-free two-stage training scheme and classifier-free guidance over the vokens, this lets a comprehension-oriented multimodal model emit interleaved text-and-image sequences while keeping both pretrained backbones frozen. The chapter argues that generation capability can be added on top of existing pretrained components through a narrow, learnable token interface rather than through end-to-end retraining. This is a much lighter form of composition than the symbolic interfaces of Part I, and a principle that carries directly into the spatial and iterative interfaces of Part III.

Chapter 4.

Interleaved Vision-and-Language Generation via Generative Voken

§ 4.1. Introduction

The development of large-scale vision-and-language models is significantly impacting a wide range of fields, like automated dialogue systems and digital content creation. With the surge in research and development in this domain, the current state-of-the-art Large Language Models (LLMs) [1, 100, 101] and vision-and-language models such as [2, 4, 102, 103] fall short in generating coherent multimodal outputs. This limitation becomes particularly evident in tasks that demand an integrated handling of vision and language, essential for the next generation of Large Language Models (LLMs).

Our work, as illustrated in Fig. 4.1, seeks to address these shortcomings by enhancing the integration of text and image generation in LLMs. The challenges in developing a multimodal LLM capable of interleaved vision and language generation are manifold. First, LLMs typically lack mechanisms to directly produce images, prompting us to introduce "generative vokens" that bridge the gap between textual and visual feature

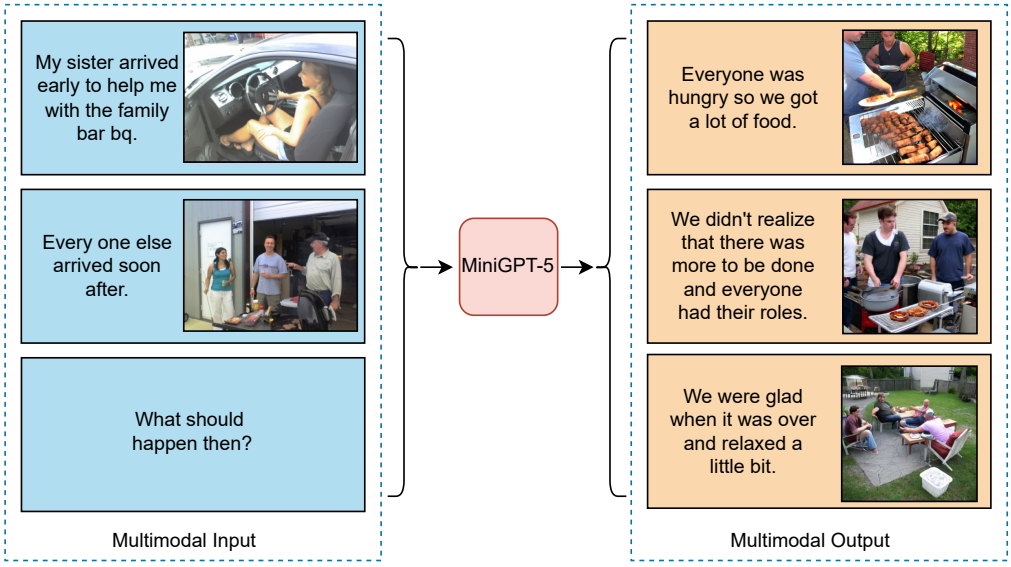


Figure 4.1.: MiniGPT-5 is a unified model for interleaved vision-and-language comprehension and generation. Besides the original multimodal comprehension and text generation abilities, MiniGPT-5 can provide appropriate, coherent multimodal outputs.

spaces. Second, the constraint of data scarcity, especially in vision-and-language tasks [104], lacking extensive, detailed captions of images [105], like descriptions of the elements inside images, is countered by our unique description-free training approach. Third, maintaining both image-text and image-image consistency poses a significant challenge, which we address through dual-loss strategies. Finally, as we push forward the boundaries with LLMs, the large memory requirements urge us to devise more efficient end-to-end strategies and create an efficient training pipeline accessible to the community, especially in downstream tasks.

Specifically, to overcome these challenges, we present MiniGPT-5, a novel approach for interleaved vision-and-language generation. By combining Stable Diffusion with LLMs through special visual tokens [106] – “generative vokens", we develop a new

approach for multimodal generation. Our two-stage training methodology emphasizes a description-free foundational phase, enabling effective model training even with limited caption-grounded images. This strategy, distinct from existing works, pivots on generic stages free from image annotations. To ensure that the generated text and images are in harmony, our dual-loss strategy comes into play, further enhanced by our innovative generative voken approach and classifier-free guidance. Our parameter-efficient fine-tuning strategy optimizes training efficiency and addresses memory constraints.

As shown in Fig. 4.2, leveraging ViT (Vision Transformer) and Qformer [4], alongside Large Language Models, we adapt multimodal inputs into generative vokens, seamlessly combined with the high-resolution Stable Diffusion 2.1 model [5] for context-aware image generation. Incorporating images as auxiliary input with instruction tuning approaches and pioneering both the text and image generation loss, we amplify the synergy between text and visuals. We experiment on the CC3M [104], VIST [105], and MMDialog [107] datasets. Notably, MiniGPT-5 shows superior performance across the two multimodal generation datasets.

In summary, our contributions are primarily threefold:

- We introduce a novel framework that leverages "generative vokens" to unify LLMs with Stable Diffusion, facilitating interleaved vision-and-language generation without relying on detailed image descriptions. We bridge the modality gap and improve the generation quality by using the loss of the latent diffusion model, the text generation loss, and the caption alignment loss together during training.

- We propose a new two-stage training strategy for description-free multimodal generation. The first stage focuses on extracting high-quality text-aligned visual

features from large text-image pairs, while the second stage ensures optimal coordination between visual and textual prompts during generation. The inclusion of classifier-free guidance during training enhances the overall generation quality.

- MiniGPT-5 achieves significant improvements over baseline methods on interleaved vision-and-language datasets, including VIST and MMDialog, and comparable results to the state-of-the-art on the single text-image pair dataset, CC3M. The human evaluation further shows that, compared with the two-stage baseline, MiniGPT-5 can provide better generation in perspectives of appropriate texts (55%), high-quality images (53%), and coherent multimodal outputs (56%).

§ 4.2. Related Work

Multimodal Large Language Models As Large Language Models (LLMs) become increasingly impactful and accessible, a growing body of research has emerged to extend these pretrained LLMs into the realm of multimodal comprehension tasks [1–4, 108–111]. For example, to reproduce the impressive multimodal comprehension ability in GPT-4 [1], MiniGPT-4 [3] proposes a projection layer to align pretrained vision component of BLIP-2 [4] with an advanced open-source large language model, Vicuna [100]. In our work, we utilize the MiniGPT-4 as the base model and extend the model's capabilities to multimodal generation.

Text-to-Image Generation To transform textual descriptions into their corresponding visual representations, text-to-image models [5, 112–121] design algorithms to bridge the gap between textual information and visual content. A notable recent contribution is Stable Diffusion V2 [5], which employs a diffusion process to generate conditional

image features and subsequently reconstructs images from these features. Our research aims to leverage this pretrained model, enhancing its capabilities to accommodate both multimodal input and output.

Multimodal Generation with Large Language Models. Recent work augments LLMs to *generate* as well as *understand* across modalities via retrieval-augmented decoders, tokenized image streams, or learned visual-token interfaces [122–129]. CM3Leon [126] uses a retrieval-augmented, decoder-only architecture for text↔image tasks. Emu [129] converts images to 1D features with EVA-CLIP [130] and fine-tunes LLaMA [131] to autoregress over text and *text-encoder–conditioned* image features (requiring subsequent T2I fine-tuning). NextGPT [127], GILL [125], and SEED [124] map "visual tokens" into the *text* feature space of a pretrained diffusion model (via encoder–decoder or Q-Former variants). A parallel line builds *unified* autoregressive models that couple understanding and synthesis in a single network by introducing large vocabularies of image tokens and performing extensive full-model (re)training, e.g., Chameleon [132], Emu3 [133], and Show-o [134]. Unlike unified models, MiniGPT-5 is *modular and parameter-efficient*, reusing pretrained MLLM and diffusion backbones while tuning only a lightweight feature mapper and LoRA. Its key novelty is *generative vokens* that project LLM states directly into the diffusion *conditional* space and are trained with a latent diffusion loss; paired with *description-free* interleaved fine-tuning and *CFG over vokens*, this yields efficient, long-horizon interleaved generation with minimal backbone changes.

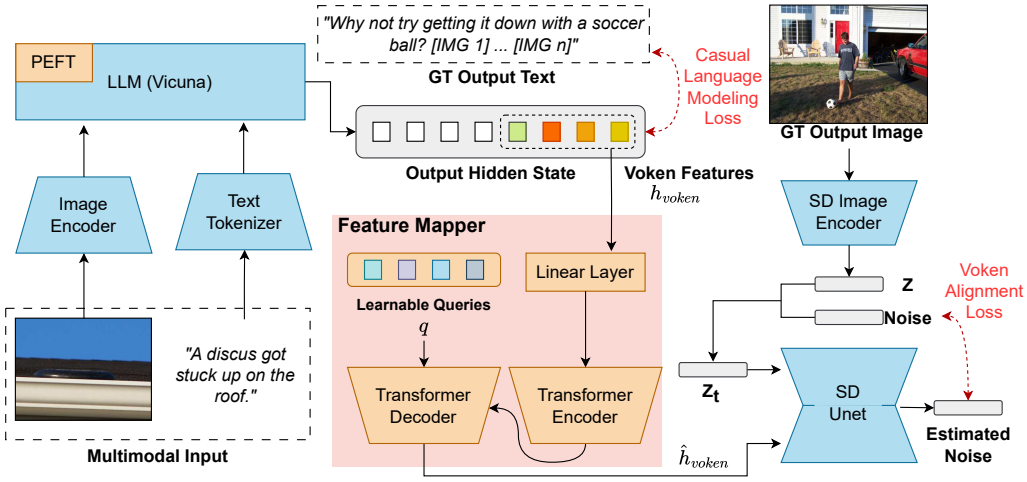


Figure 4.2.: The overview structure of MiniGPT-5 pipeline. We leverage the pretrained multimodal large language model (MiniGPT-4) and text-to-image generation model (Stable Diffusion 2.1) to create a unified multimodal generation pipeline. The input image encoder includes a ViT, Qformer, and linear layer, pretrained by MiniGPT-4. The orange blocks include learnable parameters, while the blue blocks are fixed during training. More details can be found in Section 4.3.

§ 4.3. Method

In order to endow Large Language Models with multimodal generation capabilities, we introduce a new framework that integrates pretrained multimodal Large Language Models and text-to-image generation models. Central to our approach is the introduction of "generative vokens", special visual tokens that effectively bridge the textual and visual domains during the training process. Additionally, we implement a two-stage training method combined with a classifier-free guidance strategy to enhance the quality and coherence of generated outputs. Fig. 4.2 provides an overview of our model structure. MiniGPT-5 primarily consists of two modules: the Multimodal Understanding module, utilizing the pretrained multimodal large language model (MiniGPT-4) for handling

multimodal inputs, and the Multimodal Generation module, employing Stable Diffusion for generating visual outputs.

4.3.1. Multimodal Understanding Module

Recent advancements in multimodal Large Language Models, such as MiniGPT-4 [3], have primarily concentrated on multimodal comprehension, enabling the processing of images as sequential input. The Integrated Vision-Language Encoding Module is designed to extend the capabilities of LLMs from mere comprehension to active generation in multimodal contexts. Generative vokens play a crucial role in this module, enabling the translation of raw visual inputs into a format that LLMs can process and utilize for subsequent generation tasks.

Multimodal Encoding Each text token is embedded into a vector $e_{\text{text}} \in \mathbf{R}^d$, while the pretrained visual encoder transforms each input image into the feature $e_{\text{img}} \in \mathbf{R}^{32\times d}$. These embeddings are concatenated to create the input prompt features.

Generative Vokens Since the original LLM's V vocabulary only includes the textual tokens, we need to construct a bridge between the LLM and the generative model. Therefore, we introduce a set of special tokens $V_{\text{img}} = \{[\text{IMG1}], [\text{IMG2}], \ldots, [\text{IMGn}]\}$ (by default $n = 8$) as generative vokens into the LLM's vocabulary V. The LLM's output hidden state for these vokens is harnessed for subsequent image generation, and the positions of these vokens can represent the insertion of the interleaved images. With all pretrained weights $\theta_{\text{pretrained}}$ in MiniGPT-4 fixed, the trainable parameters include extra input embedding $\theta_{\text{voken\_input}}$ and output embedding $\theta_{\text{voken\_output}}$.

Parameter-Efficient Fine-Tuning (PEFT) Parameter-efficient fine-tuning (PEFT) [135–137] is critical in training Large Language Models (LLMs), employed to adapt LLMs to

downstream tasks without the need for extensive retraining. In PEFT, rather than updating all the parameters of a model, only a small subset of parameters is trained. This subset typically includes task-specific components or lightweight layers added to the original model architecture [135, 136, 138, 139]. We apply PEFT to the MiniGPT-4 [3] encoder, enhancing its ability to process and generate multimodal content based on given instructions or prompts. More specifically, this involves the use of prefix tuning [137] and LoRA[136] over the entire language encoder – `Vicuna` [100] used in MiniGPT-4. Additionally, we implement learnable queries at the input of the transformer decoder, a conventional approach in sequence-to-sequence transformer architectures, to further improve the model's multimodal generation capabilities. We also adopted learnable queries at the input of the transformer decoder as a conventional setting for sequence-to-sequence transformer architectures [140]. Learnable queries in the decoder allow the model to have dynamic, adaptable representations for initiating the generation process. This is particularly useful when the model needs to generate outputs based on a mix of visual and textual inputs. Combined with the instruction tuning [101], it notably amplifies multimodal generation performance across various datasets.

4.3.2. Multimodal Generation Module

To accurately align the generative vokens with the text-to-image generation models, we formulate a compact mapping module for dimension matching and incorporate several supervised losses, including causal language modeling loss and voken alignment loss. The causal language modeling loss assists the model in learning the correct positioning of tokens, while the voken alignment loss directly aligns the vokens with the appropriate conditional generation features of the diffusion model. Since the gradients of generative

vokens' features can be directly calculated from images, shown on the right side of Fig. 4.2, our method does not need comprehensive descriptions of images, leading to description-free learning.

Voken Positioning We first jointly generate both text and vokens in the text space by following next-word prediction in autoregressive language model [140]. During the training, we append the vokens V_{img} to the positions of ground truth images and train the model to predict vokens within text generation. Specifically, the generated tokens are represented as $W = \{w_1, w_2, \ldots, w_m\}$, where $w_i \in V \cup V_{\text{img}}$, and the causal language modeling loss is defined as:

$$
\begin{aligned}
L_{\text{text}} := -\sum_{i=1}^{m} \log p(w_i | & e_{\text{text}}, e_{\text{img}}, w_1, \ldots, w_{i-1}; \\
& \theta_{\text{pretrained}}, \theta_{\text{voken\_input}}, \theta_{\text{voken\_output}}), \\
& \text{where } w_i \in V \cup V_{\text{img}}
\end{aligned}
\tag{4.3.1}
$$

Voken Alignment for Image Generation Next, we align the output hidden state h_{voken}, shown in Fig. 4.2, with the conditional feature space of the text-to-image generation model. To map the voken feature h_{voken} to a feasible image generation conditional feature $e_{\text{text\_encoder}} \in \mathbf{R}^{L \times \hat{d}}$ (where L is the maximum input length of text-to-image generation text encoder, and $\hat{d}$ is the dimension of encoder output feature in text-to-image generation model). We construct a feature mapper module, including a two-layer MLP model θ_{MLP}, a four-layer encoder-decoder transformer model $\theta_{\text{enc-dec}}$, and a learnable decoder feature sequence q. The mapping feature $\hat{h}_{\text{voken}}$ is then given by:

$$
\hat{h}_{\text{voken}} := \theta_{\text{enc-dec}}(\theta_{\text{MLP}}(h_{\text{voken}}), q) \in \mathbf{R}^{L \times \hat{d}}
\tag{4.3.2}
$$

To generate appropriate images, the mapping feature $\hat{h}_{\text{voken}}$ is used as a conditional input in the denoising process. Intuitively, $\hat{h}_{\text{voken}}$ should represent the corresponding conditional features that conduct the diffusion model to generate the ground truth image. We employ the latent diffusion model (LDM) loss as voken alignment loss for training the image generation module. During the training, the ground truth image is first converted to latent feature z_0 through the pretrained VAE (Variational Autoencoder) [141]. Then, we obtain the noisy latent feature z_t by adding noise ϵ to z_0. A pretrained U-Net model ϵ_θ is used to calculate the conditional LDM loss as:

$$L_{LDM} := \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1),t}\left[\left\|\epsilon - \epsilon_\theta\left(z_t, t, \hat{h}_{\text{voken}}\right)\right\|_2^2\right] \tag{4.3.3}$$

To summarize, the causal language modeling loss enables the model to learn the accurate placement of tokens. Without this component, the model lacks the essential capability to predict when vokens should be generated during inference. Additionally, the voken alignment loss ensures the direct correspondence between vokens and the appropriate conditional generation characteristics of the diffusion model. In the absence of this loss, the model is unable to learn semantic vokens from images directly. This comprehensive approach ensures a coherent understanding and generation of both textual and visual elements, leveraging the capabilities of pretrained models, specialized tokens, and innovative training techniques.

4.3.3. Training Strategy

Given the non-negligible domain shift between text and image domains, we observe that direct training on a limited interleaved text-and-image dataset can result in misalignment of generated texts and images and diminished image quality. Consequently, we adopt

a two-stage training strategy: an initial pretraining stage focusing on coarse feature alignment for unimodal generation, followed by a fine-tuning stage dedicated to intricate feature learning for multimodal generation. Furthermore, to amplify the effectiveness of the generative tokens throughout the diffusion process, we incorporate the idea of classifier-free guidance [142] technique throughout the whole training process.

Two-stage Training Strategy Recognizing the non-trivial domain shift between pure-text generation and text-image generation, we propose a two-stage training strategy: Pretraining Stage and Fine-tuning Stage. Initially, we align the voken feature with image generation features in single text-image pair datasets, such as CC3M, where each data sample only contains one text and one image, and the text is usually the caption of the image. During this stage, we utilize captions as LLM input, enabling LLM to generate vokens. Since these datasets include the image descriptive information, we also introduce an auxiliary loss to aid voken alignment, minimizing the distance between the generative feature $\hat{h}_{\text{voken}}$ and the caption feature from the text encoder τ_θ in the text-to-image generation model:

$$L_{\text{CAP}} := \text{MSE}(\hat{h}_{\text{voken}}, \tau_\theta(c)) \tag{4.3.4}$$

The pretraining stage loss is expressed as $L_{\text{Pretrain}} = \lambda_1 * L_{\text{text}} + \lambda_2 * L_{\text{LDM}} + \lambda_3 * L_{\text{CAP}}$, with selected values $\lambda_1 = 0.01, \lambda_2 = 1, \lambda_3 = 0.1$ to rescale the loss into a similar numerical range.

After the pretraining stage, the model is capable of generating images for single text descriptions but struggles with interleaved vision-and-language generation, which includes multiple text-image pairs and requires complicated reasoning for both text and

image generation. To address this, in the fine-tuning stage, we further fine-tune our model with PEFT parameters by interleaved vision-and-language datasets, such as VIST, where the data sample has several steps with text-image and texts are sequentially relevant. During this stage, we construct three types of tasks from the dataset, encompassing (1) text-only generation: given the next image, generating the related text; (2) image-only generation: given the next text, generating the related image, and (3) multimodal generation: generating a text-image pair by given context. The fine-tuning stage loss is given by $L_{\text{Fine-tune}} = \lambda_1 * L_{\text{text}} + \lambda_2 * L_{\text{LDM}}$. More implementation details can be found in Section 4.4.1.

Classifier-Free Guidance (CFG) To enhance the coherence between the generated text and images, we first leverage the idea of Classifier-free Guidance for multimodal generation. Classifier-free guidance is introduced in the text-to-image diffusion process. This method observes that the generation model P_θ can achieve improved conditional results by training on both conditional and unconditional generation with conditioning dropout. In our context, we want the model to focus directly on the output features h_{voken} from LLM. Instead of using original stable diffusion unconditional distributions (dropping $\hat{h}_{\text{voken}}$), the whole feature mapper also needs to be included during the unconditional process. Therefore, our objective is to accentuate the trainable condition h_{voken} and the generation model is fixed. During training, we replace h_{voken} with zero features $h_0 \in \mathbf{0}^{n \times d}$ with a 10% probability, obtaining the unconditional feature $\hat{h}_0 = \theta_{\text{enc-dec}}(\theta_{\text{MLP}}(h_0), q)$. During inference, $\hat{h}_0$ serves as negative prompting, and the refined denoising process is:

$$\begin{aligned} \log \widehat{\mathrm{P}_\theta}\left(\epsilon_t \mid z_{t+1}, \hat{h}_{\text{voken}}, \hat{h}_0\right) &= \log \mathrm{P}_\theta\left(\epsilon_t \mid z_{t+1}, \hat{h}_0\right) + \\ \gamma\left(\log \mathrm{P}_\theta\left(\epsilon_t \mid z_{t+1}, \hat{h}_{\text{voken}}\right) - \log \mathrm{P}_\theta\left(\epsilon_t \mid z_{t+1}, \hat{h}_0\right)\right) & \end{aligned} \tag{4.3.5}$$

§ 4.4. Experiments

To assess the efficacy of our model, we conducted a series of evaluations across multiple benchmarks. These experiments aim to address several key questions: (1) *Can our model generate plausible images and reasonable texts?* (2) *How does our model compare with state-of-the-art models in both single-turn and multi-turn interleaved vision-and-language generation tasks?* (3) *What impact does the design of each module have on overall performance?* Below we will discuss the experimental setup and present a comprehensive analysis of our model's performance. We use three datasets: CC3M [104], VIST [105], and MMDialog [107]. More details about datasets and data format can be found in Appendix C.1.

4.4.1. Implementation Details

In the pretraining stage, we introduce additional voken embeddings at both the input and output layers of the Vicuna-7B model, while keeping the embeddings of other tokens fixed. These new embeddings – denoted as $\theta_{\text{voken\_input}}$ and $\theta_{\text{voken\_output}}$ – along with the feature mapper module $(\theta_{\text{MLP}}, \theta_{\text{enc\_dec}}, q)$ are jointly trained on the CC3M dataset, which consists of single text-image pairs. Training is conducted using the AdamW optimizer over two epochs, with a batch size of 48, amounting to over 110,000 steps, and a learning rate of 2×10^{-4}.

In the subsequent fine-tuning stage, we incorporate LoRA modules – denoted as θ_{LoRA} – into Vicuna for the generation of both tokens and vokens. We keep the MLP model θ_{MLP} and decoder query q fixed. The model is then fine-tuned on interleaved vision-and-language datasets, like VIST and MMDialog. The trainable parameters

for this stage are $\theta = \{\theta_{\text{voken\_input}}, \theta_{\text{voken\_output}}, \theta_{\text{LoRA}}, \theta_{\text{enc\_dec}}\}$. Training is carried out using the AdamW optimizer over four epochs, with a batch size of 32 and a learning rate of 2×10^{-5}. Trainable parameters are nearly 6.6 million, and all training can be completed on a server equipped with 4 A6000 GPUs.

4.4.2. Experimental Setup

Baselines

For a comprehensive evaluation of our performance in multimodal generation, we conducted comparative analyses with several prominent baseline models: the Fine-tuned Unimodal Generation Models, Two-stage Baseline, GILL * [125], and Divter [122]:

- **Fine-tuned Unimodal Generation Models**: To facilitate fair comparisons in both image and text generation, we fine-tuned two separate models, Stable Diffusion 2.1 and MiniGPT-4 [3], utilizing the VIST dataset. Within the Stable Diffusion 2.1 [5] model, the U-Net parameters were fine-tuned. For MiniGPT-4's LLM part, LoRA parameters were fine-tuned.

- **Two-stage Baseline**: A common approach in multimodal generation involves first employing Large Language Models (LLMs) to create image captions, which are then fed into text-to-image models for image generation [143]. We create such a two-stage baseline for comparison with our end-to-end method by fine-tuning MiniGPT-4 for caption generation and Stable Diffusion 2.1 for text-to-image generation. Given the absence of image descriptions in the VIST dataset, we

*Given the variations in the valid data within the CC3M dataset, we made adjustments to ensure fair comparisons. Specifically, we retrained it on our specific CC3M data, following the guidelines in their official implementation (https://github.com/kohjingyu/gill).

incorporate a SOTA image captioning model, InstructBLIP-13B [108], to generate synthetic captions for supervision.

- **GILL**: GILL is a recent innovation that allows the LLM to generate vokens using a pre-trained text-to-image generation model for single-image generation, where GILL minimizes the Mean Squared Error (MSE) loss between the text-to-image text encoding feature and voken features, similar to L_{CAP} in our approach. For fine-tuning on multimodal datasets, since GILL requires image captions for training, we use Descriptions of Images-in-Isolation (DII) [105] in the VIST fine-tuning and generate captions for MMDialog fine-tuning. Contrarily, MiniGPT-5 does not related on all caption data during multimodal generation fine-tuning.

- **Divter** [122]: Divter is a state-of-the-art conversational agent developed for multimodal dialogue contexts. It introduces a customized transformer structure for generating multimodal responses. Divter's methodology includes pretraining on a vast corpus of text-only dialogues and text-image pairs, followed by fine-tuning on a selected set of multimodal response data. The MMDialog dataset regards Divter's method as the baseline.

Metrics

To comprehensively assess the model performance across image, text, and multimodal dimensions, we employ a diverse set of metrics. For evaluating the quality and diversity of generated images, we utilize the Inception Score (IS) [144], and Fréchet Inception Distance (FID) [145]. Textual performance is gauged through metrics such as BLEU [146], Rouge-L [147], METEOR [148], and Sentence-BERT (S-BERT) [149]

Table 4.1.: Image generation on VIST. Given the historical context, models need to generate images for each step. FID scores evaluate the visual diversities between generated and ground truth images within each story sequence.

Model	CLIP-I (↑)	FID (↓)
SD 2.1 [5]	0.59	393.49
Fine-tuned SD 2.1	0.61	390.25
Two-stage Baseline	0.57	403.06
GILL [125]	0.60	381.88
MiniGPT-5 (Prefix Tuning)	0.65	381.55
MiniGPT-5 (LoRA)	**0.66**	**366.62**

Table 4.2.: Narration Generation on VIST. We added LoRA fine-tuning for GILL, MiniGPT-4, and MiniGPT-5 with the same LoRA configuration. The results show that adding generative vokens does not hurt the performance on the multimodal comprehension tasks.

Model	S-BERT (↑)	Rouge-L (↑)	Meteor (↑)
GILL [125]	0.3864	0.1784	0.1951
MiniGPT-4 [3]	0.6273	**0.3401**	**0.3296**
MiniGPT-5	**0.6315**	0.3373	0.3263

scores.

From the multimodal perspective, we leverage CLIP-based metrics [5] to assess the similarities between generated content and ground truth. CLIP-I evaluates the similarity between generated and ground-truth image features. To address potential misalignments in the multimodal generation, such as when the ground truth is text-only, but the output is multimodal, we utilize MM-Relevance [107]. This metric calculates the F1 score based on CLIP similarities, providing a nuanced evaluation of multimodal coherence.

We also incorporate human evaluation to assess the model's performance. We examine the model's effectiveness from three perspectives: (1) *Language Continuity*: assessing if the produced text aligns seamlessly with the provided context; (2) *Image Quality*: evaluating the clarity and relevance of the generated image; and (3) *Multimodal Coherence*: determining if the combined text-image output is consistent with the initial context.

Table 4.3.: VIST Human Evaluation on 5,000 samples for multimodal generation from Language Continuity, Image Quality, and Multimodal Coherence aspects. The results indicate, in more than 70% cases, the MiniGPT-5 is better or on par with the two-stage baseline.

Model	MiniGPT-5	Two-stage Baseline	Tie
Language Continuity (%)	**55.22**	34.89	9.89
Image Quality (%)	**52.43**	37.79	9.78
Multimodal Coherence (%)	**56.90**	28.88	14.22

4.4.3. Main Results

In this subsection, we present the performance of different models on the VIST [105] and MMDialg [107] datasets. Our evaluations span all vision, language, and multimodality domains to showcase the versatility and robustness of the proposed models.

Unimodal Generation on VIST To evaluate the model performance on image generation and text generation, we systematically provide models with prior history context and subsequently assess the generated images and narrations at each following step. Tables 4.1 and 4.2 outline the results of these experiments on the VIST validation set, showing the performance in both image and language metrics, respectively. The findings demonstrate that MiniGPT-5 can generate coherent, high-quality images utilizing long-horizontal multimodal input prompts across all data, without compromising the original model's ability for multimodal comprehension, indicating the efficacy of our model in diverse settings.

Multimodal Generation on VIST To assess the quality of multimodal generation, we test both our model and the baselines on the VIST validation set by human evaluation. Given a preceding multimodal sequence, models are tasked with producing the subsequent

scenario for each task. We select a random sample of 5,000 sequences, with each requiring evaluation by two workers. These evaluators are tasked with determining the superior multimodal output based on three criteria: Language Continuity, Image Quality, and Multimodal Coherence. This assessment is facilitated using Amazon Mechanical Turk [150], with a representative example (Fig. C.1) provided in the Appendix. As depicted in Table 4.3, our model, MiniGPT-5, is found to generate more fitting text narrations in around 55% of instances, deliver superior image quality in around 53% of cases, and produce more coherent multimodal outputs in around 56% of the scenarios. This data distinctly showcases its enhanced multimodal generation capabilities compared to the two-stage baseline, which must generate intermediate image captions first.

Multimodal Dialog Generation on MMDialog We conduct an evaluation of our method on the MMDialog dataset to determine the effectiveness of generating precise and appropriate multimodal information in multi-turn conversational scenarios. The model is required to generate either unimodal or multimodal responses based on the previous turns during the conversations. Our results, as presented in Table 4.4, demonstrate that MiniGPT-5 outperforms the baseline model Divter in terms of generating more accurate textual responses. While the image qualities of the generated responses are similar, MiniGPT-5 excels in MM-Relevance compared to the baselines. This indicates that our model can better learn how to position image generation and produce highly coherent multimodal responses appropriately.

4.4.4. Ablation Studies

To further evaluate the effectiveness of our design, we conducted several ablation studies, and more ablation studies can be found in Appendix C.2.

Table 4.4.: Multimodal generation results on MMDialog test set. In order to compare with their baseline, we use the same metrics reported in MMDialog [107].

Model	IS (↑)	BLEU-1 (↑)	BLEU-2 (↑)	Rouge-L (↑)	MM-Relevance (↑)
Divter [122]	20.53	0.0944	0.0745	0.1119	0.62
GILL [125]	**23.78**	0.2912	0.1945	**0.1207**	0.64
MiniGPT-5	20.23	**0.3369**	**0.2323**	0.1176	**0.67**

Table 4.5.: Evaluation of different method designs for image generation qualities on the CC3M validation set.

Model	CLIP-I (↑)	CLIP-T (↑)	IS (↑)	FID (↓)
MiniGPT-5	**0.61**	**0.22**	**28.09**	**31.47**
MiniGPT-5 (w/o CFG)	0.60	0.22	23.41	33.73
MiniGPT-5 (w/o L_{CAP})	0.54	0.16	21.27	40.24
MiniGPT-5 (w/o L_{LDM})	0.58	0.20	24.79	34.65

Evaluation of Classifier-Free Guidance (CFG) To assess the effectiveness of the CFG strategy, we trained our model without CFG dropoff. During inference, the model utilized the original CFG denoising process, which utilized the empty caption feature from Stable Diffusion's text encoder as negative prompt features. The results in Table 4.5 demonstrate that all metrics are worse without CFG, indicating that the CFG training strategy improves the image generation quality.

Evaluation of Different Loss Guidance As described in Sec. 4.3.3, we introduced an auxiliary loss, denoted as L_{CAP} for CC3M training. To assess the impact of this loss and determine if the single caption loss alone can generate high-quality images like GILL, we trained our model without the caption loss L_{CAP} (alignment between the mapped generative voken features and the caption features from stable diffusion text encoder) and the conditional latent diffusion loss L_{LDM} (alignment between the mapped

Table 4.6.: Influence of prompts for image generation on CLIP-I metrics on VIST. We establish four distinct conditions for the final-step image generation: 'No Context' (solely the last step's narration), 'Text Context' (inclusive of historical textual narrations), 'Image Context' (inclusive of historical images), and 'Image-Text Context' (inclusive of both historical images and narrations). From the results, MiniGPT-5 can generate more coherent images.

Model	No Context	Text Context	Image Context	Image-Text Context
SD 2 [5] (Zero-shot)	0.57	0.59	-	-
GILL [125] (Zero-shot)	0.54	0.54	0.55	0.54
MiniGPT-5 (Zero-shot)	0.54	0.57	0.57	0.57
Fine-tuned SD 2	0.59	0.61	-	-
Two-stage Baseline	0.54	0.56	0.57	0.58
MiniGPT-5 (Prefix Tuning)	0.60	0.63	0.68	0.70
MiniGPT-5 (LoRA)	**0.61**	**0.64**	**0.69**	**0.70**

generative voken features and conditional features for latent diffusion process of ground truth images) separately. The results, as shown in Table 4.5, indicate that the caption loss significantly aids in generating better images, and the voken alignment loss further enhances coherence and image quality performance.

Influence of Input Types for Image Generation To assess the impact of various types of input data for image generation, models are tasked with generating the final-step images based on specific prompts and comparing them with ground truth images by CLIP-I metric. All models are fine-tuned on data with full multimodal context and tested on various input types. As indicated in Table 4.6, the MiniGPT-5 model exhibits exceptional proficiency in producing semantically precise images compared to other models. Furthermore, we observed increased CLIP similarities when more information was provided in the input, signifying the models' enhanced ability to process diverse, long-horizon multimodal inputs.

Influence of Model Designs for Image Generation To validate our feature mapper

Table 4.7.: Evaluation of different model designs for image generation qualities on the CC3M validation set.

Model	CLIP-I (↑)	CLIP-T (↑)	IS (↑)	FID (↓)
MiniGPT-5	**0.61**	**0.22**	28.09	31.47
MiniGPT-5 (Fixed Queries)	0.60	0.21	**28.55**	**30.56**
MiniGPT-5 (Decoder-Only)	0.58	0.20	24.74	34.88

Table 4.8.: Backbone/base-model comparison on CC3M and VIST. SD 2.1/SD3 are unimodal T2I baselines; MiniGPT-5 rows are interleaved text–image generation.

