FORGE-Tree: Diffusion-Forcing Tree Search for Long-Horizon Robot Manipulation

Yanjia Huang[1*], Shuo Liu[2*], Sheng Liu[3], Qingxiao Xu[1], Mingyang Wu[1], Xiangbo Gao[1] and Zhengzhong Tu[1]

Abstract*— Long-horizon robot manipulation tasks remain challenging for Vision-Language-Action (VLA) policies due to drift and exposure bias, often denoise the entire trajectory with fixed hyperparameters, causing small geometric errors to compound across stages and offering no mechanism to allocate extra test-time compute where clearances are tight. To address these challenges, we introduce FORGE-Tree, a plug-in control layer that couples a stage-aligned** **Diffusion Forcing** ***(DF) head with test-time **Monte Carlo Tree Diffusion** ***(MCTD). With a frozen VLA encoder, DF aligns timesteps to subtask stages; during inference we partially denoise only a target segment while keeping other tokens frozen, turning trajectory refinement into a sequence of local edits. We then apply Monte Carlo Tree Diffusion to select the next segment to refine. A scene graph supplies priors for expansion and geometry relation-aware scoring for rollouts, yielding*** **tree-structured denoising** ***whose performance scales with search budget while preserving the executed prefix. Evaluation on LIBERO, FORGE-Tree improves success rate by +13.4–+17.2 pp over the native VLA baselines with both OpenVLA and Octo-Base. Gains remain consistent under comparable compute budgets, especially on long-horizon variants.***

I. Introduction

Long-horizon, language-conditioned manipulation is a task where success hinges on *global plan coherence* across multiple sub-tasks and *local geometric precision* at stage ends and the final goal [1]–[5]. A simple instruction such as *"put both the alphabet soup and the tomato sauce in the basket"* demands stage-wise reasoning (reach–grasp–place, twice), contact-aware motion, and centimeter-level alignment with relational constraints (e.g., *in-basket*). Beyond isolated pick-and-place skills, such instructions require maintaining a consistent task narrative across partial successes and minor slips, while remaining precise enough to satisfy relation constraints that are only locally verifiable. These gaps suggest treating control not as a one-shot commitment but as an explicitly staged refinement process and reallocating attention and computation where uncertainty concentrates.

Existing methods such as Vision–Language–Action models (VLAs) learn to map images and text to actions [2], [6], and diffusion policies provide robust multimodal sequence generation [7], [8]. However, long horizons expose three recurring gaps. *First*, most decoders denoise entire trajectories with a fixed schedule, so small pose errors compound across stages. *Second*, decoding knobs (e.g., step schedule, guidance, temperature) are typically frozen *a priori*, providing no mechanism to spend additional test-time compute where clearances are tight or the scene is ambiguous. *Third*, feed-forward controllers seldom leverage *symbolic or relational* structure alongside continuous kinematics, limiting reliability on relation-heavy tasks.

To address these issues, we keep the VLA encoder fixed and upgrade only the control layer with **FORGE-Tree**. The key idea is to treat diffusion decoding not as a one-shot generator but as a *planner over partial trajectories*. This perspective decouples planning from perception-scale retraining and turns decoding into a sequence of transparent, budget-aware edits over short horizons. Concretely, FORGE-Tree (i) aligns denoising to stage structure so the model *learns to land* subgoals, (ii) casts inference as a budgeted search over *denoising meta-actions*, which segment to edit next and how aggressively to denoise it, so compute is adaptively focused where it matters, and (iii) couples a lightweight scene graph to both exploration and evaluation, linking symbolic relations (e.g., `in`, `on top of`) with continuous robot kinematics. Crucially, these priors act as soft guidance rather than hard constraints, preserving flexibility while exposing the reasons behind each refinement. Trained on EMMA-X and evaluated in simulation on LIBERO and ManiSkill under standard protocols [9]–[11], our method improves over a VLA to Diffusion Policy baseline: DF alone reduces mid-horizon drift, and MCTD delivers monotonic gains with increasing search budget, yielding higher success rate, lower terminal pose error and collisions, and higher relation satisfaction on both OpenVLA and Octo backbones.

We position **FORGE-Tree** as a control-layer upgrade rather than a new pretraining recipe, prioritizing modularity, interpretability, and test-time efficiency. **Contributions** are as follows: (1) A VLA-conditioned *Diffusion Forcing* scheme that teaches the head to *land* stage subgoals. (2) *Tree-structured denoising* through meta-actions on segment, stride, guidance, and temperature: scalable adaptive budget decoding. (3) A *dual-role scene graph* providing both expansion priors and geometry relation-aware evaluation to bridge symbols and kinematics, making decisions traceable to human-readable relations. (4) A *plug-and-play* control layer that drops into VLAs and produces consistent gains without modifying the backbone, easing adoption across existing VLA stacks.

[1]Department of Computer Science and Engineering, Texas A&M University
[2]Department of Electrical & Computer Engineering, University of Washington
[3]Karlsruhe Institute of Technology, Germany
*Equal Contributors
Corresponding Author: tzz@tamu.edu

