Jointly Learning Predicates and Actions Enables Zero-Shot Skill Composition

Author Names Omitted for Anonymous Review

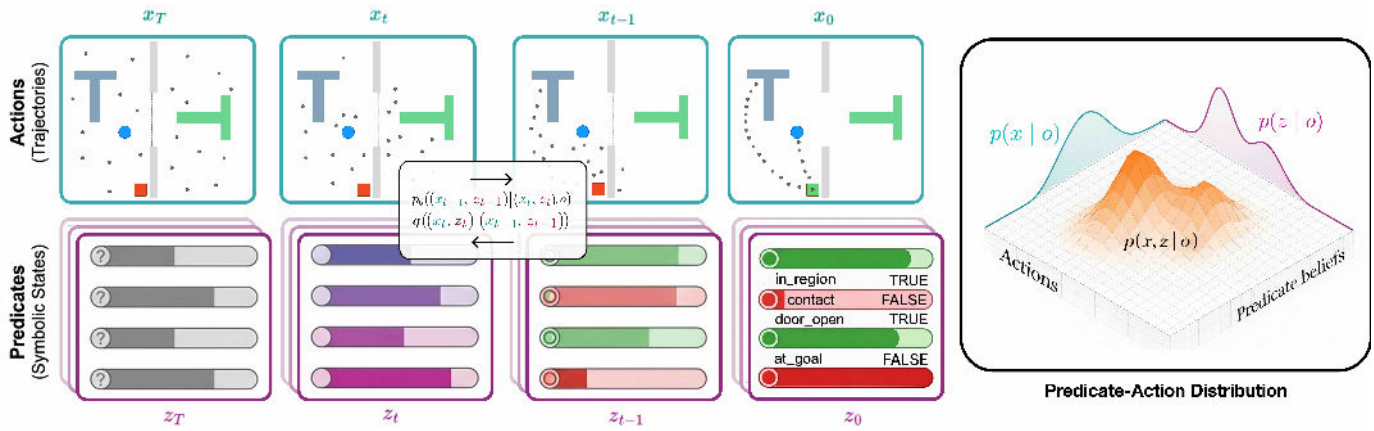

Fig. 1: **Predicate-Action Skills**. Conditioned on current observations $\mathbf{o}$, we model skills as a joint generative process over an action trajectory $\mathbf{x}$ and a predicate-belief trajectory $\mathbf{z}$ by learning the coupled distribution $p(\mathbf{x}, \mathbf{z} \mid \mathbf{o})$. Starting from noise $(\mathbf{x}_T, \mathbf{z}_T)$, our model iteratively refines both modalities to produce temporally coherent action–outcome rollouts $(\mathbf{x}_0, \mathbf{z}_0)$. The resulting predicate-belief trajectory $\mathbf{z}_0$ provides an online symbolic interface for monitoring skill execution and planning-based skill composition using off-the-shelf planners.

***Abstract*—Learning from Demonstration (LfD) enables robots to learn complex behaviors from expert examples, yet existing approaches often fail to generalize to new compositions of known skills without retraining. Modern generative policies model distributions over action trajectories alone, thus are unable to reason about the symbolic outcomes required for robust composition. We propose that skills should jointly model action trajectories and the symbolic outcomes they induce. To address this gap, we introduce Predicate-Action Skills (PACTS), a class of closed-loop visuomotor policies that model skills as a joint generative process over action and predicate belief trajectories, producing coherent action–outcome rollouts within a single model. Jointly generating actions and predicates enables PACTS to learn internal representations that improve both action generation and predicate classification. Furthermore, we demonstrate zero-shot composition of learned skills via planning by leveraging online predicate predictions from PACTS as a symbolic interface for sequencing and monitoring execution.**

I. Introduction

Learning from Demonstration (LfD) has become a standard approach for teaching robots complex behaviors. Recent generative visuomotor policies are particularly compelling in this setting: by modeling distributions over action trajectories effectively capture multi-modal behavior. Yet, the gap between learning short-horizon behaviors and solving long-horizon, multi-step tasks remains open. In long-horizon execution, distribution shift compounds over time and the space of possible task variations and compositions grows combinatorially. Closing this gap typically requires large-scale demonstration datasets that cover many such variations or retraining whenever the desired composition changes.

A promising route to long-horizon generalization is to introduce modular structure: reusable *skills* that capture short-horizon behavior, and symbolic *predicates* that describe state and skill effects in a form amenable to planning. By abstracting both temporal behavior (via skills) and state (via predicates), prior work has shown that robots can generalize to unseen task compositions by sequencing learned skills to achieve goals [1, 2]. However, these prior works typically *decouple* these components, training one model to generate actions and a separate model to classify predicates from observations. This separation obscures a central fact about manipulation: *actions and symbolic outcomes are tightly coupled*. For composition, what matters is not only which actions are plausible, but which symbolic conditions those actions will make true.

This paper starts from a simple alternative viewpoint: skills should not be represented only as a distribution over action trajectories, but as distributions over *action–outcome rollouts*. We represent outcomes as a *predicate-belief trajectory*—a compact trace of soft truth values over predicates that describes what the model expects to become true (and with what confidence) as the skill unfolds. Given an observation $\mathbf{o}$, we therefore model a coupled distribution over an action trajectory $\mathbf{x}$ and an outcome trace $\mathbf{z}$:

$$(\mathbf{x}, \mathbf{z}) \sim p(\mathbf{x}, \mathbf{z} \mid \mathbf{o}).$$

Sampling from this joint distribution yields coherent pairs, as the same sample that commits to an action mode also commits to the corresponding symbolic trace. We instantiate this idea with **Predicate-Action Skills** (PACTS), closed-loop visuomotor policies that model skills as a **joint generative process** over action trajectories and predicate-

Manuscript 3773 submitted to 2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 5, 2026.

belief trajectories. PACTS iteratively refines both modalities to produce temporally aligned rollouts $(\mathbf{x}_0, \mathbf{z}_0)$ within a single model (see Fig.1). Crucially, the resulting predicate-belief trajectory $\mathbf{z}_0$ provides a practical interface for planning-based skill composition: it can be used to check preconditions and effects, monitor execution, and trigger replanning when expectations are violated—enabling zero-shot composition of learned skills with off-the-shelf planners.