	CC3M		**VIST**	
Model	**CLIP-I** (↑)	**FID** (↓)	**CLIP-I** (↑)	**FID** (↓)
Stable Diffusion 2.1 [5]	0.64	**26.39**	0.59	393.49
Stable Diffusion 3 [151]	**0.68**	29.19	0.63	380.35
MiniGPT-5 (MiniGPT-4 + SD 2.1)	0.61	31.47	0.66	366.62
MiniGPT-5 (LLaVA-1.5 + SD 2.1)	0.62	28.96	0.65	376.58
MiniGPT-5 (Qwen2.5-VL + SD3)	0.64	29.31	**0.70**	**340.20**

encoder/decoder architecture, we tested two alternatives: a model with **Fixed Queries**, where queries are initialized but not trained, and a **Decoder-Only** model, which pads its output for compatibility with the Stable Diffusion encoder. The results of these experiments are detailed in Table 4.7. From the results of MiniGPT-5 with fixed queries, we find there exists a slight trade-off between image-text coherence and image qualities, where fixed queries can lead to higher image metrics (IS and FID) but lower CLIP similarities. Meanwhile, MiniGPT-5 consistently outperforms the Decoder-Only results in all four evaluation metrics, validating the robustness and efficacy of MiniGPT-5's transformer encoder/decoder architecture design.

Backbone and Base-Model Study We evaluate whether the interface of MiniGPT-5 depends on a particular multimodal backbone or image generator. We consider three MLLMs, MiniGPT-4 [3], LLaVA-1.5 [152], and Qwen2.5-VL [153], and two diffusion

backbones, Stable Diffusion 2.1 (SD 2.1) [5] and Stable Diffusion 3 (SD3) [151], shown in Table 4.8. On VIST, MiniGPT-5 (Qwen2.5-VL+SD3) outperforms both SD3 alone and MiniGPT-5 (MiniGPT-4+SD2.1), indicating that stronger backbones translate into better interleaved coherence through our voken-to-diffusion interface. With the generator fixed (SD 2.1), swapping MiniGPT-4 for LLaVA-1.5 yields comparable VIST results and a modest improvement on CC3M, suggesting robustness to the choice of MLLM. On CC3M, unimodal T2I baselines achieve the best single-turn extremes, while MiniGPT-5 (Qwen2.5-VL+SD3) remains competitive and uniquely supports interleaved generation; the residual gap likely stems from the added mapping from LLM states to diffusion conditions and our objective emphasizing interleaving rather than single-turn T2I. Overall, gains on VIST persist across MLLMs and improve with a stronger generator, supporting that our contributions are orthogonal to specific backbones yet benefit from newer ones.

§ 4.5. Conclusion

We introduce MiniGPT-5, designed to augment the capabilities of LLMs for multimodal generation by aligning the LLM with a pretrained text-to-image generation model. Our approach demonstrates substantial improvements. The limitation of MiniGPT-5 is that we still find the object texture is hard to maintain in the new generation. Through this work, we aspire to set a new benchmark for existing and future multimodal generative models, opening doors to applications previously deemed challenging due to the disjointed nature of existing image and text synthesis paradigms.

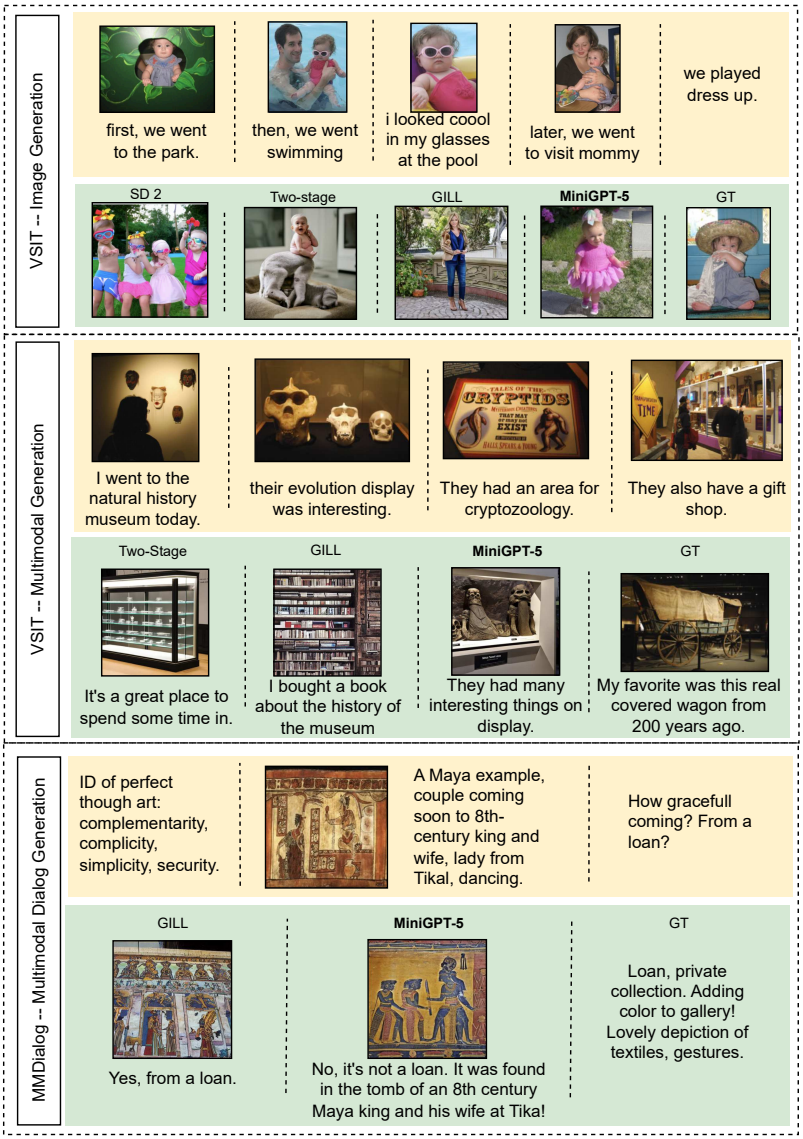


Figure 4.3.: Qualitative examples from MiniGPT-5 and baselines on the VIST and MMDialog datasets. The orange blocks indicate the input prompts, while the green blocks include model outputs. The comparisons show that MiniGPT-5 can produce coherent and high-quality multimodal output. We would like to emphasize that MiniGPT-5 does not use any caption data during fine-tuning on VIST and MMDialog, which obeys to our description-free settings. More qualitative examples can be found in the Appendix C.3.

Part III.

Generative 3D World Construction

The final part extends the composition view from 2D into 3D. Coherent 3D scene generation from a single image is harder than image generation because geometry, layout, texture, and cross-view consistency must hold simultaneously, and large-scale 3D training data is scarce. Pretrained object-centric 3D generators now produce strong isolated assets, but applying them directly to full scenes causes layout drift, resolution collapse, and texture inconsistency.

The two chapters here take complementary, training-free approaches that compose pretrained 3D, depth, and video models. SceneFuse-3D (Chapter 5) treats scene generation as a single-pass spatial composition problem: it decomposes the scene into overlapping regions, delegates each region to a pretrained object-centric generator, uses pretrained depth and detection models to set up spatial priors, and applies a masked rectified flow procedure to glue the regions into a globally coherent whole. EvoScene (Chapter 6) instead introduces an iterative interface, cycling between spatial prior initialization, pretrained 3D mesh generation, and spatial-guided novel view synthesis through a pretrained depth-conditioned video diffusion model, so that geometry and appearance progressively correct each other. Together they argue that scene-level 3D generation does not require training a new large-scale 3D backbone; it can be built by composing pretrained 2D and 3D priors under spatial and iterative interfaces, completing the composition-based view of embodied world modeling developed in this dissertation.

Chapter 5.

Constructing A 3D Scene from a Single Image

§ 5.1. Introduction

Constructing 3D scene environments serves as an essential component in simulation, robotics, digital content creation, and virtual world building. They enable scalable training for autonomous agents, immersive game environments, and rapid digital twin construction. However, constructing detailed and coherent 3D scenes typically requires either expensive 3D scanning equipment, multi-view data collection, or labor-intensive modeling. In contrast, generating 3D scenes from a single top-down image offers a lightweight and accessible alternative, making it possible to bootstrap comprehensive environments from minimal input.

Despite its appeal, generating a coherent and complex 3D scene from a single image presents several fundamental challenges. First, the synthesized scene must exhibit consistent geometry across novel views, which is difficult for pure volumetric rendering methods such as Neural Radiance Fields (NeRF) [12] or 3D Gaussian Splatting (3DGS) [13]. While these techniques excel at photorealistic appearance modeling, they

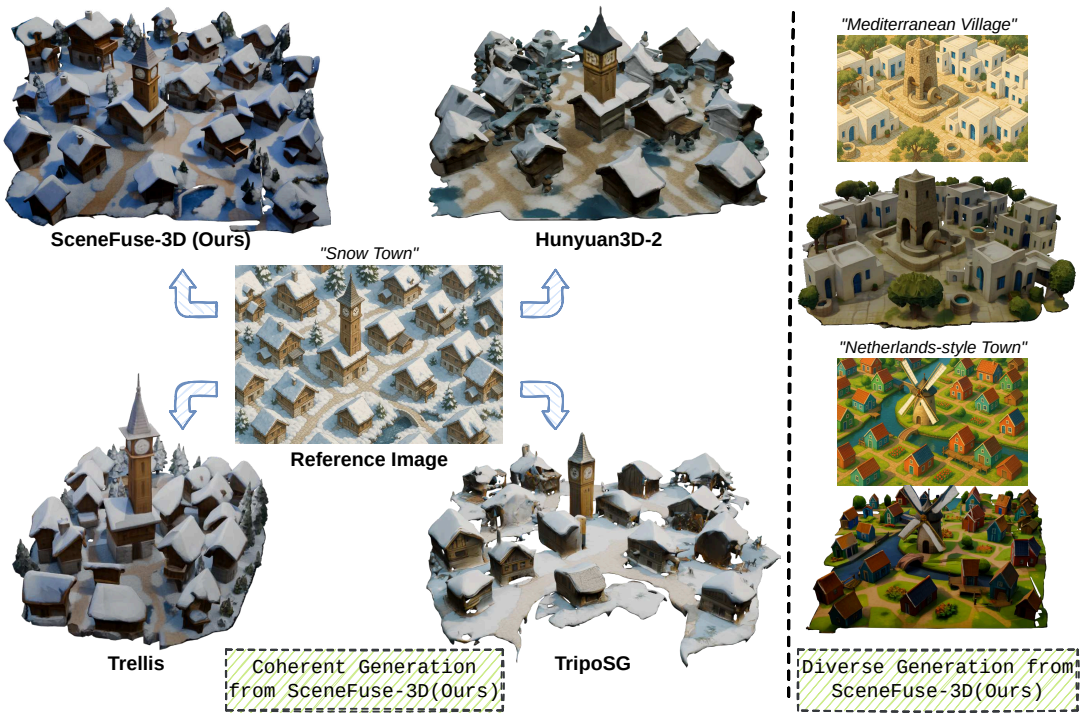


Figure 5.1.: **3D Scene Generation from a Single Image.** Given a top-down reference image (center), SceneFuse-3D generates coherent and realistic 3D scenes that preserve geometry, texture, and layout compared to other state-of-the-art end-to-end image-to-3D generation models. Our method also generalizes across diverse styles (right), producing high-quality outputs without any 3D training.

often suffer from geometry artifacts, especially in occluded or sparsely visible regions, leading to multiview inconsistencies and structural implausibility [154–157]. Second, the global layout of the generated scene must remain faithful to the input image. This is particularly challenging when considering the entire scene as a single asset and utilizing image-to-3D asset generators [6–8], which often fail to preserve the spatial relationships between elements, resulting in distorted or semantically misaligned arrangements. Third, the local fidelity of individual objects should align closely with the visual evidence in the input. Due to the resolution constraints of 3D representations and the domain shift

from object-level training to scene-level inference, previous image-to-3D generators are prone to producing low-quality meshes and misaligning textures.

To address these challenges, we propose **SceneFuse-3D**, a training-free framework for generating complex 3D scenes from a single top-down image, by enhancing the capability of image-to-3D object generators. Our method combines two core components: region-based generation and spatial-aware 3D inpainting. Each targets specific challenges in existing pipelines. We divide the scene into overlapping regions and synthesize each independently. This modular approach enables spatial upscaling and improves local alignment by grounding generation on localized image crops. To maintain global coherence and object continuity across regions, we estimate a coarse 3D structure from monocular depth and landmark detection, forming a spatial prior. A masked rectified flow mechanism then completes missing parts while preserving known content, enhancing structural consistency and object-level fidelity throughout the scene.

Through extensive experiments, we demonstrate that SceneFuse-3D generates realistic, diverse, and geometrically consistent 3D scenes from a single top-down image. Our method significantly outperforms strong baselines, including Trellis [6], Hunyuan3D-2 [7], TripoSG [8], and LGM [158], across both human preference and GPT-based evaluations. Quantitative results show notable gains in geometry quality, layout coherence, and texture fidelity, while qualitative comparisons highlight SceneFuse-3D's ability to preserve spatial structure and fine-grained detail. These results underscore the effectiveness of our modular, training-free approach to 3D scene synthesis.

Our main contributions are summarized as follows:

- We propose **SceneFuse-3D**, a training-free framework for generating structured 3D scenes from a single top-down image, leveraging pretrained object-centric

generators for zero-shot scene asset synthesis.

- We develop a modular generation strategy that combines region-wise latent synthesis with spatial-aware 3D inpainting, effectively addressing resolution bottlenecks, image-geometry misalignment, and inter-region inconsistency.
- We conduct comprehensive evaluations on diverse scenes and show that SceneFuse-3D outperforms state-of-the-art baselines in geometry quality, layout coherence, and texture realism under both human and GPT-4o-based assessments.

§ 5.2. Related Work

3D Scene Generation with 2D Generative Models Progress in 2D generative models [115, 118, 159, 160] has enabled pipelines that outpaint views and then reconstruct 3D via depth fusion or volumetric representations such as NeRF [12] and 3DGS [13]. Early work focused on indoor scenes [161–164], with later methods tackling natural scenes [154, 155, 165, 166] and improving reconstruction through stronger depth pipelines [156, 157, 167–171]. Despite compelling renders, these approaches often exhibit geometric inconsistency due to hallucinations, especially in occluded regions. Panoramic [172–176] and multiview generation [177–179] improve coverage but typically restrict camera motion and still struggle with accurate geometry. Concurrently, Syncity [180] assembles block-wise 3D generations yet produces compact layouts and does not take full scene images as input. In contrast, we directly generate geometry-consistent scene assets with arbitrary layout from a single top-down image.

3D Scene Generative Model Beyond 2D-to-3D pipelines, recent work directly generates scenes in native 3D. One line uses LLMs [181–185] or diffusion models [186–190] to predict scene layouts that are then populated with assets. These methods yield semantically plausible arrangements but are often limited to predefined categories, struggle with coherent background geometry, and can suffer inter-object collisions. A separate line models scenes directly in latent 3D spaces [191–198], e.g., BlockFusion's triplane blocks [191] and LT3SD/XCube's TUDF/voxel representations [192, 193]. While architecturally strong, these approaches require large, domain-specific 3D datasets (e.g., indoor rooms or urban layouts), limiting generalization to unseen scene types. In contrast, our method is *training-free* and modular, generating diverse 3D scene assets directly from a single top-down image.

Image-to-3D Asset Generation A parallel line of work generates *single* 3D assets from one image. Early methods combine 2D diffusion with NeRF/3DGS optimization [158, 199–203] but often trade speed for geometry fidelity. Newer approaches directly generate 3D latents via diffusion [204–208], and rectified-flow models further improve quality/efficiency [6–8, 209]. However, these methods are trained on large object-centric datasets (e.g., Objaverse-XL [210]), so applying them to full scenes faces limited 3D resolution and domain shift between objects and scenes, leading to spatial inconsistencies and layout hallucinations. We build on pretrained rectified-flow generators [6] and mitigate these issues with a *region-based* strategy plus *spatial-aware 3D inpainting*, yielding scene assets with high geometric fidelity and global layout coherence from a single top-down image.

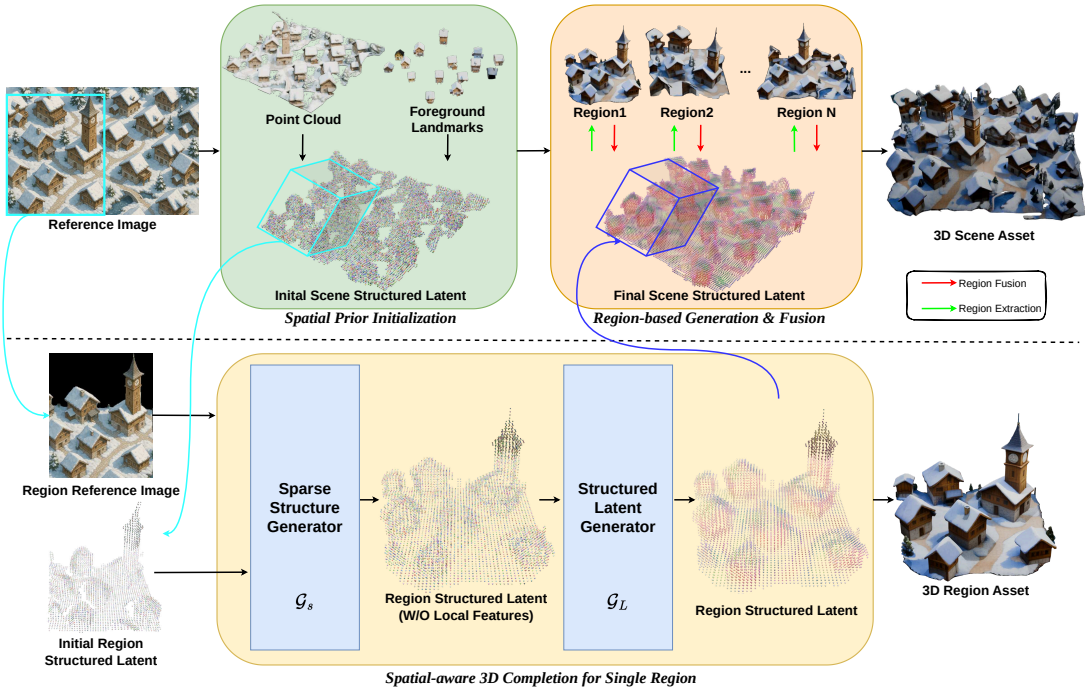


Figure 5.2.: **Overview of the SceneFuse-3D Pipeline.** Given a single top-down image, we first estimate a coarse scene structure via monocular depth and landmark extraction to initialize the scene latent (*Spatial Prior Initialization*). The scene is divided into overlapping regions for localized synthesis and progressively fused into a coherent global latent (*Region-based Generation & Fusion*). Each region is completed using a two-stage masked rectified flow pipeline with a sparse structure generator $\mathcal{G}_s$ and a structured latent generator $\mathcal{G}_L$ (*Spatial-aware 3D Completion*). The final 3D scene is decoded from the completed structured latent.

§ 5.3. Method

Given a single top-down image of an unknown scene, our objective is to synthesize a high-quality 3D scene that is geometrically and visually consistent with the input view. Figure 5.2 illustrates an overview of our pipeline.

From the scene image, we first construct the scene latents with structured latent representations (Sec. 5.3.1) from spatial priors (Sec. 5.3.2). Then, we divide the scene-level latent into region-level latents for sequential processing. For each region, we extract

the latents from the latest scene-level structured latents and take those as priors for spatial-aware 3D completion (Sec. 5.3.3) by using the base 3D generator, Trellis [6]. Later, we fuse the updated region latents to the scene latents for cross-region consistency (Sec. 5.3.4). After finishing all regions, we leverage pretrained object decoders on the complete scene structured latents to obtain a 3D scene asset. Details are explained below.

5.3.1. Structured Latent Representation

To leverage the pretrained knowledge of the base 3D generator, we construct the scene with a structured latent representation z [6]:

$$z = \{(p_i, f_i)\}_{i=1}^{L}, \quad p_i \in \{0, 1, \ldots, K-1\}^3, \quad f_i \in \mathbb{R}^C, \tag{5.3.1}$$

where p_i denotes the positional index of an active voxel in the 3D grid, and f_i represents the associated latent feature vector of dimension C. Here, K is the resolution of the voxel grid, and L is the total number of active voxels. In general, p_i captures the coarse structural layout of the object, while f_i encodes fine-grained local appearance and shape information. For the pretrained models, the resolution K should be equal to $N = 64$. To upscale the resolution for the scene, we construct scene structured latents with resolution M, where $M > N$ ($M = 2N$ by default).

For the general image-to-3D asset generation, there are two pretrained rectified flow transformers: a sparse structure generator $\mathcal{G}_s$ and a structured latent generator $\mathcal{G}_L$. At inference time, $\mathcal{G}_s$ first generates the active voxel positions $\{p_i\}^L$ from the noisy grids V_T with Gaussian noise. These positions are then used by $\mathcal{G}_L$ to generate the corresponding latent features $\{f_i\}^L$ from noisy features F_T. Both generators are conditioned on an input image condition C_I, encoded by DINOv2 [211]. Finally, the

structured latent representation z is decoded into a 3D object O using object decoders, which include different sparse 3D VAE decoders for 3D Gaussians, Radiance Fields, and mesh generation. The overall process is described as follows:

$$\{p_i\}^L = \mathcal{G}_s(V_T \mid C_I), \quad V_T \sim \mathcal{N}(0, I)^{[N,N,N]} \tag{5.3.2}$$

$$\{f_i\}^L = \mathcal{G}_L(F_T \mid C_I, \{p_i\}^L), \quad F_T \sim \mathcal{N}(0, I)^{[L,C]} \tag{5.3.3}$$

$$O = \text{ObjectDecoder}(z), \quad z = \{(p_i, f_i)\}_{i=1}^L \tag{5.3.4}$$

5.3.2. Spatial Prior Initialization

Given a top-down image, the image-to-3D generator may produce outputs in arbitrary orientations. To resolve this ambiguity and provide a consistent structural prior, we initialize the scene using point clouds. Specifically, we employ a monocular depth estimator [212] to predict a depth image and infer camera parameters, from which we construct pixel-wise point clouds. Due to occlusions, these point clouds contain missing regions, which will later be filled by the image-to-3D generator.

However, occluded areas of complex objects (e.g., buildings) may have multiple plausible completions. To enforce cross-region consistency in such cases, we propose first generating landmark objects independently and then conditioning subsequent generations on their geometry. To achieve this, we use Florence2 [213] to propose landmark bounding boxes and SAM2 [214] to extract instance masks. Each detected landmark is then processed individually using the 3D generator to obtain instance-level meshes. These meshes are aligned with the raw point clouds using ICP [215], replacing the original landmark regions with mesh-derived point clouds.

After normalization, we voxelize the aggregated scene point clouds at resolution M to

obtain the initial voxelized scene, including foreground voxels V_0^f, background voxels V_0^b, and the full scene voxel set V_0:

$$V_0 = \{V_0^b, V_0^f\} = \{p_i\}^L, \quad p_i \in \{0, 1, \ldots, M-1\}^3 \tag{5.3.5}$$

Finally, we initialize the corresponding voxel features F_0 as zeros and construct the initial structured latent representation for the scene as $z_0 = \{(V_0, F_0)\}$, shown in the top-left of Figure 5.2.

5.3.3. Region-based Generation

We adopt a region-based strategy to overcome the limitations of applying pretrained object-centric 3D generators directly to full scenes. These models are trained on single-object data, where each object occupies the full latent space at a fixed resolution N. When extended to an entire scene, this limited capacity results in low-resolution geometry and missing details. Moreover, direct scene-level generation from a single top-down image often leads to layout distortions and semantic hallucinations, as the model struggles to maintain spatial relationships or align 3D content with image cues. To address this, we divide the scene into overlapping regions and condition each on its corresponding image crop, enabling locally grounded and high-fidelity generation.

To implement this, we divide the initial scene voxel grid V_0 (of resolution M) into a set of overlapping region-level subgrids $\{V_0^{(r)}\}_{r=1}^R$, each with shape N^3, where N is the resolution used by the pretrained 3D generator. For each region r, we also extract the corresponding image crop to obtain a localized image conditioning input $C_I^{(r)}$. This ensures that the generation in each region is locally grounded in the image evidence.

Spatial-aware 3D Completion While region-based generation improves local fidelity, it introduces a new challenge: how to maintain global consistency, especially across overlapping regions. Furthermore, since the 3D generator may create assets from any orientation, we need to alleviate the misalignment between image conditions and region latents to enable further region fusion. To address these challenges, we draw inspiration from training-free inpainting methods in 2D diffusion models, such as RePaint [216], and adapt a similar approach for 3D generation. We propose to use a masked rectified flow pipeline for 3D completion, which treats the partially completed global scene latent as a constraint and performs conditional generation over the current region by completing only the unknown parts.

Given a region-level subgrids $V_0^{(r)}$, we designate known active voxels as positions $\{p_{i,0}^{(r)}\}^{L_{r,0}}$ with corresponding latent features $\{f_{i,0}^{(r)}\}^{L_{r,0}}$. For the coarse structure generation, we define a binary mask $m_s^{(r)}$ where inactive voxels are marked for regeneration. We use the sparse structure generator $\mathcal{G}_s$ to complete the region structure and obtain active voxel positions $\{p_i^{(r)}\}^{L_r}$. Next, for the fine-grained local features generation, we retain original features for positions overlapping with known voxels; otherwise, features are initialized with Gaussian noise. A second binary mask $m_L^{(r)}$ identifies unknown features for regeneration. We then use the structured latent generator $\mathcal{G}_L$ to obtain the inpainted local features $\{f_i^{(r)}\}^{L_r}$. Finally, we can construct the region structured latent $z^{(r)} = \{(p_i^{(r)}, f_i^{(r)})\}^{L_r}$. These two completions both leverage the masked rectified flow pipeline (Details in the next paragraph). The whole process is shown at the bottom of

Figure 5.2 and can be formally written as follows:

$$\{p_i^{(r)}\}^{L_r} = \mathcal{G}_s(V_T \,|C_I^{(r)}, \{p_{i,0}^{(r)}\}^{L_{r,0}}, m_s^{(r)}), \quad V_T \sim \mathcal{N}(0, I)^{[N,N,N]} \tag{5.3.6}$$

$$\{f_i^{(r)}\}^{L_r} = \mathcal{G}_L(F_T \,|C_I^{(r)}, \{p_i^{(r)}\}^{L_r}, \{f_{i,0}^{(r)}\}^{L_{r,0}}, m_L^{(r)}), \quad F_T \sim \mathcal{N}(0, I)^{[L_r,C]} \tag{5.3.7}$$

$$z^{(r)} = \{(p_i^{(r)}, f_i^{(r)})\}^{L_r} \tag{5.3.8}$$

Masked Rectified Flow for Completion We adopt a masked generation strategy based on rectified flow to complete the unknown regions of a structured 3D latent. Let x_{known} denote the known latent values to preserve, and let $m \in \{0, 1\}$ be a binary mask that indicates which parts of the latent should be regenerated ($m = 1$) and which should remain fixed ($m = 0$).

We initialize the latent variable $x_T \sim \mathcal{N}(0, I)$ with Gaussian noise, representing the unknown region at the final time step. For each timestep $t = T, T-1, \ldots, 1$, we perform U resampling steps to improve stability and smoothness [216]. At each resampling iteration u, we first compute the flow field $v_\theta(x_t, t)$ using the rectified flow model and apply an Euler update to obtain the intermediate latent:

$$x_{t_{\text{prev}}} = x_t - \Delta t \cdot v_\theta(x_t, t), \quad \text{where } \Delta t = 1. \tag{5.3.9}$$

We then re-noise the known region using a forward noise operator:

$$\texttt{forward\_step}(x, t) = (1 - t) \cdot x + [\sigma_{\min} + (1 - \sigma_{\min}) \cdot t] \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{5.3.10}$$

and merge it back into the latent using the mask m:

$$x_{t_{\text{prev}}} \leftarrow m \odot x_{t_{\text{prev}}} + (1 - m) \odot \texttt{forward\_step}(x_{\text{known}}, t_{\text{prev}}). \tag{5.3.11}$$

σ_{min} denotes the minimum noise scale used by the pretrained rectified model.

If $t > 1$ and the latent needs to be resampled, we apply additional forward noise to the merged latent: $x_t \leftarrow \texttt{forward\_step}(x_{t_{\text{prev}}}, \Delta t)$.

Otherwise, we simply continue with $x_t \leftarrow x_{t_{\text{prev}}}$. This masked rectified flow process iterates until $t = 0$, at which point the completed latent x_0 is returned. The full procedure is outlined in Algorithm 5 in Appendix D.2.

5.3.4. Region Fusion

For each generated region, we update the scene-level structured latent z_0 by replacing the corresponding part with the region-level latent $z^{(r)}$. Because regions are extracted using a patchification strategy, some may contain only partial observations of foreground landmarks. To preserve landmark integrity, we discard those structured latents corresponding to partial foregrounds during fusion.

Each region is extracted from the latest version of the scene-level latent, ensuring consistency across regions. If a region overlaps with previously generated ones, its overlapping voxels are constrained to match the existing content during generation. This enforces continuity and avoids inconsistencies in overlapping areas, leading to smooth transitions between adjacent regions while preserving already synthesized content.

Once all regions have been processed, the final scene-level latent is decoded using object decoders to produce scene-level meshes and 3D Gaussians. The complete textured scene is then rendered using a combination of physically based rendering (PBR) baking and Gaussian Splatting. Additional implementation details, including the patchification strategy, are provided in Appendix D.2.

§ 5.4. Experiments

5.4.1. Experimental Setup

Benchmark To the best of our knowledge, there is no established benchmark for 3D outdoor scene mesh generation from single images. Therefore, we construct a custom test set by prompting GPT-4o [217] to generate 100 diverse top-down scene images in a variety of styles, such as "snow village," "desert town," and more. The generation details can be found in Appendix D.3.

Metrics Without ground-truth meshes, we evaluate pairwise model comparisons per reference image on three criteria, *geometry quality*, *layout coherence*, and *texture coherence*, assessing detail fidelity, spatial arrangement, and texture–image alignment, respectively. Each pair is annotated by two AMT workers to yield a **human-preference win rate**. We also report a **GPT-4o weighted win rate**: given the same pair, GPT-4o outputs token probabilities for "A/B"; we use the probability of the chosen option as a soft vote and average over pairs. All scenes are rendered to RGB images with identical Blender settings; further GPT prompt details appear in Appendix D.3. Additionally, we report *rendered-view* image metrics between each method's render and its reference image: **CLIP** similarity (`openai/clip-vit-base-patch32`, higher is better) and **FID** (`Inception-V3`, lower is better), averaged over the test set; these complement the human/GPT criteria by measuring image-level fidelity.

Baselines We compare SceneFuse-3D with four image-to-3D generation methods: *Trellis* [6], *Hunyuan3D-2* [7], *TripoSG* [8], and *LGM* [158]. While *Trellis*, *Hunyuan3D-2* and *TripoSG* represent the state-of-the-art end-to-end 3D transformer models, *LGM* represents the multi-view generation-based 3D generator. All models are evaluated in

Table 5.1.: **Quantitative comparisons of SceneFuse-3D and baselines.** We report human preference win rates and GPT-4o-based weighted win rates (%) across geometry, layout, and texture quality. SceneFuse-3D consistently outperforms all baselines by large margins in both evaluations.

	Human Preference Win Rate (Percentage)			GPT-4o-based Weighted Win Rate (Percentage)		
Models	Geometry Quality	Layout Coherence	Texture Coherence	Geometry Quality	Layout Coherence	Texture Coherence
Trellis [6]	31.50%	30.00%	33.00%	17.58%	19.04%	14.75%
SceneFuse-3D (Ours)	**68.50%**	**70.00%**	**67.00%**	**82.42%**	**80.96%**	**85.25%**
Hunyuan3D-2 [7]	33.50%	33.50%	31.50%	11.66%	12.13%	7.67%
SceneFuse-3D (Ours)	**66.50%**	**66.50%**	**88.34%**	**88.34%**	**87.87%**	**92.33%**
TripoSG [8]	22.50%	23.00%	26.00%	24.60%	22.34%	22.79%
SceneFuse-3D (Ours)	**77.50%**	**77.00%**	**74.00%**	**75.40%**	**77.66%**	**77.21%**
LGM [158]	0.00%	0.00%	0.00%	0.60%	0.00%	0.00%
SceneFuse-3D (Ours)	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**	**100.00%**

a zero-shot setting using official pretrained checkpoints. To ensure fair comparison, background removal is disabled, and the full input image is encoded for all methods. We also experimented with the progressive novel-view method *WonderWorld* [218]. As it does not yield a consistent editable mesh, it is not included in our mesh-based quantitative tables; we provide qualitative comparisons and discussion in Appendix D.1.2.

5.4.2. Main Results

Table 5.1, 5.2 and Figure 5.3 present the main quantitative and qualitative comparisons between SceneFuse-3D and baseline methods. The results clearly demonstrate that SceneFuse-3D consistently outperforms Trellis [6], Hunyuan3D-2 [7], TripoSG [8], and LGM [158] across geometry, layout, and texture quality, as evaluated by both human annotators and GPT-4o. These improvements can be attributed to our region-based design and spatially guided generation strategy, which together promote better alignment between image features and 3D content, while preserving scene-wide consistency.

Figure 5.3.: **Qualitative comparisons between SceneFuse-3D and baselines.** Given a single top-down image (left column), we compare 3D scene outputs generated by SceneFuse-3D, Trellis [6], Hunyuan3D-2 [7], TripoSG [8], and LGM [158]. SceneFuse-3D consistently produces globally coherent scenes with fine-grained geometry, accurate object layouts, and realistic textures across a variety of styles and environments. In contrast, Trellis often produces oversimplified geometry; Hunyuan3D-2 suffers from structural inconsistencies and domain mismatch; TripoSG exhibits repetition artifacts and layout drift; LGM cannot generate consistent multi-view scene images for 3D construction.

Quantitative Analysis SceneFuse-3D's region-wise decomposition aligns each latent block with a localized image crop, reducing the domain gap between object-centric training and scene-level inference. This design boosts texture fidelity, with GPT-4o assigning a 92.3% win rate versus only 7.7% for Hunyuan3D-2. The upscaled resolution further enhances structural detail, reflected in geometry improvements of +37 points over Trellis (68.5% vs. 31.5%) and +55 points over TripoSG (77.5% vs. 22.5%). Spatial priors and masked 3D inpainting stabilize layouts and smooth inter-region transitions, yielding

higher layout coherence (70.0% vs. 30.0% for Trellis in human study; 87.9% vs. 12.1% for Hunyuan3D-2 in GPT-4o evaluation). LGM, leveraging multi-view image generation, fails to provide consistent geometry in this setting and collapses to oversimplified outputs.