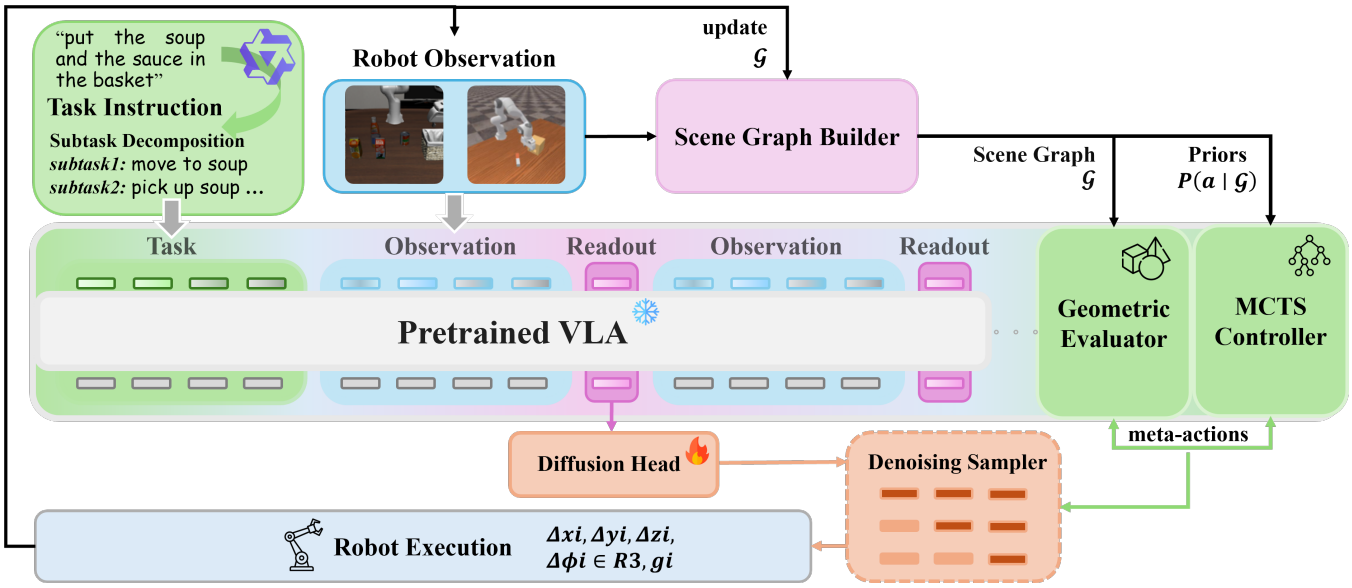

Fig. 1. **System overview.** Pretrained VLAs encode the instruction and observations, builds a scene graph $\mathcal{G}$, our diffusion head predicts noise, the partial-denoising sampler edits a selected future segment with meta-actions $a = (k, m, s, w, \tau)$, and MCTD evaluates candidates with geometry-aware rewards before executing the first segment and replanning.

II. Related Work

A. Vision-Language-Action Models

Recent advances in Large Language Models and Vision Language Models have shown that transformer models trained on internet-scale data possess remarkable capabilities in text generation, multimodal content understanding, and reasoning [12]–[14] . Inspired by these developments, researchers have explored using transformer models to directly map language instructions and visual observations to physical robot actions [1], [2], [6], [15]–[17]. These models, trained on large-scale robotics datasets [17]–[19] via behavior cloning, can generalize to unseen scenes and tasks beyond their training data distribution. However, their autoregressive architecture—designed to predict the next action or action chunk [20]—struggles with cumulative errors and typically fails in long-horizon tasks requiring sequential operations. Our approach builds a test-time framework that enhances pre-trained VLA models with Monte Carlo Tree Search [21] and scene graphs [22], leveraging structured search to enable effective long-term planning.

B. Diffusion Models for Robotics

Diffusion models are generative models that learn data distributions through iterative noise addition and denoising processes, showing excellent results in image and video generation [23]–[25]. Building on this success, researchers have applied these models to robotics [7], [26], where they demonstrate powerful capabilities in multimodal action generation and planning [27], [28]. Some approaches combine VLM backbones with diffusion heads to generate smooth motion trajectories and use multimodal conditions to guide the denoising process [29]–[32]. In our work, inspired by the Diffusion Forcing mechanism [33], we introduce a diffusion action head conditioned on VLA readout information. This is combined with search mechanisms to schedule the denoising process, integrating next token prediction with the diffusion model's denoising time-steps, resulting in more flexible control and improved geometric alignment.

C. Long-horizon manipulation tasks

Long-horizon tasks remain a persistent challenge in robot manipulation. Traditional approaches like Task and Motion Planning (TAMP) [34] and skill chaining [28], [35] require perfect perception of world states and precise dynamics modeling, which can rarely met in real-world environments. Recently, researchers have leveraged LLMs and VLMs as planners for open-world robot manipulation due to their powerful prior knowledge [36]–[39]. Yet these approaches still require integration with motion planners to achieve physical robot control. Most similar to our approach is VLAPS [40], they use MCTS and world model with pretrained VLA models to handle long-horizon tasks. Different from this, our method combines VLA models' ability to directly map instructions and observations to actions with VLMs and scene graphs serving as abstract world state representations. We integrate these components with search algorithms to effectively solve long-horizon operation problems.

III. Method

A. Problem Setup and Notation

We consider language-conditioned, long-horizon manipulation where the agent must output a continuous action sequence $\mathbf{x}_{1:L} \in \mathbb{R}^{L \times d_a}$ (we use d_a=7) that satisfies intermediate *stage* constraints and a final goal. Let u be the instruction, $\mathbf{o}$ the visual observation(s), and $\mathbf{c} = f_{\text{VLA}}(\mathbf{o}, u) \in \mathbb{R}^{d_c}$ the embedding from a frozen or LoRA-tuned Octo or OpenVLA