Our formulation supports both diffusion-style denoising and flow-matching objectives, and instantiates naturally with different sequence backbones (e.g., UNet- and Transformer-based policies). We systematically evaluate PACTS on a compositional 2D benchmark for comprehensive ablations and on two 3D RoboMimic tasks [3, 4], spanning a broad grid of model variants. Across these settings, jointly generating actions and predicate beliefs maintains competitive performance and often improves both action performance and predicate prediction relative to action-only generative policies and pipelines that pair action generation with a separate predicate predictor. Finally, we demonstrate planning-based composition of learned subskills in simulation and in a real-world environment.

In summary, we make three contributions: **(1)** a joint generative skill formulation that models action trajectories and predicate traces to produce coherent action–outcome rollouts; **(2)** planning-based skill composition using predicate-belief traces as an online interface for sequencing and monitoring with off-the-shelf planners; and **(3)** an open-source skill segmentation and labeling toolkit that converts monolithic demonstrations into skill-centric datasets for joint $(\mathbf{x}, \mathbf{z})$ learning in real-world deployments.

II. Related Work

a) Generative visuomotor policies for imitation learning: Behavior cloning learns skills by mapping observations to actions [5], and recent work increasingly models *distributions* over action trajectories to capture multi-modality and long-horizon structure. Diffusion-based policies [6] and conditional flow matching [7] are prominent instantiations, with complementary approaches improving data efficiency via pretrained vision-language representations [8, 9] or architectural/representational structure (e.g., equivariance and object-/affordance-centric models) [10, 11, 12]. Diffusion has also been used for planning: Diffuser [13] denoises state–action trajectories to enable flexible test-time conditioning, whereas Diffusion Policy [6] treats control as learning the distribution over action sequences, conditioning on observations without denoising them for efficient real-time inference. We adopt this conditional-policy perspective, but depart from action-only modeling by learning a coupled generator $p(\mathbf{x}, \mathbf{z} \mid \mathbf{o})$ that jointly samples actions and a low-dimensional predicate-belief rollout $\mathbf{z}$ aligned with PDDL preconditions/effects, yielding an outcome-aware interface for execution-time monitoring and planning-based composition. Vision–language–action models similarly couple control with language semantics, but their semantics are typically part of the conditioning interface rather than an explicit planner-consumable outcome trajectory.

b) Skills, abstraction, and planning-based composition: Automated task planning in robotics [14] specifies a domain (often in PDDL [15]) in terms of abstract actions with preconditions and effects, and uses search to synthesize sequences that achieve goal conditions. In robotic systems, these abstract actions are typically instantiated as hand-designed controllers (e.g., finite-state machines [16] or behavior trees [17]), and predicates are grounded through engineered perception and state estimation modules [18]. Several frameworks support such symbolic planning architectures [19, 20, 21]. Recent learning-based approaches aim to reduce manual engineering by learning both skill policies and predicate evaluators from data: policies are learned via behavior cloning or reinforcement learning, while predicates are grounded either with supervised classifiers or with pretrained vision–language models that infer symbolic propositions from observations. A common design pattern in learned planning pipelines is to train a policy separately from a predicate evaluator (classifier or pretrained model) and use the latter as the planning/monitoring interface. However, in long-horizon manipulation, actions and symbolic outcomes are tightly coupled: different action *modes* correspond to different predicate transitions, and treating them as separate predictions can lead to inference-time inconsistency. PACTS addresses this coupling directly by modeling a skill as a *joint* generative process over action trajectories and predicate-belief trajectories. By sampling $(\mathbf{x}, \mathbf{z})$ from a single model, the predicted outcome trace remains aligned with the sampled action mode.

III. Jointly Modeling Predicates and Actions

A. Problem definition: joint action–outcome skill model

We model a closed-loop skill as a conditional distribution over a horizon-H action trajectory $\mathbf{x} = (\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1}) \in \mathbb{R}^{H \times d_a}$, where each $\mathbf{a}_{t+h} \in \mathbb{R}^{d_a}$ is a single control action. To support planning-based skill composition, we additionally model the symbolic outcomes induced by actions using a *predicate-belief trajectory* $\mathbf{z} = (\mathbf{z}_{t+1}, \ldots, \mathbf{z}_{t+H}) \in [0,1]^{H \times J}$, where $\mathbf{z}_{t+h,j}$ is the model's confidence that predicate $j \in \{1, \ldots, J\}$ will hold at future step $t + h$.

Given the current observation $\mathbf{o}_t$, PACTS learn the coupled distribution $p_\theta(\mathbf{x}, \mathbf{z} \mid \mathbf{o}_t)$ so that a single sample yields a coherent action–outcome rollout. We implement this by defining the joint trajectory variable $\mathbf{y} \triangleq [\mathbf{x} \,|\, \mathbf{z}] \in \mathbb{R}^{H \times (d_a + J)}$ and learning a single conditional generative model $p_\theta(\mathbf{y} \mid \mathbf{o}_t)$. This joint modeling choice is the key distinction from decoupled or multi-task baselines: $\mathbf{z}$ is not produced by an independent classifier head, but is part of the generated trajectory sample.

a) Predicate-belief representation.: In the dataset, predicates are provided as binary labels in $\{0,1\}^J$. We treat predicate channels as continuous during generation by linearly normalizing labels to $[-1, 1]$ and learning to generate them jointly with actions. At inference, we map the generated predicate channels back to $[0, 1]$, yielding soft beliefs that can be thresholded when discrete predicate evaluations are required (e.g., checking preconditions/effects during planning).

Manuscript 3773 submitted to 2026 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 5, 2026.