Rendered-view metrics in Table 5.2 support these findings. SceneFuse-3D achieves the highest CLIP similarity (0.8030), indicating stronger semantic and visual alignment with the input than all baselines. It also records the best FID (258.74), showing the closest distributional match in image structure, while Trellis follows at 288.23, and the others exceed 300. Together, these results confirm that SceneFuse-3D not only excels in human and GPT-4o preference studies but also in reference-based image metrics, underscoring its advantages in both semantic fidelity and distributional realism.

Qualitative Analysis Qualitatively, SceneFuse-3D produces scene assets with clear structure, consistent layout, and fine-grained surface details that closely match the reference top-down image. In contrast, Trellis often generates overly centralized, low-resolution structures and lacks peripheral detail. Hunyuan3D-2 exhibits notable issues with layout distortion and geometry hallucinations despite acceptable textures in isolated parts. TripoSG maintains some compositional structure but frequently introduces repeated objects and ignores the layout evidence within the reference image. LGM can only produce a hollow cube with the scene texture mapped onto its faces. SceneFuse-3D's region-wise generation and spatial inpainting pipeline helps it avoid these artifacts while maintaining both global coherence and local fidelity.

These findings confirm that spatial decomposition and prior-guided inpainting are effective principles for lifting single-view image inputs into coherent, high-quality 3D scenes. Additional qualitative comparisons are available in Figure D.1 in Appendix D.1.1.

Table 5.2.: **Rendered-view metrics on the reference images.** CLIP similarity and FID are computed between each method's render and the corresponding reference image.

Models	CLIP ↑	FID ↓
SceneFuse-3D (Ours)	**0.8030**	**258.74**
Trellis [6]	0.7760	288.23
Hunyuan3D-2 [7]	0.7716	302.26
TripoSG [8]	0.7203	353.78
LGM [158]	0.7510	353.86

Table 5.3.: **Ablation Study Results.** Win rates for geometry, layout, and texture show that removing region-based generation or landmark conditioning degrades performance, highlighting the importance of both components in SceneFuse-3D.

	Human Preference Win Rate (Percentage)			GPT-4o-based Weighted Win Rate (Percentage)		
Models	Geometry Quality	Layout Coherence	Texture Coherence	Geometry Quality	Layout Coherence	Texture Coherence
SceneFuse-3D (w/o regions)	20.00%	17.00%	13.50%	6.11%	8.48%	6.08%
SceneFuse-3D	**80.00%**	**83.00%**	**86.50%**	**92.89%**	**91.52%**	**92.92%**
SceneFuse-3D (w/o landmarks)	41.50%	46.00%	42.00%	36.90%	44.21%	38.26%
SceneFuse-3D	**59.50%**	**54.00%**	**58.00%**	**64.10%**	**56.79%**	**61.74%**

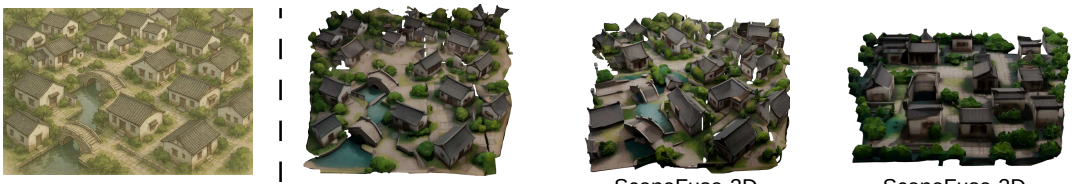


Figure 5.4.: **Qualitative Ablation Results.** Left: Reference image. Middle: SceneFuse-3D without landmark conditioning. Right: SceneFuse-3D without region-based generation. Landmark conditioning ensures consistency for foreground objects, especially for objects across regions, while region-based generation preserves overall detail and coherence.

5.4.3. Ablation Study

We conduct ablation studies to evaluate the contributions of key components in SceneFuse-3D: the region-based generation strategy and the use of pre-generated landmarks. Both ablation studies still apply the spatial-aware 3D completion. Quantitative results are shown in Table 5.3, and qualitative comparisons are illustrated in Figure 5.4.

Without Region-Based Generation In this setting, the entire scene latent is directly passed into the pretrained 3D generator, without being split into localized regions. This leads to severe performance drops across all metrics. The results suggest that holistic generation fails to make full use of the pretrained model's capacity, which was originally trained on single-object inputs. Without localized conditioning, the model struggles to resolve spatial context and image-to-3D correspondence, producing low-resolution and spatially incoherent outputs. As illustrated in Figure 5.4 (right), buildings lose structural sharpness and alignment, and the overall layout becomes underspecified.

Without Landmark Conditioning Removing landmark-aware initialization (depth-only prior) degrades geometry and layout, especially around large foreground structures (e.g., gates/towers). Landmarks act as semantic and geometric anchors across patches: they fix orientation/scale and provide reliable context for masked completion. Without them, region completions drift at boundaries, yielding duplicated facades or misaligned parts across neighboring patches (Fig. 5.4, middle).

Overall, these ablations show complementary roles: *region-wise decomposition* keeps generation within the base model's effective receptive field and preserves local detail, while *landmark anchors* enforce cross-patch continuity and stabilize global structure—both are necessary for coherent, high-fidelity scenes.

§ 5.5. Conclusion

To address the challenge of generating high-quality, coherent 3D scenes from a single image, we proposed SceneFuse-3D, a training-free framework that decomposes scenes

into overlapping regions and guides generation with spatial priors. A spatial-aware 3D completion with masked rectified flow preserves local object fidelity while enforcing global coherence. Empirically, SceneFuse-3D outperforms existing methods across geometry, texture, and layout, underscoring the promise of modular, spatially grounded generation for stereotype 3D scene synthesis from minimal input.

Chapter 6.

Self-Evolving 3D Scene Generation from a Single Image

§ 6.1. Introduction

Games, films, animation, simulation, and many other digital applications rely heavily on high-quality 3D visuals. While recent video generation models can take a single image and render visuals as if a full 3D scene existed, their outputs remain purely image-space: they lack consistent geometry, exhibit viewpoint-dependent drift, and cannot be converted into editable or physically grounded 3D assets. In practice, producing such 3D assets still requires extensive manual modeling and texturing, which demands significant artistic effort and is difficult to scale. This makes an automated approach for generating complete 3D scenes from a single image highly desirable.

Despite its appeal, single-image 3D scene generation remains fundamentally challenging. Limited observations constrain coverage: methods [219–221] that synthesize novel views along a fixed camera path leave large portions of the scene unseen, leading to holes or hallucinated regions with weak geometric grounding. Ensuring high-

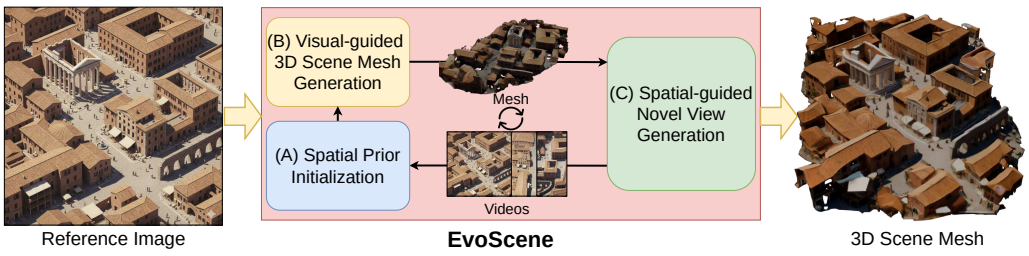


Figure 6.1.: **Self-evolving 3D scene generation from a single image.** Given a single input photograph, EvoScene progressively builds a complete 3D scene through a virtuous cycle where geometry and appearance mutually refine each other. Our method synthesizes photorealistic novel views and produces high-quality 3D representations (Gaussians and meshes) with substantially improved angular coverage and completeness. Project page: https://eric-ai-lab.github.io/evoscene.github.io/

resolution, globally consistent textures adds further difficulty. Existing image-to-3D generators [6–8, 222], predominantly trained on object-level data, struggle when scaled to complex scenes, producing coarse geometry and misaligned or blurry textures. Extensions [180, 223, 224] that operate on overlapping latent patches expand spatial coverage but still suffer from texture seams and inconsistent fine details. End-to-end scene generation models [191, 192] attempt a direct mapping from images to 3D scenes, but the scarcity of diverse, high-quality scene datasets restricts them to narrow domains and limits their general applicability.

To address these challenges, we propose a self-evolving, iterative pipeline, **EvoScene**, that progressively improves the 3D scene rather than attempting to recover everything from a single pass, shown in Figure 6.1. EvoScene operates by cycling through three stages: In Stage A, Spatial Prior Initialization, EvoScene builds a spatial prior, a coarse geometric representation of the scene, by estimating depth from the 2D scene images (initially the input image) and fusing it into a point cloud. This step provides an initial understanding of the scene's layout and major structures. In Stage B, Visual-guided 3D

Scene Mesh Generation, the point-cloud prior is lifted into an explicit 3D mesh using an existing 3D generation model. This mesh captures key surfaces and topology, forming a stable geometric scaffold with editable textures. In Stage C, Spatial-guided Novel View Generation, the reconstructed mesh guides a video generation model to synthesize new multi-view images that reveal unseen regions and supply additional structure and appearance cues. These new views are then fed back into the next iteration, allowing geometry and texture to improve step by step.

The entire pipeline is training-free, combining complementary knowledge already learned by existing models through its stages. The stages that lift 2D observations into 3D structure (Stages A and B) leverage the geometric reasoning of depth estimation and a 3D generation model to produce a consistent scene-level mesh. The stage that converts this mesh into synthesized multi-view imagery (Stage C) draws on the visual and textural knowledge of a video generation model to achieve detailed, view-consistent appearances beyond mesh-resolution limits. Together, these stages provide wide viewpoint coverage, reliable structure, and high-quality textures in a 3D mesh representation ready for downstream use.

Through extensive experiments across diverse scenes, we demonstrate that EvoScene produces complete textured 3D scene meshes with superior geometric quality, spatial layout coherence, and photorealistic appearance compared to state-of-the-art image-to-3D baselines, achieving human preference win rates of 78.5%-90.5% across all evaluation criteria. Ablations validate the importance of our core design choices, confirming that both iterative geometry-appearance co-evolution and depth-conditioned video guidance are essential for achieving high-quality scene generation. Both quantitative metrics and qualitative comparisons highlight EvoScene's ability to generate geometrically consistent

scenes with fine-grained details across varying complexity.

Our main contributions are:

- We propose **EvoScene**, a self-evolving framework for single-image 3D scene generation that progressively expands spatial coverage and refines complete textured meshes through iterative cycles between geometry and appearance across 2D and 3D domains.
- We introduce a three-stage modular design that synergistically combines point-cloud spatial priors, SLAT-based 3D diffusion for mesh generation, and depth-conditioned video generation, enabling effective integration of geometric knowledge from 3D models with visual priors from video models.
- Comprehensive experiments with human preference and GPT-4o-based evaluations on diverse scenes demonstrate that EvoScene achieves superior geometric completeness, layout coherence, and texture fidelity compared to state-of-the-art image-to-3D baselines, with ablations confirming substantial improvements over pose-only world-model variants.

§ 6.2. Related Work

Image-to-3D Scene Generation While object-level image-to-3D methods [6–8, 158, 199–203, 222] achieve impressive reconstruction quality, they struggle when scaled to complex scenes due to limited spatial coverage. Scene-level generation requires handling larger spatial extents and maintaining global coherence. Neural scene representations like NeRF [12] and 3D Gaussians [13] provide powerful rendering frameworks. Recent approaches leverage 2D diffusion models [115, 118, 159, 160] to outpaint novel views

and reconstruct via depth fusion or volumetric representations. Early work focused on indoor scenes [161–164], with later methods tackling natural scenes [154–157, 165–171]. Panoramic [172–176] and multi-view generation [177–179] improve coverage but typically restrict camera motion and struggle with geometric accuracy due to hallucinations in occluded regions. Alternative approaches directly generate scenes in native 3D through layout prediction [181–190] or latent 3D spaces [191–196], but often require large domain-specific datasets and struggle with generalization. There are some recent works [180, 209, 223–225] that apply object-level 3D generators to scene-level mesh synthesis without training. Critically, while these methods achieve impressive single-pass results, they lack mechanisms to progressively refine and expand coverage through iterative multi-view observations, leading to incomplete spatial coverage and limited detail in initially unseen areas. Our self-evolving approach addresses this through geometry-appearance co-evolution across multiple iterations.

Video Generation for 3D Scene Reconstruction Video diffusion models [226–232] capture powerful visual priors for generating photorealistic imagery with temporal coherence. World models extend this to novel view synthesis by conditioning on camera trajectories [179, 233–235], enabling controllable viewpoint generation. Video generation with 3D priors [219–221, 236, 237] further incorporates depth or geometry conditioning to improve multi-view consistency. However, these approaches face key limitations: world models with initial depth conditioning [219–221] are constrained by predefined camera trajectories that limit full scene coverage, and lack mechanisms to accumulate scene knowledge for iterative refinement. Mesh texturing methods [236, 237] require pre-existing complete 3D meshes rather than performing reconstruction. Unlike these approaches that use video generation as a final rendering step, we leverage

depth-conditioned video diffusion within an iterative reconstruction framework. Our method progressively refines and expands scene geometry through generated multi-view observations, accumulating spatial knowledge across iterations to achieve complete coverage. As our ablations demonstrate, this strategy substantially outperforms pose-only world models, producing geometrically consistent results suitable for 3D mesh reconstruction.

Iterative and Self-Evolving Methods Iterative refinement is widely used in 3D generation. Multi-stage pipelines separate geometry and texture generation [238–241], while iterative optimization methods alternate between shape and appearance [242, 243]. Recent work explicitly closes the loop between 2D and 3D domains: Ouroboros3D [244] integrates multi-view diffusion and reconstruction in a recursive diffusion process, Cycle3D [245] alternates between 2D generation and 3D reconstruction, GeoDream [246] disentangles diffusion and cost-volume priors for iterative refinement, and GenFusion [247] leverages reconstruction-driven video diffusion. However, these methods rely primarily on multi-view generation models for geometric reasoning, lacking explicit integration of 3D generative priors that encode structural knowledge. Unlike prior methods, our self-evolving framework synergistically combines knowledge from both 3D generation models (for geometric structure) and video generation models (for photorealistic appearance): each iteration uses a 3D diffusion model to complete scene geometry, then leverages this completed mesh to guide video synthesis for texture generation, progressively refining both modalities through their iterative interaction to achieve complete scene generation from a single input image.

§ 6.3. Method

6.3.1. Overview

Given a single input image I_0, our goal is to generate a complete and view-consistent 3D scene representation. Single-image generation is challenging due to occlusions and depth ambiguity. We leverage two complementary sources of prior knowledge: pretrained *3D generation models* that encode geometric reasoning for plausible structure, and pretrained *video generation models* that capture visual knowledge for realistic textures. Neither alone is sufficient, as geometry-based approaches struggle with high-fidelity textures, while video-based approaches lack explicit 3D consistency.

EvoScene addresses this through a self-evolving framework with three coupled stages forming a virtuous cycle (Figure 6.2). *Stage A: Spatial Prior Initialization* extracts geometric priors by estimating depth maps and back-projecting them into point clouds. *Stage B: Visual-guided 3D Scene Mesh Generation* leverages these priors with a pretrained 3D diffusion model to generate a complete mesh filling occluded regions. *Stage C: Spatial-guided Novel View Generation* uses the mesh to guide a video generation model, synthesizing photorealistic multi-view images from novel viewpoints. These newly generated views are converted back into geometric priors through depth re-estimation and fed into the next iteration. Through this cycle, geometry and appearance mutually refine each other, progressively evolving a single image into a complete 3D scene.

6.3.2. Self-Evolving 3D Scene Generation Stages

Let t index the iteration number ($t = 0, 1, \ldots, T$), S denote voxel resolution, and P denote patch size. At each iteration, we maintain three key data structures that evolve throughout

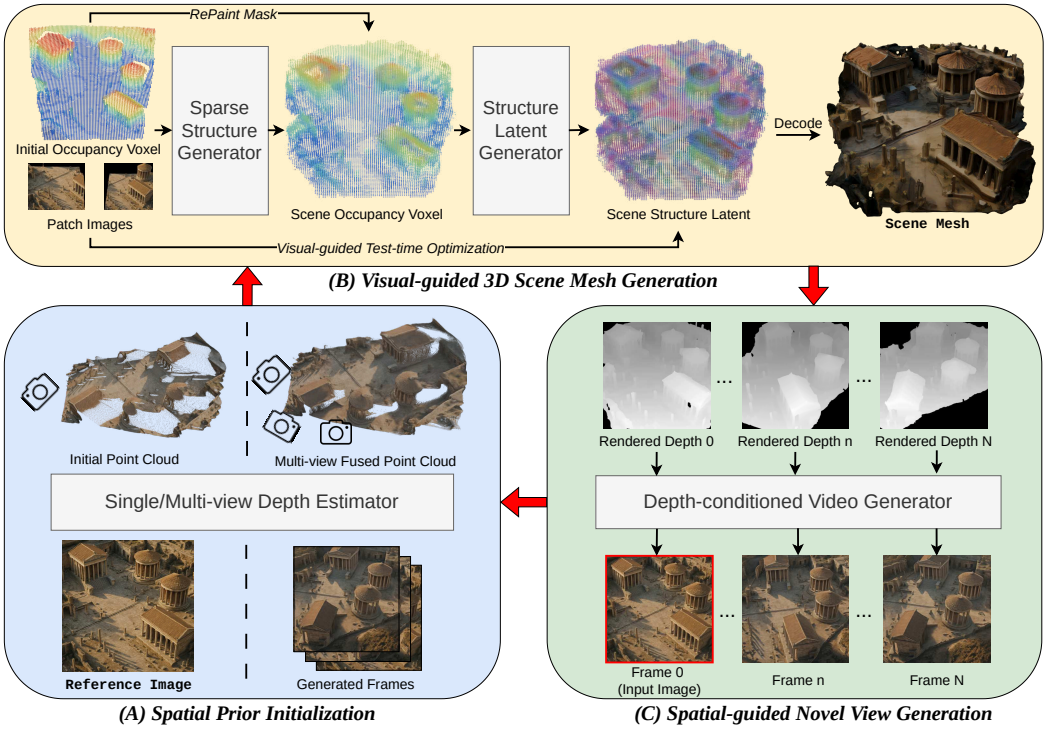


Figure 6.2.: **Overview of EvoScene.** Our self-evolving framework consists of three coupled stages that form a virtuous cycle: (A) *Spatial Prior Initialization* extracts depth from 2D observations and back-projects to 3D point clouds, providing geometric constraints. (B) *Visual-guided 3D Scene Mesh Generation* uses a 3D diffusion model with test-time rendering optimization to complete the mesh guided by point cloud priors. (C) *Spatial-guided Novel View Generation* renders depth from the mesh to guide depth-conditioned video diffusion, synthesizing photorealistic multi-view images. These new views are converted back to point clouds (via depth re-estimation and multi-view filtering) and fed into the next iteration. This cycle progressively refines geometry and appearance to produce a complete 3D scene from a single input image.

the process:

(*i*) **Multi-view observations** $\mathcal{V}_t = \{(I_k, C_k)\}_{k=1}^{N_t}$, where I_k are RGB images and C_k are camera parameters. Initially, $\mathcal{V}_0 = \{(I_0, C_0)\}$ contains only the input image. New views are added in each iteration from Stage C.

(*ii*) **Point cloud prior** $\mathcal{P}_t$ with per-point confidence scores, derived from depth estimation and multi-view filtering. This provides sparse, metric-scale geometric constraints extracted from the accumulated observations.

(*iii*) **Latent 3D representation** $\mathcal{G}_t$ encoded as SLAT [6] (Structured LATent), a sparse voxel grid where each occupied voxel stores a latent feature vector. This representation can be decoded into explicit geometries (meshes, 3D Gaussians) for rendering, optimization, and editing.

Each iteration consists of three coupled stages that transform data between these representations:

(A) Spatial Prior Initialization. To build a 3D geometric understanding from 2D observations, we extract depth information and lift it into 3D space. In the first iteration ($t = 0$), we have only the input image I_0. We apply a pretrained monocular depth estimator to obtain a depth map d_0. Using the estimated camera intrinsics K_0 and extrinsics E_0, we back-project the depth map to obtain an initial point cloud:

$$\mathcal{P}_0 = \text{BackProject}(d_0, K_0, E_0). \tag{6.3.1}$$

Each point is assigned a confidence score based on local depth gradient magnitude, reflecting the reliability of the depth estimation. Points in regions with smooth depth transitions receive higher confidence, while those near depth discontinuities or uncertain

areas receive lower confidence.

In subsequent iterations ($t > 0$), we have access to multi-view observations $\mathcal{V}_t = \{(I_k, C_k)\}_{k=1}^{N_t}$ accumulated from previous cycles. For each view, we re-estimate depth maps $\{d_k\}$ and back-project them into point clouds. To ensure geometric consistency and filter out erroneous depth estimates, we apply multi-view geometric voting: for each candidate 3D point, we project it into all available views and check depth agreement. Only points with sufficient cross-view support are retained. The filtered points from all views are merged to form the updated point cloud:

$$\mathcal{P}_t = \mathcal{P}_{t-1} \cup \text{Filter}\Big(\bigcup_{k=1}^{N_t} \text{BackProject}(d_k, K_k, E_k) \Big), \tag{6.3.2}$$

where Filter$(\cdot)$ denotes the multi-view voting procedure that removes outliers based on cross-view depth consistency.

This accumulated point cloud $\mathcal{P}_t$ serves as the spatial prior for the next stage, providing metric-scale geometric constraints that anchor the 3D generation process.

(B) Visual-guided 3D Scene Mesh Generation. While the point cloud $\mathcal{P}_t$ from Stage A provides sparse geometric constraints, it remains incomplete due to occlusions and limited viewpoint coverage. Large regions of the scene are unobserved, and the point cloud lacks surface topology. To address this, we leverage a pretrained 3D diffusion model to complete the geometry and generate a full 3D mesh representation.

We voxelize the point cloud $\mathcal{P}_t$ at resolution S^3 to obtain a sparse occupancy grid $O_t \in \{0, 1, 2\}^{S^3}$, where values indicate observed regions (from $\mathcal{P}_t$), carved free-space (ray visibility), and unknown occluded regions to be hallucinated. To handle large scenes, we decompose the grid into overlapping P^3 patches $\{\mathcal{B}_j\}$ and extract corresponding

image crops $\{I_j\}$ from accumulated observations.

We employ a two-stage flow-matching diffusion generator based on SLAT [6]: We first use a sparse structure generator to denoise a binary occupancy latent Z_t^{SS} with RePaint-style [216] masking: observed voxels are re-injected from the ground truth encoding while unknown voxels evolve freely, conditioned on image crops $\{I_j\}$:

$$\frac{d}{dt}Z_t^{\mathrm{SS}} = v^{\mathrm{SS}}(Z_t^{\mathrm{SS}}, \{I_j\}, t), \quad \text{s.t. } Z_t^{\mathrm{SS}}|_{O_t=2} = \mathcal{E}(O_t), \tag{6.3.3}$$

where $\mathcal{E}$ encodes the occupancy grid and v^{SS} is the learned velocity field. This yields a completed occupancy $\hat{O}_0$ filling occluded regions while preserving observed structure.

Next, we employ a structured latent generator to generate latent features $\{\mathbf{z}_i\}$ for each occupied voxel, given $\hat{O}_0$ and image crops $\{I_j\}$, by denoising Z_t^{SLAT}. Inspired by Extend3D [225], we apply test-time optimization using multi-view rendering feedback. We periodically decode the current SLAT into 3D Gaussians, render from all accumulated views $\mathcal{V}_t$, and minimize the multi-view reconstruction loss:

$$\begin{aligned}\mathcal{L}_{\mathrm{mv}} = \sum_{k=1}^{N_t} \Big[&\lambda_1 \|\Pi(\mathcal{G}_t; C_k) - I_k\|_1 + \lambda_2 \mathrm{LPIPS}(\Pi(\mathcal{G}_t; C_k), I_k) \\ &+ \lambda_3 (1 - \mathrm{SSIM}(\Pi(\mathcal{G}_t; C_k), I_k)) \Big],\end{aligned} \tag{6.3.4}$$

where $\Pi(\cdot; C_k)$ denotes the differentiable rendering operator and $\lambda_1, \lambda_2, \lambda_3$ are loss weights (1 by default). The gradient $\nabla_{Z_t^{\mathrm{SLAT}}} \mathcal{L}_{\mathrm{mv}}$ guides denoising toward photometrically consistent geometry.

The output $\mathcal{G}_t$ (encoded as SLAT) can be decoded into 3D Gaussians for differentiable rendering and an explicit mesh (via Marching Cubes) for editing, providing a scaffold for the next stage.

(C) Spatial-guided Novel View Generation. While Stage B produces a geometrically complete mesh, its appearance quality is limited by the 3D diffusion model's training data and resolution. To overcome this and expand viewpoint coverage, we leverage the mesh $\mathcal{G}_t$ as a spatial scaffold to guide a pretrained video diffusion model in synthesizing photorealistic multi-view images.

We render N novel viewpoints along an orbital camera trajectory, obtaining depth maps $\{D^{(k)}\}_{k=1}^{N}$ from the mesh geometry. These depth maps encode the 3D spatial layout, providing structural guidance for video generation. We feed them to a pretrained depth-conditioned video diffusion model with the input image I_0 as the first frame and an optional text prompt p:

$$\mathcal{V}_{t+1} = \text{VideoDiffusion}\big(I_0, \{D^{(k)}\}_{k=1}^{N}, p\big). \tag{6.3.5}$$

The model generates temporally coherent frames $\mathcal{V}_{t+1} = \{I_k^{\text{new}}\}_{k=1}^{N}$ that are photorealistic and geometrically consistent, hallucinating plausible textures for previously occluded regions.

These synthesized images serve dual purposes: (*i*) they provide appearance supervision for refining mesh texture, and (*ii*) they are fed back into Stage A of the next iteration, where depth re-estimation converts them into geometric priors. This closes the self-evolving loop, enabling geometry and appearance to mutually refine each other across iterations.

Self-evolution schedule. The three stages form a *self-evolving cycle* where geometry and appearance co-evolve through iterative refinement. The cycle operates as follows:

- **Stage A → B:** Sparse point cloud priors (from depth estimation) provide geometric constraints that anchor the 3D diffusion model, ensuring the completed mesh

Table 6.1.: **Quantitative comparisons of EvoScene and baselines.** We report human preference win rates and GPT-4o-based weighted win rates (%) across geometry, layout, and texture quality. EvoScene consistently outperforms all baselines by large margins in both evaluations.

	Human Preference Win Rate (Percentage)			GPT-4o-based Weighted Win Rate (Percentage)		
Models	Geometry Quality	Layout Coherence	Texture Coherence	Geometry Quality	Layout Coherence	Texture Coherence
Trellis [6]	19.50%	16.00%	20.00%	9.47%	6.62%	7.75%
EvoScene (Ours)	**80.50%**	**84.00%**	**80.00%**	**90.53%**	**93.38%**	**92.25%**
Hunyuan3D-2.1 [7]	16.50%	11.50%	15.50%	9.72%	5.13%	6.42%
EvoScene (Ours)	**83.50%**	**88.50%**	**84.50%**	**90.28%**	**94.87%**	**93.58%**
TripoSG [8]	15.00%	9.50%	14.00%	9.60%	6.12%	9.28%
EvoScene (Ours)	**85.00%**	**90.50%**	**86.00%**	**90.40%**	**92.88%**	**90.72%**

preserves observed structure.

- **Stage B → C:** The completed mesh provides spatial guidance (via rendered depth maps) that steers video generation toward geometrically consistent, photorealistic appearance.
- **Stage C → A (next iteration):** Synthesized multi-view images are converted back into geometric priors through depth re-estimation and multi-view filtering, providing richer constraints for the next iteration's mesh completion.

We iterate this cycle T times ($t = 1, \ldots, T$). To maximize scene coverage, each iteration generates videos from complementary camera trajectories (for example, alternating between positive and negative azimuth angles, e.g., $+30^\circ$ in iteration 1, -30° in iteration 2, or varying elevation angles). This strategy progressively expands angular coverage, revealing previously unseen regions and refining both geometry and appearance with each cycle. As iterations progress, the accumulated multi-view observations $\mathcal{V}_t$ grow richer, the point cloud $\mathcal{P}_t$ becomes denser, and the mesh quality improves, ultimately converging to a complete, photorealistic 3D scene.

Mesh extraction. After the final iteration, we decode $\mathcal{G}_T$ into both a 3D Gaussian representation and a mesh (via Marching Cubes on the occupancy grid). We bake multi-view textures onto the mesh using the accumulated observations and export as a GLB file for rendering and editing.

6.3.3. Implementation Details

We employ pretrained models for all components: HunyuanWorld-Mirror [248] for single/multi-view 3D reconstruction, Trellis [6] for SLAT-based 3D mesh generation, and Wan2.2-Fun-5B-Control [226] for depth-conditioned video generation. We set voxel resolution $S = 128$ with patch size $P = 64$, and render video frames at 1024×1024 resolution with 121 frames. We run $T = 3$ self-evolution iterations, generating videos from complementary azimuth angles (e.g., $+30°$ and $-30°$ with uniform view sampling) to progressively expand angular coverage. Every iteration requires 6 minutes with the highest 68G VRAM on one H100 GPU. All models are used off-the-shelf without fine-tuning. For more method details, please refer to Appendix E.1.

§ 6.4. Experiments

6.4.1. Experimental Setup

Benchmark and Metrics. We evaluate on 100 diverse scene images generated by GPT-4o [217] and GPT-Image-1 [249]. We conduct pairwise comparisons on three criteria: geometry quality, layout coherence, and texture coherence, annotated by two AMT workers (**human preference win rate**) for each pair and GPT-4o with token

Table 6.2.: **Rendered-view metrics on the reference images.** CLIP similarity and FID are computed between each method's render and the corresponding reference image.

Models	CLIP ↑	FID ↓
EvoScene (Ours)	**0.8643**	**188.68**
Trellis [6]	0.7454	280.67
Hunyuan3D-2.1 [7]	0.7312	322.53
TripoSG [8]	0.7152	366.34

Table 6.3.: **Ablation Study Results.** Human preference and GPT-4o win rates demonstrate the importance of iterative refinement and depth-conditioned video guidance in EvoScene.

	Human Preference Win Rate (Percentage)			GPT-4o-based Weighted Win Rate (Percentage)		
Models	Geometry Quality	Layout Coherence	Texture Coherence	Geometry Quality	Layout Coherence	Texture Coherence
EvoScene (Single Iteration)	46.00%	43.50%	47.50%	41.78%	39.52%	47.08%
EvoScene (Full)	**54.00%**	**56.50%**	**52.50%**	**58.22%**	**60.48%**	**52.92%**
EvoScene (Pose-only Video Gen.)	6.50%	10.00%	6.00%	1.36%	0.88%	2.14%
EvoScene (Full)	**93.50%**	**90.00%**	**94.00%**	**98.64%**	**99.12%**	**97.86%**

probability weighting (**GPT-4o weighted win rate**). We also report **CLIP** similarity and **FID** between rendered views and reference images. More details can be found in Appendix E.2.

Baselines. We compare against three state-of-the-art image-to-3D models: *Trellis* [6], *Hunyuan3D-2.1* [250], and *TripoSG* [8]. All models are evaluated zero-shot with official checkpoints using complete input images without background removal.

6.4.2. Main Results

Quantitative Analysis Table 6.1 demonstrates EvoScene's substantial advantages across three state-of-the-art baselines. In human preference evaluations, EvoScene achieves win rates of 78.5%-85.0% for geometry quality, 84.0%-90.5% for layout coherence, and 84.5%-88.0% for texture coherence. The particularly strong performance in

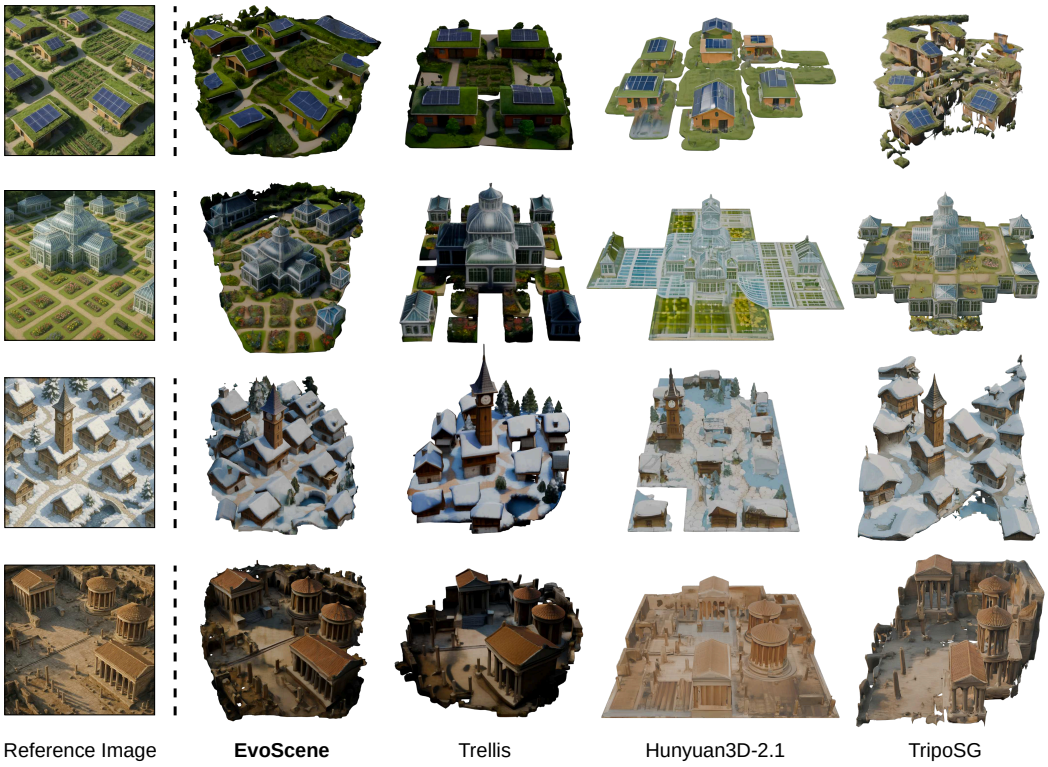


Figure 6.3.: **Qualitative comparisons across diverse scene types.** From left to right: reference images, EvoScene (ours), Trellis, Hunyuan3D-2.1, and TripoSG. Scenes include residential neighborhoods (rows 1-2), urban architecture (row 3), and historical landmarks (row 4). EvoScene produces complete geometry with preserved architectural details and photorealistic textures, while baselines exhibit fragmentation, flattened representations, or severe geometric distortions.

layout coherence indicates our iterative approach effectively preserves spatial relationships and scene structure. GPT-4o-based evaluations show even stronger advantages, with weighted win rates consistently exceeding 90% across all criteria and baselines, reaching up to 94.9% for layout coherence against Hunyuan3D-2.1. This suggests that when quality differences are weighted by magnitude, EvoScene's superiority becomes more pronounced.