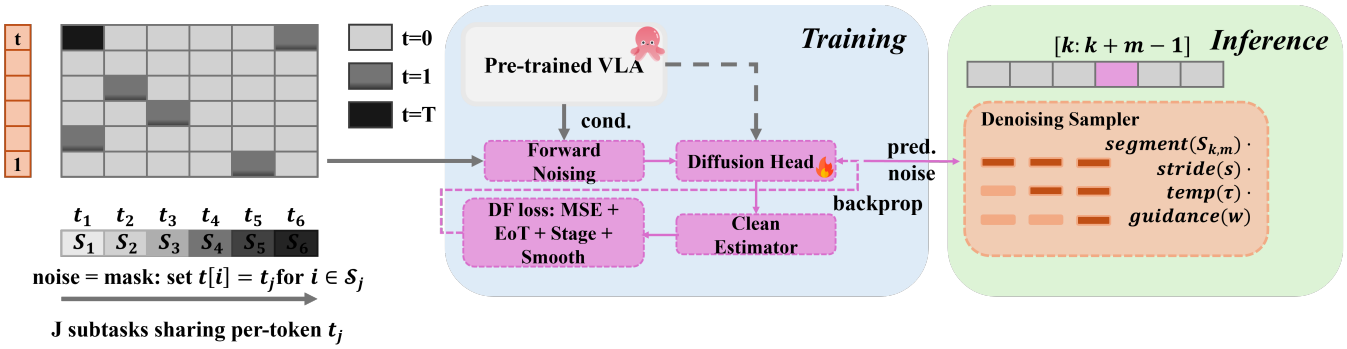

Fig. 2. **Diffusion Forcing (DF) with partial denoising.** Left: *Noise as masking*—subtasks $\mathscr{S}_j$ share per-subtask timesteps $t[i]{=}t_j$ (darker cells indicate larger t). Middle (training): conditioned on $c{=}f_{\text{VLA}}(o,u)$, the diffusion head predicts $\varepsilon_\theta(x_t[i],c,t[i])$, recovers $\hat{x}_0[i]$, and is supervised by the DF loss (noise MSE + end-of-trajectory and stage-end geometric terms + smoothness) [7], [33]. Right (inference): we *partially denoise* a selected segment $S_{k,m}$ using jumpy DDIM [41] with geometry guidance $w\nabla_{x_t}U(\hat{x}_0;\mathscr{G})$; only the segment evolves while the complement is frozen, and the first segment is committed before replanning.

TABLE I

NOTATION USED IN THE PAPER.

Symbol	Meaning
$\mathbf{x}_{1:L}$, $\mathbf{x}_i \in \mathbb{R}^{d_a}$	Action tokens (position/orientation/gripper)
$\mathbf{c} \in \mathbb{R}^{d_c}$	VLA embedding (Octo or OpenVLA)
$t[i] \in \{1,\dots,T\}$	Per-token diffusion timestep
α_t, $\bar{\alpha}_t$	Noise schedule and cumulative product
$\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon}_\theta$	Gaussian noise or predicted noise
$\hat{\mathbf{x}}_0$	Predicted clean actions
$\mathscr{S}$, $\mathbf{G}$, $\mathbf{g}_{\text{final}}$	Stage indices, stage goals, final goal
$\mathscr{G} = (\mathscr{V},\mathscr{E})$	Scene graph (objects/relations)
$a = (k,m,s,w,\tau)$	Meta action (start/length, stride, guidance, temperature)
$N(n), W(n,a), Q(n,a)$	Visit count, cumulative return, mean action value at n
$r_{\text{fast}}(n,a)$	Fast reward for unexpanded/evaluable children
$S_{k,m}$	Segment operator selecting tokens $[k:k{+}m{-}1]$

encoder [2], [6]. Each trajectory optionally contains a stage-end mask $\mathbf{m} \in \{0,1\}^L$ and stage goals $\mathbf{G} \in \mathbb{R}^{L\times 3}$ (EE geometry at stage ends), with $\mathscr{S} = \{i \mid m_i = 1\}$, and a final positional goal $\mathbf{g}_{\text{final}} \in \mathbb{R}^3$. We denote by t_{now} the index of the last executed token (prefix), and by $\hat{\mathbf{x}}_0^{(i)}$ the clean estimate truncated at step i; see Table I.

B. VLA-Conditioned Diffusion Head

We parameterize a Transformer+FiLM diffusion head f_θ that predicts per-token noise conditioned on $\mathbf{c}$. For token i at timestep $t[i]$,

$$\boldsymbol{\varepsilon}_\theta\big(\mathbf{x}_t[i],\mathbf{c},t[i]\big) \in \mathbb{R}^{d_a}. \tag{III.1}$$

FiLM derives token-wise scales/shifts/gates from $\mathbf{c}$ (and optionally stage or subtask embeddings) to modulate each block [7]. Following variance-preserving diffusion [23], for each token i:

$$\mathbf{x}_t[i] \;=\; \sqrt{\bar{\alpha}_{t[i]}}\,\mathbf{x}_0[i] \;+\; \sqrt{1-\bar{\alpha}_{t[i]}}\,\boldsymbol{\varepsilon}[i], \qquad \boldsymbol{\varepsilon}[i] \sim \mathscr{N}(\mathbf{0},\mathbf{I}), \tag{III.2}$$

and the clean estimator is

$$\hat{\mathbf{x}}_0[i] \;=\; \frac{1}{\sqrt{\bar{\alpha}_{t[i]}}}\Big(\mathbf{x}_t[i] - \sqrt{1-\bar{\alpha}_{t[i]}}\,\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t[i],\mathbf{c},t[i])\Big). \tag{III.3}$$

Eqs. (III.2)–(III.3) apply independently across tokens, enabling non-uniform noise levels across the sequence ("noise = mask") [33].

C. Diffusion Forcing (DF) Objective

We align the denoising schedule with subtask structure instead of treating all tokens equally. Given an ECoT-style stage mask $\mathbf{m} \in \{0,1\}^L$ and boundaries $\mathscr{S} = \{i : m_i = 1\}$, we assign a *single* timestep t_j to all tokens within subtask j ("noise = mask"), i.e., $t[i]{=}t_j$ if i lies in subtask j [33]. This yields clean predictions $\hat{\mathbf{x}}_0[i]$ via (III.2)–(III.3). We then supervise with a stage- and geometry-aware objective:

$$\begin{aligned}\mathscr{L}_{\text{DF}} = \underbrace{\tfrac{1}{L}\sum_{i=1}^{L}\big\|\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t[i],\mathbf{c},t[i]) - \boldsymbol{\varepsilon}[i]\big\|_2^2}_{\text{noise MSE}} \\ + \lambda_{\text{EoT}}\big\|\text{pos3}(\hat{\mathbf{x}}_0^{(L)}) - \mathbf{g}_{\text{final}}\big\|_2^2 \\ + \lambda_{\text{stage}}\sum_{i\in\mathscr{S}}\big\|\text{pos3}(\hat{\mathbf{x}}_0^{(i)}) - \mathbf{G}_i\big\|_2^2 \\ + \lambda_{\text{traj}}\tfrac{1}{L}\sum_{i=1}^{L}\big\|\hat{\mathbf{x}}_0^{(i)} - \mathbf{x}_0^{(i)}\big\|_2^2 \\ + \lambda_{\text{smooth}}\tfrac{1}{L}\sum_{i=2}^{L}\big\|\Delta\hat{\mathbf{x}}_0^{(i)}\big\|_2^2.\end{aligned} \tag{III.4}$$

Implementation. We use a VP schedule with $T{=}1000$ steps and sample a single per-subtask timestep $t_j \sim \mathscr{U}[400,900]$ ("noise = mask"). This range balances denoising difficulty across stages and worked robustly across all suites.