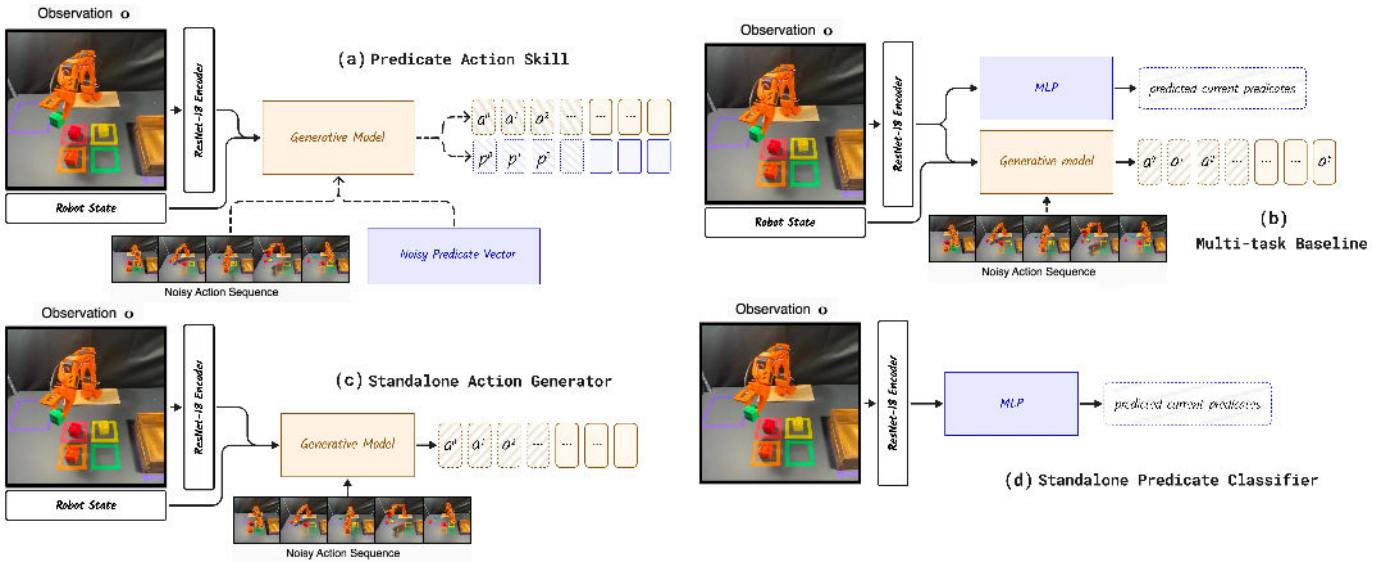

Fig. 2: **Architectures and baselines. (a)** PACTS uses a *single* conditional generative model to sample *joint* action–predicate rollouts, refined together via DDPM denoising or flow matching. To *disentangle* joint modeling from multi-task learning, we include a multi-task baseline **(b)** that shares the same observation encoder but predicts predicates and actions with *separate* heads trained with a combined loss. Standalone baselines isolate each component: **(c)** a generative *action-only* generator that generates actions, and **(d)** a *predicate-only* model predicts predicates from observations.

B. DDPM formulation

Following diffusion policy [6], we model the conditional trajectory distribution $p(\mathbf{y} \mid \mathbf{o}_t)$: we condition on $\mathbf{o}_t$ but do not denoise observations or infer future observation (or state), avoiding the cost of modeling observations and trajectories [13]. We apply denoising diffusion probabilistic modeling (DDPM) [22] to the joint variable $\mathbf{y} = [\mathbf{x} \,|\, \mathbf{z}]$. Let $q(\mathbf{y}^k \mid \mathbf{y}^{k-1})$ denote the forward noising process with a standard noise schedule $\{\alpha_k\}_{k=1}^K$. We train a noise-prediction network ϵ_θ, conditioned on $\mathbf{o}_t$, to predict the injected noise:

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{\mathbf{o}_t, \mathbf{y}^0, k, \epsilon}\left[\left\|\epsilon_\theta(\mathbf{o}_t, \mathbf{y}^k, k) - \epsilon\right\|^2\right], \tag{1}$$

where $\mathbf{y}^k$ is obtained by noising the clean joint trajectory $\mathbf{y}^0 = [\mathbf{x} \,|\, \mathbf{z}]$ at *denoising step* k. At inference, we sample $\mathbf{y}^K \sim \mathcal{N}(0, I)$ and iteratively denoise to obtain $\mathbf{y}^0$.

C. Conditional Flow Matching formulation

The joint object $\mathbf{y} = [\mathbf{x} \,|\, \mathbf{z}]$ can also be learned via *conditional flow matching* (CFM) [23, 24, 9], which targets the same conditional distribution $p_\theta(\mathbf{y} \mid \mathbf{o}_t)$ but uses a different objective and sampler. Let $\mathbf{y}^1 \sim \mathcal{N}(0, I)$ and define the optimal transport path between data $\mathbf{y}^0$ and noise $\mathbf{y}^1$ as

$$\mathbf{y}_\tau = \tau\,\mathbf{y}^1 + (1-\tau)\,\mathbf{y}^0, \qquad \tau \in [0,1], \tag{2}$$

with constant target velocity $\mathbf{u}_\tau = \frac{d}{d\tau}\mathbf{y}_\tau = \mathbf{y}^1 - \mathbf{y}^0$. A conditional vector field $v_\theta(\mathbf{o}_t, \mathbf{y}_\tau, \tau)$ is trained by regression:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{\mathbf{o}_t, \mathbf{y}^0, \mathbf{y}^1, \tau}\left[\left\|v_\theta(\mathbf{o}_t, \mathbf{y}_\tau, \tau) - (\mathbf{y}^1 - \mathbf{y}^0)\right\|^2\right]. \tag{3}$$

At inference, samples are obtained by integrating from noise (τ=1) to data (τ=0), yielding $\mathbf{y}^0$.

Importantly, DDPM and flow matching instantiate the same modeling choice—learning the coupled action–outcome distribution $p_\theta(\mathbf{x}, \mathbf{z} \mid \mathbf{o}_t)$—but provide complementary training lenses with different sampling trade-offs.

D. Architecture

As shown in Fig. 2(a), PACTS factorizes into (i) an observation encoder and (ii) a conditional trajectory generator. The observation encoder maps the current observation $\mathbf{o}_t$ into a latent embedding used to condition the generator. The generator implements either a DDPM noise-prediction network $\epsilon_\theta(\mathbf{o}_t, \mathbf{y}^k, k)$ (Sec. III-B) or a CFM velocity field $v_\theta(\mathbf{o}_t, \mathbf{y}_\tau, \tau)$ (Sec. III-C), operating on the joint trajectory $\mathbf{y} = [\mathbf{x} \,|\, \mathbf{z}]$.

We instantiate the trajectory generator with two backbones: (i) a temporal UNet that applies 1D convolutions over the horizon dimension, and (ii) a Transformer that treats time steps as tokens and models long-range temporal dependencies via self-attention. In both cases, conditioning enters through (a) the observation embedding and (b) the diffusion/flow time parameter (discrete k for DDPM, continuous τ for CFM), injected via standard conditional modulation.

Like prior trajectory-generating visuomotor policies, PACTS can be executed in a receding-horizon manner: at each control cycle, the policy conditions on the current observation $\mathbf{o}_t$, samples a length-H action chunk $\mathbf{x}$ (and its paired predicate-belief trajectory $\mathbf{z}$), executes a short prefix of $\mathbf{x}$, then re-observes and re-samples.