Table 6.2 provides automatic metrics measuring semantic fidelity and perceptual quality. EvoScene achieves CLIP similarity of 0.8643 (15.9% improvement over Trellis) and FID of 188.68 (32.8% improvement), demonstrating superior semantic alignment and visual realism. These results validate our self-evolving approach, where iterative refinement enables geometry and appearance to mutually reinforce each other while progressively expanding scene coverage.

Qualitative Analysis Figure 6.3 provides visual comparisons across diverse scene types. For geometry quality, EvoScene produces complete, coherent meshes with well-preserved architectural details including building facades, rooftops, and terrain structures. In contrast, Trellis generates incomplete geometry with missing regions and lacks fine structural details, while TripoSG suffers from severe fragmentation where scene elements appear disconnected. Hunyuan3D-2.1 produces oversimplified geometry missing important features. For layout coherence, EvoScene maintains correct spatial relationships with proper depth ordering, while Hunyuan3D-2.1 generates flattened representations where depth relationships are compressed. For texture quality, EvoScene synthesizes photorealistic, multi-view consistent appearance through depth-conditioned video generation, exhibiting sharp details and smooth surface transitions. Baselines suffer

from various artifacts: Trellis produces blurry textures, TripoSG shows inconsistent coloring, and Hunyuan3D-2.1 lacks realism. EvoScene demonstrates consistent quality across varying scene complexity. More experiment results can be found in Appendix E.2.

6.4.3. Ablation Study

We conduct ablation studies to validate two core design choices: (1) iterative refinement versus single-iteration generation, and (2) depth-conditioned versus pose-only video generation. These ablations isolate the contributions of our self-evolving mechanism and geometric guidance. Quantitative results are in Table 6.3, and qualitative comparisons in Figures 6.4 and 6.5.

Impact of Iterative Refinement. We compare single-iteration reconstruction (using only the input image) versus our full multi-iteration approach. Figure 6.4 reveals critical differences: the single-iteration mesh (Mesh 0) exhibits noticeable geometric errors in the highlighted region, including distorted architectural structures and incomplete surfaces. This stems from limited viewpoint coverage, as a single image provides insufficient constraints to accurately reconstruct occluded regions. Through iterative refinement (Mesh T), our method progressively accumulates multi-view observations, providing increasingly rich geometric constraints. Each iteration's video generation reveals new viewpoints, which inform subsequent 3D generation stages, enabling the system to correct initial errors and complete previously unseen regions with high fidelity.

Table 6.3 quantifies this improvement with the full method achieving 54.0%-56.5% win rates in human evaluations and 58.2%-60.5% in GPT-4o assessments. While these margins may appear modest, they reflect the inherent challenge that even single-iteration

results are reasonably plausible. The consistent preference for multi-iteration results across all three criteria (geometry, layout, texture) confirms that iterative geometry-appearance co-evolution is essential for achieving high-quality scene completion with accurate detail preservation.

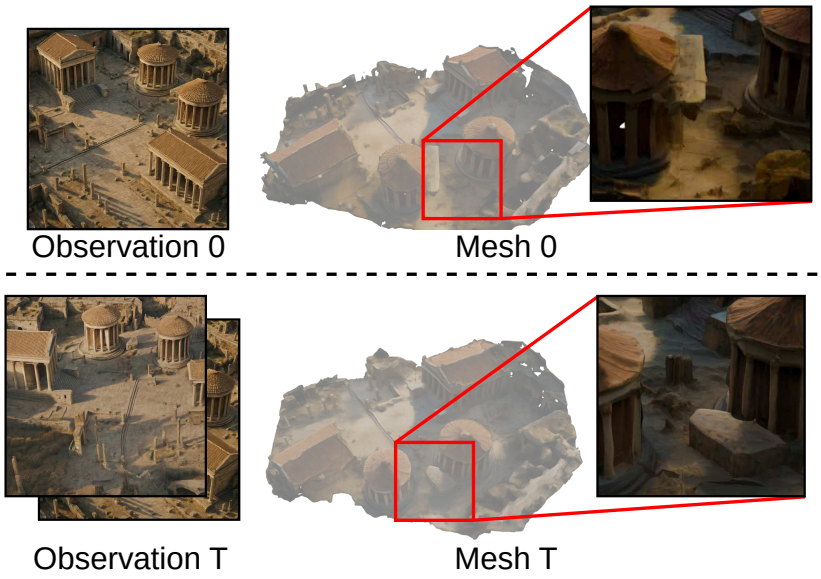


Figure 6.4.: **Ablation: Impact of iterative refinement.** Comparison between single-iteration (Mesh 0) and multi-iteration (Mesh T) reconstruction. The single-iteration mesh exhibits geometric errors in the highlighted region (distorted architectural structures), which are corrected after iterative refinement through accumulated multi-view observations and geometry-appearance co-evolution.

Impact of Depth-Conditioned Video Guidance. We compare our depth-conditioned video generation against a pose-only world model baseline (FlashWorld [221] generating views from $-30°$ to $+30°$). Figure 6.5 reveals a striking disparity: while the pose-only model generates visually plausible individual frames, the reconstructed 3D mesh exhibits catastrophic geometric failures with severe distortions and broken surfaces.

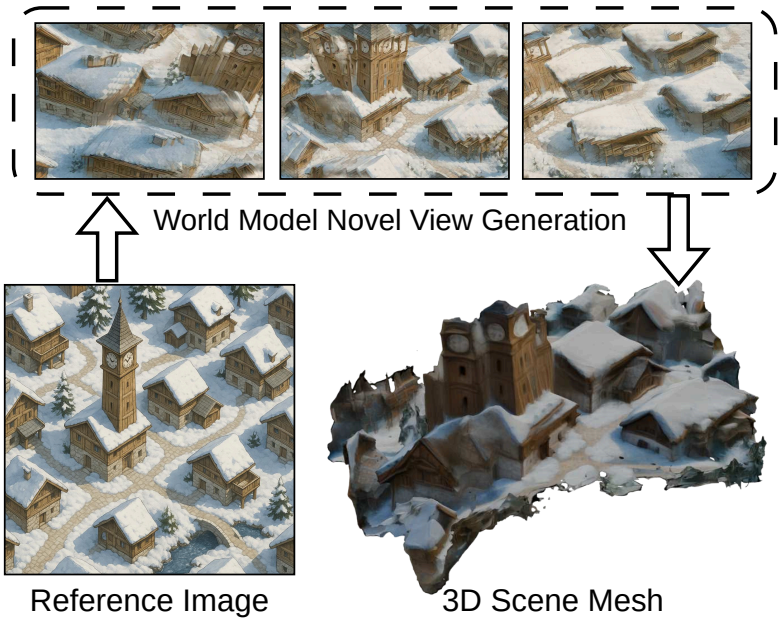


Figure 6.5.: **Ablation: Impact of depth-conditioned video guidance.** Comparison between pose-only world model (FlashWorld [221]) and our depth-conditioned video generation. The pose-only baseline generates visually plausible frames but produces geometrically inconsistent 3D meshes with severe distortions and broken surfaces (right). Our depth conditioning provides geometric scaffolding that ensures multi-view consistency and stable 3D reconstruction.

Without explicit geometric guidance, the model hallucinates appearance details that look plausible in 2D but violate 3D consistency constraints. Our depth-conditioned approach addresses this by rendering depth maps from the progressively refined mesh to guide video generation, providing geometric scaffolding that ensures multi-view consistency. Table 6.3 shows overwhelming win rates of 90.0%-94.0% in human evaluations and 97.9%-99.1% in GPT-4o assessments, confirming depth conditioning is absolutely critical for geometrically stable reconstruction under large viewpoint variations.

§ 6.5. Conclusion

We presented EvoScene, a self-evolving framework for single-image 3D scene generation that progressively refines geometry and appearance through iterative cycles between 2D and 3D domains. By synergistically combining geometric knowledge from 3D generation models with visual priors from video generation models, our training-free approach addresses key limitations of prior single-pass methods. Through three coupled stages, spatial prior initialization, visual-guided mesh generation, and spatial-guided novel view synthesis, EvoScene achieves complete full scene coverage with photorealistic textures. Comprehensive experiments demonstrate substantial improvements over state-of-the-art baselines, with human preference win rates reaching 78.5%-90.5% and ablations confirming the critical importance of both iterative refinement and depth-conditioned video guidance. We hope our work inspires further research into self-evolving approaches that bridge complementary strengths of different generative models for 3D content creation.

Chapter 7.

Conclusion and Future Work

§ 7.1. Summary of Contributions

This dissertation set out to study how to build embodied agents that ground language in perception and action, generate coherent multimodal content, and construct editable 3D worlds from limited input. The central proposal of the dissertation is that the most effective route to these capabilities is not to train a single new embodied system, but to compose existing pretrained foundation models through deliberately designed interfaces. The five works in the preceding chapters develop four families of such interfaces, and together they sketch one methodology for embodied AI built on top of pretrained knowledge.

Part I examined how pretrained perception and language models can be composed for embodied reasoning. VLMbench (Chapter 2) formalized language-guided manipulation as a compositional problem, organizing tasks around motion constraints and object-centric representations, and exposing where pretrained vision-language components fail to ground compositional instructions in physically constrained motion. JARVIS

(Chapter 3) then tackled long-horizon, dialog-based task completion by treating pretrained language and perception models as sensors and binding their outputs through symbolic task- and action-level reasoning. The two chapters argue that compositional benchmarks and symbolic interfaces over pretrained components are complementary: the former diagnoses the failure modes of direct use of pretrained models, while the latter shows how structured composition can recover what end-to-end pipelines cannot.

Part II moved from symbolic to learnable interfaces. MiniGPT-5 (Chapter 4) introduced generative vokens, a small set of learnable tokens that bridge a pretrained multimodal language model and a pretrained diffusion model, enabling description-free interleaved text-and-image generation while keeping both backbones frozen. This chapter argued that coherent multimodal output does not require monolithic retraining; a narrow, well-supervised token interface between pretrained components is sufficient to turn a comprehension model into a generation model.

Part III extended composition into 3D. SceneFuse-3D (Chapter 5) showed that high-fidelity scene-level 3D generation can be built on top of a pretrained object-centric 3D generator without any 3D training, by decomposing the scene into overlapping regions, using pretrained depth and detection models as spatial priors, and applying spatial-aware completion to preserve global layout. EvoScene (Chapter 6) then introduced an iterative interface, cycling between spatial prior initialization, visual-guided mesh generation, and spatial-guided novel view synthesis through a pretrained depth-conditioned video diffusion model, so that geometry and appearance progressively refine each other. Together these chapters point to a training-free, modular recipe for 3D scene synthesis in which pretrained 3D, depth, and video models cooperate rather than compete.

Across all three parts, a single message emerges. Embodied competence does not

reduce to scaling a single model or collecting a single dataset; it benefits from the deliberate *composition of pretrained foundation models* through interfaces matched to the structure of each problem: symbolic glue in JARVIS, learnable token interfaces in MiniGPT-5, spatial priors in SceneFuse-3D, and iterative 2D–3D feedback in EvoScene. The systems studied here gain interpretability, generalization, and data efficiency precisely because they decline to collapse perception, generation, and 3D construction into a single monolithic model, and instead reuse the foundation models that already encode much of the needed knowledge.

§ 7.2. Future Work

While these contributions advance embodied world modeling along perception, generation, and 3D construction, they also make the remaining gaps more visible. I outline three directions that I believe are essential for extending this line of work toward more capable embodied systems.

7.2.1. Interactive 4D Dynamic World Simulators

The generative 3D frameworks developed in this dissertation, including SceneFuse-3D (Chapter 5) and EvoScene (Chapter 6), focus on spatial geometry and visual fidelity, but treat the world as essentially static. The physical world is inherently temporal and governed by contact, articulation, and deformation. A natural next step is to move from static or purely visual 3D environments to *interactive 4D world simulators* that model both time and physical interaction.

An ideal 4D simulator should predict physically plausible future states of the environ-

ment conditioned on an agent's continuous action stream, rather than only synthesizing a passive visual sequence. This requires integrating generative priors with physics-aware dynamics so that opening a door, pouring a liquid, or deforming a cloth produces temporally coherent consequences that are consistent with the scene's geometry and material properties. Such simulators would close the loop between the generative 3D scene construction in Part III and the embodied interaction studied in Part I: instead of evaluating agents only in hand-authored simulators, we could train and evaluate them inside generated worlds that are as diverse as the 2D and 3D generative priors that built them.

7.2.2. Training World Action Models via Generative World Models

A recurring obstacle across the manipulation and task-completion chapters is data scarcity. Real robot trajectories are expensive to collect, and conventional simulators require significant engineering per task. Once generative world models are capable enough to synthesize diverse and physically plausible interactions, they become a natural training substrate for *world action models* — policies that learn to act by rolling out trajectories inside the generated world.

This direction reframes generative world models as data engines rather than visualization tools. Instead of relying solely on human demonstrations or handcrafted simulators, an agent can practice long-horizon planning, tool use, and precise manipulation inside a generative simulator, and the learned policies can then be transferred to physical robots. The central research questions here are not whether rollouts look realistic, but whether their *action–consequence statistics* are reliable enough to drive transfer: which generative priors preserve the causal structure of the real world, and which quietly distort

it. Answering this will likely require evaluation protocols that go beyond visual fidelity toward physical and behavioral fidelity.

7.2.3. MLLM-driven Self-Correction and Evolution Loop

Both neuro-symbolic agents and generative world models still depend heavily on human supervision: annotated sub-goals, preference data, and manual failure analysis. As these systems scale, that supervision becomes the bottleneck. A third direction is to close the loop through an *MLLM-driven self-correction and evolution* process, in which multimodal large language models act as automatic evaluators and feedback providers rather than only as generators.

Concretely, an MLLM could critique an agent's executed trajectory, diagnose physical or semantic inconsistencies in a generated scene, or score the alignment between an instruction and its resulting multimodal output, and feed structured, language-level feedback back into the underlying policy or generator. This is a natural continuation of the self-evolving cycle prototyped in EvoScene (Chapter 6), but pushed from geometry and appearance into task, dialog, and physics. The open problems here are trust and grounding: when should the MLLM critique be trusted, how should it be calibrated against ground-truth signals, and how can disagreements between the critic and the environment be resolved without drift? Addressing these will be key to turning one-shot generation systems into continuously improving embodied systems.

Taken together, these directions extend the composition methodology of this dissertation: 4D simulators by composing pretrained 3D, video, and physics priors; action models trained inside generative simulators built from pretrained world models; and self-

correcting embodied systems in which pretrained multimodal models act as critics in a closed loop. The frameworks in this dissertation treat embodied AI as a problem of composing pretrained components, and the next generation of embodied agents will benefit from pushing that composition further.

Appendix

A. Appendix of VLMbench

§ A.1. Task Details

Table A.1.: All task variations except shape used in VLMbench. The shape variation of each task can be found in the detail descriptions of each task category.

Variations	Totals	Values
Color	25	seen:red, maroon, lime, green, blue,navy, yellow, cyan, magenta, silver, gray, olive, purple, teal, azure, violet, rose, black, white unseen: brown, gold, pink, chocolate, coral
Size	5	larger, smaller, large, medium, small
Relative Position	5	top, front, rear, left, right
Level	3	top, middle, bottom
Amount	2	fully, slightly
Action Type	2	open, close

In the VLMbench, we show eight task categories:“Pick & Place objects", “Stack objects", “Drop pencil", “Put into shape sorter", “Pour water", “Wiper table", “Use drawer", and “Use door". Here, we list variations used for these tasks in Table. A.1. For each demonstration, all things in the scene will change the pose at the beginning. When building an instance-level task with one variation, the other variations will also

Table A.2.: All object models used in VLMbench. The number behind the object class indicate the instance number of that class.

Object type	Number of classes	Classes
Basic model	3	cube (1), triangular prism (1), cylinder (1)
Special model	9	star (1), moon (1), cross (1), flower (1), letter 't' (1), pencil (1), basket (1), box container(1), shape sorter (1)
Planar model	6	rectangle (1), circle (1), triangle (1), star (1), cross (1), flower (1)
Functional model	2	mug (6), sponge (1)
Articulated model	2	door with one rotatble handle (2), cabinet with three vertical drawers (3)

randomly change. For example, in the demonstrations of "Pick & Place objects" with "size" variation, all objects' color and relative positions, including targets and distractors, will randomly change. In the dataset, we have five types of objects, shown in Table A.2. We will explain each task in detail as follows. Visualizations can be found on the project website.

A.1.1. Pick & Place Objects

Task Definition: The agent needs to distinguish the specific object to grasp and then place it into a particular container. The object can be placed anywhere with any orientation inside the container.

Task Templates: Unit task sequence: *(Control, target object)+(M1[a1,b1], target object)*; Goal Conditions: A 3D bounding box without orientation constraints inside the empty space of target container, for *(M1[a1,b1], target object)*.

Success Conditions: The object detector inside the target container only can be triggered by the specific object. When the detector is triggered, the task considers a success.

Instruction Templates: High-level instructions: "Pick up [target object description] and place it into [target container description]."; Low-level instructions: ("Move to the

top of [target object description]; Grasp [target object description].", "Move the object into [target container description]; Release the gripper.").

Object models: One container model is used in both seen and unseen. In the seen settings, five object models: star, triangular, cylinder, cube, and moon. In the unseen settings, four object models: cube, the letter 't,' cross, and flower.

Variations and scene settings:

All objects are randomly changing colors, size, and positions in each demonstration.

Color: There are two same-shape objects and two same-shape containers in the scene initialization. All colors are randomly sampled from the color library. The object description is "[color] object"; The container description is "[color] container."

Size: There are two same-shape objects and two same-shape containers in the scene initialization. One object and one container are randomly magnified while others are randomly shrunk. The object description is "[larger/smaller] object"; The container description is "[larger/smaller] container."

Relative Position: There are two same-shape objects and two same-shape containers in the scene initialization. All objects are randomly sampled in the workspace until they obey the predefined relative positions. The object description is "[front/rear/left/right] object"; The container description is "[front/rear/left/right] container."

Shape: There are two same-shape containers and more than two objects with different shapes in the scene initialization. The number of objects varies from two to the length of the object library. The object description is "[shape]"; The container descriptions is "[color] container."

A.1.2. Stack Objects

Task Definition: The agent needs to distinguish the specific two objects and then stack them in a particular sequence. Since the objects have different shapes and some surfaces are hard to maintain for stacking, we use plane surfaces. Therefore, the goal pose of the above object should be inside the top plane of the below object and only can rotate along the axis which is perpendicular to the plane.

Task Templates. Unit task sequence: *(Control, target object)+(M1[a2,b2], above object)*; Goal Conditions: A plane on the surface of the below object with orientation constraints along the perpendicular axis, for *(M1[a2,b2], above object)*.

Success Conditions. The object detector attached to the bottom object will be triggered when the specific thing is above.

Instruction Templates. High-level instructions: "Stack [below object description] and [above object description] in sequence."; Low-level instructions: ("Move to the top of [above object description]; Grasp [above object description].", "Move the object on [below object description]; Release the gripper.").

Object models: In the seen settings, five object models: star, triangular, cylinder, cube, moon. In the unseen settings, four object models: cube, the letter 't', cross, flower.

Variations and scene settings:

All objects are randomly changing colors, size, and positions in each demonstration.

Color: There are four same-shape objects in the scene initialization. All colors are randomly sampled from the color library. The object descriptions are all "[color] [shape]."

Size: There are three same-shape objects in the scene initialization. One model is randomly magnified while another is randomly shrunk. The object description is

"[large/medium/small] [shape]."

Relative Position: There are four objects with two different shapes in the scene initialization. All objects are randomly sampled in the workspace until the relative position of the same-shape objects obey the predefined relative positions. The below object description is "[front/rear/left/right] [shape 1]"; The above object description is "[front/rear/left/right] [shape 2]."

Shape: There are more than two objects with different shapes in the scene initialization. The number of objects varies from two to the length of the object library. The below object description is "[shape 1]"; The above object description is "[shape 2]."

A.1.3. Drop Pencil

Task Definition: The agent must distinguish the specific pencil and then drop it into an upright container. Only when the pencil is vertical down above the container can it fall appropriately.

Task Templates. Unit task sequence: *(Control, target pencil)+(M1[a3,b2], target pencil)*; Goal Conditions: A line, which is perpendicular to the opening of target container, with orientation constraints along the line, for *(M1[a3,b2], target pencil)*.

Success Conditions. The object detector attached to the target container will be triggered when the specific pencil is inside.

Instruction Templates. High-level instructions: "Drop [target pencil description] into [target container description]."; Low-level instructions: ("Move to the top of [target pencil description]; Grasp [target pencil description].", "Move the object above [target container description]; Release the gripper.").

Object models: One pencil and one basket for both seen and unseen.

Variations and scene settings:

All objects are randomly changing colors and positions in each demonstration.

Color: There are two same-shape pencils and two same-shape baskets in the scene initialization. All colors are randomly sampled from the color library. The object description is “[color] pencil"; The container description is “[color] container."

Size: There are two same-shape pencils and two same-shape baskets in the scene initialization. One pencil and one container are randomly magnified while others are randomly shrunk. The object description is “[larger/smaller] pencil"; The basket description is “[larger/smaller] container."

Relative Position: There are two same-shape pencils and two same-shape baskets in the scene initialization. All objects are randomly sampled in the workspace until they obey the predefined relative positions. The object description is “[front/rear/left/right] object"; The basket description is “[front/rear/left/right] container."

A.1.4. Put Into Shape Sorter

Task Definition: There is a shape sorter in the environment, and the agent needs to distinguish the specific object and then put it through the hole of the sorter. Only when the object’s shape and pose fit the hole can it fall.

Variations: Color, Shape, and Relative Position. These variations work on all objects except the sorter.

Task Templates. Unit task sequence: *(Control, target pencil)+(M1[a4,b3], target object)*; Goal Conditions: A fixed pose related to the sorter for each object shape, for *(M1[a4,b3], target object)*.

Success Conditions. The object detector attached to the sorter will be triggered when

the specific object is inside.

Instruction Templates. High-level instructions: “Put [target object description] through the hole of [target hole description]."; Low-level instructions: (“Move to the top of [target object description]; Grasp [target object description].", “Move the object to the hole of [target hole description]; Release the gripper.").

Object models: One shape sorter container for both seen and unseen. In the seen settings, four object models: star, triangular, cylinder, and cube. In the unseen settings, five object models: star, triangular, cylinder, cube, and moon.

Variations and scene settings: All objects are randomly changing colors and positions in each demonstration.

Color: There are three same-shape objects and one shape sorter in the scene initialization. All colors are randomly sampled from the color library. The object description is “[color] [shape]."

Relative Position: There are two same-shape objects and one shape sorter in the scene initialization. All objects are randomly sampled in the workspace until they obey the predefined relative positions. The object description is “[front/rear/left/right] [shape]."

Shape: There are one shape sorter and more than two objects with different shapes in the scene initialization. The number of objects varies from two to the length of the object library. The object description is "[shape]."

A.1.5. Pour Water

Task Definition: The agent needs to pour the water from the specific source mug into the particular container mug. Here we use 50 small particles to simulate the water.

Variations: Color, Size, and Relative Position. These variations work on all mugs.

A. Appendix of VLMbench

Task Templates. Unit task sequence: *(Control, source mug)+(M2[a1,b2], source mug)+(M2[a4, b4], source mug)*; Goal Conditions: For *(M2[a1,b2], source mug)*, we need to keep the opening of the source mug is upward with the rotations along the axis of the opening directions. Further, the goal position of this step needs to make the horizontal distance between the geometry centers of two mugs less than the half-height of the source mug, and the vertical distance should be larger than the height of the container mug. For *(M2[a4, b4], source mug)*, the source mug needs to make the opening point to the container mug. Therefore, the end-effector should follow a particular path, keeping the source mug in the same position but rotating it to the required orientation. Therefore, the goal condition is a set of poses, calculated by a function using the geometry and poses of two mugs.

Success Conditions. The object detector attached to the container mug will be triggered when more than half particles are inside the mug.

Instruction Templates. High-level instructions: “Pour the water from [source mug description] to [container mug description]."; Low-level instructions: (“Move to the [relative position] of [source mug description]; Grasp [source mug description].", “Move the object to the top of [container mug description] with the opening upwards.", “Rotate [source mug description] toward [container mug description]").

Object models: Four different mugs in seen scenes and two different mugs in unseen scenes.

Variations and scene settings: All objects are randomly changing colors and positions in each demonstration.

Color: There are three same-shape mugs in the scene initialization. All colors are randomly sampled from the color library. The object description is “[color] mug."

Relative Position: There are two same-shape mugs in the scene initialization. All objects are randomly sampled in the workspace until they obey the predefined relative positions. The object description is "[front/rear/left/right] mug."

Size: There are two same-shape mugs in the scene initialization. One mug is randomly magnified while another is randomly shrunk. The object description is "[larger/smaller] mug."

A.1.6. Wipe Table

Task Definition: The agent needs to wipe the particular area with a sponge, which means moving the sponge from one side of the area to another with keeping it contacting with the area.

Task Templates. Unit task sequence: *(Control, sponge)+(M1[a4,b2], sponge) +(M2[a2, b2], sponge)*; Goal Conditions: For *(M1[a4,b2], sponge)*, the sponge needs to move to one side of the target area with the surface face contacting the area. For *(M2[a2, b2], sponge)*, the sponge needs to move from the current side to another side with keeping contact. The goal conditions are both a set of poses with the fixed position and rotations along the axis perpendicular to the table.

Success Conditions. We create 50 small invisible particles in the target area, and these particles will be removed if the sponge surface touches them. The successor attached to the target area will be triggered when more than half particles have been removed.

Instruction Templates. High-level instructions: "Wipe [target area description] with a sponge."; Low-level instructions: ("Move to the top of the sponge; Grasp the sponge.", "Move the object to the side of [target area description].", "Move the object along the

main direction of [target area description]").

Object models: One sponge for both seen and unseen. The four planes in the seen scenes: rectangle, round, star, and triangle; The two planes in the unseen scenes: cross, flower.

Variations and scene settings: All objects are randomly changing colors and positions in each demonstration.

Color: There are two same-shape planes and one sponge in the scene initialization. All colors are randomly sampled from the color library. The plane description is "[color] area."

Size: There are two same-shape planes and one sponge in the scene initialization. One plane is randomly magnified while another is randomly shrunk. The plane description is "[larger/smaller] area."

Relative Position: There are two same-shape planes and one sponge in the scene initialization. All planes are randomly sampled in the workspace until they obey the predefined relative positions. The object description is "[front/rear/left/right] area."

Direction: There are two same-shape directional planes and one sponge in the scene initialization. One plane is horizontal to the width of table while another is vertical. The plane description is "[horizontal/vertical] area."

Shape: There are one sponge and more than two planes with different shapes in the scene initialization. The number of planes varies from two to the length of the object library. The plane description is "[shape] area."

A.1.7. Use Drawer

Task Definition: The agent needs to figure out the exact level of the drawers to use and execute the correct action with appropriate degrees. The initial states of the drawer will change according to the action types. For example, if the task is to open the top drawer, the top drawer will be closed at first.

Task Templates. Unit task sequence: *(Control, target drawer handle)+(M2[a3,b3], target drawer joint)*; Goal Conditions: For *(M2[a3,b3], target drawer)*, the agent needs to move the target drawer to a fixed position with the linear physic constraints on the joints of the drawer. Therefore, the goal conditions are a set of the end-effector poses whose positions are along an axis, and the orientations are fixed.

Success Conditions. The successor will be triggered when the joint of the target drawer have moved a particular length.

Instruction Templates. High-level instructions: "[Amount] [Action Type] the [Level] drawer."; Low-level instructions: ("Move to the front of the handle of [target drawer description]; Grasp the handle of [target drawer description].", "Move along the axis of [target drawer description] for [Amount] [Action Type].").

Object models: Two cabinets in seen scenes and one cabinet in unseen scenes. The cabinets have different appearance but all have three vertical drawers.

Variations and scene settings: The cabinet is randomly changing the pose in the workspace for each demonstration.

Level, Action Type, and Amount: One cabinet is sampled from the object list. These variations are all working on the drawer at the same time. For example, we regard "Fully open the top drawer." as one situation.

A.1.8. Use Door

Task Definition: The agent needs to execute the correct action with appropriate degrees for the door. If the door is closed, the agent needs to rotate the door handle to unlock it for opening it. Like the use drawer tasks, the initial states will change according to the action type.

Task Templates. Unit task sequence: For closing the door, *(Control, door handle)+(M2[a4,b4],door plank joint)*. For opening the door, *(Control, door handle)+(M2[a4,b4],door handle joint)+(M2[a4,b4],door plank joint)*; Goal Conditions: For *(M2[a4,b4],door handle joint)*, the agent needs to rotate the joint, which connects the handle and plank, to a degree for unlock. Therefore, the goal conditions are a set of the end-effector poses whose positions are along an arc, and the orientations are depended on the positions. For *((M2[a4,b4], door plank joint)*, the goal conditions are the same as the previous one, but the joint connects the plank and base.

Success Conditions. The successor will be triggered when the plank and base joint has reached a particular angle.

Instruction Templates. High-level instructions: "[Amount] [Action Type] the door."; Low-level instructions: ("Move to the front of the handle of the door; Grasp the handle of the door.", "Rotate around the axis of the handle joint.", "Rotate around the axis of the door joint for [Amount] [Action Type].").

Object models: One door in seen scenes and one door in unseen scenes. The doors have different appearance but all have the rotatable handle.

Variations and scene settings: The door is randomly changing the pose in the workspace for each demonstration.

Action Type and Amount: One door is sampled from the object list. These variations

are all working on the door at the same time. For example, we regard “Fully open the door." as one situation.

§ A.2. Implementation Details

A.2.1. AMSolver

Task Creation To generate demonstrations for semantic tasks, we need to use the predefined unit task sequence combined with object configurations and goal conditions. The goal conditions are the feasible goal pose sets of the current manipulated object for each unit task. The goal conditions can be formatted as bounding boxes, planes, lines, joint angles, or generated by functions for special pose sets. For example, a rearrangement task *“Put the ball into the basket"* needs to transit a ball into the empty space of the basket without any constraints requirement, which means the goal condition of the ball is a set of poses inside the blank space of basket. Then, we can format this task as $(Control, ball) + (M1[a1, b1], ball)$. Note that *ball* can be replaced with other objects, so $(Control, object) + (M1[a1, b1], object)$ can represent any pick-and-place task without specific constraints. Another example is *“Pour the water from one mug to another"*. This task needs to keep the mug opening upward while moving toward the top of another mug. We can format this task as $(Control, source\ mug) + (M2[a1, b2], source\ mug) + (M2[a4, b4], source\ mug)$. For the first $(M2[a1, b2], source\ mug)$, the goal condition should make the horizontal distance between the geometry centers of two mugs less than the half-height of the source mug, and the vertical distance should be larger than the height of the container mug. Therefore, the goal condition is a set of poses calculated by a function using the geometry and poses of two mugs.

Implementation The AMSolver is built inside the CoppeliaSim [46] environment and written in Python and Lua based on RLbench [9] and PyRep [47]. The agent is a Franka Emika Panda robot arm with 7 DoF, standing on a wooden table. To achieve the grasping ability for any object shape, we implement one version of the algorithm of ten Pas et al. [50]. The generator will calculate the normals and principal axis of partial point clouds. Then, the normals, principal axis, and cross-product will establish the grasp poses. Finally, the agent will calculate inverse kinematics to check the validation. More details can be found in [50]. Each unit task will generate one waypoint as the sub-goal of the end-effector or one fixed cartesian path as the fixed sub-trajectory for the end-effector in the environment. We use the OMPL library [251] to find a valid trajectory between two waypoint configurations. In the end, we will get a valid path from the beginning to the task finish, consisting of the critical waypoints information. The object detector used in the simulator is a 3D box, where the poses (positions and orientations) and properties (length, width, height) can be modified. The object detector is placed in the target destinations of the task. When the target object mesh overlaps with the detector, it will return the true value for checking the object.

A.2.2. VLMbench

Dataset Generation We collect the VLMbench dataset in the environment of RLbench with AMSolver. There are five RGB-D cameras in the environment: front view, left view, right view, overhead view, and wrist view. We use the image resolution of 360×360 in this dataset. As mentioned in section 3.4, the AMSolver will output the trajectories with waypoints. Therefore, the robot arm will use motion planning to transfer from one waypoint to another, and the simulator will record the observation with the time step 50

ms. Each camera can produce an RGB-D image with point cloud and instance masks for observation. We also record the low-level states of the robot joints, including joint velocity, joint torque, joint position, and end-effector pose. In addition, the observations record the object pose at each time step and the corresponding relationship between the frames and waypoints so that the demonstrations can be automatically divided into the sub-demonstrations with the critical frame. The objects will change the poses among different demonstrations, and the distractors will be randomly added to the environment.

For language instructions, we predefined some templates for each task category to quickly generate various language instructions by filling the object properties' descriptions, shown in Fig. 2.3. For example, the "Pick & Place objects" task has the template "Pick [Object] and place it into [Container]," where [Object] denotes the target object descriptions corresponding to variations mentioned above. Meanwhile, by defining the structured language templates on the unit task, we can simultaneously generate the low-level instructions in the dataset, which structurally describe the basic actions corresponding to the frames, e.g. "Move to [relative position] [manipulated object]; Grasp [manipulated object]" correspond to the pre-grasp action and grasp action in the Control unit task. We hope this information will be helpful for further research in this area.