D. Partial Denoising with a Segment Operator

At test time we only evolve a future segment while freezing the executed prefix. Instead of a diagonal matrix, we use a binary *mask vector* $s{\in}\{0,1\}^L$ for the chosen segment $[k:k{+}m{-}1]$:

$$s_i = \mathbb{1}[k \le i \le k{+}m{-}1], \quad \bar{s} = \mathbf{1} - s.$$

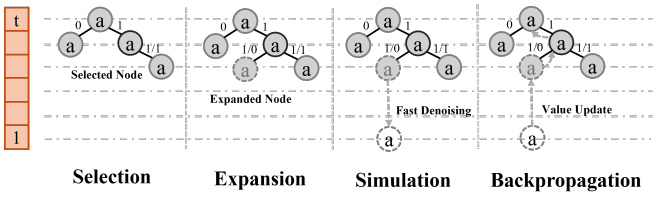

Fig. 3. **Stage-aware MCTD.** For each stage we run an MCTS whose edges are meta-actions $a=(k,m,s,w,\tau,t)$. Selection uses P-UCT with scene-graph priors, [21], and unvisited edges are ordered by a fast heuristic $r_{\text{fast}}(n,a)$. Simulation performs *partial denoising* on $S_{k,m}$ (jumpy DDIM, guidance strength w) to produce a candidate $\hat{x}_0$, which is scored by the geometry-aware return $R(\hat{x}_0;\mathscr{G})$. Returns are backed up to update (N,W,Q); the best child is executed for one segment, observations update $\mathscr{G}$, and the process recedes to the next stage.

With a schedule $t_{J-1}=\max(0,t_J-s)$ and temperature τ **Meta-parameters searched by MCTD.** segment start $k\in\{t_{\text{now}}+1,\dots\}$, length $m\in\{8,12,16\}$, stride $s\in\{2,4,8\}$, guidance $w\in[0,5]$, temperature $\tau\in[0.5,1.0]$. We restrict k to start within 16 tokens before the next stage boundary to focus search near subgoal interfaces.

1) Geometry Relation-Aware Guidance: We define a potential $U(\hat{\mathbf{x}}_0;\mathscr{G})$ as a non-negative weighted sum of terminal alignment, stage anchoring, relation violation penalties from the scene graph, and collision costs; lower values indicate candidates that precisely land subgoals, satisfy symbolic relations, and remain collision-free. We form a guided noise with a practical Jacobian that ignores cross-token coupling:

$$\tilde{\boldsymbol{\varepsilon}} \;=\; \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t,\mathbf{c},t) \;-\; w\,\underbrace{\Big(\operatorname{diag}\big(1/\sqrt{\bar{\alpha}_{t[i]}}\big)\Big)^{\top}}_{\text{stop-grad on }\boldsymbol{\varepsilon}_\theta}\nabla_{\hat{\mathbf{x}}_0}U,$$

and substitute $\tilde{\boldsymbol{\varepsilon}}$ into the partial DDIM update (III.2). In practice we compute $\nabla_{\hat{\mathbf{x}}_0}U$ with autograd on the differentiable predicates (next paragraph) and cache it across the m active tokens for the current jump.

E. Scene Graph and Differentiable Predicates

We maintain $\mathscr{G}=(\mathscr{V},\mathscr{E})$ via detection and VLM parsing [22]. Each relation $r\in\mathscr{R}$ induces a differentiable predicate ϕ_r (for guidance) and a satisfaction score $\text{sat}_r\in[0,1]$ (for reward), e.g.,

$$\begin{aligned}\phi_{\text{on-top}}(A,B) \;&=\; \Big[\max\big(0,\,\delta_\perp-\|p_A^\perp-p_B^\perp\|\big)\Big]^2 \\ &\quad+\;\Big[\max\big(0,\,h_{\min}-(p_A^z-p_B^z)\big)\Big]^2, \qquad \text{(III.5)}\\ \text{sat}_{\text{on-top}} \;&=\; \sigma\big(-\gamma\,\phi_{\text{on-top}}(A,B)\big).\end{aligned}$$

Other relations (*in/left-of/aligned-with*) follow analogous distance or angle forms; collisions use SDF or capsule distances. A stage-level textual parser (Qwen2.5-VL) extracts candidate relations (in, on top of, left-of, aligned) and anchors. For guidance and reward, each relation maps to a differentiable predicate based on signed distances and angular offsets. Collision penalties use capsule–capsule distances with safety margins (EE radius r_{EE}, object radii r_o).

F. Stage-Aware MCTD with Dual Rewards

a) Dual rewards: fast heuristic vs. true geometry return.: We use two complementary signals. The *fast* heuristic scores a child before we spend rollout compute: it combines (i) a language-consistent likelihood from the VLA (token-average log-probability of the meta-action, capturing whether the segment choice matches the instruction and visual context), (ii) a scene-graph prior that prefers segments which end near stage boundaries or make measurable progress toward unmet relations (e.g., moving a can toward the basket opening), and (iii) a cheap collision proxy based on minimum capsule or SDF margin along a straight-line waypoint proposal. This fast score is normalized, lightweight, and used to order unvisited children and to define the sampling distribution during simulation.

After we actually *partially denoise* a segment and obtain a concrete candidate, we compute the *true* geometry return on that candidate. It aggregates: terminal pose accuracy (position/orientation at the segment end or episode end), relation satisfaction from the scene graph (higher is better), collision penalties (lower is better), and a mild smoothness term around the edited boundary. All terms are scaled to comparable ranges and combined with fixed validation-tuned weights into a single scalar. The true return is the value that is backed up along the path to update (N,W,Q) in MCTD. In short, the fast score guides exploration cheaply, while the true return provides faithful evaluation once a candidate has been generated.