IV. PLANNING-BASED SKILL COMPOSITION WITH PREDICATE BELIEFS

A key benefit of modeling skills as joint action–outcome rollouts is that the generated predicate-belief trajectory provides a symbolic interface for composition. We assume a PDDL symbolic planning domain specifying skill-level operators with preconditions and effects over predicates, and a PDDL problem instance specifying an initial predicate state and a goal. Given a library of learned skills, an off-the-shelf planner synthesizes a sequence of skills to achieve a

Algorithm 1 Planning with Predicted Predicate Beliefs

Require: Skill library $\mathcal{S}$ with PACTS $\{p_\theta(\mathbf{x}, \mathbf{z} \mid \mathbf{o}, s)\}_{s \in \mathcal{S}}$; planning domain $\mathcal{D}$; PDDL problem instance $\mathcal{P}$ with initial predicate state $\hat{\mathbf{p}}_0$ and goal g

$t \leftarrow 0$; $\hat{\mathbf{p}}_t \leftarrow \hat{\mathbf{p}}_0$; observe $\mathbf{o}_t$
while goal g not satisfied under $\hat{\mathbf{p}}_t$ **do**
  $\pi \leftarrow$ PLAN$(\mathcal{D}, \hat{\mathbf{p}}_t, g)$ {symbolic plan in predicate space}
  $s \leftarrow \pi[0]$ {next planned skill}
  while not EFFECTSSATISFIED$(s, \hat{\mathbf{p}}_t)$ **and not** TERMINATESKILL(s) **do**
    Sample rollout $(\mathbf{x}, \mathbf{z}) \sim p_\theta(\mathbf{x}, \mathbf{z} \mid \mathbf{o}_t, s)$
    Execute first action (or short prefix) from $\mathbf{x}$ and observe $\mathbf{o}_{t+1}$
    $\hat{\mathbf{p}}_{t+1} \leftarrow$ THRESH$(\mathbf{z}_{t+1})$ {predicted next predicate state}
    $t \leftarrow t + 1$; $\hat{\mathbf{p}}_t \leftarrow \hat{\mathbf{p}}_{t+1}$; $\mathbf{o}_t \leftarrow \mathbf{o}_{t+1}$
  end while
end while=0

goal. During execution, PACTS produces online predicate-belief estimates that serve as the planner state, enabling the system to (i) check whether a skill's intended effects are satisfied and (ii) replan from an updated symbolic state.

a) Predicate beliefs as an execution interface: At each decision point, the robot observes $\mathbf{o}_t$ and (for the selected skill s) samples a joint rollout $(\mathbf{x}, \mathbf{z}) \sim p_\theta(\mathbf{x}, \mathbf{z} \mid \mathbf{o}_t, s)$. The belief trajectory $\mathbf{z} \in [0, 1]^{H \times J}$ provides a compact prediction of how symbolic conditions are expected to evolve as the skill unfolds. We obtain a discrete predicate valuation by thresholding beliefs (e.g., $\mathbf{z}_{t+h,j} > 0.5$), and use the predicted next-step predicates $\hat{\mathbf{p}}_{t+1} = \text{THRESH}(\mathbf{z}_{t+1})$ as the symbolic state for effect checking and replanning. Although $\hat{\mathbf{p}}_{t+1}$ is obtained from the model's prediction $\mathbf{z}_{t+1}$ (made when sampling at time t, before observing $\mathbf{o}_{t+1}$), beliefs are re-grounded at every step because the next rollout is conditioned on the latest observation $\mathbf{o}_{t+1}$.

b) Plan–execute–check–replan: Algorithm 1 summarizes our composition loop. We replan at a regular cadence (after each executed skill), using the updated predicted predicate state. This yields a simple integration with off-the-shelf planners: the planner proposes a next skill based on the current symbolic state, and execution proceeds step-by-step until the skill's effects are satisfied (or a termination condition is met), at which point we replan.

A. Skill segmentation and labeling toolkit.

To make real-world deployment tractable, we introduce a skill segmentation and labeling toolkit that converts monolithic demonstrations into training-ready, skill-centric datasets for joint $(\mathbf{x}, \mathbf{z})$ learning. The toolkit operates natively on LeRobot [25] episodic datasets (state/action logs with video observations) and provides three core capabilities: (i) sparse keyframe annotation of grounded predicates defined directly from the PDDL domain/problem files, (ii) automatic conversion of sparse annotations into dense per-timestep predicate labels via forward-fill interpolation, and (iii) skill-level segmentation by identifying goal-achieving predicate configurations and backward-filling executing phases to extract contiguous per-skill segments).

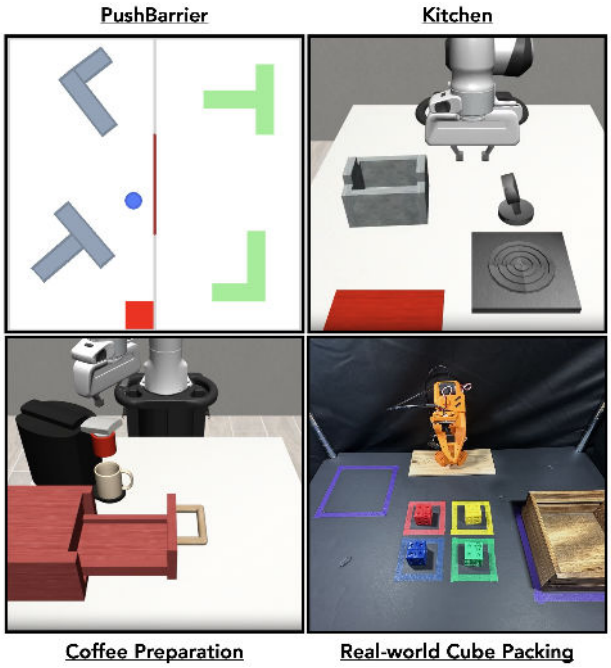

Fig. 3: **Evaluation environments.** We evaluate PACTS across (a) **PushBarrier**, a controlled 2D compositional environment for systematic ablations and planner-based skill sequencing; (b) **Kitchen** and (c) **Coffee Preparation** from RoboMimic/MimicGen, which test joint action–predicate modeling under realistic 3D visual manipulation with simulator predicate oracles; and (d) a **real-world cube packing** task, where a robot packs four colored cubes into a container.