A.2.3. 6D-CLIPort

In this section, we provide hyperparameter values in our 6D-CLIPort training. We have trained each agent on eight A5000 GPUs for one hour with a batch size of 16, an epoch of 15, and a learning rate of 1e-3. The pixel-wise feature maps from the attention module have one dimension. The output feature maps of the value and key module have four

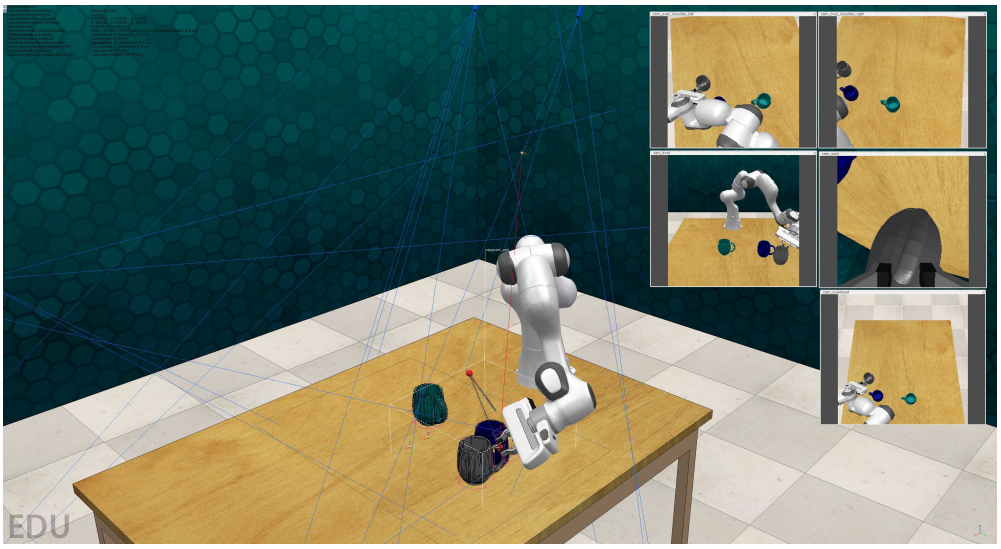


Figure A.1.: The environment used in VLMbench. A 7 DoF robot arm stands on the table with five cameras from different views. Meanwhile, the images from the camera views are shown aside. We also can see the predefine of the paths in the environment, which are the ground truth waypoints generated by the AMSolver.

dimensions in each pixel to estimate 2D positions and the rotation along the perpendicular axis. Meanwhile, The output feature maps have 12 dimensions in each pixel for the other three regressors and are chunked into three parts for separate convolutions, where the convolutions of value and key feature maps are input to each regressor.

§ A.3. Dataset Details

We sample VLMbench in the CoppeliaSim simulator, shown in Fig. A.1. The simulator collects the data at a 20 Hz frequency, and at each step, the state record all visual, robot, and object information. To illuminate the dataset's structure, we have a folder for each instance-level task, e.g., pouring water with color variation. Each variation value creates

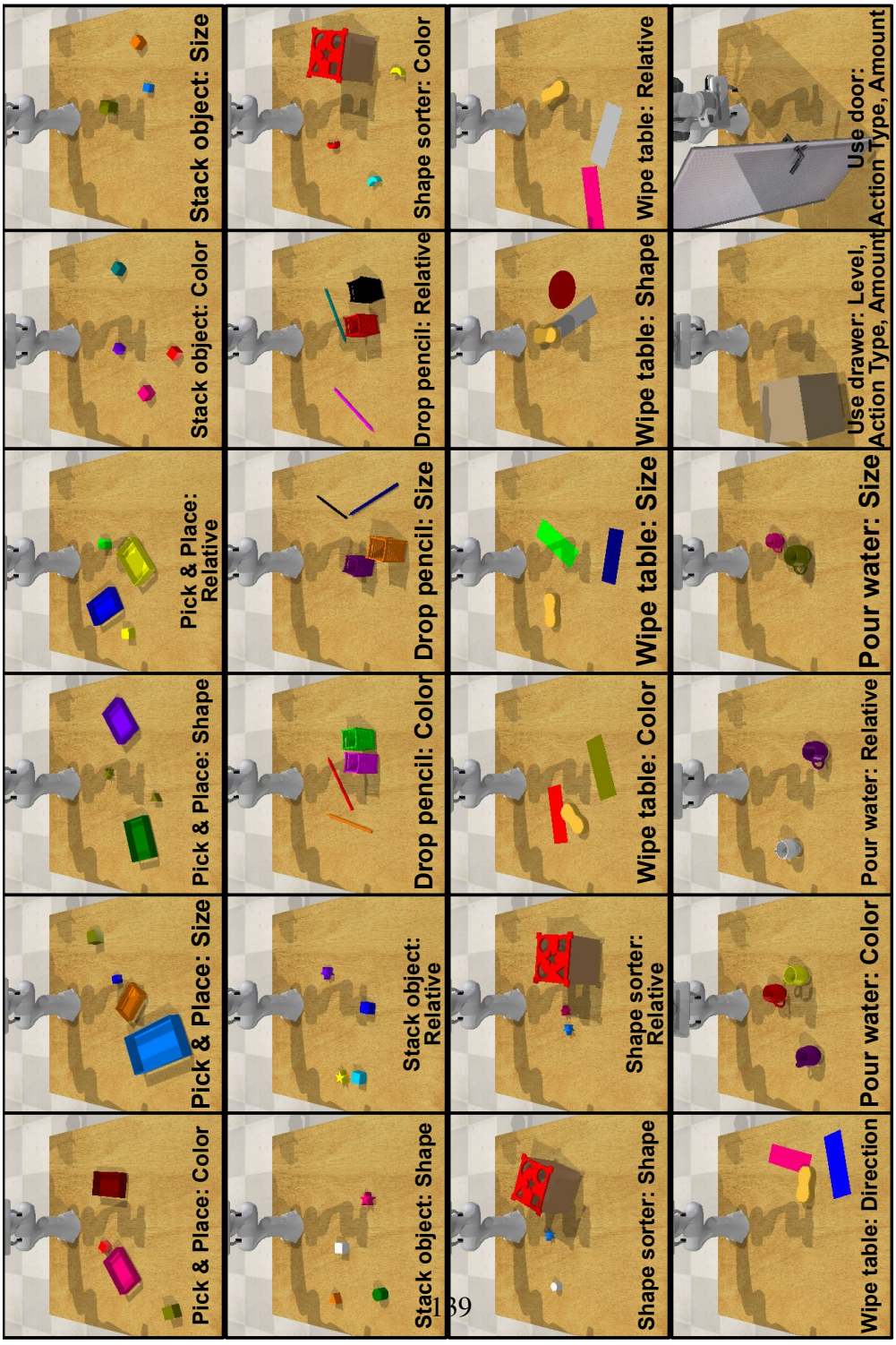


Figure A.2.: All instance-level tasks in VLMbench.

a folder under the task folder, e.g., the target mug is red in the pouring water with color variation. Inside one variation value folder, we have demonstration folders that contain the state record for demonstrations. Then, we save the RGB-D, segmentation, and point cloud from each camera in one demonstration folder. Another file contains all low-level robot states, object states, and waypoint information for every step. All instance-level tasks are shown in Fig. A.2. Demonstration demos can be found in the video on the website.

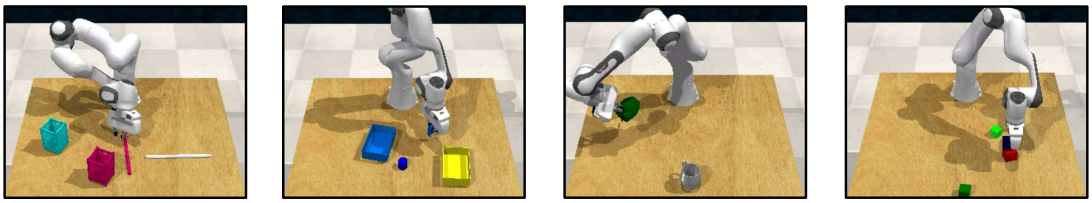
Task Statistics In total, we have 24 instance-level tasks with 234 variations, which include 120 color variations, 18 size variations, 60 relative position variations, 18 shape variations, 2 direction variations, and 16 variations of the combinations of level, amount, and action type. The benchmark have 6,658 manipulation demonstrations which contain 4,783 train demonstrations, 1,170 seen validation demonstrations, and 705 unseen validation demonstrations. Since the agent test in the online simulator, we have generated 2,380 seen test settings and 2,380 unseen test settings for all instance-level tasks. The detailed numbers can be found in Table A.3. All demonstrations are generated by two servers with 64-Core CPUs and 8 GPUs within 24 hours.

§ A.4. Additional Experiments

We have shown task success rates of each baseline agent in both seen and unseen settings. Here, we provide the results of step success for both seen and unseen settings. Then, we want to provide more explanation and analysis of failure cases and ablations study.

A.4.1. Goal-Conditioned Success Rates

In Table A.4 and A.5, we show the success rates of each step for every agent. For tasks in the VLMbench, each task can be considered as a multi-step task, since each task at least consists of one grasping step and one following movement step. Therefore, we used two other metrics for the step's success: Goal-condition Grasp (G-G) success rate and Goal-conditioned Movement (G-M) success rate. The Goal-conditioned Grasp (G-G) success rates indicate whether the agent has grasped the correct object for the first step. The Goal-conditioned Movement (G-M) success rates indicate whether the agent can satisfy the success conditions in the following movement by using the pre-generated grasp poses for the grasping step. We can find that the agent has high grasping successes in both seen and unseen settings for most tasks, including pick, stack, drop, shape sorter, pour, and wiper, which indicates that 6D-CLIPort can successfully reason the correct grasping objects with both seen and unseen settings. For these tasks, the prominent failure cases come from the following movement. The conclusion can also be obtained from the minor difference between the goal-conditioned movement and the total success rate. In the wiping tasks, we can see high grasping success rates in all settings since we only have one sponge shape model in both training and testing. We also can find that the Vision-Only agent has a higher grasping success rate than 6D-CLIPort in the "Drawer" tasks with seen settings. However, it has a much lower grasping success rate in the unseen settings, which indicates that the Vision-Only agent has overfitted on the grasping step in the "Drawer" tasks. For opening/closing drawer and door tasks, we find that goal-conditioned movement success rates are similar for seen and unseen settings, meaning the agent performs similarly after the agent has correct grasping. In opening/closing drawer tasks, we also find that the goal-condition grasp is smaller than



(a) **Wrong Grasping Pose** "Drop the pink pencil into the pink container."

(b) **Wrong Grasping Object** "Pick up the cylinder and place it into the yellow container."

(c) **Wrong Destination Pose** "Pour water from the green mug to the gray mug."

(d) **Wrong Destination Object** "Stack the green cube and the navy cube in sequence."

Figure A.3.: Four failure cases in the results of the 6D-CLIPort. Details can be found in A.4.2.

total success, which means some tasks reach the success conditions without grasping the handle of doors. It also makes sense that closing the drawer can be finished by pushing instead of grasping the handle first.

A.4.2. Failure Cases

The results show that the 6D-CLIPort agent has failed in many tasks. Except for the unreachable poses due to the errors in the position and orientation estimations, we summarize the four kinds of failure cases, shown in Fig. A.3: (a) Wrong grasping pose, which means the agent cannot grasp any object. (b) Wrong grasping object, which means the agent grasps the incorrect object. (c) Wrong destination pose, which means the destination pose of the agent cannot finish any semantic task. (d) Wrong destination object, which means the estimation destination is inconsistent with the instructions. More visualization can be found in the attached video.

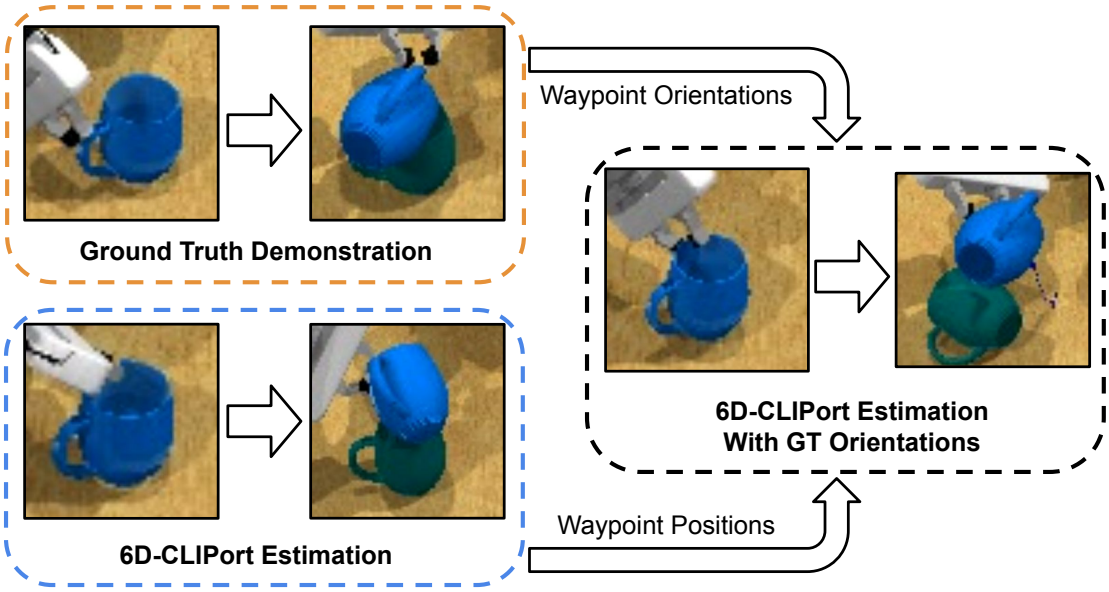


Figure A.4.: 6D-CLIPort with ground truth waypoint information. The instruction here is “Pour water from the blue mug to the green mug." Since the orientations and positions are unmatched, we can see 6D-CLIPort with ground truth orientation can pour the water but have collisions with the container mug. In this situation, 6D-CLIPort with ground truth positions cannot find a valid path by motion planning.

A.4.3. Partially Modal Agents

In the experiment section, we have tested the 6D-CLIPort with the ground truth waypoints' positions or orientations. We can see the agent still failed the tasks even with this privileged information. The reason is the unmatched positions and orientations. The ground truth positions and orientations are obtained from the pre-generated waypoints. Therefore, this ground truth information is not the optimal parameter associated with the estimation. The unmatched positions and orientations can lead to a failure or even an unreachable pose where the motion planner cannot find a feasible path. For the tasks that need the cooperation of position and orientation, such as dropping, pouring, and opening the door, we can figure out that giving partially ground truth still cannot finish the task correctly. We shown a example in Fig A.4.Since the estimation trajectory and ground truth trajectory can be valid but different solutions for the same task. The combination of the positions and orientations will fail the task. Furthermore, the wrong estimation of positions can lead to task failure, even given the ground truth orientation, since the tasks in the VLMbench require correct compositional reasoning. More visualization can be found in the video on the project web page.

Table A.3.: The number of episodes for each task in the dataset.

Task category	Variation	Train	Valid		Test	
			Seen	Unseen	Seen	Unseen
Pick	Color	400	100	25	100	100
	Relative	320	80	80	96	96
	Shape	100	25	20	100	100
	Size	80	20	20	100	100
Stack	Color	400	100	25	100	100
	Relative	320	80	80	96	96
	Shape	100	25	20	100	100
	Size	120	30	30	96	96
Drop	Color	395	100	25	100	100
	Relative	320	80	80	96	96
	Size	80	20	20	100	100
Place	Color	400	100	25	100	100
	Relative	80	20	20	100	100
	Shape	80	20	25	100	100
Wipe	Color	400	100	25	100	100
	Relative	80	20	20	100	100
	Shape	80	20	20	100	100
	Size	40	10	10	100	100
	Direction	40	10	10	100	100
Pour	Color	388	100	25	100	100
	Relative	80	20	20	100	100
	Size	40	10	10	100	100
Door	action&amount	200	20	20	100	100
Drawer	action&amount&level	240	60	60	96	96
	Total	4783	1170	705	2380	2380

A. Appendix of VLMbench

Table A.4.: Results of all tasks, including both seen and unseen settings. In the table, G-G denotes Goal-conditioned Grasp success rate, which reflects whether the agent can correctly grasp the target. G-M denotes Goal-conditioned Movement success rate, which reflects whether the agent can finish the task by using the pre-generated grasping poses. SC denotes success rates for the whole estimation trajectory. More explanations can be found in section A.4.1.

Agent	**Pick&Place**						**Stack**						**Drop**						**Shape Sorter**					
	Seen			Unseen			Seen			Unseen			Seen			Unseen			Seen			Unseen		
	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC
Language-Only	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Vision-Only	29.55	29.80	6.31	36.62	26.52	9.85	34.95	32.65	6.89	32.65	22.45	1.79	10.47	0.00	0.00	16.89	0.00	0.00	8.33	8.67	0.00	4.00	6.33	0.33
6D-CLIPort	**58.33**	**43.33**	**28.28**	**63.14**	**41.16**	**27.53**	**53.06**	**44.64**	**22.19**	**48.21**	**28.06**	**18.37**	**70.61**	**9.12**	**6.42**	**65.88**	**8.78**	**6.42**	**73.00**	**23.67**	**17.33**	**66.67**	**18.67**	**12.33**
6D-CLIPort (GT Ori)	60.61	42.93	28.03	65.40	36.87	26.26	56.38	44.64	26.53	51.02	48.98	26.02	65.88	31.08	17.91	62.16	28.72	16.22	72.67	32.67	24.00	67.33	24.00	15.67
6D-CLIPort (GT Pos)	94.95	88.64	83.84	95.96	83.33	75.25	96.94	69.39	58.93	96.43	47.70	50.51	95.61	13.85	16.89	94.26	13.18	11.82	92.00	19.33	18.00	94.00	22.67	17.33

	Pour						**Wipe**						**Door**						**Drawer**					
	Seen			Unseen			Seen			Unseen			Seen			Unseen			Seen			Unseen		
	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC
Language-Only	0.00	0.76	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.21	15.63	4.17	3.13	13.54	1.04
Vision-Only	2.67	2.00	0.00	4.00	**1.33**	0.00	99.20	19.40	19.80	**96.00**	21.20	20.80	0.00	0.00	0.00	0.00	0.00	0.00	**26.04**	21.88	14.58	1.04	18.75	7.29
6D-CLIPort	**61.00**	**2.00**	**1.00**	**69.67**	1.00	**1.00**	**99.20**	**21.60**	**22.40**	95.20	**21.60**	**21.80**	**29.00**	**14.00**	**6.00**	**5.00**	**15.00**	**5.00**	22.92	**25.00**	**22.92**	**8.33**	**21.88**	**15.63**
6D-CLIPort (GT Ori)	70.00	4.00	3.67	76.67	3.67	3.67	99.60	25.40	25.80	97.20	25.20	25.20	49.00	14.00	6.00	5.00	15.00	5.00	31.25	30.00	23.96	8.33	23.96	17.71
6D-CLIPort (GT Pos)	61.67	3.00	0.33	67.33	0.67	0.67	100.00	61.00	60.20	99.40	57.40	53.40	80.00	46.00	27.00	83.00	49.00	27.00	79.17	43.75	43.75	87.50	50.00	52.08

Table A.5.: Results of all variations. The notations are used the same as in Table 2.3.

Agent	**Color**						**Shape**						**Size**						**Relative Position**					
	Seen			Unseen			Seen			Unseen			Seen			Unseen			Seen			Unseen		
	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC
Language-Only	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Vision-Only	29.67	14.17	6.00	30.00	13.00	5.00	42.75	21.00	6.00	46.25	20.00	8.50	36.49	15.12	6.05	36.09	12.76	5.10	31.46	19.73	6.80	31.97	12.76	5.10
6D-CLIPort	**69.83**	**19.67**	**15.17**	**69.00**	**17.50**	**13.00**	**69.75**	**33.25**	**23.00**	**71.25**	**27.25**	**19.50**	**70.56**	**25.40**	**18.35**	**64.31**	**21.77**	**14.92**	**68.03**	**27.72**	**15.31**	**69.56**	**20.75**	**15.82**
6D-CLIPort (GT Ori)	72.00	24.67	18.67	70.83	26.00	17.67	70.25	39.75	28.00	72.50	33.75	27.25	71.57	29.23	20.97	66.53	28.02	17.74	70.58	33.33	21.43	71.09	29.42	18.37
6D-CLIPort (GT Pos)	90.67	42.33	40.17	90.33	37.50	35.00	96.00	59.75	56.25	96.75	53.25	51.25	90.32	47.58	44.96	91.53	41.33	38.51	89.12	41.50	38.61	90.99	36.39	34.86

	Direction						**Level**						**Action Type**						**Amount**					
	Seen			Unseen			Seen			Unseen			Seen			Unseen			Seen			Unseen		
	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC	G-G	G-M	SC
Language-Only	0.00	0.00	0.00	0.00	0.00	0.00	5.21	15.63	4.17	3.13	13.54	1.04	2.55	7.65	2.04	1.53	6.63	0.51	2.55	7.65	2.04	1.53	6.63	0.51
Vision-Only	**99.00**	15.00	21.00	95.00	22.00	24.00	26.04	21.88	14.58	1.04	18.75	7.29	12.76	10.71	7.14	0.51	9.18	3.57	12.76	10.71	7.14	0.51	9.18	3.57
6D-CLIPort	98.00	**19.00**	**21.00**	**100.00**	**22.00**	**26.00**	**22.92**	**25.00**	**22.92**	**8.33**	**21.88**	**15.63**	**26.02**	**19.39**	**14.29**	**6.63**	**18.37**	**10.20**	**26.02**	**19.39**	**14.29**	**6.63**	**18.37**	**10.20**
6D-CLIPort (GT Ori)	99.00	26.00	26.00	98.00	29.00	27.00	31.25	30.00	23.96	8.33	23.96	17.71	40.31	22.45	14.80	6.63	19.39	11.22	40.31	22.45	14.80	6.63	19.39	11.22
6D-CLIPort (GT Pos)	100.00	63.00	53.00	100.00	56.00	41.00	79.17	43.75	43.75	87.50	50.00	52.08	79.59	44.90	35.20	85.20	49.49	39.29	79.59	44.90	35.20	85.20	49.49	39.29

B. Appendix of JARVIS

§ B.1. Implementation Details

JARVIS takes advantage of both connectionism and symbolism. Here we first introduce the learning details of the deep learning modules in JARVIS, followed by the symbolic commonsense reasoning. Then we introduce the task completion process of the JARVIS framework, shown in the Algorithm 1.

B.1.1. Learning Modules

Here we elaborate on the implementation details of the learning and symbolic reasoning modules in our framework, including language understanding and planning, semantic world representation, and goal transformer.

Language Understanding and Planning

For data collection of the EDH task, we collect one data sample from each EDH instance. In each data sample, the input contains a history dialogue between the commander and the follower and history sub-goals that have been executed, and the output is the future sub-goals. We create special tokens <COM>, <FOL> to concatenate different utterances of the dialogue, and <HIS> to concatenate the history sub-goals and dialogue. Note that

the history sub-goals are not available in the training set, so we translate the provided history actions into history sub-goals by excluding all the navigation. For the TfD task, we collect the whole dialog sequence and the whole ground truth future sub-goal sequence from each TfD instance as input and output for training. We collect the same data scheme from seen validation EDH/TfD instances for validation. We adapt the pre-trained BART-LARGE [55] model and fine-tune the BART model separately on the collected training data for each task. We follow the Huggingface [252] implementation for the BART model as well as the tokenizer. The BART model is finetuned for 50 epochs and selected by the best validation performance. We use the Adam optimizer with a learning rate of 5×10^{-5}. The maximum length of generated subgoal is set to 300, and the beam size in beam search is set to 4.

Semantic World Representation

For Mask-RCNN [54], we use a model pre-trained on MSCOCO [253] data and fine-tune it on collected data from training environments of TEACh dataset. We get the ground truth data samples from the provided interface of AI2THOR. During training, the loss function is as follows:

$$L = L_{cls} + L_{box} + L_{mask} \tag{B.1.1}$$

Where L_{cls} is the cross entropy loss of object class prediction. L_{box} is a robust L_1 loss of bounding box regression. And L_{mask} is the average binary cross entropy loss of mask prediction, where the neural network predict a binary mask for each object class. Following [254], we use the batch size of 4 and learning rate of 5×10^{-3} for Mask-RCNN training. We train for 10 epochs and select by the best performance on data in unseen

validation environments.

For the depth prediction model, we use an Unet-based model architecture same as [88]. We get the images and ground truth depth frames from the training environments of TEACh dataset by AI2THOR [255] too. The range of depth prediction is from 0-5 meters, with an interval of 5 centimeters. Thus, the depth prediction problem is formulated as a classification problem over 50 classes on each pixel. During training, the loss function is a pixel-wise cross entropy loss:

$$L = \mathrm{CE}(D_{predict}, D_{gt}) \tag{B.1.2}$$

D_{gt} stands for ground truth depth and $D_{predict}$ stands for predicted depth. During training, we follow the training details in [88] and using the batch size of 4, learning rate of 1×10^{-4}. We train for 4 epochs and select by the best performance on data in unseen validation environments.

For semantic map construction, we first project the depth and semantic information predicted by learned models into a 3-D point cloud voxel, based on which we do vertical projection and build a semantic map of 240×240 for each object class and obstacle map. Each pixel represents a $5cm \times 5cm$ patch in the simulator.

Goal Transformer

We collect one data sample from each EDH instance to train the Goal Transformer. The Goal Transformer takes the ground truth future sub-goals, history images, and history actions as inputs. We modify the prediction head of the Episodic Transformer to generate the actions in TEACh benchmarks [11]. The Goal Transformer uses a language encoder to encode the future sub-goals, an image encoder (a pre-trained ResNet-50) to encode

the current and history images, and an action embedding layer to encode the history actions in order. The model is trained to predict future actions autoregressively. During training, we use the cross entropy loss between the ground truth future action A_{gt} and predicted future action, as is $A_{predict}$:

$$L = \text{CE}(A_{predict}, A_{gt}) \tag{B.1.3}$$

We follow the episodic transformer (E.T.) [86] training details for the goal transformer model. The batch size is 8 and the training epoch is 20. We use the Adamw optimizer [256] with a learning rate of 1×10^{-4}.

Symbolic Reasoning

Here, we provide some details of the implementations of the symbolic reasoning part. In task-level commonsense reasoning, we mainly check sub-goals from two perspectives: properties and causality. For properties, we need to check whether the action is affordable for the object. Therefore, we define some object collections depending on the properties, including movable, sliceable, openable, toggleable, and supportable. We can determine the unreasonable sub-goals by checking whether the planned object can afford the corresponding action. Then, causality means whether the sub-goal sequence obey the causal relations, like "a knife in hand" should always be the prerequisite of "slice a bread". To achieve this purpose, we define some rules of prerequisites and solutions according to commonsense, including placing the in-hand object before picking something, removing the placing action if nothing in hand, etc. To check the prerequisites, we assume all the previous sub-goals are completed and check if the agent's states matches the desired states. If the current sub-goal and the agent's state have causal conflict, we will use

predefined solutions to deal with it. For example, we will remove *Place(agent, x)* if there is nothing in the hand. We will add *Pick(agent, "Knife")* before *Slice(agent, "Bread")* if the agent do not grasp a knife in hand. The whole process can be found in Algorithm 2.

We consider both the semantic map and action states for action-level commonsense reasoning. In general, the agent updates the semantic map according to the observation of every step. We use the pixels change to detect whether the previous action success. To determine the next step, the agent must check whether the target object has been observed. If the target object has been observed, an FMM algorithm will plan a path to the closest feasible space. Otherwise, we will use the Goal Transformer to determine the next action for exploration. The action space of the Goal Transformer is all motion actions, including forwarding, backward, panning left, panning right, turning right, and turning left. During the movement to the target position, if the agent counterfaces unexpected collisions due to the error of the semantic map, the agent will save the current pose and the action from causing the collision. Then, the agent will first consider using the estimated motion action from Goal Transformer. But if the output action of GT is the same as the previous one leading to the collision, it will take a random motion action. After the agent reaches the target position, the agent will rotate and move around the place if the target object cannot be found in the current observation. If all attempts have been made and the object is still unobservable, the agent will consider it as a false detection situation and add the corresponding signal in the semantic map. The whole process can be found in Algorithm 4.

B.1.2. Task Completion Process of the JARVIS framework

We elaborate logic predicates for symbolic commonsense reasoning in Table B.1. In Algorithm 1, we show the overall process when our JARVIS framework tries to finish an EDH or TfD tasks, which includes two algorithms to implement symbolic commonsense reasoning predicates: Algorithm. 2 describes semantic map building process and language planning process, which including the task-level commonsense reasoning. Algorithm. 4 describes the detailed action generation process with action-level commonsense reasoning.

B.1.3. Experiment Details for Two Agent Task Completion (TATC)

We experiment with our JARVIS framework on TATC task in three different settings, where there are different constraints about how much information will be available to the *Commander* for generating instructions. As in Table. 3.2, in the first setting named full info. setting, the *Commander* has all information about the current subgoal, eg. *pickup knife*, target object location and the ground truth segmentation of the target object in the view. In this setting, same as the setting in [11], the *Commander* can specifically instructs where the *Follower* should interact to finish the current subgoal. In the second setting, we eliminate the ground truth segmentation of the target object from the information provided to the *Commander*. As a result, the *Commander* could still instruct the *Follower* to arrive near the target object but will not be able to explicitly tell the follower the ground truth location of the target object in the view. In the third setting, we further restrict the target object location (goal location) from the information to the *Commander* and thus the instruction generated from the *Commander* will not include

Table B.1.: Logic predicates for symbolic commonsense reasoning.

Task Common Sense		**Action Common Sense**	
$Movable(x)$	True if x can be moved by the agent.	$IsEmpty(x)$	True if location x is empty in map.
$Slicedable(x)$	True if x can be sliced into parts.	$Observe(x)$	True if x is observed.
$Openable(x)$	True if x can be opened or closed.	$Success(x, y)$	True if action x can be executed at state y.
$Toggleable(x)$	True if x can be toggled on or off.	$Near(x, y)$	True if the distance from the agent to x is smaller than y.
$IsReceptacle(x)$	True if x can support other objects.	$Move(x, y)$	$\exists P \subseteq V, \forall i \in P, IsEmpty(i)$ where V is the all possible paths from y to x.
$IsGrasped(agent)$	True if agent has grasped something.	$Collision(x, y)$	$Move(x, y) \wedge \neg Success(x, y)$
$Pick(agent, x)$	$\neg IsGrasped(agent) \wedge Moveable(x)$	$Target(x, y)$	$Observe(x) \wedge Move(x, y)$
$Place(agent, x)$	$IsGrasped(agent) \wedge IsReceptacle(x)$	$Interactive(action, x)$	$Observe(x) \wedge Near(x, 0.5)$
$Slice(agent, x)$	$Pick(agent, \text{“}Knife\text{”}) \wedge Sliceable(x)$	$Ignore(action, x)$	$\neg Observe(x) \wedge Near(x, 0.5)$ after exploring around x for serval times.

any ground truth information about where the target object is.

§ B.2. Notation Table

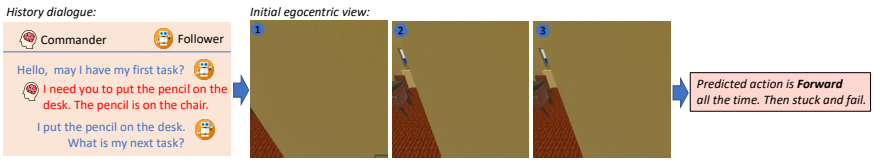
We describe the notation used in this paper in Table. B.2.

Table B.2.: Primary notation table.

Symbol	Description
$\mathcal{A}$	The action space.
A	A random variable representing the action sequence.
a_t	A specific action at time t.
$\int$	The environment state space.
s_i	The initial state.
s_f	The final state.
D	Dialogues consisting of pairs of players and utterance.
p	The players which can only be *Commander* or *Follower*.
u	The utterance.
V	A random variable representing the visual input of the *Follower*.
G	The sequence of sub-goals which is an intermediate guidance used for making decisions.

B. Appendix of JARVIS

(a)



(b)

Figure B.1.: EDH example. (a) shows an example of our JARVIS in EDH task, where the inputs are dialog history and sub-goal history (converted from action history input). The inputs are first interpreted by the Language Parsing Module to become sub-goals. Then, our Symbolic Reasoning Module will generate action predictions. The predicted actions will change the follower's egocentric views and the semantic map will be built up and completed gradually. 1 shows the agent is opening the fridge 2 shows the agent has placed the knife and navigate back to the fridge. 3 shows the agent is picking up the potato. (b) is an example demonstrating a typical way of how Episodic Transformer fails on EDH task. In this case, the E.T. model predicts "Forward" repetitively even facing the wall, therefore stuck at the current position.

Algorithm 1 Task Completion Process of the JARVIS framework

Require: History observations V_{history}, History agent actions A_{history}, Utterance u

$G_{\text{future}}, M_{\text{semantic}} \leftarrow$ Algorithm 2$(V_{\text{history}}, A_{\text{history}}, u)$
$pointer \leftarrow 0$
$fail\_times \leftarrow 0$
$step \leftarrow 0$
$need\_stop \leftarrow$ False
$a_{\text{prev}} \leftarrow A_{\text{history}}[-1]$ ▷ Previous action is the last in history
$v_{\text{curr}}, success\_execute \leftarrow$ Simulator.Step(a_{prev})

while not *need_stop* **do**
    $a_{\text{next}}, M_{\text{semantic}}, pointer \leftarrow$ Algorithm 4$(a_{\text{prev}}, v_{\text{curr}}, M_{\text{semantic}}, pointer, G_{\text{future}})$
    $v_{\text{curr}}, success\_execute \leftarrow$ Simulator.Step(a_{next})
    $a_{\text{prev}} \leftarrow a_{\text{next}}$
    $step \leftarrow step + 1$
    if not *success_execute* **and** a_{prev} is an interaction action **then**
        $fail\_times \leftarrow fail\_times + 1$
    end if
    if $fail\_times \geqslant 30$ **or** $step \geqslant 1000$ **or** a_{next} **is** `"Stop"` **then**
        $need\_stop \leftarrow$ True
    end if
end while

§ B.3. Case Study

In the EDH task, the agent can process history dialog and execute sub-goals, and plan future actions to complete the task. With well-designed Symbolic Commonsense Reasoning module, the agent can efficiently navigate to the target location and execute planned actions, as shown in Figure. B.1. In the TfD task, the agent is able to understand the dialog and break down the whole task into future sub-goals, and execute it correctly, as in Figure. B.2.

We also show an example of one classic error of episodic transformer in B.1b. The

Algorithm 2 Semantic Map Building and Language Planning

Require: History observations V_{history}, History agent actions A_{history}, Utterance u
Ensure: Future Subgoals G_{future}, Semantic map M_{semantic}

$G_{\text{history}} \leftarrow [\,]$, $M_{\text{semantic}} \leftarrow \emptyset$, *pointer* $\leftarrow 0$, *need_nav* $\leftarrow$ False
▷ Part 1: Process History and Update Semantic Map
for $i \leftarrow 0$ **to** $t-1$ **do**
  $a_i \leftarrow A_{\text{history}}[i]$, $v_i \leftarrow V_{\text{history}}[i]$
  if $a_i \in$ *interaction_action* **then**
    Target_object $\leftarrow a_i$.target_object
    if *need_nav* **then**
      $G_{\text{history}} \leftarrow G_{\text{history}}$ + [Navigate, *Target_object*] + [a_i.action_name, *Target_object*]
    else
      $G_{\text{history}} \leftarrow G_{\text{history}}$ + [a_i.action_name, *Target_object*]
    end if
    need_nav $\leftarrow$ False
  else
    need_nav $\leftarrow$ True
    $M_{\text{semantic}} \leftarrow \textsc{ObservationProject}(v_i, a_i, M_{\text{semantic}})$
  end if
end for
▷ Part 2: Planning and Common Sense Processing
$G_{\text{future}} \leftarrow \textsc{LanguagePlanner}(G_{\text{history}}, u)$
$G_{\text{future}} \leftarrow \textsc{TaskCommonSenseProcess}(G_{\text{future}})$ ▷ Detailed in Algorithm 3
Initialize Goal_Transformer with $V_{\text{history}}, A_{\text{history}}, G_{\text{history}}$

E.T. model will repetitively predict “Forward" when facing the wall, or “Pickup Place" in some other cases. This is because the agent can not correctly and robustly infer correct actions from all the input information.