b) Robust P-UCT selection.: We select actions with Laplace smoothing to avoid $\log(0)$ and division by zero:

$$\text{Score}(n,a)=Q(n,a)+c_{\text{puct}}\,P(n,a)\sqrt{\frac{\log\big(N(n)+1\big)}{N(n,a)+1}}, \qquad \text{(III.6)}$$

where $P(n,a)\propto\exp\{\psi(a;\mathscr{G})\}$. If there exist unvisited children $\mathscr{U}=\{a:N(n,a)=0\}$, we pick among them by the fast reward; otherwise we use Eq. (III.6):

$$a^\star=\begin{cases}\arg\max\limits_{a\in\mathscr{U}} r_{\text{fast}}(n,a), & \text{if } \mathscr{U}\neq\emptyset,\\ \arg\max\limits_{a\in\mathscr{A}(n)} \text{Score}(n,a), & \text{otherwise.}\end{cases} \qquad \text{(III.7)}$$

c) Simulation with normalized sampling.: During rollout, when a *sampling* policy is enabled, we pick children by a softmax over fast rewards:

$$p(a\mid n)=\frac{\exp\big(r_{\text{fast}}(n,a)-\max_{a'} r_{\text{fast}}(n,a')\big)}{\sum_{a''}\exp\big(r_{\text{fast}}(n,a'')-\max_{a'} r_{\text{fast}}(n,a')\big)}. \qquad \text{(III.8)}$$

which is well-defined even when all fast rewards are equal; a greedy policy is recovered by $\arg\max_a r_{\text{fast}}(n,a)$.

IV. Experiments

A. Benchmarks and Baselines

*a) **Benchmarks.**:* We train on **EMMA-X** (stage-annotated manipulation trajectories) [9] and evaluate in simulation on LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long (10 tasks per suite; 500 expert demos per

TABLE II

Model	SR (%)				Average
	Libero-Spatial	Libero-Object	Libero-Goal	Libero-Long	
From scratch					
Diffusion Policy [7]	78.3	92.5	68.3	50.5	72.4
MDT [42]	78.5	87.5	73.5	64.8	76.1
Seer (scratch) [43]	–	–	–	78.7	–
Fine-tuned from pretrained VLMs					
Seer (fine-tuned) [43]	–	–	–	87.7	–
Dita / DiT Policy [44]	84.2	96.3	85.4	63.8	82.4
TraceVLA [45]	84.6	85.2	75.1	54.1	74.8
SpatialVLA [46]	88.2	89.9	78.6	55.5	78.1
π_0 [1]	96.8	**98.8**	95.8	85.2	94.2
GR00T-N1 [47]	94.4	97.6	93.0	90.6	93.9
Discrete Diffusion VLA	**97.2**	98.6	**97.4**	**92.0**	**96.3**
OpenVLA [2]	84.7	88.4	79.2	53.7	76.5
Octo-Base [6]	78.9	85.7	84.6	51.1	75.1
FORGE-Tree + OpenVLA	**92.4**	**94.7**	**89.3**	**83.2**	**89.9**
FORGE-Tree + Octo	**96.6**	**88.2**	**93.4**	**91.0**	**92.3**

Algorithm 1 PartialDenoise($\mathbf{x}_{t_K}$, $a=(k,m,s,w,\tau)$, f_θ,$\mathbf{c}$,$\mathscr{G}$)

Build grid $\{t_J\}_{J=K}^{0}$ with stride s; set $S \leftarrow S_{k,m}$, $\bar{S} \leftarrow I - S$
for $J = K, K-1, \ldots, 1$ **do**
  Compute $\boldsymbol{\varepsilon}_\theta$ and $\hat{\mathbf{x}}_0$ by (III.3)
  $\tilde{\boldsymbol{\varepsilon}} \leftarrow \boldsymbol{\varepsilon}_\theta - w \nabla_{\mathbf{x}_{t_J}} U(\hat{\mathbf{x}}_0; \mathscr{G})$
  Sample $\boldsymbol{\eta} \sim \mathcal{N}(0, \tau^2 \mathbf{I})$
  $\mathbf{x}_{t_{J-1}} \leftarrow \bar{S}\mathbf{x}_{t_J} + S\big(\sqrt{\bar{\alpha}_{t_{J-1}}}\,\hat{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t_{J-1}}}\,\boldsymbol{\eta}\big)$
return $\hat{\mathbf{x}}_0$ from the last iteration

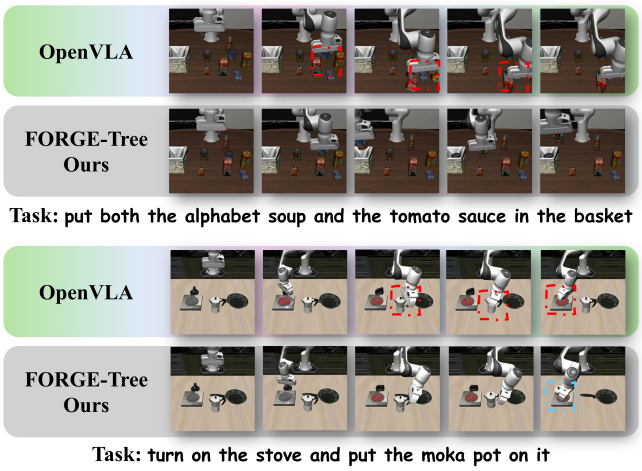

Fig. 4. Top: OpenVLA collides (red dashed boxes). Bottom: **FORGE-Tree** edits only the upcoming segment with geometry guidance and places actions inside the goal region with proper clearance.