V. EVALUATION

We evaluate PACTS across three(3) tasks in simulation and a real-world multi-step task. Our experiments are designed to answer three (3) key questions:

1) How does joint predicate–action generation affect action performance and predicate classification accuracy?
2) How do modeling choices–DDPM denoising versus conditional flow matching, and UNet versus Transformer backbones–impact performance?
3) Can the learned predicate-belief interface support planning-based recomposition and replanning-based recovery on novel goals without retraining?

A. Tasks and datasets

We evaluate PACTS in three settings that trade off control, realism, and compositional complexity (see Fig. 3).

2D PushBarrier (compositional benchmark). We introduce a multi-step 2D manipulation task adapted from the popular Push-T environment [6]. The workspace contains two objects (a `T` block and an `L` block) that must each be pushed from their respective start regions to corresponding goal regions. Successful completion requires not only reaching the goal regions

but also achieving the correct final *orientation/alignment* of each object with its goal marker. A barrier separates the start and goal regions, and the robot must first press a button to open the barrier before either object can be transferred to the goal side. This environment provides a controlled setting with clear compositional structure and is our primary benchmark for systematic evaluations and ablations.

We use PushBarrier in two complementary evaluation modes. To address Questions (1) and (2) in an end-to-end setting, we train *single* long-horizon policies on the full task and compare task performance and predicate classification accuracy against our baselines. To address Questions (3) in a compositional setting, we construct a library of short-horizon skills and a task-relevant predicate set, segment demonstrations into per-skill datasets, and train a separate PACTS model for each skill. We then evaluate planning-based compositions that require sequencing skills, including *novel skill sequences* not observed during training.

RoboMimic (3D manipulation). To evaluate under realistic visual and geometric complexity, we additionally evaluate on two RoboMimic [3] tasks with MimicGen [4] demonstration data: **Kitchen** and **Coffee Preparation**. These tasks involve long-horizon manipulation under visually grounded observations and serve as a complementary test of whether joint modeling improves or maintains performance beyond the controlled 2D benchmark. For each task, we define a predicate set aligned with task structure and implement predicate oracles inside the simulator that compute ground-truth predicate values from simulator state. These predicate labels are used for both training and evaluation, enabling direct comparisons of predicate classification accuracy across methods. We use the base-difficulty (`d0`) variants, converted to absolute action space, with 200 demonstrations per task: `kitchen_d0` and `coffee_preparation_d0`.

Real-world demonstration We include a real-world on a cube packing task. A tabletop robot must pick and place four colored cubes from fixed start regions into a container. We collect *monolithic*, long-horizon demonstrations that complete the full task (packing all four cubes), then use our segmentation and labeling toolkit to decompose each episode into skill-centric subdemonstrations corresponding to per-cube packing behaviors (e.g., `pack_red`, `pack_green`, etc.). We train a separate PACTS model for each subskill and compose them with an off-the-shelf PDDL planner. This enables recompositions beyond the original demonstrations, such as executing individual packing skills in isolation or packing only a specified subset of cubes (e.g., "pack red and yellow"), providing a concrete test of zero-shot compositionality in the real world.

B. Baselines and ablations

We design strong baselines to isolate the benefit of modeling the joint action–outcome distribution $p_\theta(\mathbf{x}, \mathbf{z} \mid \mathbf{o}_t)$ and sampling paired action–predicate rollouts in PACTS. We begin with baselines that model each conditional in isolation (action-only and predicate-only). We then include a shared-encoder multi-task baseline that predicts actions and predicates with separate heads trained end-to-end, which isolates the effect of *joint sampling* from standard multi-task representation sharing.

a) Action-only generative policy (Fig. 2(c)).: A diffusion/flow policy that models only $p(\mathbf{x} \mid \mathbf{o}_t)$ by generating an action trajectory $\mathbf{x}$ without predicate channels. This represents the standard action-only generative-policy baseline and tests whether adding outcome modeling degrades (or improves) action quality.

b) Standalone predicate predictor (Fig. 2(d)).: A predicate-only model that predicts predicate beliefs from observations, modeling $p(\mathbf{z} \mid \mathbf{o}_t)$. This quantifies predicate prediction performance in isolation and mirrors common pipelines where predicates are treated purely as a discriminative output.

c) Multi-task baseline (Fig. 2(b)).: To disentangle joint modeling from multi-task learning, we include a baseline with a shared observation encoder and two output heads trained end-to-end with a combined loss. This corresponds to learning a factorized conditional model

$$p(\mathbf{x}, \mathbf{z} \mid \mathbf{o}_t) \approx p(\mathbf{x} \mid \mathbf{o}_t)\, p(\mathbf{z} \mid \mathbf{o}_t), \tag{4}$$

where the action head is a conditional diffusion/flow policy trained to model $p(\mathbf{x} \mid \mathbf{o}_t)$ (using the same DDPM/CFM objectives as Sec. III-B–III-C), and the predicate head is a discriminative predictor trained with binary cross-entropy on ground-truth predicates. The combined objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{action}} + \mathcal{L}_{\text{predicate}}. \tag{5}$$

Although the encoder is shared, the two heads produce outputs independently at inference, so this baseline cannot encourage action–predicate consistency at sampling time.

Objective and backbone sweeps. All methods (PACTS and baselines) are evaluated across both training objectives (DDPM and CFM) and both backbone choices (UNet and Transformer), so differences can be attributed to joint modeling rather than a particular generative objective or architecture.

C. Metrics

We report the following metrics.

a) Policy evaluation: We evaluate methods by rolling out policies in the environment and reporting task performance. In PushBarrier we report *Max Reward* (↑) and *Steps to Max Reward* (↓). In RoboMimic, we follow the benchmark protocol and report rollout success/reward statistics. Results are mean $\pm$ standard error over seven (7) seeded rollouts.

b) Predicate classification evaluation: We compare predicted predicates to ground-truth oracle labels and report macro *F1*, *recall*, *precision*, and *accuracy*. For belief outputs $\mathbf{z} \in [0,1]^{H \times J}$, we threshold at 0.5 to obtain $\hat{\mathbf{p}} \in \{0,1\}^J$. Predicate metrics are *macro-averaged across predicates* so that each predicate contributes equally, preventing frequent predicates from dominating the score.