Figure. B.3 illustrates how our JARVIS framework can be adapted to the TATC task under the setting that the commander can acquire state changes needed to be complete, ground truth segmentation, and object location. The symbolic commonsense reasoning action module of our JARVIS-based follower can generate actions to complete the sub-goals provided by the commander.

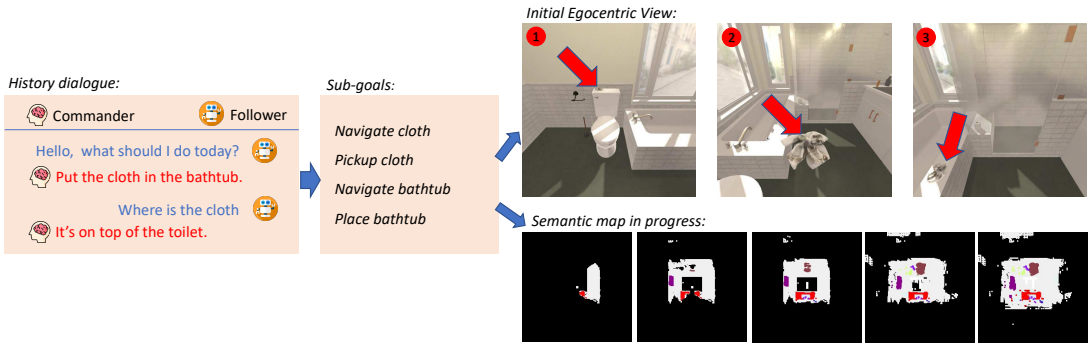


Figure B.2.: Successful TfD example from our JARVIS framework. According to the dialog, the language planner estimate four future sub-goals: ("Navigate cloth", "PickUp Cloth", "Navigate Bathtub", "Place Bathtub"). Then with the symbolic reasoning module, interaction and navigation actions are predicted. 1 shows the agent is finding the cloth. 2 shows the agent has picked up the cloth and then found the bathtub. 3 shows the agent can correctly put the cloth on the bathtub.

§ B.4. Error Analysis

Table B.3.: Average action failure rate on validation set for EDH, TfD and TATC tasks. When the time of action failure in a session reaches 30, the session will be forced to end, causing a task failure. The action failure exist widely in sessions.

Failure Mode	EDH(%)	TfD(%)	TATC(%)
No action failure	4.0	17.0	5.8
Action failures exist but are less than 30 times	78.8	79.9	69.0
Action failures reaches 30 times	17.2	3.1	25.2

According to Table. B.3, we find that all failed tasks include at least one action failure. For the shorter tasks, like EDH, the dominant situation is “Action failures are less than 30 times", while the longer tasks, like TfD and TATC, include even more action failures due to more sub-goals. To this aspect, we further analyze the action failure reason as in

Table B.4.: Action failure categorization result on a sub-set of validation set for all three tasks. The numbers show the rate of a certain failure action occurs among the total action failure time.

Action failure reason	EDH(%)	TfD(%)	TATC(%)
Hand occupied when picking up	3.20	1.46	1.04
Knife not in hand when cutting	3.12	1.82	1.36
Target object not support open/close	0.21	0.36	0.83
Cannot open when running	0.07	0.80	0.31
Blocked when moving	40.81	40.74	47.34
Collide when rotating	5.68	3.57	0.63
Collide when picking up	0.14	0.00	0.10
Invalid position for placing the held object	9.37	7.94	4.28
Too far to interact	8.52	10.13	4.28
Pouring action not available for the held object	1.13	1.82	0.10
Object not found at location or cannot be picked up	21.50	19.39	28.26
Cannot place without holding any object	6.17	11.30	8.03
Collide when placing	0.00	0.00	0.21
Target receptacle full when placing	0.07	0.66	3.23

Table. B.4, which shows a quantitative analysis of the failed actions categories in the validation set. We randomly sampled 50 failure episodes in both seen and unseen splits and computed the average ratio of each action failure category in all the failed actions. From Table. B.4, we notice that 'Blocked when moving' is the most frequent cause of failure action in all the three tasks, EDH, TfD, and TATC. The main reason is that errors in depth prediction cause semantic obstacle map errors. Besides, 'Too far to interact' is also mainly caused by wrongly predicting the 3D location of an object. To solve this, we need to improve the precision of the depth prediction model or design a more delicate and robust algorithm to update the semantic map. The second frequent failure action is 'Object not found at the location or cannot be picked up,' which could be due to the wrong predicted location of an object by Mask R-CNN. Moreover, it is also likely caused

by the situation where the target object is inside a receptacle that needs to be open first, as in Figure. B.4a. 'Invalid position for placing the held object' is also mainly caused by the segmentation error of Mask R-CNN. Thus, improving the ability of two perception models is the most effective way to avoid failure actions.

Other frequent reasons for failure are 'Cannot place without holding any object,' 'Hand occupied when picking up,' and 'Knife not in hand when cutting.' These are because of the wrong judgment of whether the former 'Place' and 'Pickup' actions have succeeded. For example, the agent might think it has picked up the knife, while it failed to pick up in fact, which causes the latter 'Knife not in hand when cutting.' We need to further refine our Symbolic Commonsense Reasoning Action module for these cases.

Instead of the failed actions, a miss-recognized object will also lead to an unsuccessful task. As in Figure B.4b, the tomato is falsely recognized as an apple, which leads to a failure when the target object is about "Tomato". Additionally, the false sub-goals estimations also contribute to the failures in task completion. In Section 3.4.5, we quantitatively analyze the performance of the Language Planning Module in the JARVIS framework. Here, we show a typical qualitative error result in Fig. B.5. According to the dialog, the ground truth future sub-goals need the follower to put all potatoes on the plate. However, the estimated sub-goals indicate one slice of potatoes and tomatoes will be put on the plate, which will cause the unsatisfied state changes.

Algorithm 3 Task Common Sense Processing

```
function TaskCommonSenseProcess(G)
    picked_object ← None
    for i ← 0 to G.length − 1 do
        a_i ← G[i].action, object_i ← G[i].target_object
        if a_i is "PickUp" then
            if Movable(object_i) then
                if picked_object ≠ None then
                    Add [Place, CounterTop] into G before step i
                end if
                picked_object ← object_i
            else
                Remove a_i from G
            end if
        else if a_i is "Place" then
            if IsReceptacle(object_i) then
                picked_object ← None
            else
                Remove a_i from G
            end if
        else if a_i is "Slice" then
            if Sliceable(object_i) then
                if picked_object ≠ "Knife" then
                    Add [...] into G before step i          ▷ Simplified for space
                    picked_object ← "Knife"
                end if
            else
                Remove a_i from G
            end if
        else if (a_i involves invalid Open/Close/Toggle) then
            Remove a_i from G
        end if
    end for
    return G
end function
```

Algorithm 4 Action Generation with Action Commonsense Reasoning

Require: Previous action a_{t-1}, Current observation v_t, Semantic map M_{t-1}, Subgoal pointer $pointer_{t-1}$, Future subgoal G_{future}

Ensure: Next action a_t, Updated semantic map M_t, Updated subgoal pointer $pointer_t$

stop_navigate ← False
a_t ← None
previous_success ← CheckSuccess(v_t) ▷ True if pixel changes > threshold

if *previous_success* **then**
    $pointer_t \leftarrow pointer_{t-1} + 1$
    ExecutePostActionProcess(a_{t-1}) ▷ Update planner states
    $M_t \leftarrow$ ObservationProject(v_t, a_{t-1}, M_{t-1})
else
    $pointer_t \leftarrow pointer_{t-1}$
    $M_t \leftarrow$ UpdateCollision(a_{t-1}, M_{t-1})
end if

$g_t \leftarrow G_{\text{future}}[pointer_t]$
Goal_ET_action ← Goal_Transformer.GetNextAction(v_t, a_{t-1}, g_t)
$object_t \leftarrow g_t$.target_object
target_observed ← Observe($object_t$) ▷ True if target is in current v_t
target_found ← FindInMap($object_t, M_t$) ▷ True if target is in M_t

if *target_found* **then**
    stop_navigate ← Near($object_t$, 0.5) ▷ True if distance to target < 0.5m
    if *stop_navigate* **then**
        if *target_observed* **then**
            $a_t \leftarrow$ Interaction(g_t.action, $object_t$)
        else
            $a_t \leftarrow$ FindTargetNearDestination(M_t) ▷ Perform delta moves around destination
        end if
    else
        $a_t \leftarrow$ FMM($object_t, M_t$)
    end if
else
    $a_t \leftarrow$ *Goal_ET_action*
end if

if not *previous_success* **and** $a_t = a_{t-1}$ **then**
    $a_t \leftarrow$ GenerateRandomFesibleAction(M_t) ▷ Random non-colliding movement
end if
return $a_t, M_t, pointer_t$

B. Appendix of JARVIS

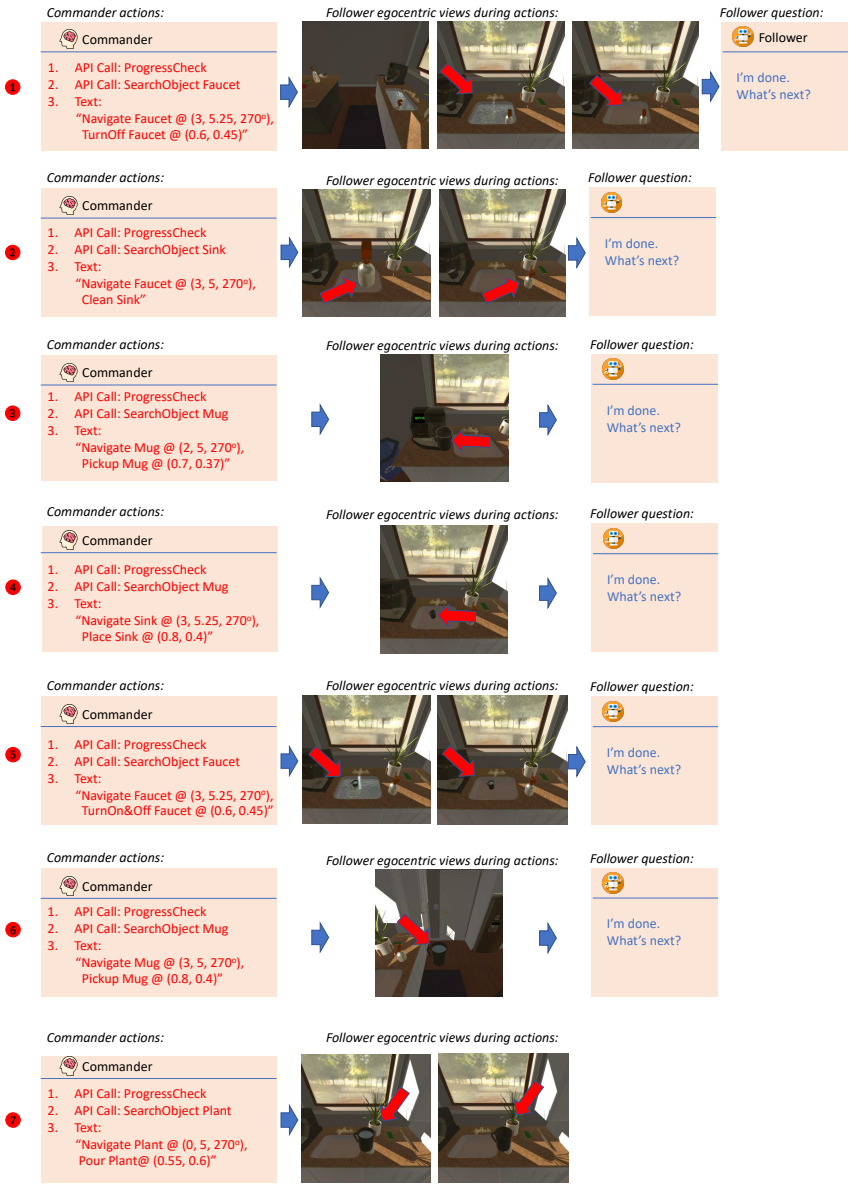


Figure B.3.: TATC "Water the plant" sequence. At each step the commander calls SearchObject to get an optimal navigation pose (e.g. 1: Navigate to Faucet @ (3, 5.25, 270°)), then uses ground-truth segmentation to compute an interaction target (e.g. 3: Pickup Mug @ (0.7, 0.37)). Steps 1–6 repeat for Faucet, Sink, Mug, Sink, Faucet, and Mug; step 7 pours water on the Plant @ (0.55, 0.6). After each sub-goal the follower executes in its egocentric view and reports completion.

(a) Object not found at location (b) Miss-recognized target object

Figure B.4.: Action failure examples

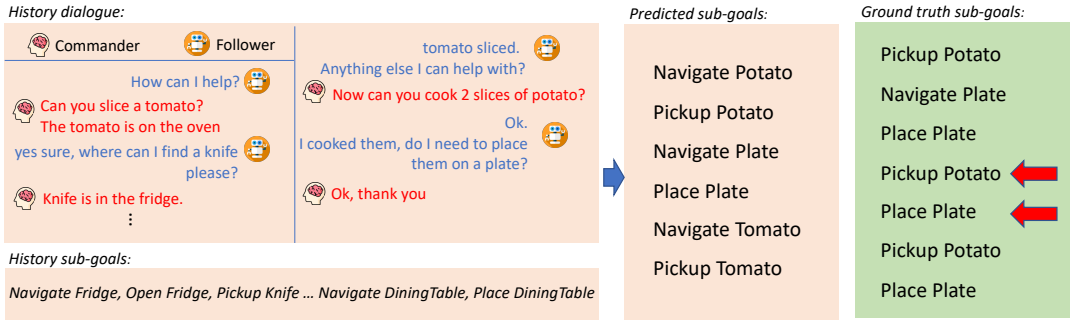


Figure B.5.: An example of sub-goal estimation failure. The predicted sub-goal lacks two important sub-goal as pointed by the red arrows, which causes the task failure in the end.

C. Appendix of MiniGPT-5

§ C.1. Experimental Settings

C.1.1. Datasets

CC3M [104]**:** Conceptual Captions (CC3M) dataset represents a remarkable collection of high-quality image captions, amassing approximately 3.3 million pairs of text and images from the internet. The CC3M dataset's diverse content, quality assurance, and support for multimodal learning make it a valuable asset for researchers and AI enthusiasts. Each dataset sample consists of an image accompanied by a corresponding text description, reflecting the richness of human language and visual perception. However, after accounting for license restrictions and eliminating invalid image links, the dataset comprises approximately 2.2 million data pairs suitable for training purposes and 10 thousand data pairs designated for validation.

VIST [105]**:** Visual Storytelling (VIST) dataset is an innovative compilation of visual narratives. The VIST dataset's engaging content, narrative structure, and emphasis on sequential understanding position it as an essential resource for researchers focusing on sequential image understanding. Each sequence within this dataset consists of five images accompanied by corresponding textual narratives, showcasing the intricate interplay

between visual imagery and storytelling. Designed to foster creativity and challenge conventional image-captioning models, the dataset provides a platform for training and validating algorithms capable of generating coherent and contextually relevant stories. After eliminating the invalid image links, we got over 65 thousand unique photos organized into more than 34 thousand storytelling sequences for training and 4 thousand sequences with 8 thousand images for validation.

MMDialog [107]**:** Multi-Modal Dialogue (MMDialog) dataset stands as the largest collection of multimodal conversation dialogues. The MMDialog dataset's extensive scale, real human-human chat content, and emphasis on multimodal open-domain conversations position it as an unparalleled asset for researchers and practitioners in artificial intelligence. Each dialogue within this dataset typically includes 2.59 images, integrated anywhere within the conversation, showcasing the complex interplay between text and visual elements. Designed to mirror real-world conversational dynamics, the dataset is a robust platform for developing, training, and validating algorithms capable of understanding and generating coherent dialogues that seamlessly blend textual and visual information.

C.1.2. Data Format

Pretraining Stage In the pretraining stage, we aim to synchronize the generative voken with the text-to-image model's conditional feature, focusing on single-turn text-image pairs. To achieve this, we utilize data from the CC3M dataset, constructing training samples by appending vokens as image placeholders after the captions, such as "a big black dog [IMG1] . . . [IMGn]." The Language Model (LLM) is then tasked with only

generating these placeholders for text creation, and the corresponding output hidden features are further employed to compute the conditional generation loss with the ground truth image.

Fine-tuning Stage In this stage, we utilize the VIST and MMDialog datasets, which contain multi-turn multimodal data. During training, we integrate placeholders for input images, such as '<Img><ImageHere></Img>', into the input text prompts when applicable. These prompts also encompass various instructions corresponding to different task types, with outputs manifesting as pure-text, pure-voken, or text-voken combinations. Below, we present example templates in the VIST dataset to illustrate the different task types:

- **Text Generation:** Input: "<History Context> What happens in the next scene image: <Img><ImageHere></Img>"; Output: "<Text Description>"
- **Image Generation:** Input: "<History Context> Generate an image with the scene description: [Text Description]"; Output: "[IMG1]...[IMGn]"
- **Text-Image Generation:** Input: "<History Context> What should happen then?"; Output: "<Text Description> [IMG1]...[IMGn]"

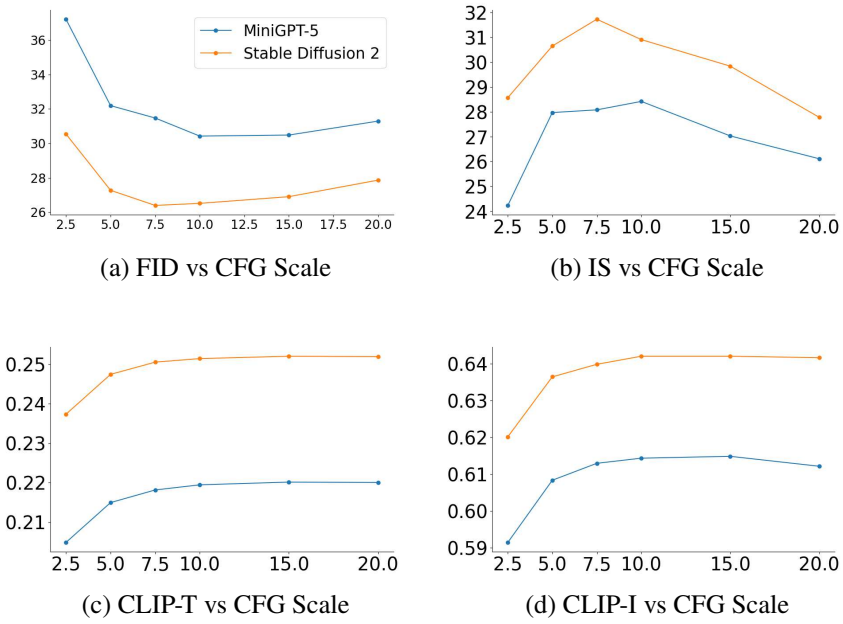
By structuring the input and output in this manner, we create a flexible framework that accommodates various multimodal tasks, enhancing the model's ability to interpret and generate textual and visual content. The history context in the VIST dataset includes all previous story steps with texts and images. In the MMDialog dataset, due to the limitation of computational resources, we only use up to one previous turn as the history context, and all data are formatted into the dialog.

Figure C.1.: Screenshot for human evaluation interface on the Amazon Mechanical Turk crowdsource evaluation platform. Output 1 is generated by MiniGPT-5, while output 2 is generated by the two-stage baseline.

§ C.2. More Experiments

C.2.1. Evaluation of Guidance Scale

Since our model incorporates CFG, evaluating how different guidance scales affect image generation is crucial. Therefore, we plotted several line charts in Fig C.2 to depict the changes in metrics with varying guidance scales. The figures reveal that the stable diffusion model and our model generate better images as the guidance scale increases. However, when the scale exceeds 10, the image semantic coherence stabilizes while the



(a) FID vs CFG Scale

(b) IS vs CFG Scale

(c) CLIP-T vs CFG Scale

(d) CLIP-I vs CFG Scale

Figure C.2.: Line charts for various metrics vs Classifier-free Guidance (CFG) scale on CC3M. The results suggest that our CFG strategy can exhibit comparable effectiveness to the CFG strategy employed in SD2, with the appropriate CFG scale significantly enhancing both image quality and coherence.

image quality declines. This suggests that the guidance scale should be set within a reasonable range for optimal image generation.

C.2.2. Evaluation of Voken Number

The voken features in our model are directly utilized as conditions in the text-to-image model, leading to the expectation that an increase in the number of vokens would enhance the model's representative capabilities. To validate this hypothesis, we experimented by training the model with varying numbers of vokens, ranging from 1 to 8. As illustrated in Fig C.3, the model's performance consistently improves with adding more vokens.

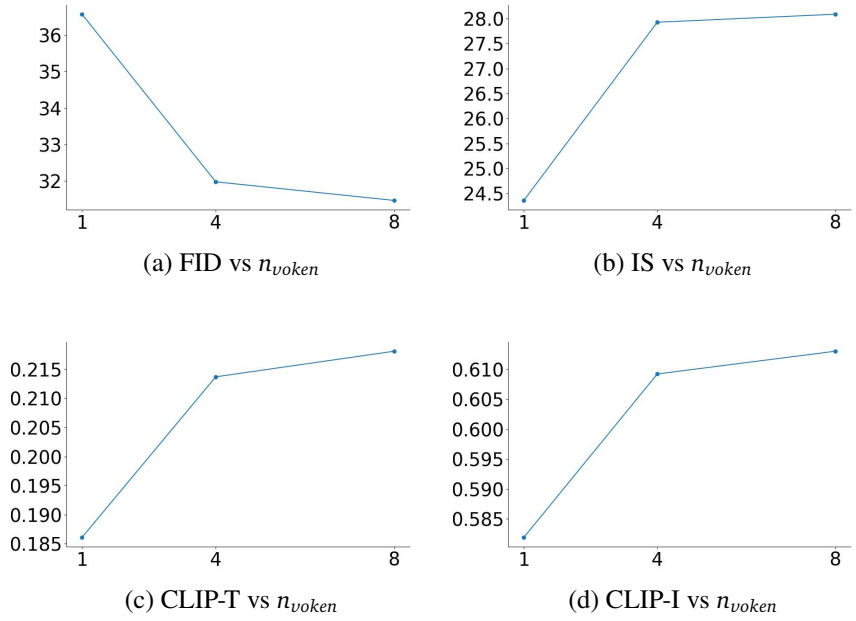
Table C.1.: Texture preservation on VIST (final–step).

Model	**FID (↓)**
GILL [125]	61.85
MiniGPT-5 (MiniGPT-4 + SD 2.1)	59.48
MiniGPT-5 (Qwen2.5-VL + SD3)	**56.32**

This improvement is particularly noticeable when the number of vokens is increased from 1 to 4, highlighting the significant role that vokens play in enhancing the model's effectiveness.

C.2.3. Texture Preservation via FID

To assess whether models preserve the textures present in prior images when generating the last image of a story, we evaluate on VIST in a "final–step" setting: for each story, all preceding step images and narrations are provided as context, and the model must produce the final image. We then compute FID between the set of generated final images and the ground-truth final images from VIST. This directly measures the realism and low-level appearance consistency of the predicted finals relative to the true finals under an identical multimodal context, shown in Table C.1. In this setting, ViLGen (Qwen2.5-VL + SD3) attains the lowest FID, indicating stronger retention of low-level textures in the generated final images.



(a) FID vs n_{voken} (b) IS vs n_{voken}

(c) CLIP-T vs n_{voken} (d) CLIP-I vs n_{voken}

Figure C.3.: Line charts for various metrics vs the number of vokens on CC3M. As the number of vokens increases, the image quality and CLIP scores improve. In this work, our default voken number is 8.

§ C.3. More Qualitative Examples

In this section, we provide additional qualitative examples to further demonstrate the capabilities of MiniGPT-5. Figures C.4,C.5,C.6, and C.7 showcase these examples across various datasets and settings.

Figure C.4 presents a comparative analysis on the VIST validation set, illustrating how MiniGPT-5 outperforms baseline models in terms of image generation quality and alignment with multimodal inputs. The examples highlight the superiority of MiniGPT-5 in generating images that closely match the given text prompts.

In Figure C.5, we focus on the performance of MiniGPT-5 in free multimodal generation scenarios. The results clearly indicate an improvement over the Two-Stage

baseline, emphasizing MiniGPT-5's ability to perform consistent and creative multimodal generation.

Figure C.6 showcases the application of MiniGPT-5 in the context of the MMDialog test set. Here, the emphasis is on free multimodal dialog generation, with MiniGPT-5 displaying a decent performance in generating coherent and contextually relevant multimodal dialogues.

Lastly, Figure C.7 highlights MiniGPT-5's performance in single text-to-image generation tasks on the CC3M validation set. The examples underline the model's proficiency in generating visually accurate and contextually appropriate images from textual descriptions, surpassing the performance of baseline models.

Each figure includes a clear depiction of input prompts (indicated in orange blocks) and the corresponding model outputs (in green blocks), providing a comprehensive view of MiniGPT-5's capabilities across different multimodal generation tasks.

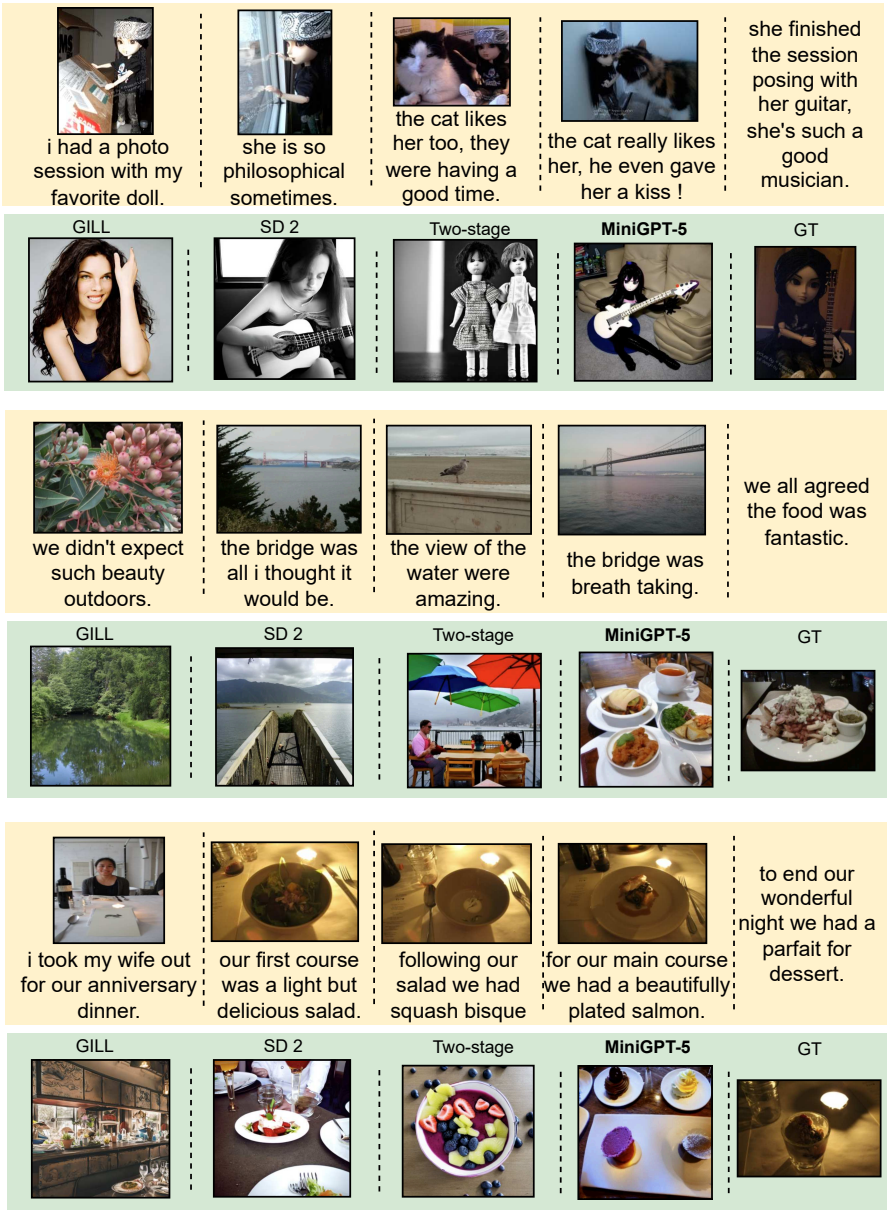


Figure C.4.: Comparative examples from MiniGPT-5 and baselines on the VIST validation set for image generation with multimodal input. Orange blocks denote input prompts, while green blocks show model outputs.

Figure C.5.: More qualitative examples from MiniGPT-5 and baselines on VIST validation set for free multimodal generation.

C. *Appendix of MiniGPT-5*

Figure C.6.: More qualitative examples from MiniGPT-5 on MMDialog test set for free multimodal dialog generation.

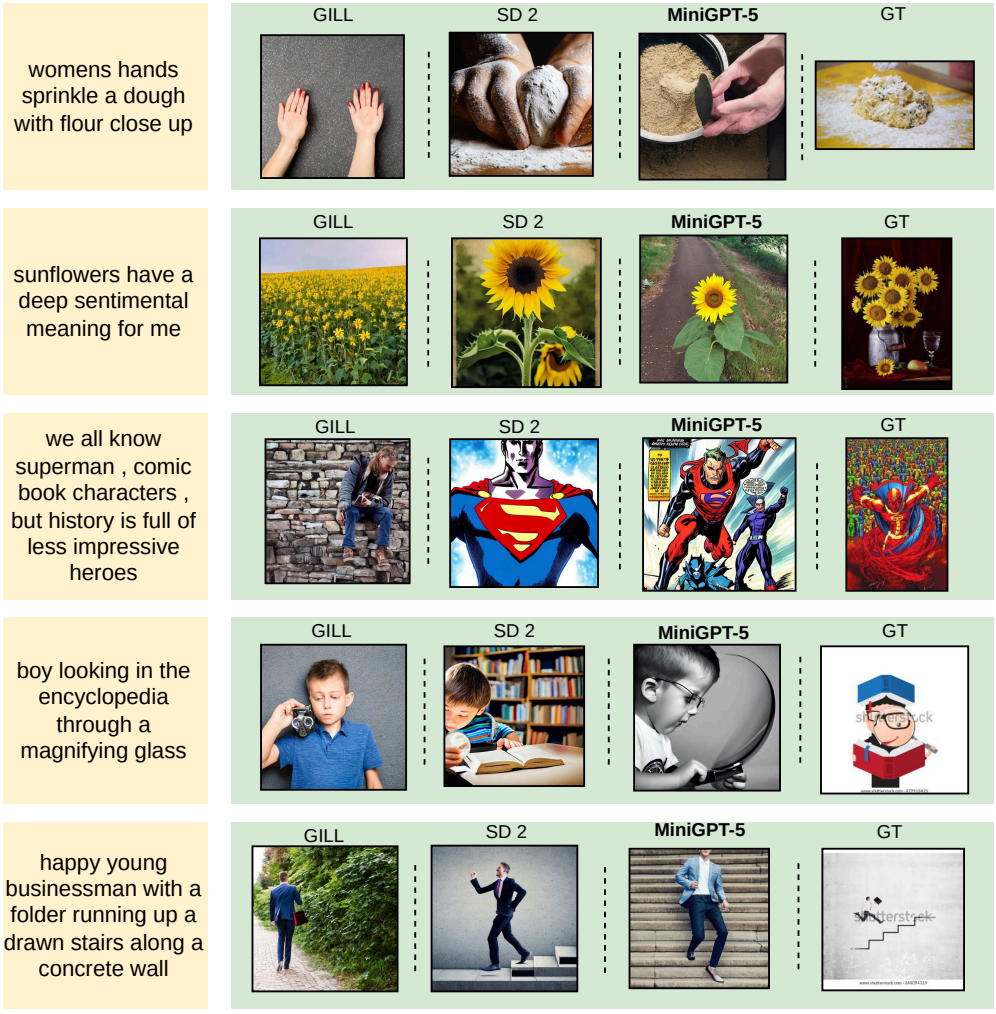


Figure C.7.: More qualitative examples from MiniGPT-5 and baselines on CC3M validation set for single text-to-image generation.

D. Appendix of SceneFuse-3D

§ D.1. More Results

D.1.1. Additional Qualitative Results

Figure D.1 presents additional qualitative results comparing SceneFuse-3D with Trellis [6], Hunyuan3D-2 [7], and TripoSG [8] across a diverse set of visual scenes. Besides, we collect some top-down view images from the internet for qualitative comparisons on real image inputs, shown in Figure D.2. These examples further demonstrate the robustness and generality of our approach across different architectural styles, spatial layouts, and artistic domains.

SceneFuse-3D consistently produces scene assets that are geometrically detailed, visually coherent, and well-aligned with the input reference images. In contrast, baseline models frequently suffer from artifacts such as repeated structures, layout collapse, or low-resolution textures. These comparisons further highlight the benefits of our region-based generation and spatial-aware inpainting pipeline in generating diverse 3D scenes from a single image without training.

Figure D.1.: **More qualitative comparisons between SceneFuse-3D and baselines.** From the image, we can find that SceneFuse-3D can generate more coherent scenes from diverse scene images. LGM is skipped since it fails to generate structured scenes for all inputs.