Algorithm 2 Stage-Aware MCTD with Dual Rewards

Require: Stages $\{\mathsf{stage}_1, \ldots, \mathsf{stage}_S\}$, VLA f_{VLA}, diffusion head f_θ, budget per stage B
$\mathbf{c} \leftarrow f_{\mathrm{VLA}}(\mathbf{o}, u)$; $\mathscr{G} \leftarrow \mathrm{BuildSceneGraph}(\mathbf{o}, u)$
Initialize root n_0 with current prefix and $\mathbf{x}_{t_K}$; n_0.children $\leftarrow [\,]$
for $s = 1$ **to** S **do** ▷ stage-wise planning
  Create stage node $n^{(s)}$ (parent n_0) and append to n_0.children
  for $b = 1$ **to** B **do**
    $n \leftarrow n^{(s)}$, store path $\mathscr{P} \leftarrow [\,]$
    while n expandable **do**
      Generate $\mathscr{A}(n)$ with scene-graph prior $P(\cdot \mid n)$
      if $\exists a \in \mathscr{A}(n)$ with $N(n,a)=0$ **then** $a^\star \leftarrow \arg\max_{a:N(n,a)=0} r_{\mathrm{fast}}(n,a)$
      else $a^\star \leftarrow \arg\max_{a \in \mathscr{A}(n)} \mathrm{Score}(n,a)$ using Eq. (III.6)
      Append $(n, a^\star)$ to $\mathscr{P}$ and move to child $n \leftarrow \mathrm{Child}(n, a^\star)$ (create if absent)
    $\hat{\mathbf{x}}_0 \leftarrow \textsc{PartialDenoise}(\mathbf{x}_{t_K}, a^\star, f_\theta, \mathbf{c}, \mathscr{G})$
    $R \leftarrow R(\hat{\mathbf{x}}_0; \mathscr{G})$
    for (n,a) **in** $\mathscr{P}$ **do** update (N, W, Q)
  Choose best child of n_0 for stage s by visits or value; commit its first segment; observe, update $\mathscr{G}$
return concatenated (tasks, states, actions) trace for all stages

suite) and ManiSkill (diverse physics tasks) following each suite's official protocol (episode budgets, success definitions, and resets) [10], [11]. Stage boundaries come from an ECoT-style parser; we also extract a *Qwen2.5-VL* [48] embedding as additional context for stage conditioning and scene-graph construction. Stage parsing is used only to align training targets on EMMA-X; at test time, no stage labels are used, the search selects segments from decoded tokens alone. Both the parser and the Qwen2.5-VL encoder are frozen and are not fine-tuned on LIBERO or ManiSkill. All methods use the same observations (RGB only unless noted), action space (7-DoF EE pose + gripper), and demonstration splits. We train *only* the control head on EMMA-X and *do not* fine-tune on LIBERO or ManiSkill. All results are *zero-shot evaluations* of the control layer on top of frozen VLAs (OpenVLA and Octo). Baselines that report numbers using LIBERO

or ManiSkill demonstrations are taken from their official protocols or re-runs; we do *not* use these demonstrations to train our head.

b) ***Backbones.:*** Unless otherwise stated, the Vision–Language-Action encoder is *frozen* OpenVLA or Octo; optional LoRA uses the official OpenVLA implementation. Our controller is the proposed VLA-conditioned diffusion head trained with Diffusion Forcing (DF) and executed with MCTD at test time.

c) ***Baselines.:*** We compare against representative policies spanning the two dominant paradigms—*autoregressive (AR) token decoders* and *continuous diffusion or flow-matching* policies—covering both models trained from scratch and fine-tuned from large pretrained bases. **AR action decoders.** RT-1-X / RT-2-X [19], OpenVLA [2], Octo-Small / Octo-Base [6], HPT [49], TraceVLA [45], and SpatialVLA [46] instantiate AR-style generation of discrete action tokens on a unified VLM backbone. **Continuous diffusion or flow-matching.** Diffusion Policy [7], MDT [42], DiT Policy (DiTA) [44], $\pi 0$ [1], and GR00T-N1 [47] implement denoising or flow-matching heads over continuous action trajectories; Seer [43] is reported in both scratch and fine-tuned forms.

Control-layer comparison. Because our method is a *plug-in control layer* over frozen VLA backbones, we additionally report *OpenVLA and Octo + FORGE-Tree* (DF-only and DF+MCTD) alongside native baselines to highlight the incremental gains achievable without changing the encoder. Numbers marked "+ FORGE-Tree" reuse the exact frozen checkpoint and differ only in the control layer at inference.

B. Performance Comparisons

a) ***Training details.:*** We train *only* the diffusion control head with Diffusion Forcing (DF) on top of a frozen VLA encoder; unless stated, there is no benchmark-specific fine-tuning on LIBERO or ManiSkill. Inputs follow the backbone's preprocessing (RGB 224×224, backbone normalization). Actions are 7-D (EE x,y,z, orientation, gripper), standardized per-dataset and clipped to $[-3\sigma,3\sigma]$. From EMMA-X we sample windows of L=64 actions (pad/trunc as needed) with ECoT-derived stage boundaries. For each window, we draw 1–2 stages and assign a constant per-stage timestep $t_j \sim \mathscr{U}[400,900]$ (others t=0); diffusion horizon T=1000 with a cosine schedule. The DF objective uses fixed weights λ_{EoT}=5.0, λ_{stage}=3.0, λ_{traj}=1.0, λ_{smooth}=0.1; padding is fully masked. Optimization: AdamW (β=(0.9, 0.999), weight decay 0.05), peak LR 3×10^{-4} (control head) with cosine decay and 2k warmup, total 200k steps, global-norm clipping 1.0, EMA 0.999. Training uses bf16 and PyTorch DDP on 4×48 GB Ada GPUs, per-GPU batch 64 with ×2 accumulation (effective 512). Augmentations are mild: color jitter 0.2, horizontal flip p=0.5 when mirror-invariant, and Gaussian action noise $\mathscr{N}(0,0.01)$; we avoid heavy geometric transforms to preserve metric geometry.

b) ***LIBERO results.:*** Table II reports success rates (SR) on the four LIBERO suites. As a *control-layer* add-on, **FORGE-Tree** delivers substantial gains over its underlying VLAs without changing the encoder: on **OpenVLA** it lifts the average SR from 76.5% to 89.9% (**+13.4 pp**), with per-suite improvements of **+7.7** (Spatial), **+6.3** (Object), **+10.1** (Goal), and **+29.5** pp (Long). On **Octo**, it raises the average from 75.1% to 92.3% (**+17.2 pp**), with suite-wise gains of **+17.7**, **+2.5**, **+8.8**, and **+39.9** pp, respectively. The largest improvements occur on *LIBERO-Long*, consistent with our stage-wise partial denoising and geometry-aware planning designed for multi-step, contact-rich sequences.

For context, state-of-the-art discrete-diffusion decoders report an average SR of 96.3% on these suites; FORGE-Tree *narrows the gap to within* **4–6 pp** while operating as a plug-in control layer on frozen OpenVLA or Octo backbones. Compared with from scratch diffusion or flow-matching policies, our results remain competitive across suites, highlighting that test-time, tree-structured denoising can recover a large fraction of long-horizon performance without retraining the vision–language encoder.