TABLE I: **Single-policy rollout evaluation on PushBarrier.** We compare PACTS to action-only, predicate-only, and shared-encoder multi-task baselines when learning a *single* policy for the full task. This isolates whether joint action–outcome modeling degrades action performance and whether it improves predicate prediction in the absence of any skill segmentation or planning. Policy metrics are mean±standard error over seven (7) seeded rollouts

Setting	**Approach**	Policy Evaluation	Predicate Classification Evaluation				
(Objective + Backbone)		Max Reward ↑	F1 ↑	Recall ↑	Precision ↑	Accuracy ↑	Observations
DDPM + UNet	Action-Only Generator + Predicate-Only Predictor	0.83 ± 0.09	0.9476	0.9547	0.9471	79.40 %	2403
	Multi-task Baseline (Shared Encoder)	0.77 ± 0.13	0.9764	0.9712	0.9834	89.39 %	2705
	Predicate Action Skill (PACTS)	**1.00 ± 0.00**	**0.9885**	**0.9924**	**0.9849**	**93.78** %	2234
DDPM + Transformer	Action-Only Generator + Predicate-Only Predictor	0.45 ± 0.10	0.9364	0.9355	0.9561	84.18 %	2800
	Multi-task Baseline (Shared Encoder)	0.33 ± 0.08	0.8684	0.8705	0.8679	85.43 %	2800
	Predicate Action Skill (PACTS)	**0.58 ± 0.15**	**0.9872**	**0.9925**	**0.9822**	**91.98** %	2606
CFM + UNet	Action-Only Generator + Predicate-Only Predictor	0.31 ± 0.11	0.8297	0.7985	0.8775	82.11 %	2800
	Multi-task Baseline (Shared Encoder)	0.45 ± 0.11	0.9551	0.9595	0.9559	84.88 %	2784
	Predicate Action Skill (PACTS)	**0.56 ± 0.12**	**0.9819**	**0.9945**	**0.9712**	**92.18** %	2800
CFM + Transformer	Action-Only Generator + Predicate-Only Predictor	0.62 ± 0.12	0.8969	0.9194	0.8869	75.15 %	2612
	Multi-task Baseline (Shared Encoder)	0.45 ± 0.12	0.8914	0.9387	0.8573	81.43 %	2800
	Predicate Action Skill (PACTS)	**0.72 ± 0.12**	**0.9813**	**0.9932**	**0.9706**	**92.21** %	2759

TABLE II: **Single-policy rollout evaluation on RoboMimic (Kitchen_d0 + Coffee_d0).** We evaluate PACTS and baselines on two RoboMimic manipulation tasks under realistic visual and geometric complexity, training a *single* policy per task (no planning or composition). Results are averaged across Kitchen and Coffee Preparation, weighting each environment equally. Policy metrics report mean±SE over seeded rollouts, with standard errors propagated across environments. Predicate classification metrics are macro-averaged across predicates within each environment, then averaged across environments.

Setting	**Approach**	Policy	Predicate Classification				
(Objective + Backbone)		Max Reward ↑	F1 ↑	Recall ↑	Precision ↑	Accuracy ↑	Observations
DDPM + UNet	Action-Only Generator + Predicate-Only Predictor	0.93 ± 0.02	0.9194	0.9095	0.9479	83.98 %	10476
	Predicate Action Skill (PACTS)	**0.96 ± 0.02**	**0.9340**	**0.9271**	**0.9537**	**90.74** %	10246
DDPM + Transformer	**Action-Only Generator + Predicate-Only Predictor**	**0.97 ± 0.02**	**0.9451**	**0.9360**	**0.9643**	**88.36** %	10087
	Predicate Action Skill (PACTS)	0.93 ± 0.04	0.9185	0.9050	0.9531	88.03 %	10103
CFM + UNet	**Action-Only Generator + Predicate-Only Predictor**	**0.99 ± 0.01**	**0.9467**	**0.9407**	**0.9574**	**88.26** %	10243
	Predicate Action Skill (PACTS)	0.91 ± 0.03	0.8884	0.8824	0.9296	82.47 %	10543
CFM + Transformer	Action-Only Generator + Predicate-Only Predictor	**0.97 ± 0.02**	0.9324	0.9206	**0.9612**	86.07 %	9927
	Predicate Action Skill (PACTS)	0.89 ± 0.03	**0.9386**	**0.9287**	0.9587	**89.78** %	10369

c) Predicate–action coherence: To test whether predicted outcomes match the outcomes induced by executed actions, we compute horizon-indexed coherence curves over open-loop action chunks. At each prediction cycle τ, the policy predicts $\hat{\mathbf{z}}_{\tau+1:\tau+H}$ and executes its generated action chunk; we then compare $\hat{\mathbf{z}}_{\tau+h}$ to oracle predicates $\mathbf{p}_{\tau+h}$ for $h \in \{1, \ldots, H\}$. We report CROSSCONS(h) (thresholded agreement, ↑), CROSSBRIER(h) (MSE on beliefs, ↓), and CROSSNLL(h) (BCE on beliefs, ↓), along with majority/base-rate reference baselines.

VI. RESULTS AND DISCUSSION

Single-policy analysis. We first study PACTS in the simplest setting: training a *single* long-horizon policy per task. This isolates the impact of jointly generating action–outcome rollouts on policy performance and predicate prediction independent of planning, or composition. Crucially, our goal is not to establish PACTS as the best monolithic policy learner, but to test whether we can *add a planner-aligned outcome interface while preserving closed-loop control* and obtain an outcome trace that is (i) coherent under execution and (ii) suitable for downstream composition.

a) PushBarrier (2D compositional benchmark).: Table I reports single-policy performance and predicate classification on PushBarrier across objectives and backbones. Across all settings, PACTS achieves strong task reward while consistently improving predicate prediction and policy performance relative to baselines. Notably, the shared-encoder multi-task baseline often attains strong predicate scores, but its factorized heads yield weaker control; in contrast, PACTS samples *paired* action–predicate rollouts from $p_\theta(\mathbf{x}, \mathbf{z} \mid \mathbf{o}_t)$ and remains competitive (often best) on reward. This supports our primary modeling claim: adding outcome channels does *not* require trading off control quality for symbolic prediction. Instead, learning the coupled object $p_\theta(\mathbf{x}, \mathbf{z} \mid \mathbf{o}_t)$ provides a useful inductive bias, encouraging the policy to represent *which* action modes correspond to *which* symbolic outcomes rather than learning actions and predicates as loosely related outputs.