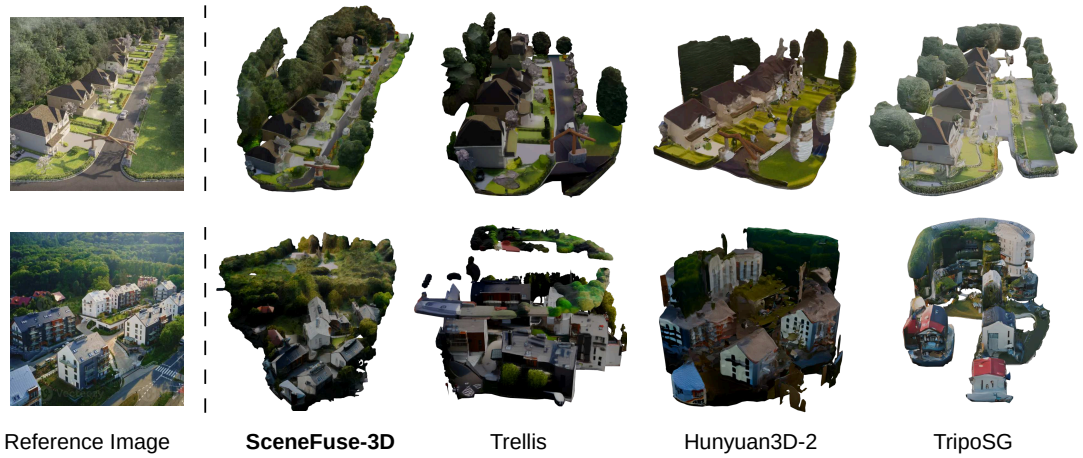


Figure D.2.: **Qualitative comparisons between SceneFuse-3D and baselines in real images.** We sample some top-down view images from the internet for comparisons. From the image, we can find that SceneFuse-3D can generate more coherent scenes.

D.1.2. Additional Baseline

WonderWorld [218] enables novel view synthesis of 3D scenes from a single image through text-guided inpainting, which is designed to be effective when a coarse single-view Gaussian representation is initialized and can be refined and enriched across different views. In our task of constructing a consistent 3D scene, we set up the camera trajectory to rotate along the vertical normal to the horizontal plane of the reference images for a comprehensive scene synthesis. However, WonderWorld struggles to generate view-consistent structures and often fails to update the Gaussians effectively due to suboptimal inpainting results based on the existing observations, shown in Fig. D.3.

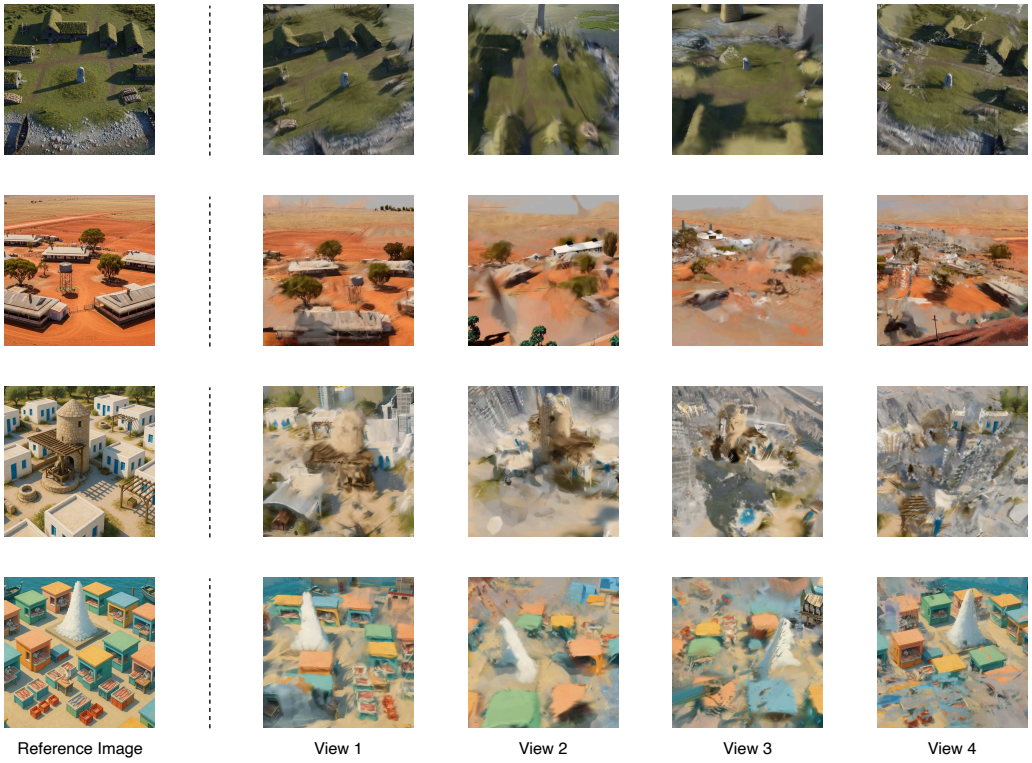


Figure D.3.: **Results of the WonderWorld.** As long as the viewpoint changes, we can find that WonderWorld cannot generate consistent geometry information for 3D scene generation.

§ D.2. Method Details

Region Extraction Strategy. To generate overlapping regions with balanced seams, we first compute the tight bounding box of all occupied voxels and tile it with a base grid of non-overlapping patches of size (p_x, p_y, p_z). For the vertical (z) axis, start positions are evenly interpolated so that the entire height is covered with the minimum number of full patches. Next, we insert *seam patches* to equalise overlap: for every pair of neighbouring base-grid origins we create an additional patch whose origin is the

Algorithm 5 Masked Rectified Flow for Completion Pipeline

Require: Learned flow field $v_\theta(x, t)$; known latent x_{known}; mask $m \in \{0, 1\}$; total steps T; resampling count U; minimum noise scale $\sigma_{\min}$
Ensure: Regenerated latent x_0

Sample $\epsilon \sim \mathcal{N}(0, I)$
Define forward_step$(x, t) = (1 - t)x + [\sigma_{\min} + (1 - \sigma_{\min})t]\epsilon$
Initialize $x_T \sim \mathcal{N}(0, I)$
for $t = T, T - 1, \ldots, 1$ **do**
    $t_{\text{prev}} \leftarrow t - 1$, $\Delta t \leftarrow t - t_{\text{prev}}$
    for $u = 1, \ldots, U$ **do**
        $v \leftarrow v_\theta(x_t, t)$
        $x_{t_{\text{prev}}} \leftarrow x_t - \Delta t\, v$
        $\hat{x}_{t_{\text{prev}}} \leftarrow$ forward_step$(x_{\text{known}}, t_{\text{prev}})$
        $x_{t_{\text{prev}}} \leftarrow m \odot x_{t_{\text{prev}}} + (1 - m) \odot \hat{x}_{t_{\text{prev}}}$
        if $u < U$ and $t > 1$ **then**
            $x_t \leftarrow$ forward_step$(x_{t_{\text{prev}}}, \Delta t)$
        else
            $x_t \leftarrow x_{t_{\text{prev}}}$
        end if
    end for
end for
return x_0

midpoint between them—first along the x-axis (keeping y, z fixed), then along the y-axis (keeping x, z fixed). The union of base and seam origins is deduplicated and sorted, after which the corresponding voxel sub-volumes are extracted. The procedure returns the list of patch origins and their binary masks, and is invoked once per scene to define the region schedule used in our region-wise generation and fusion pipeline.

Masked Rectified Flow Algorithm For completeness, we provide the full algorithmic details of the masked rectified flow completion process used in our spatial-aware 3D inpainting pipeline. While the core formulation is introduced in Section 5.3.3, this pseudocode (Algorithm 5) clarifies the iterative update, re-noising, and resampling

procedures that enable conditional generation of unknown regions while preserving the known latent structure.

Implementation Details For all rectified flow generators, we step the sampling time step $T = 50$. The classifier-free guidance scales are 7.5 and 5 for the spare structure generator and structured latent generator. Resampling time U during the masked rectified flow is set to 2. The whole pipeline can be loaded with an NVIDIA RTX A5000 GPU with 24G VRAM.

Runtime Analysis All timing measurements are obtained on NVIDIA RTX A5000 GPUs with 24G VRAM, using the same software stack as in our main experiments. The total runtime of SceneFuse-3D depends primarily on the number of detected landmarks and the number of non-empty regions. In our current implementation, we select at most 10 landmark instances; each landmark requires running a full Trellis image-to-3D generation followed by point-cloud alignment, which takes roughly 1 minute per landmark. For the scene body, we divide the scene into 8 overlapping regions and apply the sparse structure generator and structured latent generator with masked rectified flow to each region, resulting in approximately 40 seconds per regional completion. In practice, depending on how many landmarks are actually detected and how many regions contain active voxels, the end-to-end runtime per scene ranges from about 4 minutes (few landmarks, sparse regions) to about 14 minutes (many landmarks, dense regions). For comparison, under the same hardware and resolution settings, a single-pass run of Trellis takes about 50 seconds, Hunyuan3D-2 about 2 minutes, and TripoSG about 1.5 minutes for one scene. While our region-wise pipeline incurs additional overhead compared to these one-shot generators, it enables significantly improved geometry quality and layout

coherence at scene scale by reusing a fixed-capacity object-centric backbone.

Pipeline Decomposition and Modularity To clarify which components of our method are reused from existing work and which are newly introduced, we decompose SceneFuse-3D into three stages: *Scene Latent Initialization*, *Region Generation*, and *Region Fusion*. In the first stage, we build spatial priors and a scene-level structured latent from a single top-down image using off-the-shelf monocular depth estimation, Florence2, and SAM2, followed by point-cloud alignment and voxelization into a Trellis-style structured latent representation. In the second stage, we perform region-wise sparse structure and latent feature completion in 3D by reusing Trellis's sparse structure generator G_s and structured latent generator G_L, but adapting them with a masked rectified-flow scheme and local image conditioning to support spatially constrained, training-free completion at scene scale. In the final stage, we fuse all updated regions back into the global scene latent with landmark-aware fusion rules that enforce cross-region consistency and preserve key structures, and then decode the resulting latent into meshes and 3D Gaussians using Trellis object decoders and standard rendering. Table D.1 summarizes these three stages and highlights which parts are off-the-shelf and which are introduced in this work.

§ D.3. Experiment Settings

Test Set Generation To evaluate models' performance under stylistically diverse conditions, we curated a human-verified synthesized image test set with 100 top-down views. Given an example image, we ask ChatGPT-o3 [257] to generate image prompts for top-down scene views with these requirements: *1280 × 720 resolution, quasi-*

Table D.1.: Summary of the three-stage SceneFuse-3D pipeline.

Stage	**Main function**	**Components / status**
Scene Latent Initialization	Build spatial priors and a scene-level structured latent from a single top-down image	Monocular depth, Florence2, SAM2, point-cloud alignment and voxelization into a Trellis-style structured latent; mostly off-the-shelf with a scene-level extension (ours).
Region Generation	Region-wise sparse structure and latent feature completion in 3D	Trellis G_s and G_L applied on cropped regions with masked rectified flow and local image conditioning; our region-based masked 3D completion built on Trellis.
Region Fusion	Enforce cross-region consistency and decode the final 3D scene	Landmark-aware fusion of the scene latent (ours) plus Trellis object decoders and standard rendering (off-the-shelf).

orthographic three-quarter ("isometric") camera, one hero landmark at the image centre, 10–20 surrounding buildings, daylight illumination, and "no far-away object". Then, we used GPT-4o [217] to generate scene images according to the corresponding prompts.

Human Evaluation Given a reference image and observations of two generated scenes, we ask the human annotator to answer these three questions:

- Which scene, A or B, has geometry that is more detailed, precise, and closer to the reference image?
- Which scene, A or B, demonstrates a spatial layout and arrangement of objects that is more coherent and closely aligned with the layout in the reference image?
- Which scene, A or B, exhibits textures that are significantly more coherent and

consistent with the reference image?

GPT-4o-based Evaluation For GPT-based automatic evaluation, we ask the same questions as the human evaluation and prompt the model to directly return the answer with top-5 token log probability. Then, we extract the token probability of 'A' and 'B' (0 if not included) as the answer weights. We treat $P(\text{A})$ as a soft vote when our method is option A (and analogously for B). Weighted win rate is computed as $P(\text{win})$ if our model wins, and $1 - P(\text{lose})$ otherwise, then averaged across all pairs.

§ D.4. Limitation

Although SceneFuse-3D delivers strong scene-level results, several challenges remain. The pretrained 3D generator we adopt is trained on single-object imagery; even after region decomposition, the underlying distribution mismatch can lead to patch-level hallucinations—for example, duplicated facades, unrealistic roof shapes, or occasionally missing geometry in otherwise well-observed areas. Future work could mitigate this via scene-level fine-tuning or domain adaptation.

Geometric holes in our outputs currently arise from two main sources. First, our coarse spatial prior contains many voids where occlusions obscure geometry; regions dominated by such unobserved space sometimes inherit empty or over-smoothed surfaces from the generator. Second, distribution shift can cause the backbone to under-fill thin or high-frequency structures even in observed regions, leading to *observed-space* holes despite valid image evidence. Integrating uncertainty-aware depth completion, multi-view cues may yield denser scaffolds and more reliable inpainting.

E. Appendix of EvoScene

§ E.1. Method Details

Depth Estimation and Point Cloud Construction. We employ HunyuanWorld-Mirror [248] for metric depth estimation and camera parameter prediction from both single images and video frames. For the initial image I_0, the model predicts a metric depth map and camera intrinsics/extrinsics, which we back-project to obtain the initial point cloud $\mathcal{P}_0$. In subsequent iterations, we extract frames from generated videos and process each frame through HunyuanWorld-Mirror to obtain per-frame depth maps and camera parameters. To ensure geometric consistency, we apply multi-view geometric voting: for each candidate 3D point from frame k, we project it into all other frames and verify depth agreement within a tolerance of 0.1 meters. Points with support from at least 3 views are retained and assigned confidence scores based on the number of supporting views and local depth gradient magnitude (lower gradients indicate more reliable estimates). The filtered point clouds from all frames are merged with previous iterations' point clouds $\mathcal{P}_{t-1}$ through spatial binning at 5cm resolution, keeping points with highest confidence in each bin.

E. Appendix of EvoScene

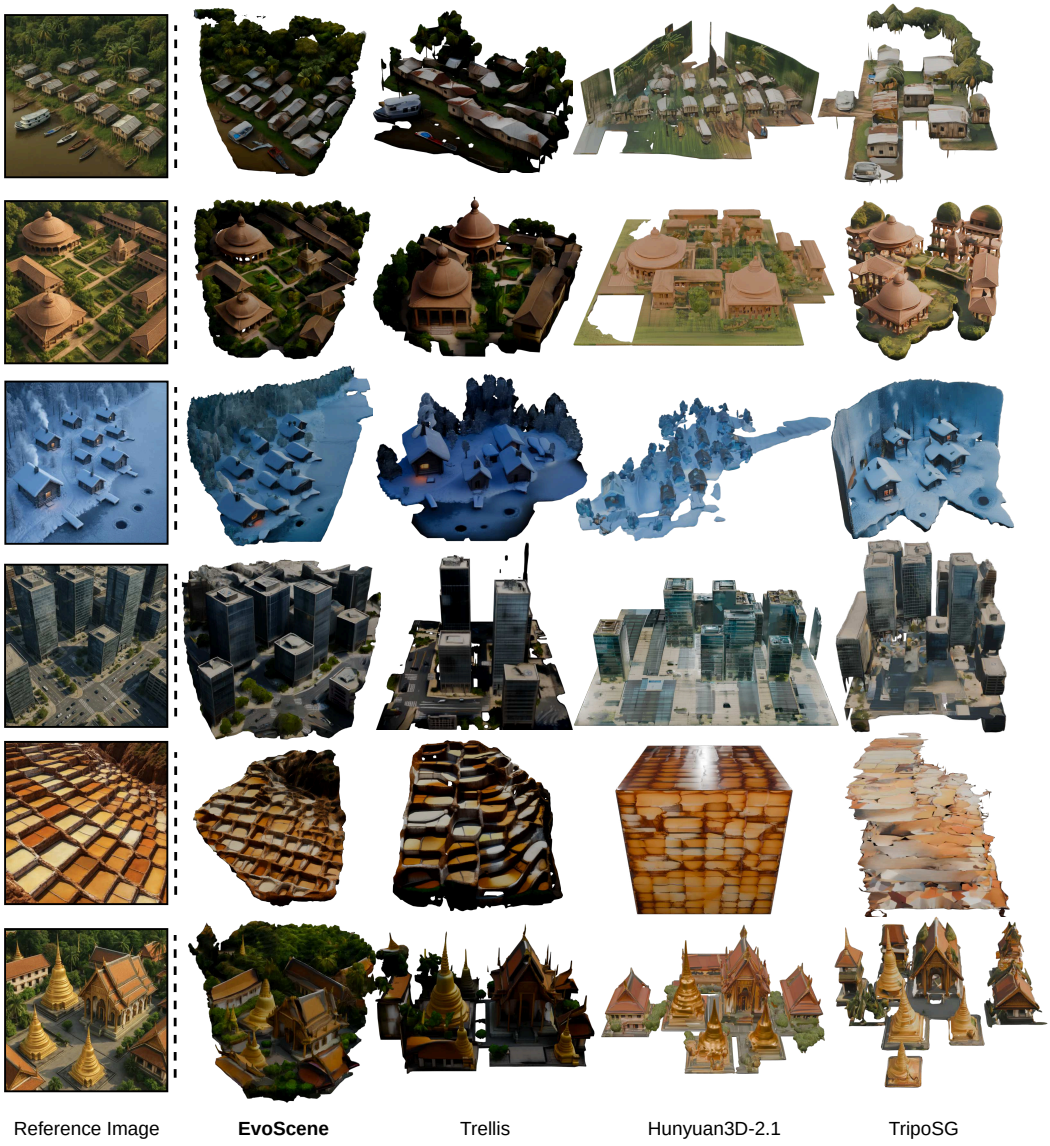


Figure E.1.: **Additional qualitative comparisons across diverse scene types.** From left to right: reference images, EvoScene (ours), Trellis, Hunyuan3D-2.1, and TripoSG. EvoScene produces complete geometry with preserved architectural details and photorealistic textures, while baselines exhibit fragmentation, flattened representations, or severe geometric distortions.

3D Scene Mesh Generation. We leverage Trellis [6], a SLAT-based 3D diffusion model, to generate complete scene meshes. The accumulated point cloud $\mathcal{P}_t$ is voxelized at resolution 128^3 to produce an occupancy grid O_t with three states: observed voxels (from point cloud), carved free-space (along viewing rays), and unknown occluded regions. For large scenes, we decompose the grid into overlapping 64^3 patches with 48-voxel overlap and extract corresponding image crops from accumulated observations. We employ Trellis's two-stage flow-matching generator: the sparse structure generator first completes the binary occupancy. The structured latent generator then produces per-voxel latent features conditioned on the completed occupancy and image crops. During generation, we apply test-time optimization with multi-view rendering: every denoising steps, we decode the current SLAT into 3D Gaussians, render from all accumulated views, and compute gradients of the photometric loss $\mathcal{L}_{\text{mv}}$ (Eq. 6.3.4) with weights $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 1.0$. These gradients guide 5 optimization steps (learning rate 1.0) before resuming diffusion. The final SLAT is decoded into both a mesh (via Marching Cubes) and 3D Gaussians for differentiable rendering.

Depth-Conditioned Video Generation. For novel view synthesis in Stage C, we render the completed mesh from $N = 121$ viewpoints along an orbital trajectory spanning the target rotation angle. Camera trajectories alternate between iterations: iteration 1 uses azimuth range $[0°, +45°]$ and iteration 2 uses $[0°, -45°]$, with elevation and radius fixed to the camera pose of the original reference image. For each viewpoint, we render a disparity map (inverse depth) from the mesh, normalize it to $[0, 1]$, and resize to 1024×1024. These disparity maps serve as geometric conditioning for Wan2.2-Fun-5B-Control [226], a depth-conditioned video diffusion model. We construct the prompt

as: "Camera orbiting around a static 3D scene, filling the blank space with detailed and realistic details, geometry consistent, high quality, 8k. The video shows [caption]", where the optional caption is generated by Qwen3-VL-2B [258] from the input image. Video generation uses 50 DDIM sampling steps with classifier-free guidance scale 7.5, and the conditioning strength (controlnet scale) is set to 0.4 to balance geometric fidelity and visual quality. The input image I_0 is injected as the first frame to ensure appearance consistency.

Implementation and Computational Cost. All experiments are conducted on a system with one NVIDIA H100 (80GB) GPUs. Each pipeline iteration takes approximately 8 minutes: depth estimation and point cloud construction ($\sim$ 1 min), mesh generation with test-time optimization ($\sim$ 3 min), and video generation ($\sim$ 3 min). The complete three-iteration pipeline requires approximately 20 minutes per scene (No video generation for the last iteration). Peak GPU memory usage is approximately 68GB during mesh generation (for batch processing of 64^3 patches) and 45GB during video generation. All models are used off-the-shelf without fine-tuning, demonstrating the training-free nature of our approach.

§ E.2. More Experiment Results

Evaluation Metrics We employ both human evaluation and GPT-4o-based automatic evaluation to comprehensively assess 3D scene generation quality across three key dimensions: geometric quality, layout coherence, and texture consistency. Given a reference image and observations of two generated scenes (A and B), we ask annotators

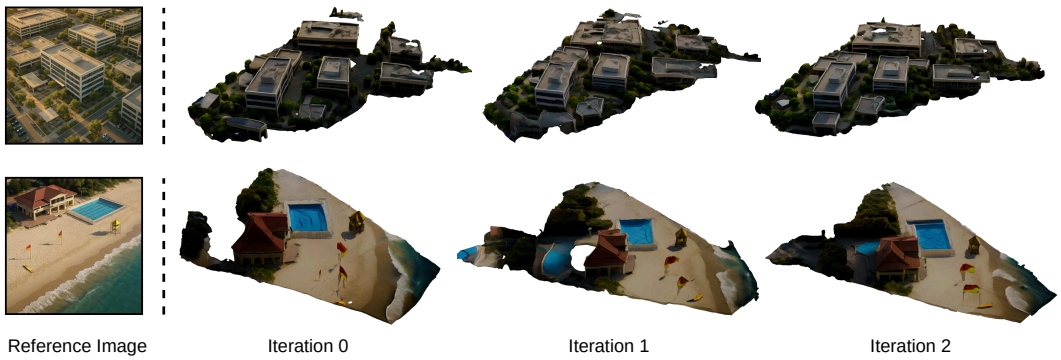


Figure E.2.: **Qualitative comparisons across different iterations.** The image shows the mesh differences between different iterations. With the iteration increasing, EvoScene can progressively improve the geometry and appearance quality, especially for unseen regions within the reference images.

(or GPT-4o) to answer three questions: *(i)* Which scene has geometry that is more detailed, precise, and closer to the reference image? *(ii)* Which scene demonstrates a spatial layout and arrangement of objects that is more coherent and closely aligned with the layout in the reference image? *(iii)* Which scene exhibits textures that are significantly more coherent and consistent with the reference image? For human evaluation, annotators select either A or B for each question, and we compute win rate as the percentage of times our method is preferred. For GPT-4o-based evaluation, we prompt the model to directly return the answer and extract top-5 token log probabilities, obtaining $P(\text{A})$ and $P(\text{B})$ (set to 0 if not in top-5). When our method is option A, we treat $P(\text{A})$ as a soft vote; when our method is option B, we use $P(\text{B})$. The weighted win rate is computed as $P(\text{win})$ if our method wins, and $1 - P(\text{lose})$ if it loses, then averaged across all comparisons to capture both preference frequency and confidence strength.

Extended Baseline Comparisons Figure E.1 presents additional qualitative comparisons between EvoScene and three strong baselines (Trellis [6], Hunyuan3D-2.1 [7],

and TripoSG [8]) across diverse scene categories including urban environments, natural landscapes, architectural landmarks, and residential areas. As discussed in the main paper (Section 4.2), EvoScene consistently produces complete geometry with preserved architectural details and photorealistic textures, while baselines exhibit fragmentation (TripoSG), flattened representations (Hunyuan3D-2.1), or incomplete geometry with missing regions (Trellis). These additional examples further validate our method's superior performance across varying scene complexity and demonstrate robust generalization to diverse visual content.

Iterative Refinement Visualization Figure E.2 illustrates the progressive improvement of geometry and appearance through our three-iteration self-evolving process (iterations 0, 1, 2). The initial iteration 0 reconstruction from only the input image produces reasonable but incomplete meshes with geometric errors in occluded regions and limited texture detail. Iteration 1 adds synthesized views from azimuth range $[0°, +45°]$, revealing previously occluded regions and improving background structure accuracy. Iteration 2 synthesizes complementary views from $[0°, -45°]$, achieving near-complete coverage with substantially refined geometric accuracy and texture consistency across the entire scene. The progression clearly demonstrates how our iterative geometry-appearance co-evolution progressively corrects initial errors, expands spatial coverage, and enhances both structural accuracy and visual realism, with the largest improvements occurring between iterations 0 and 1 when multi-view observations dramatically expand scene knowledge.

Bibliography

[1] OpenAI, *Gpt-4 technical report*, 2023.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, and Malcolm Reynolds, *Flamingo: a visual language model for few-shot learning*, Advances in Neural Information Processing Systems **35** (2022), 23716–23736.

[3] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny, *Minigpt-4: Enhancing vision-language understanding with advanced large language models*, arXiv preprint arXiv:2304.10592 (2023).

[4] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi, *Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models*, arXiv preprint arXiv:2301.12597 (2023).

[5] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer, *High-resolution image synthesis with latent diffusion models*, Cvpr, 2022.

[6] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang, *Structured 3d latents for scalable and versatile 3d generation*, arXiv preprint arXiv:2412.01506 (2024).

[7] Tencent Hunyuan3D Team, *Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation*, 2025.

[8] Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, and Wanli Ouyang, *Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models*, arXiv preprint arXiv:2502.06608 (2025).

[9] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison, *Rlbench: The robot learning benchmark & learning environment*, IEEE Robotics and Automation Letters **5** (2020), no. 2, 3019–3026.

[10] Mohit Shridhar, Lucas Manuelli, and Dieter Fox, *Cliport: What and where pathways for robotic manipulation* (2022), 894–906.

[11] Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur, *Teach: Task-driven embodied agents that chat*, arXiv preprint arXiv:2110.00534 (2021).

[12] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng, *Nerf: Representing scenes as neural radiance fields for view synthesis*, Communications of the ACM **65** (2021), no. 1, 99–106.

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis, *3d gaussian splatting for real-time radiance field rendering.*, ACM Trans. Graph. **42** (2023), no. 4, 139–1.

[14] Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer, *Causalworld: A robotic manipulation benchmark for causal structure and transfer learning*, arXiv preprint arXiv:2010.04296 (2020).

[15] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine, *Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning* (2020), 1094–1100.

[16] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, and Oleksandr Maksymets, *Habitat 2.0: Training home assistants to rearrange their habitat*, Advances in Neural Information Processing Systems **34** (2021).

[17] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox, *ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks* (2020).

[18] Kiana Ehsani, Winson Han, Alvaro Herrasti, Eli VanderBilt, Luca Weihs, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi, *Manipulathor: A framework for visual object manipulation* (2021), 4497–4506.

[19] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, and Karen Liu, *Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments* (2022), 477–490.

[20] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard, *Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks*, IEEE Robotics and Automation Letters (RA-L) **7** (2022), no. 3, 7327–7334.

[21] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín, *robosuite: A modular simulation framework and benchmark for robot learning*, arXiv preprint arXiv:2009.12293 (2020).

[22] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín, *What matters in learning from offline human demonstrations for robot manipulation*, arXiv preprint arXiv:2108.03298 (2021).

[23] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh, *Vqa: Visual question answering* (2015), 2425–2433.

[24] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra, *Embodied question answering* (2018), 1–10.

[25] Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang, *Vatex: A large-scale, high-quality multilingual dataset for video-and-language research*, The ieee international conference on computer vision (iccv), 2019October.

[26] Lianli Gao, Zhao Guo, Hanwang Zhang, Xing Xu, and Heng Tao Shen, *Video captioning with attention-based lstm and semantic consistency*, IEEE Transactions on Multimedia **19** (2017), no. 9, 2045–2055.

[27] Xin Wang, Wenhu Chen, Jiawei Wu, Yuan-Fang Wang, and William Yang Wang, *Video captioning via hierarchical reinforcement learning*, The ieee conference on computer vision and pattern recognition (cvpr), 2018.

[28] Qi Zhang, Zhen Lei, Zhaoxiang Zhang, and Stan Z Li, *Context-aware attention network for image-text retrieval* (2020), 3536–3545.

[29] Hui Chen, Guiguang Ding, Xudong Liu, Zijia Lin, Ji Liu, and Jungong Han, *Imram: Iterative matching with recurrent attention memory for cross-modal image-text retrieval* (2020), 12655–12663.

[30] Runtao Liu, Chenxi Liu, Yutong Bai, and Alan L Yuille, *Clevr-ref+: Diagnosing visual reasoning with referring expressions* (2019), 4185–4194.

[31] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum, *Clevrer: Collision events for video representation and reasoning*, arXiv preprint arXiv:1910.01442 (2019).

[32] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel, *Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments* (2018), 3674–3683.

[33] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang, *Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation*, Proceedings of the ieee conference on computer vision and pattern recognition, 2019, pp. 6629–6638.

[34] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang, *Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation*, The european conference on computer vision (eccv), 2018September.

[35] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel, *Reverie: Remote embodied visual referring expression in real indoor environments*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2020, pp. 9982–9991.

[36] Alexander Pashevich, Cordelia Schmid, and Chen Sun, *Episodic transformer for vision-and-language navigation* (2021), 15942–15952.

[37] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Eric Wang, *Vision-and-language navigation: A survey of tasks, methods, and future directions*, Proceedings of the 60th annual meeting of the association for computational linguistics (acl), 2022.

[38] Kaizhi Zheng, Kaiwen Zhou, Jing Gu, Yue Fan, Jialu Wang, Zonglin Li, Xuehai He, and Xin Eric Wang, *Jarvis: A neuro-symbolic commonsense reasoning framework for conversational embodied agents*, arXiv preprint arXiv:2208.13266 (2022).

[39] Oier Mees, Lukas Hermann, and Wolfram Burgard, *What matters in language conditioned robotic imitation learning*, arXiv preprint arXiv:2204.06252 (2022).

[40] Weiyu Liu, Chris Paxton, Tucker Hermans, and Dieter Fox, *Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects*, arXiv preprint arXiv:2110.10189 (2021).

[41] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor, *Language-conditioned imitation learning for robot manipulation tasks*, 2020.

[42] Corey Lynch and Pierre Sermanet, *Language conditioned imitation learning over unstructured data*, arXiv preprint arXiv:2005.07648 (2020).

[43] Hanbo Zhang, Yunfan Lu, Cunjun Yu, David Hsu, Xuguang La, and Nanning Zheng, *Invigorate: Interactive visual grounding and grasping in clutter*, arXiv preprint arXiv:2108.11092 (2021).

[44] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg, *Concept2robot: Learning manipulation concepts from instructions and human demonstrations*, The International Journal of Robotics Research **40** (2021), no. 12-14, 1419–1434.

[45] Prasoon Goyal, Raymond J Mooney, and Scott Niekum, *Zero-shot task adaptation using natural language*, arXiv preprint arXiv:2106.02972 (2021).

[46] E. Rohmer, S. P. N. Singh, and M. Freese, *V-rep: A versatile and scalable robot simulation framework* (2013), 1321–1326.

[47] Stephen James, Marc Freese, and Andrew J. Davison, *Pyrep: Bringing v-rep to deep robot learning*, arXiv preprint arXiv:1906.11176 (2019).

[48] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, and Tanish Jain, *Igibson 2.0: Object-centric simulation for robot learning of everyday household tasks*, arXiv preprint arXiv:2108.03272 (2021).

[49] Wentao Yuan, Chris Paxton, Karthik Desingh, and Dieter Fox, *Sornet: Spatial object-centric representations for sequential manipulation*, Conference on Robot Learning (2022), 148–157.

[50] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt, *Grasp pose detection in point clouds*, The International Journal of Robotics Research **36** (2017), no. 13-14, 1455–1473.

[51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, and Jack Clark, *Learning transferable visual models from natural language supervision* (2021), 8748–8763.

[52] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee, *Transporter networks: Rearranging the visual world for robotic manipulation*, Conference on Robot Learning (CoRL) (2020).

[53] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, *An image is worth 16x16 words: Transformers for image recognition at scale*, International conference on learning representations, 2021.

[54] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick, *Mask r-cnn*, Proceedings of the ieee international conference on computer vision (iccv), 2017Oct.

[55] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer, *BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, Proceedings of the 58th annual meeting of the association for computational linguistics, July 2020, pp. 7871–7880.

[56] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu, *Plug and play language models: A simple approach to controlled text generation*, International conference on learning representations, 2020.

[57] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi, *Target-driven visual navigation in indoor scenes using deep reinforcement learning*, 2017 ieee international conference on robotics and automation (icra), 2017, pp. 3357–3364.

[58] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi, *Visual semantic navigation using scene priors*, 2019.

[59] Dongshin Kim, Jie Sun, Sang Min Oh, J.M. Rehg, and A.F. Bobick, *Traversability classification using unsupervised on-line visual learning for outdoor robot navigation*, Proceedings 2006 ieee international conference on robotics and automation, 2006. icra 2006., 2006, pp. 518–525.

[60] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel, *Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments*, Proceedings of the ieee conference on computer vision and pattern recognition (cvpr), 2018June.

[61] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge, *Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding*, Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp), November 2020, pp. 4392–4412.

[62] Wang Zhu, Hexiang Hu, Jiacheng Chen, Zhiwei Deng, Vihan Jain, Eugene Ie, and Fei Sha, *Babywalk: Going farther in vision-and-language navigation by taking baby steps*, Proceedings of the 58th annual meeting of the association for computational linguistics, July 2020, pp. 2539–2556.

[63] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi, *Touchdown: Natural language navigation and spatial reasoning in visual street environments*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr), 2019June.

[64] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel, *Reverie: Remote embodied visual referring expression in real indoor environments*, Proceedings of the ieee conference on computer vision and pattern recognition (cvpr), 2020.

[65] Arun Balajee Vasudevan, Dengxin Dai, and Luc Van Gool, *Talk2nav: Long-range vision-and-language navigation with dual attention and spatial memory*, Int. J. Comput. Vision **129** (2021jan), no. 1, 246–266.

[66] Keji He, Yan Huang, Qi Wu, Jianhua Yang, Dong An, Shuanglin Sima, and Liang Wang, *Landmark-rxr: Solving vision-and-language navigation with fine-grained alignment supervision*, Advances in neural information processing systems, 2021, pp. 652–663.

[67] Jing Gu, Eliana Stefani, Qi Wu, Jesse Thomason, and Xin Wang, *Vision-and-language navigation: A survey of tasks, methods, and future directions*, Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: Long papers), 2022, pp. 7606–7623.

[68] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer, *Vision-and-dialog navigation*, Conference on robot learning (corl), 2019.

[69] Shurjo Banerjee, Jesse Thomason, and Jason J. Corso, *The robotslang benchmark: Dialog-guided robot localization and navigation*, 4th conference on robot learning, corl 2020, 16-18 november 2020, virtual event / cambridge, ma, USA, 2020, pp. 1384–1393.