TABLE III

EFFECT OF REMOVING STAGE GUIDANCE (OPENVLA). ABSOLUTE SR AND RELATIVE CHANGE VS. FULL.

	LIBERO–Goal		**LIBERO–Long**	
Variant	**SR (%)**	**ΔSR**	**SR (%)**	**ΔSR**
Full (ours)	**[88.2]**	0	**[83.2]**	0
w/o Stage guidance	[75.9]	**-14**	[60.7]	**-27**

C. Ablation Study

We compare the full model against (i) *no stage guidance* in DF and (ii) *no scene graph* in MCTD. Removing stage guidance consistently lowers SR and increases terminal pose error, most notably on LIBERO–Goal and Long, which decrease by **14%** and **27%** average each, showing that stage-aligned supervision is essential for landing intermediate subgoals. Removing the scene graph reduces relation satisfaction, increases collisions, and requires larger search budgets to match SR, demonstrating that $\mathscr{G}$ supplies both effective exploration priors and geometry-aware evaluation. These results confirm that stage-aware DF and scene-graph structure are key to long-horizon reliability.

D. Qualitative Failure Analysis

We observed three recurring failures for FORGE-TREE. (i) *Scene-graph errors.* False/weak detections (e.g., a shallow rim or occluded handle) inject noisy relations, so search refines a non-critical segment while the terminal segment remains suboptimal. (ii) *Search overhead and slow rewards.* Tree expansion can be compute-heavy when end-of-segment evaluation relies on expensive geometry checks, delaying value estimates and reducing effective breadth. (iii) *Stall at the first grasp.* When the initial pick fails to establish stable contact, the policy keeps revisiting the same short segment with conservative edits, making little physical progress.

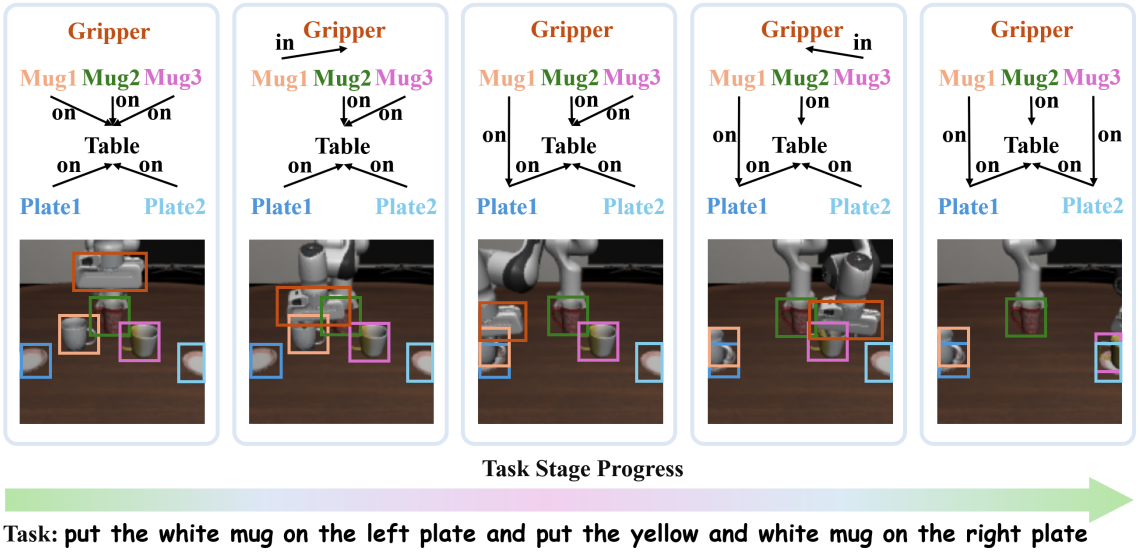

Fig. 5. Left→right: snapshots from a LIBERO task ("put the white mug on the left plate and put the yellow mug on the right plate") with **FORGE-Tree**. In each panel, the top diagram shows the current scene graph $\mathcal{G}$ with nodes (Gripper, Mug1–3, Plate1–2, Table) and edges labeled by relations (`in`, `on`); the bottom image shows the corresponding RGB frame with color-matched boxes. As the episode advances, relations update (e.g., `in`(Gripper, Mug), `on`(Mug, Plate)), marking subgoal completion. FORGE-Tree uses this graph both to bias expansion (priors over meta-actions) and to score candidates via geometry relation predicates, enabling stage-wise planning that satisfies the relational goals.

V. CONCLUSION

This work introduces **FORGE-Tree**, a control-layer framework that couples a VLA-conditioned *Diffusion Forcing* head with test-time *Monte Carlo Tree Diffusion*. The key idea is to plan in the space of *editable trajectory segments*: training aligns denoising with stage structure so the decoder learns to land intermediate subgoals, while inference performs *tree-structured denoising* over meta-actions (segment, stride, guidance, temperature) guided by a scene graph. This turns diffusion decoding into a budget-scalable planner that preserves a good prefix and allocates compute where geometry is tight.

Our study suggests a practical path to stronger long-horizon manipulation by *augmenting* rather than replacing pretrained VLAs. Looking forward, we aim to (i) deploy on real hardware and study sim-to-real effects, (ii) jointly learn scene-graph perception with task priors, (iii) amortize meta-action proposals to reduce planning latency, and (iv) extend to partially observed, multi-object settings with tighter contact modeling. We hope this bridges symbolic task structure and continuous diffusion control in a way that remains modular, interpretable, and compute-aware.

REFERENCES

[1] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "*pi*_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

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

[3] Y. Mu, Q. Zhang, M. Hu, W. Wang, M. Ding, J. Jin, B. Wang, J. Dai, Y. Qiao, and P. Luo, "Embodiedgpt: Vision-language pre-training via embodied chain of thought," *Advances in Neural Information Processing Systems*, vol. 36, pp. 25 081–25 094, 2023.

[4] Y. Mu, T. Chen, S. Peng, Z. Chen, Z. Gao, Y. Zou, L. Lin, Z. Xie, and P. Luo, "Robotwin: Dual-arm robot benchmark with generative digital twins (early version)," in *European Conference on Computer Vision*. Springer, 2024, pp. 264–273.