Objective/backbone choices shift absolute numbers but not the conclusion: joint modeling remains strong across DDPM/CFM and UNet/Transformer. Consistent with prior work, UNet-style backbones are a reliable default, whereas Transformer variants can be more tuning-sensitive [6]; in our evaluation, all models are trained for the same number of epochs without per-setting hyperparameter sweeps. We therefore view DDPM vs. CFM and UNet vs. Transformer primarily as implementation trade-offs (e.g., sampling speed and tuning sensitivity), with the primary differentiator being the *joint* modeling target.

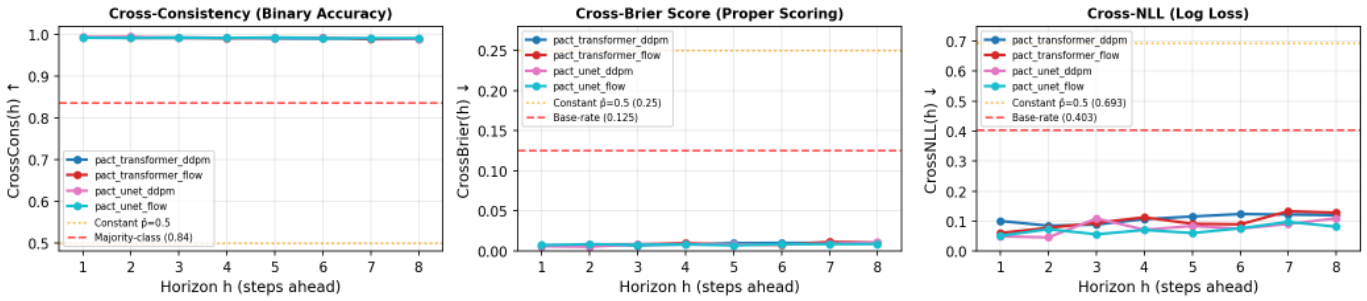

Fig. 4: **Predicate–action coherence of predicted belief rollouts.** We measure how well PACTS's predicted predicate-belief rollout matches the predicate outcomes realized after executing its open-loop action chunks. The x-axis reports lookahead h steps into the chunk. We plot (left) CROSSCONS(h) (thresholded agreement; higher is better), (middle) CROSSBRIER(h) (Brier score; lower is better), and (right) CROSSNLL(h) (log loss; lower is better). Horizontal lines are dataset-statistics baselines: constant $\hat{p}$=0.5 (dotted) and a base-rate/majority predictor from label frequencies (dashed). Since these baselines ignore both observations and planned actions, beating them indicates the beliefs reflect action-conditioned outcomes rather than priors. Across objectives and backbones, PACTS maintains high agreement and low losses over the horizon.

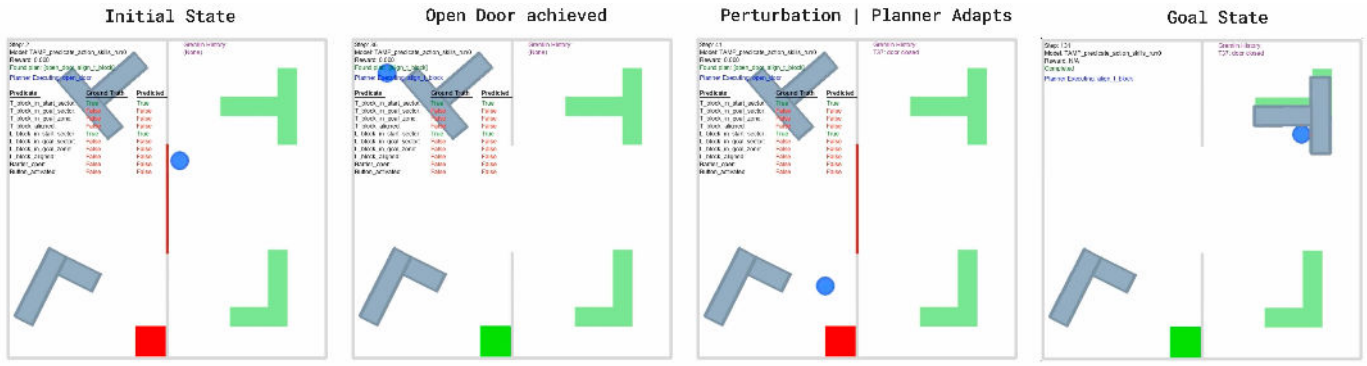

Fig. 5: **Skill recomposition and adversarial recovery in PushBarrier.** Left to right: starting from the initial state, the planner synthesizes a sequence for a *novel* subgoal (`open_door` → `align_t_block`). After `open_door` succeeds, an adversarial perturbation (*Gremlin*) closes the door, which invalidates the current plan while aligning the T block. Using online predicate-belief estimates to check effects, the system triggers replanning, re-invokes `open_door`, and then resumes `align_t_block` to reach the goal.

b) RoboMimic (Kitchen_d0 + Coffee_d0).: Table II reports single-policy evaluation averaged across both RoboMimic tasks. The action-only baseline remains strong on return across several objective/backbone choices, underscoring that explicit outcome modeling is not strictly necessary to achieve high reward when training end-to-end on a fixed task distribution. Our objective, however, is planning-based composition, which requires a *planner-aligned outcome interface* that stays coupled to the policy's sampled actions and remains meaningful under execution—without sacrificing closed-loop control in realistic 3D manipulation. In this setting, PACTS remains competitive in return and frequently improves predicate quality, with the strongest overall trade-off in the DDPM+UNet configuration.

c) Predicate–action coherence.: An outcome interface is only useful for composition if it remains *coupled* to what the policy actually does. Per-step predicate metrics alone (Tables I–II) do not test whether predicted outcomes match what the model's *own* generated actions induce. Figure 4 therefore evaluates predicate–action coherence: whether predicted belief rollouts agree with oracle predicate outcomes *under the model's executed actions* across lookahead horizons. We report CROSSCONS(h) and two proper scoring rules, CROSSBRIER(h) and CROSSNLL(h), alongside constant and base-rate/majority reference baselines computed from dataset label statistics (i.e., priors). Across objectives and backbones, PACTS achieves near-perfect CROSSCONS and substantially lower CROSSBRIER/CROSSNLL than these reference baselines across the full horizon, indicating that predicted belief rollouts track outcomes realized under execution rather than reflecting label priors. This provides empirical support for our central claim that $\mathbf{z}$ functions as an *action-dependent outcome trace*, rather than an auxiliary prediction.