[70] Khanh Nguyen, Debadeepta Dey, Chris Brockett, and Bill Dolan, *Vision-based navigation with language-based assistance via imitation learning with indirect intervention*, The ieee conference on computer vision and pattern recognition (cvpr), 2019June.

[71] Khanh Nguyen and Hal Daumé III, *Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning*, Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp), November 2019, pp. 684–695.

[72] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox, *Alfred: A benchmark for interpreting grounded instructions for everyday tasks*, Ieee/cvf conference on computer vision and pattern recognition (cvpr), 2020June.

[73] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi, *Mapping instructions to actions in 3d environments with visual goal prediction*, Proceedings of the 2018 conference on empirical methods in natural language processing, October 2018, pp. 2667–2678.

[74] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi, *Iqa: Visual question answering in interactive environments*, Proceedings of the ieee conference on computer vision and pattern recognition (cvpr), 2018June.

[75] Anjali Narayan-Chen, Prashant Jayannavar, and Julia Hockenmaier, *Collaborative dialogue in Minecraft*, Proceedings of the 57th annual meeting of the association for computational linguistics, July 2019, pp. 5405–5415.

[76] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang, *Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr), 2019June.

[77] Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould, *Vln bert: A recurrent vision-and-language bert for navigation*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr), 2021June, pp. 1643–1653.

[78] Hao Tan, Licheng Yu, and Mohit Bansal, *Learning to navigate unseen environments: Back translation with environmental dropout*, Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers), June 2019, pp. 2610–2621.

[79] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell, *Speaker-follower models for vision-and-language navigation*, Neurips, 2018.

[80] Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid, *Airbert: In-domain pretraining for vision-and-language navigation*, Proceedings of the ieee/cvf international conference on computer vision (iccv), 2021October, pp. 1634–1643.

[81] Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev, *History aware multimodal transformer for vision-and-language navigation*, Neurips, 2021.

[82] Xin Eric Wang, Vihan Jain, Eugene Ie, William Yang Wang, Zornitsa Kozareva, and Sujith Ravi, *Environment-agnostic multitask learning for natural language grounded navigation*, Computer vision – eccv 2020: 16th european conference, glasgow, uk, august 23–28, 2020, proceedings, part xxiv, 2020, pp. 413–430.

[83] Yi Zhu, Fengda Zhu, Zhaohuan Zhan, Bingqian Lin, Jianbin Jiao, Xiaojun Chang, and Xiaodan Liang, *Vision-dialog navigation by exploring cross-modal memory*, Ieee/cvf conference on computer vision and pattern recognition (cvpr), 2020June.

[84] Hyounghun Kim, Jialu Li, and Mohit Bansal, *NDH-full: Learning and evaluating navigational agents on full-length dialogue*, Proceedings of the 2021 conference on empirical methods in natural language processing, November 2021, pp. 6432–6442.

[85] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das, *Habitat-web: Learning embodied object-search strategies from human demonstrations at scale*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2022, pp. 5173–5183.

[86] Alexander Pashevich, Cordelia Schmid, and Chen Sun, *Episodic transformer for vision-and-language navigation*, Proceedings of the ieee/cvf international conference on computer vision (iccv), 2021October, pp. 15942–15952.

[87] So Yeon Min, Devendra Singh Chaplot, Pradeep Kumar Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov, *FILM: Following instructions in language with modular methods*, International conference on learning representations, 2022.

[88] Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi, *A persistent spatial semantic representation for high-level natural language instruction execution*, 5th annual conference on robot learning, 2021.

[89] Chan Hee Song, Jihyung Kil, Tai-Yu Pan, Brian M. Sadler, Wei-Lun Chao, and Yu Su, *One step at a time: Long-horizon vision-and-language navigation with milestones*, Ieee/cvf conference on computer vision and pattern recognition (cvpr), 2022June.

[90] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers), June 2019, pp. 4171–4186.

[91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep residual learning for image recognition*, 2016 ieee conference on computer vision and pattern recognition (cvpr), 2016, pp. 770–778.

[92] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu, *The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision*, International conference on learning representations, 2019.

[93] Zhenfang Chen, Jiayuan Mao, Jiajun Wu, Kwan-Yee K Wong, Joshua B. Tenenbaum, and Chuang Gan, *Grounding physical concepts of objects and events through dynamic visual reasoning*, International conference on learning representations, 2021.

[94] Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner, *Neural module networks for reasoning over text*, International conference on learning representations, 2020.

[95] Drew Hudson and Christopher D Manning, *Learning by abstraction: The neural state machine*, Advances in neural information processing systems, 2019.

[96] Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi, *proscript: Partially ordered scripts generation via pre-trained language models*, arXiv preprint arXiv:2104.08251 (2021).

[97] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch, *Language models as zero-shot planners: Extracting actionable knowledge for embodied agents*, arXiv preprint arXiv:2201.07207 (2022).

[98] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, *U-net: Convolutional networks for biomedical image segmentation*, International conference on medical image computing and computer-assisted intervention, 2015, pp. 234–241.

[99] James A Sethian, *Fast marching methods*, SIAM review **41** (1999), no. 2, 199–235.

[100] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing, *Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality*, 2023.

[101] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, and Alex Ray, *Training language models to follow instructions with human feedback*, Advances in Neural Information Processing Systems **35** (2022), 27730–27744.

[102] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan, *Visual chatgpt: Talking, drawing and editing with visual foundation models*, arXiv preprint arXiv:2303.04671 (2023).

[103] Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill, *Multimodal few-shot learning with frozen language models*, Advances in Neural Information Processing Systems **34** (2021), 200–212.

[104] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut, *Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning*, Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers), 2018, pp. 2556–2565.

[105] Ting-Hao K. Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Aishwarya Agrawal, Ross Girshick, Xiaodong He, Pushmeet Kohli, and Dhruv Batra, *Visual storytelling*, 15th annual conference of the north american chapter of the association for computational linguistics (naacl 2016), 2016.

[106] Hao Tan and Mohit Bansal, *Vokenization: Improving language understanding with contextualized, visual-grounded supervision*, arXiv preprint arXiv:2010.06775 (2020).

[107] Jiazhan Feng, Qingfeng Sun, Can Xu, Pu Zhao, Yaming Yang, Chongyang Tao, Dongyan Zhao, and Qingwei Lin, *Mmdialog: A large-scale multi-turn dialogue dataset towards multi-modal open-domain conversation*, arXiv preprint arXiv:2211.05719 (2022).

[108] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi, *Instructblip: Towards general-purpose vision-language models with instruction tuning*, 2023.

[109] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu, *Otter: A multi-modal model with in-context instruction tuning*, arXiv preprint arXiv:2305.03726 (2023).

[110] Dongxu Li, Junnan Li, Hung Le, Guangsen Wang, Silvio Savarese, and Steven C.H. Hoi, *LAVIS: A one-stop library for language-vision intelligence*, Proceedings of the 61st annual meeting of the association for computational linguistics (volume 3: System demonstrations), July 2023, pp. 31–41.

[111] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, and Jun Tang, *Qwen2. 5-vl technical report*, arXiv preprint arXiv:2502.13923 (2025).

[112] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee, *Generative adversarial text to image synthesis*, International conference on machine learning, 2016, pp. 1060–1069.

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

[114] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, and Tim Salimans, *Photorealistic text-to-image diffusion models with deep language understanding*, Advances in Neural Information Processing Systems **35** (2022), 36479–36494.

[115] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer, *High-resolution image synthesis with latent diffusion models*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2022, pp. 10684–10695.

[116] Jing Gu, Yilin Wang, Nanxuan Zhao, Tsu-Jui Fu, Wei Xiong, Qing Liu, Zhifei Zhang, He Zhang, Jianming Zhang, HyunJoon Jung, and Xin Eric Wang, *Photoswap: Personalized subject swapping in images*, 2023.

[117] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen, *Glide: Towards photorealistic image generation and editing with text-guided diffusion models*, arXiv preprint arXiv:2112.10741 (2021).

[118] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever, *Zero-shot text-to-image generation*, International conference on machine learning, 2021, pp. 8821–8831.

[119] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, and Burcu Karagol Ayan, *Scaling autoregressive models for content-rich text-to-image generation*, arXiv preprint arXiv:2206.10789 **2** (2022), no. 3, 5.

[120] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, and Michael Rubinstein, *Muse: Text-to-image generation via masked generative transformers*, arXiv preprint arXiv:2301.00704 (2023).

[121] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, and Frederic Boesel, *Scaling rectified flow transformers for high-resolution image synthesis*, Forty-first international conference on machine learning, 2024.

[122] Qingfeng Sun, Yujing Wang, Can Xu, Kai Zheng, Yaming Yang, Huang Hu, Fei Xu, Jessica Zhang, Xiubo Geng, and Daxin Jiang, *Multimodal dialogue response generation*, arXiv preprint arXiv:2110.08515 (2021).

[123] Emanuele Aiello, Lili Yu, Yixin Nie, Armen Aghajanyan, and Barlas Oguz, *Jointly training large autoregressive multimodal models*, arXiv preprint arXiv:2309.15564 (2023).

[124] Yuying Ge, Yixiao Ge, Ziyun Zeng, Xintao Wang, and Ying Shan, *Planting a seed of vision in large language model*, arXiv preprint arXiv:2307.08041 (2023).

[125] Jing Yu Koh, Daniel Fried, and Ruslan Salakhutdinov, *Generating images with multimodal language models*, arXiv preprint arXiv:2305.17216 (2023).

[126] Lili Yu, Bowen Shi, Ramakanth Pasunuru, Benjamin Muller, Olga Golovneva, Tianlu Wang, Arun Babu, Binh Tang, Brian Karrer, and Shelly Sheynin, *Scaling autoregressive multi-modal models: Pretraining and instruction tuning*, arXiv preprint arXiv:2309.02591 (2023).

[127] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua, *Next-gpt: Any-to-any multimodal llm*, CoRR **abs/2309.05519** (2023).

[128] Runpei Dong, Chunrui Han, Yuang Peng, Zekun Qi, Zheng Ge, Jinrong Yang, Liang Zhao, Jianjian Sun, Hongyu Zhou, and Haoran Wei, *Dreamllm: Synergistic multimodal comprehension and creation*, arXiv preprint arXiv:2309.11499 (2023).

[129] Quan Sun, Qiying Yu, Yufeng Cui, Fan Zhang, Xiaosong Zhang, Yueze Wang, Hongcheng Gao, Jingjing Liu, Tiejun Huang, and Xinlong Wang, *Emu: Generative pretraining in multimodality*, The twelfth international conference on learning representations, 2023.

[130] Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao, *Eva-clip: Improved training techniques for clip at scale*, arXiv preprint arXiv:2303.15389 (2023).

[131] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, and Faisal Azhar, *Llama: Open and efficient foundation language models*, arXiv preprint arXiv:2302.13971 (2023).

[132] Chameleon Team, *Chameleon: Mixed-modal early-fusion foundation models* (2024), available at 2405.09818.

[133] X. Wang, *Emu3: Next-token prediction is all you need* (2024), available at 2409.18869.

[134] J. Xie, *Show-o: One single transformer to unify multimodal understanding and generation*, Iclr, 2025.

[135] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly, *Parameter-efficient transfer learning for nlp*, International conference on machine learning, 2019, pp. 2790–2799.

[136] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen, *Lora: Low-rank adaptation of large language models*, arXiv preprint arXiv:2106.09685 (2021).

[137] Xiang Lisa Li and Percy Liang, *Prefix-tuning: Optimizing continuous prompts for generation*, arXiv preprint arXiv:2101.00190 (2021).

[138] Renrui Zhang, Rongyao Fang, Peng Gao, Wei Zhang, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li, *Tip-adapter: Training-free clip-adapter for better vision-language modeling*, arXiv preprint arXiv:2111.03930 (2021).

[139] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer, *Qlora: Efficient finetuning of quantized llms*, arXiv preprint arXiv:2305.14314 (2023).

[140] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, *Attention is all you need*, Advances in neural information processing systems (2017), 5998–6008.

[141] Diederik P Kingma and Max Welling, *Auto-encoding variational bayes*, arXiv preprint arXiv:1312.6114 (2013).

[142] Jonathan Ho and Tim Salimans, *Classifier-free diffusion guidance*, arXiv preprint arXiv:2207.12598 (2022).

[143] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan, *Visual chatgpt: Talking, drawing and editing with visual foundation models*, arXiv preprint arXiv:2303.04671 (2023).

[144] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, *Improved techniques for training gans*, Advances in neural information processing systems **29** (2016).

[145] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter, *Gans trained by a two time-scale update rule converge to a local nash equilibrium*, Advances in neural information processing systems **30** (2017).

[146] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, *Bleu: a method for automatic evaluation of machine translation*, Proceedings of the 40th annual meeting on association for computational linguistics, 2002, pp. 311–318.

[147] Chin-Yew Lin, *Rouge: A package for automatic evaluation of summaries*, Text summarization branches out, 2004, pp. 74–81.

[148] Satanjeev Banerjee and Alon Lavie, *Meteor: An automatic metric for mt evaluation with improved correlation with human judgments*, Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, 2005, pp. 65–72.

[149] Nils Reimers and Iryna Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, arXiv preprint arXiv:1908.10084 (2019).

[150] Kevin Crowston, *Amazon mechanical turk: A research tool for organizations and information systems scholars*, Shaping the future of ict research. methods and approaches: Ifip wg 8.2, working conference, tampa, fl, usa, december 13-14, 2012. proceedings, 2012, pp. 210–221.

[151] P. Esser, *Scaling rectified flow transformers for high-resolution image synthesis*, Icml, 2024.

[152] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee, *Improved baselines with visual instruction tuning*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2024, pp. 26296–26306.

[153] S. Bai, *Qwen2.5-vl technical report* (2025), available at 2502.13923.

[154] Zhengqi Li, Qianqian Wang, Noah Snavely, and Angjoo Kanazawa, *Infinitenature-zero: Learning perpetual view generation of natural scenes from single images*, European conference on computer vision, 2022, pp. 515–534.

[155] Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee, *Luciddreamer: Domain-free generation of 3d gaussian splatting scenes*, arXiv preprint arXiv:2311.13384 (2023).

[156] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao, *Text2nerf: Text-driven 3d scene generation with neural radiance fields*, IEEE Transactions on Visualization and Computer Graphics **30** (2024), no. 12, 7749–7762.

[157] Hong-Xing Yu, Haoyi Duan, Junhwa Hur, Kyle Sargent, Michael Rubinstein, William T Freeman, Forrester Cole, Deqing Sun, Noah Snavely, and Jiajun Wu, *Wonderjourney: Going from anywhere to everywhere*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2024, pp. 6658–6667.

[158] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu, *Lgm: Large multi-view gaussian model for high-resolution 3d content creation*, European conference on computer vision, 2024, pp. 1–18.

[159] Jonathan Ho, Ajay Jain, and Pieter Abbeel, *Denoising diffusion probabilistic models*, Advances in neural information processing systems **33** (2020), 6840–6851.

[160] Li-Wen Chang, Wenlei Bao, Qi Hou, Chengquan Jiang, Ningxin Zheng, Yinmin Zhong, Xuanrun Zhang, Zuquan Song, Chengji Yao, and Ziheng Jiang, *Flux: fast software-based communication overlap on gpus through kernel fusion*, arXiv preprint arXiv:2406.06858 (2024).

[161] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson, *Synsin: End-to-end view synthesis from a single image*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2020, pp. 7467–7477.

[162] Jing Yu Koh, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson, *Pathdreamer: A world model for indoor navigation*, Proceedings of the ieee/cvf international conference on computer vision, 2021, pp. 14738–14748.

[163] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner, *Text2room: Extracting textured 3d meshes from 2d text-to-image models*, Proceedings of the ieee/cvf international conference on computer vision, 2023, pp. 7909–7920.

[164] Jing Yu Koh, Harsh Agrawal, Dhruv Batra, Richard Tucker, Austin Waters, Honglak Lee, Yinfei Yang, Jason Baldridge, and Peter Anderson, *Simple and effective synthesis of indoor 3d scenes*, Proceedings of the aaai conference on artificial intelligence, 2023, pp. 1169–1178.

[165] Shengqu Cai, Eric Ryan Chan, Songyou Peng, Mohamad Shahbazi, Anton Obukhov, Luc Van Gool, and Gordon Wetzstein, *Diffdreamer: Towards consistent unsupervised single-view scene extrapolation with conditional diffusion models*, Proceedings of the ieee/cvf international conference on computer vision, 2023, pp. 2139–2150.

[166] Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel, *Scenescape: Text-driven consistent scene generation*, Advances in Neural Information Processing Systems **36** (2023), 39897–39914.

[167] Songchun Zhang, Yibo Zhang, Quan Zheng, Rui Ma, Wei Hua, Hujun Bao, Weiwei Xu, and Changqing Zou, *3d-scenedreamer: Text-driven 3d-consistent scene generation*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2024, pp. 10170–10180.

[168] Yiying Yang, Fukun Yin, Jiayuan Fan, Xin Chen, Wanzhang Li, and Gang Yu, *Scene123: One prompt to 3d scene generation via video-assisted and consistency-enhanced mae*, arXiv preprint arXiv:2408.05477 (2024).

[169] Jaidev Shriram, Alex Trevithick, Lingjie Liu, and Ravi Ramamoorthi, *Realmdreamer: Text-driven 3d scene generation with inpainting and depth diffusion*, arXiv preprint arXiv:2404.07199 (2024).

[170] Paul Engstler, Andrea Vedaldi, Iro Laina, and Christian Rupprecht, *Invisible stitch: Generating smooth 3d scenes with depth inpainting*, arXiv preprint arXiv:2404.19758 (2024).

[171] Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T Freeman, and Jiajun Wu, *Wonderworld: Interactive 3d scene generation from a single image*, arXiv preprint arXiv:2406.09394 (2024).

[172] Gabriela Ben Melech Stan, Diana Wofk, Scottie Fox, Alex Redden, Will Saxton, Jean Yu, Estelle Aflalo, Shao-Yen Tseng, Fabio Nonato, and Matthias Muller, *Ldm3d: Latent diffusion model for 3d*, arXiv preprint arXiv:2305.10853 (2023).

[173] Wenrui Li, Fucheng Cai, Yapeng Mi, Zhe Yang, Wangmeng Zuo, Xingtao Wang, and Xiaopeng Fan, *Scenedreamer360: Text-driven 3d-consistent scene generation with panoramic gaussian splatting*, arXiv preprint arXiv:2408.13711 (2024).

[174] Tianhao Wu, Chuanxia Zheng, and Tat-Jen Cham, *Panodiffusion: 360-degree panorama outpainting via diffusion*, arXiv preprint arXiv:2307.03177 (2023).

[175] Jonas Schult, Sam Tsai, Lukas Höllein, Bichen Wu, Jialiang Wang, Chih-Yao Ma, Kunpeng Li, Xiaofang Wang, Felix Wimbauer, and Zijian He, *Controlroom3d: Room generation using semantic proxy rooms*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2024, pp. 6201–6210.

[176] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen, *Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2024, pp. 6517–6526.

[177] Aoming Liu, Zhong Li, Zhang Chen, Nannan Li, Yi Xu, and Bryan A Plummer, *Panofree: Tuning-free holistic multi-view image generation with cross-view self-guidance*, European conference on computer vision, 2024, pp. 146–164.

[178] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan, *Emergent correspondence from image diffusion*, Advances in Neural Information Processing Systems **36** (2023), 1363–1389.

[179] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole, *Cat3d: Create anything in 3d with multi-view diffusion models*, arXiv preprint arXiv:2405.10314 (2024).

[180] Paul Engstler, Aleksandar Shtedritski, Iro Laina, Christian Rupprecht, and Andrea Vedaldi, *Syncity: Training-free generation of 3d worlds*, arXiv preprint arXiv:2503.16420 (2025).

[181] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang, *Layoutgpt: Compositional visual planning and generation with large language models*, Advances in Neural Information Processing Systems **36** (2023), 18225–18250.

[182] Xiaoyu Zhou, Xingjian Ran, Yajiao Xiong, Jinlin He, Zhiwei Lin, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang, *Gala3d: Towards text-to-3d complex scene generation via layout-guided generative gaussian splatting*, arXiv preprint arXiv:2402.07207 (2024).

[183] Ata Çelen, Guo Han, Konrad Schindler, Luc Van Gool, Iro Armeni, Anton Obukhov, and Xi Wang, *I-design: Personalized llm interior designer*, arXiv preprint arXiv:2404.02838 (2024).

[184] Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi, *Scenecraft: An llm agent for synthesizing 3d scenes as blender code*, Forty-first international conference on machine learning, 2024.

[185] Chunyi Sun, Junlin Han, Weijian Deng, Xinlong Wang, Zishan Qin, and Stephen Gould, *3d-gpt: Procedural 3d modeling with large language models*, arXiv preprint arXiv:2310.12945 (2023).

[186] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner, *Diffuscene: Denoising diffusion models for generative indoor scene synthesis*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2024, pp. 20507–20518.

[187] Chenguo Lin and Yadong Mu, *Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior*, arXiv preprint arXiv:2402.04717 (2024).

[188] Guangyao Zhai, Evin Pınar Örnek, Dave Zhenyu Chen, Ruotong Liao, Yan Di, Nassir Navab, Federico Tombari, and Benjamin Busam, *Echoscene: Indoor scene generation via information echo over scene graph diffusion*, European conference on computer vision, 2024, pp. 167–184.

[189] Alexander Vilesov, Pradyumna Chari, and Achuta Kadambi, *Cg3d: Compositional generation for text-to-3d via gaussian splatting*, arXiv preprint arXiv:2311.17907 (2023).

[190] Léopold Maillard, Nicolas Sereyjol-Garros, Tom Durand, and Maks Ovsjanikov, *Debara: Denoising-based 3d room arrangement generation*, Advances in Neural Information Processing Systems **37** (2024), 109202–109232.

[191] Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, and Hongdong Li, *Blockfusion: Expandable 3d scene generation using latent tri-plane extrapolation*, ACM Transactions on Graphics (TOG) **43** (2024), no. 4, 1–17.

[192] Quan Meng, Lei Li, Matthias Nießner, and Angela Dai, *Lt3sd: Latent trees for 3d scene diffusion*, ArXiv **abs/2409.08215** (2024).

[193] Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams, *Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies*, 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023), 4209–4219.

[194] Yuheng Liu, Xinke Li, Xueting Li, Lu Qi, Chongshou Li, and Ming-Hsuan Yang, *Pyramid diffusion for fine 3d large scene generation*, ArXiv **abs/2311.12085** (2023).

[195] Jumin Lee, Sebin Lee, Changho Jo, Woobin Im, Juhyeong Seon, and Sung-Eui Yoon, *Semcity: Semantic scene generation with triplane diffusion*, 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2024), 28337–28347.

[196] Lucy Chai, Richard Tucker, Zhengqi Li, Phillip Isola, and Noah Snavely, *Persistent nature: A generative model of unbounded 3d worlds*, 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023), 20863–20874.

[197] Han Yan, Yang Li, Zhennan Wu, Shenzhou Chen, Weixuan Sun, Taizhang Shang, Weizhe Liu, Tian Chen, Xiaqiang Dai, and Chao Ma, *Frankenstein: Generating semantic-compositional 3d scenes in one tri-plane*, Siggraph asia 2024 conference papers, 2024, pp. 1–11.

[198] Han-Hung Lee, Qinghong Han, and Angel X Chang, *Nuiscene: Exploring efficient generation of unbounded outdoor scenes*, arXiv preprint arXiv:2503.16375 (2025).

[199] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall, *Dreamfusion: Text-to-3d using 2d diffusion*, ArXiv **abs/2209.14988** (2022).

[200] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin, *Magic3d: High-resolution text-to-3d content creation*, 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022), 300–309.

[201] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen, *Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior*, 2023 IEEE/CVF International Conference on Computer Vision (ICCV) (2023), 22762–22772.

[202] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick, *Zero-1-to-3: Zero-shot one image to 3d object*, 2023 IEEE/CVF International Conference on Computer Vision (ICCV) (2023), 9264–9275.

[203] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng, *Dreamgaussian: Generative gaussian splatting for efficient 3d content creation*, ArXiv **abs/2309.16653** (2023).

[204] Anchit Gupta and Anchit Gupta, *3dgen: Triplane latent diffusion for textured mesh generation*, ArXiv **abs/2303.05371** (2023).

[205] Bojun Xiong, Si-Tong Wei, Xin-Yang Zheng, Yan-Pei Cao, Zhouhui Lian, and Peng-Shuai Wang, *Octfusion: Octree-based diffusion models for 3d shape generation*, ArXiv **abs/2408.14732** (2024).

[206] Weiyu Li, Jiarui Liu, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long, *Craftsman: High-fidelity mesh generation with 3d native generation and interactive geometry refiner*, ArXiv **abs/2405.14979** (2024).

[207] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis, *Lion: Latent point diffusion models for 3d shape generation*, Advances in Neural Information Processing Systems **35** (2022), 10021–10039.

[208] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu, *Clay: A controllable large-scale generative model for creating high-quality 3d assets*, ACM Transactions on Graphics (TOG) **43** (2024), 1 –20.

[209] Lin Li, Zehuan Huang, Haoran Feng, Gengxiong Zhuang, Rui Chen, Chunchao Guo, and Lu Sheng, *Voxhammer: Training-free precise and coherent 3d editing in native 3d space*, arXiv preprint arXiv:2508.19247 (2025).

[210] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram S. Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi, *Objaverse-xl: A universe of 10m+ 3d objects*, ArXiv **abs/2307.05663** (2023).

[211] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, and Alaaeldin El-Nouby, *Dinov2: Learning robust visual features without supervision*, arXiv preprint arXiv:2304.07193 (2023).

[212] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny, *Vggt: Visual geometry grounded transformer*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2025.

[213] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan, *Florence-2: Advancing a unified representation for a variety of vision tasks*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2024, pp. 4818–4829.

[214] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, and Laura Gustafson, *Sam 2: Segment anything in images and videos*, arXiv preprint arXiv:2408.00714 (2024).

[215] Szymon Rusinkiewicz and Marc Levoy, *Efficient variants of the icp algorithm*, Proceedings third international conference on 3-d digital imaging and modeling, 2001, pp. 145–152.

[216] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool, *Repaint: Inpainting using denoising diffusion probabilistic models*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2022, pp. 11461–11471.

[217] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, and Alec Radford, *Gpt-4o system card*, arXiv preprint arXiv:2410.21276 (2024).

[218] Hong-Xing Yu, Haoyi Duan, Charles Herrmann, William T Freeman, and Jiajun Wu, *Wonderworld: Interactive 3d scene generation from a single image*, Proceedings of the computer vision and pattern recognition conference, 2025, pp. 5916–5926.

[219] Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao, *Gen3c: 3d-informed world-consistent video generation with precise camera control*, Proceedings of the computer vision and pattern recognition conference, 2025, pp. 6121–6132.

[220] Tianyu Huang, Wangguandong Zheng, Tengfei Wang, Yuhao Liu, Zhenwei Wang, Junta Wu, Jie Jiang, Hui Li, Rynson WH Lau, and Wangmeng Zuo, *Voyager: Long-range and world-consistent video diffusion for explorable 3d scene generation*, arXiv preprint arXiv:2506.04225 (2025).

[221] Xinyang Li, Tengfei Wang, Zixiao Gu, Shengchuan Zhang, Chunchao Guo, and Liujuan Cao, *Flashworld: High-quality 3d scene generation within seconds*, arXiv preprint arXiv:2510.13678 (2025).

[222] Zhihao Li, Yufei Wang, Heliang Zheng, Yihao Luo, and Bihan Wen, *Sparc3d: Sparse representation and construction for high-resolution 3d shapes modeling*, arXiv preprint arXiv:2505.14521 (2025).

[223] Kaizhi Zheng, Ruijian Zhang, Jing Gu, Jie Yang, and Xin Eric Wang, *Constructing a 3d town from a single image*, arXiv preprint arXiv:2505.15765 (2025).

[224] Hanke Chen, Yuan Liu, and Minchen Li, *Trellisworld: Training-free world generation from object generators*, arXiv preprint arXiv:2510.23880 (2025).

[225] Seungwoo Yoon, Jinmo Kim, and Jaesik Park, *Extend3d: Town-scale 3d generation*, 2025.

[226] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu, *Wan: Open and advanced large-scale video generative models*, arXiv preprint arXiv:2503.20314 (2025).

[227] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi, *Ltx-video: Realtime video latent diffusion*, arXiv preprint arXiv:2501.00103 (2024).

[228] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, and Jianwei Zhang, *Hunyuanvideo: A systematic framework for large video generative models*, arXiv preprint arXiv:2412.03603 (2024).

[229] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, and Jianfeng Gao, *Sora: A review on background, technology, limitations, and opportunities of large vision models*, arXiv preprint arXiv:2402.17177 (2024).

[230] Zhihao Hu and Dong Xu, *Videocontrolnet: A motion-guided video-to-video translation framework by using diffusion model with controlnet*, arXiv preprint arXiv:2307.14073 (2023).

[231] Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu, *Vace: All-in-one video creation and editing*, arXiv preprint arXiv:2503.07598 (2025).

[232] Bohao Peng, Jian Wang, Yuechen Zhang, Wenbo Li, Ming-Chang Yang, and Jiaya Jia, *Controlnext: Powerful and efficient control for image and video generation*, arXiv preprint arXiv:2408.06070 (2024).

[233] Zhouxia Wang, Ziyang Yuan, Xintao Wang, Yaowei Li, Tianshui Chen, Menghan Xia, Ping Luo, and Ying Shan, *Motionctrl: A unified and flexible motion controller for video generation*, Acm siggraph 2024 conference papers, 2024, pp. 1–11.

[234] Jensen (Jinghao) Zhou, Hang Gao, Vikram Voleti, Aaryaman Vasishta, Chun-Han Yao, Mark Boss, Philip Torr, Christian Rupprecht, and Varun Jampani, *Stable virtual camera: Generative view synthesis with diffusion models*, arXiv preprint arXiv:2503.14489 (2025).

[235] Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang, *Cameractrl: Enabling camera control for text-to-video generation*, arXiv preprint arXiv:2404.02101 (2024).

[236] Xuyang Chen, Zhijun Zhai, Kaixuan Zhou, Zengmao Wang, Jianan He, Dong Wang, Yanfeng Zhang, Rüdiger Westermann, Konrad Schindler, and Liqiu Meng, *Mess: City mesh-guided outdoor scene generation with cross-view consistent diffusion*, arXiv preprint arXiv:2508.15169 (2025).

[237] Geonung Kim, Janghyeok Han, and Sunghyun Cho, *Videofrom3d: 3d scene video generation via complementary image and video diffusion models*, arXiv preprint arXiv:2509.17985 (2025).

[238] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia, *Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation*, Proceedings of the ieee/cvf international conference on computer vision (iccv), 2023October, pp. 22246–22256.

[239] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or, *Latent-nerf for shape-guided generation of 3d shapes and textures*, arXiv preprint arXiv:2211.07600 (2022).

[240] Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long, *Craftsman3d: High-fidelity mesh generation with 3d native generation and interactive geometry refiner*, arXiv preprint arXiv:2405.14979 (2024).

[241] Haibo Yang, Yang Chen, Yingwei Pan, Ting Yao, Zhineng Chen, Chong-Wah Ngo, and Tao Mei, *Hi3d: Pursuing high-resolution image-to-3d generation with video diffusion models*, Proceedings of the 32nd acm international conference on multimedia, 2024, pp. 6870–6879.

[242] Jie Long Lee, Chen Li, and Gim Hee Lee, *Disr-nerf: Diffusion-guided view-consistent super-resolution nerf*, Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2024, pp. 20561–20570.

[243] Nuri Ryu, Jiyun Won, Jooeun Son, Minsu Gong, Joo-Haeng Lee, and Sunghyun Cho, *Elevating 3d models: High-quality texture and geometry refinement from a low-quality model*, Proceedings of the special interest group on computer graphics and interactive techniques conference conference papers, 2025, pp. 1–12.

[244] Hao Wen, Zehuan Huang, Yaohui Wang, Xinyuan Chen, and Lu Sheng, *Ouroboros3d: Image-to-3d generation via 3d-aware recursive diffusion*, Proceedings of the computer vision and pattern recognition conference, 2025, pp. 21631–21641.

[245] Zhenyu Tang, Junwu Zhang, Xinhua Cheng, Wangbo Yu, Chaoran Feng, Yatian Pang, Bin Lin, and Li Yuan, *Cycle3d: High-quality and consistent image-to-3d generation via generation-reconstruction cycle*, Proceedings of the aaai conference on artificial intelligence, 2025, pp. 7320–7328.

[246] Baorui Ma, Haoge Deng, Junsheng Zhou, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang, *Geodream: Disentangling 2d and geometric priors for high-fidelity and consistent 3d generation*, arXiv preprint arXiv:2311.17971 (2023).

[247] Sibo Wu, Congrong Xu, Binbin Huang, Andreas Geiger, and Anpei Chen, *Genfusion: Closing the loop between reconstruction and generation via videos*, Proceedings of the computer vision and pattern recognition conference, 2025, pp. 6078–6088.

[248] Yifan Liu, Zhiyuan Min, Zhenwei Wang, Junta Wu, Tengfei Wang, Yixuan Yuan, Yawei Luo, and Chunchao Guo, *Worldmirror: Universal 3d world reconstruction with any-prior prompting*, arXiv preprint arXiv:2510.10726 (2025).

[249] OpenAI, *Gpt image 1*, 2025. OpenAI API model documentation, accessed: 2025-11-13.

[250] Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, and Yunfei Zhao, *Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material*, arXiv preprint arXiv:2506.15442 (2025).

[251] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki, *The Open Motion Planning Library*, IEEE Robotics & Automation Magazine **19** (2012December), no. 4, 72–82. https://ompl.kavrakilab.org.

[252] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush, *Huggingface's transformers: State-of-the-art natural language processing*, arXiv, 2019.

[253] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, *Microsoft coco: Common objects in context*, Computer vision – eccv 2014, 2014, pp. 740–755.

[254] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht, *ALFWorld: Aligning Text and Embodied Environments for Interactive Learning*, Proceedings of the international conference on learning representations (iclr), 2021.

[255] Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi, *AI2-THOR: an interactive 3d environment for visual AI*, CoRR **abs/1712.05474** (2017), available at 1712.05474.

[256] Ilya Loshchilov and Frank Hutter, *Decoupled weight decay regularization*, arXiv preprint arXiv:1711.05101 (2017).

[257] OpenAI, *Introducing openai o3 and o4-mini*, 2025.

[258] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, and Chenxu Lv, *Qwen3 technical report*, arXiv preprint arXiv:2505.09388 (2025).