[5] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine, "Robotic control via embodied chain-of-thought reasoning," *arXiv preprint arXiv:2407.08693*, 2024.

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

[7] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[8] Y. Huang, R. Li, and Z. Tu, "Pandora: Diffusion policy learning for dexterous robotic piano playing," 2025. [Online]. Available: https://arxiv.org/abs/2503.14545

[9] Q. Sun, P. Hong, T. D. Pala, V. Toh, U. Tan, D. Ghosal, S. Poria *et al.*, "Emma-x: An embodied multimodal action model with grounded chain of thought and look-ahead spatial reasoning," *arXiv preprint arXiv:2412.11974*, 2024.

[10] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 44 776–44 791, 2023.

[11] T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su, "Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations," *arXiv preprint arXiv:2107.14483*, 2021.

[12] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[13] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[14] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[15] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[16] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*. PMLR, 2023, pp. 2165–2183.

[17] J. Lee, J. Duan, H. Fang, Y. Deng, S. Liu, B. Li, B. Fang, J. Zhang, Y. R. Wang, S. Lee *et al.*, "Molmoact: Action reasoning models that can reason in space," *arXiv preprint arXiv:2508.07917*, 2025.

[18] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," *arXiv preprint arXiv:2403.12945*, 2024.

[19] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[20] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[21] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[22] H. Li, G. Zhu, L. Zhang, Y. Jiang, Y. Dang, H. Hou, P. Shen, X. Zhao, S. A. A. Shah, and M. Bennamoun, "Scene graph generation: A comprehensive survey," *Neurocomputing*, vol. 566, p. 127052, 2024.

[23] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[24] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[25] Y. Liu, K. Zhang, Y. Li, Z. Yan, C. Gao, R. Chen, Z. Yuan, Y. Huang, H. Sun, J. Gao *et al.*, "Sora: A review on background, technology, limitations, and opportunities of large vision models," *arXiv preprint arXiv:2402.17177*, 2024.

[26] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," *arXiv preprint arXiv:2403.03954*, 2024.

[27] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *arXiv preprint arXiv:2205.09991*, 2022.

[28] U. A. Mishra, S. Xue, Y. Chen, and D. Xu, "Generative skill chaining: Long-horizon skill planning with diffusion models," in *Conference on Robot Learning*. PMLR, 2023, pp. 2905–2925.

[29] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu, "Rdt-1b: a diffusion foundation model for bimanual manipulation," *arXiv preprint arXiv:2410.07864*, 2024.

[30] J. Wen, M. Zhu, Y. Zhu, Z. Tang, J. Li, Z. Zhou, C. Li, X. Liu, Y. Peng, C. Shen *et al.*, "Diffusion-vla: Generalizable and interpretable robot foundation model via self-generated reasoning," *arXiv preprint arXiv:2412.03293*, 2024.

[31] J. Wen, Y. Zhu, J. Li, M. Zhu, Z. Tang, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen *et al.*, "Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation," *IEEE Robotics and Automation Letters*, 2025.

[32] M. Zhu, Y. Zhu, J. Li, J. Wen, Z. Xu, N. Liu, R. Cheng, C. Shen, Y. Peng, F. Feng *et al.*, "Scaling diffusion policy in transformer to 1 billion parameters for robotic manipulation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 10 838–10 845.

[33] B. Chen, D. Martí Monsó, Y. Du, M. Simchowitz, R. Tedrake, and V. Sitzmann, "Diffusion forcing: Next-token prediction meets full-sequence diffusion," *Advances in Neural Information Processing Systems*, vol. 37, pp. 24 081–24 125, 2024.

[34] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, no. 1, pp. 265–293, 2021.

[35] U. A. Mishra, Y. Chen, and D. Xu, "Generative factor chaining: Coordinated manipulation with diffusion-based factor graph," in *ICRA 2024 Workshop {\textemdash} Back to the Future: Robot Learning Going Probabilistic*, 2024.

[36] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning," *arXiv preprint arXiv:2307.06135*, 2023.

[37] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[38] J. Duan, W. Yuan, W. Pumacay, Y. R. Wang, K. Ehsani, D. Fox, and R. Krishna, "Manipulate-anything: Automating real-world robots using vision-language models," *arXiv preprint arXiv:2406.18915*, 2024.

[39] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[40] C. Neary, O. G. Younis, A. Kuramshin, O. Aslan, and G. Berseth, "Improving pre-trained vision-language-action policies with model-based search," *arXiv preprint arXiv:2508.12211*, 2025.

[41] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[42] M. Reuss, Ö. E. Yağmurlu, F. Wenzel, and R. Lioutikov, "Multimodal diffusion transformer: Learning versatile behavior from multimodal goals," *arXiv preprint arXiv:2407.05996*, 2024.

[43] Y. Tian, S. Yang, J. Zeng, P. Wang, D. Lin, H. Dong, and J. Pang, "Predictive inverse dynamics models are scalable learners for robotic manipulation," *arXiv preprint arXiv:2412.15109*, 2024.

[44] Z. Hou, T. Zhang, Y. Xiong, H. Duan, H. Pu, R. Tong, C. Zhao, X. Zhu, Y. Qiao, J. Dai *et al.*, "Dita: Scaling diffusion transformer for generalist vision-language-action policy," *arXiv preprint arXiv:2503.19757*, 2025.

[45] R. Zheng, Y. Liang, S. Huang, J. Gao, H. Daumé III, A. Kolobov, F. Huang, and J. Yang, "Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies," *arXiv preprint arXiv:2412.10345*, 2024.

[46] D. Qu, H. Song, Q. Chen, Y. Yao, X. Ye, Y. Ding, Z. Wang, J. Gu, B. Zhao, D. Wang *et al.*, "Spatialvla: Exploring spatial representations for visual-language-action model," *arXiv preprint arXiv:2501.15830*, 2025.

[47] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, "Gr00t n1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[48] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang *et al.*, "Qwen2. 5-vl technical report," *arXiv preprint arXiv:2502.13923*, 2025.

[49] L. Wang, X. Chen, J. Zhao, and K. He, "Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers," *Advances in neural information processing systems*, vol. 37, pp. 124 420–124 450, 2024.