Planning-based composition and recovery. We demonstrate planning-based recomposition of learned skills and replanning-based recovery under perturbations using the predicate-belief interface. Figure 5 shows a PushBarrier run with two key properties. First, the goal is *novel*: it specifies only a subgoal of the full demonstrated behavior (e.g., achieve `align_t_block` without completing the entire monolithic task). Second, execution is perturbed adversarially: the door is closed mid-execution during `align_t_block`. By checking symbolic effects using updated predicate beliefs, the system detects that the current plan is no longer applicable, triggers replanning, and recomposes a corrected sequence (`open_door` → `align_t_block`) to reach the goal.

This illustrates two benefits of joint action–outcome skills: (i) *Recomposition*: skills expose planner-aligned predicates,

allowing an off-the-shelf planner to synthesize sequences for goals not demonstrated end-to-end; (ii) *Recovery*: predicate beliefs make failures legible at the symbolic level, enabling replanning-based correction without retraining.

VII. Limitations

PACTS inherits several limitations from predicate and skill learning. First, performance depends on the choice and coverage of the predicate set: incomplete or mis-specified predicates that are misaligned with task structure can make planning brittle and limit the expressivity of the outcome interface. Second, while joint modeling improves action–outcome coherence, it does not eliminate errors from perception and state aliasing—especially under occlusions and long horizons—and execution can still fail under stochasticity, partial observability, or distribution shift. Finally, our composition results assume a manually defined symbolic domain (e.g., PDDL operators and goals); automatically discovering predicates, operators, or abstractions remains an open research direction. A promising avenue is to leverage pretrained vision–language models to propose and evaluate candidate predicates directly from demonstrations [26], then distill a compact planner-compatible set for downstream planning.

VIII. Conclusion

We introduced PACTS, a class of outcome-aware robot skills that jointly model action trajectories and predicate-belief trajectories by learning the coupled distribution $p_\theta(\mathbf{x}, \mathbf{z} \mid \mathbf{o})$. Treating predicates as generated belief channels (rather than as a separate classifier output) yields *paired* action–outcome rollouts that expose a planner-aligned interface while remaining compatible with modern trajectory generators. Overall, our results suggest that the right modeling object for composable skills is not action trajectories alone, but joint action–outcome rollouts whose symbolic trace is generated together with the policy's sampled action mode. We hope this perspective motivates future work on scaling predicate vocabularies and symbolic interfaces (e.g., via foundation-model priors) and learning richer abstractions that make planning-based composition reliable over longer horizons.

References

[1] G. Konidaris, "On the necessity of abstraction," *Current Opinion in Behavioral Sciences*, vol. 29, pp. 1–7, 2019, artificial Intelligence. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352154618302080

[2] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," 2020. [Online]. Available: https://arxiv.org/abs/2010.01083

[3] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *arXiv preprint arXiv:2108.03298*, 2021.

[4] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," in *7th Annual Conference on Robot Learning*, 2023.

[5] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," 2018. [Online]. Available: https://arxiv.org/abs/1805.01954

[6] 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," 2024. [Online]. Available: https://arxiv.org/abs/2303.04137

[7] E. Chisari, N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada, "Learning robotic manipulation policies from point clouds with conditional flow matching," *Conference on Robot Learning (CoRL)*, 2024.

[8] J. Zhang, J. Huang, S. Jin, and S. Lu, "Vision-language models for vision tasks: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2304.00685

[9] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, "π_0: A vision-language-action flow model for general robot control," 2024. [Online]. Available: https://arxiv.org/abs/2410.24164

[10] D. Wang, S. Hart, D. Surovik, T. Kelestemur, H. Huang, H. Zhao, M. Yeatman, J. Wang, R. Walters, and R. Platt, "Equivariant diffusion policy," 2024. [Online]. Available: https://arxiv.org/abs/2407.01812

[11] W. Yuan, C. Paxton, K. Desingh, and D. Fox, "Sornet: Spatial object-centric representations for sequential manipulation," 2022. [Online]. Available: https://arxiv.org/abs/2109.03891

[12] K. Rana, J. Abou-Chakra, S. Garg, R. Lee, I. Reid, and N. Suenderhauf, "Affordance-centric policy learning: Sample efficient and generalisable robot policy learning using affordance-centric task frames," 2024. [Online]. Available: https://arxiv.org/abs/2410.12124

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

[14] M. Ghallab, D. Nau, and P. Traverso, *Automated planning and acting*. Cambridge University Press, 2016.

[15] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. Smith, Y. Sun, and D. Weld, "Pddl - the planning domain definition language," 08 1998.

[16] D. Harel, "Statecharts: a visual formalism for complex systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167642387900359

[17] M. Colledanchise and P. Ögren, "Behavior trees in robotics and ai," Jul. 2018. [Online]. Available: http://dx.doi.org/10.1201/9780429489105

[18] T. Migimatsu, W. Lian, J. Bohg, and S. Schaal, "Symbolic state estimation with predicates for contact-rich manipulation tasks," 2022. [Online]. Available: https://arxiv.org/abs/2203.02468

[19] M. Diab, M. Pomarlan, D. Beßler, A. Akbari, J. Rosell, J. Bateman, and M. Beetz, "Skillman — a skill-based robotic manipulation framework based on perception and reasoning," *Robotics and Autonomous Systems*, vol. 134, p. 103653, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889020304930

[20] F. Martín, J. Ginés, V. Matellán, and F. J. Rodríguez, "Plansys2: A planning system framework for ros2," 2021. [Online]. Available: https://arxiv.org/abs/2107.00376

[21] M. Mayr, F. Rovida, and V. Krueger, "Skiros2: A skill-based robot control platform for ros," 2023. [Online]. Available: https://arxiv.org/abs/2306.17030

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

[23] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio, "Improving and generalizing flow-based generative models with minibatch optimal transport," *arXiv preprint arXiv:2302.00482*, 2023.

[24] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[25] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, S. Palma, P. Kooijmans, M. Aractingi, M. Shukor, D. Aubakirova, M. Russi, F. Capuano, C. Pascal, J. Choghari, J. Moss, and T. Wolf, "Lerobot: State-of-the-art machine learning for real-world robotics in pytorch," https://github.com/huggingface/lerobot, 2024.

[26] A. Athalye, N. Kumar, T. Silver, Y. Liang, J. Wang, T. Lozano-Pérez, and L. P. Kaelbling, "From pixels to predicates: Learning symbolic world models via pretrained vlms," *IEEE Robotics and Automation Letters*, 2026.