HDFlow: Hierarchical Diffusion-Flow Planning for Long-horizon Robotic Assembly

Gireesh Nandiraju[1,2] **Yuanliang Ju**[3] **Chaoyi Xu**[2] **He Wang**[1,2,*]
[1]Peking University, [2]Galbot, [3]University of Toronto

Abstract

Long-horizon manipulation tasks represent a significant challenge in robotics, demanding both strategic, high-level reasoning and fast, precise, low-level control. While recent advances in generative models have shown promise in generating behavior plans for long-horizon tasks, they often lack a principled framework for hierarchical decomposition and struggle with the computational demands of real-time execution, due to their iterative denoising process. In this work, we introduce **Hierarchical Diffusion-Flow** (`HDFlow`), a novel hierarchical planning framework that optimally leverages the strengths of *diffusion* and *rectified flow* models. `HDFlow` employs a high-level diffusion planner to generate sequences of strategic subgoals in a learned latent space, capitalizing on diffusion's powerful exploratory capabilities. These subgoals then guide a low-level rectified flow planner that generates smooth and dense trajectories, exploiting the speed and efficiency of ordinary differential equation (ODE)-based trajectory generation. This hybrid approach synergistically combines the strengths of both models to overcome the limitations of single-paradigm generative planners, enabling robust and efficient long-horizon planning. We evaluate `HDFlow` on four challenging furniture assembly tasks, where it significantly outperforms state-of-the-art methods. Our work demonstrates that a hybrid generative planner provides a powerful solution for long-horizon robotic assembly. Project website: https://hdflow-page.github.io/

1 Introduction

Robotic manipulation for complex, long-horizon tasks such as robotic assembly [30, 56, 33, 24, 4, 5] remains a significant challenge requiring not only understanding multi-stage instructions and spatial relationships but also executing precise, contact-rich motions over extended periods. Traditional planning methods struggle with long-horizon problems because small inaccuracies in state estimation, dynamics prediction, or control execution accumulate over time, compounding into significant deviations that ultimately lead to task failure. This has motivated a shift towards hierarchical planning [48, 31, 52, 28], which decomposes a complex goal into a sequence of simpler, more manageable subgoals. A powerful approach within this paradigm is to perform planning in the latent space of a learned world model [16–18]. By forecasting future states in a compressed representation, world models allow planners to reason efficiently and abstract away from high-dimensional, noisy observations [19, 59]. However, standard world models, trained primarily on reconstruction and dynamics prediction, do not guarantee that the learned latent space is semantically structured for planning. The distance between states in this space does not correlate with progress toward a goal, making it difficult for a planner to navigate effectively.

The advent of generative models, particularly denoising diffusion models [53, 26, 55], have achieved strong results across various domains [42, 39, 35, 15, 6]. Building upon these successes, they

*Corresponding author (email: hewang@pku.edu.cn), Author (email: 2401112103@stu.pku.edu.cn)

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Embodied World Models for Decision Making.

have recently revolutionized planning in robotics [27, 2, 38]. By treating planning as a conditional generation problem, these models can produce diverse and high-quality trajectories. However, their iterative denoising process is computationally intensive [11], making them ill-suited for the fast, low-level control required for real-time robotic interaction. Applying diffusion models naively at all levels of a hierarchy [7, 34, 21] inherits this critical drawback, creating a bottleneck at the trajectory generation stage. This raises a fundamental question: ***Is a single generative modeling paradigm optimal for all levels of a planning hierarchy?***

We empirically show that the answer is no. The requirements for high-level strategic planning are fundamentally different from those of low-level trajectory generation. High-level planning demands exploration and multi-modal diversity to discover viable sequences of subgoals. In contrast, low-level planning demands speed, precision, and deterministic execution to translate a chosen subgoal into a smooth, dense trajectory.

In this paper, we introduce **H**ierarchical **D**iffusion-**F**low (**HDFlow**), a new hierarchical framework for long-horizon manipulation that is built on this core insight. `HDFlow` leverages a hybrid generative architecture that assigns the right tool to the right job. For high-level, strategic planning, we employ a *diffusion model* to generate diverse sequences of subgoals within a learned latent space of a world model. While standard conditional diffusion models can generate plans that are consistent with the goal, they lack an explicit mechanism to assess the quality or long-term viability of those plans. In complex, sparse-reward settings, many plausible-looking sequences of subgoals can lead to irreversible failures. To address this, we introduce an *energy-based model* (EBM) to provide explicit guidance. The EBM is trained to assign low energy to successful strategies and high energy to failing ones, effectively learning a dense reward signal. This energy function then steers the diffusion planner away from potential dead ends and towards high-quality solutions, which is critical for robust, long-horizon performance. Furthermore, to mitigate the failures caused by inaccurate guidance, we enhance the standard EBM guidance with a two-step *manifold-aware* process [32]. For low-level, tactical planning, we introduce a *rectified flow model* to rapidly generate dense latent trajectories to reach each subgoal. The underlying world model itself is trained with a *contrastive objective* to create a semantically structured latent space, which facilitates more effective planning by organizing representations of intermediate states from successful episodes closer to the final goal and pushing them away from failure cases, creating a smoother, more monotonic representation of task progress that is crucial for effective long-horizon planning.

Furthermore, while prior work has been limited to simple tasks with very short-horizons like Maze2D, and AntMaze [14], we evaluate our proposed method on four contact-rich assembly tasks from the FurnitureBench [24] benchmark, including tasks with very long horizons of upto $\sim$1500 timesteps, involving 11 phases, and assemblies of up to 4 parts to be precisely grasped, oriented, and inserted.

In summary, our contributions are as follows:

- We propose `HDFlow`, a novel, hybrid hierarchical planner that combines a diffusion model for high-level exploration and a rectified flow model for low-level trajectory generation.
- We introduce a two-stage training process featuring a contrastive-trained world model for structured representation learning and a manifold-aware EBM for explicit guidance of the high-level planner, enabling robust planning in sparse-reward environments.
- We demonstrate state-of-the-art performance on four challenging tasks from the FurnitureBench benchmark.

2 Related Works

2.1 World Models

Learning models of the world to enable planning has a long history in robotics and reinforcement learning [57]. The idea is to learn a compressed representation of the environment's dynamics, which allows an agent to simulate future outcomes and plan entirely in a low-dimensional latent space. This approach was popularized by [50, 49, 16], who demonstrated that a compact world model could be trained to solve classic RL control tasks. The Dreamer series of works [17, 18, 20] significantly advanced this paradigm by introducing the Recurrent State-Space Model (RSSM) and demonstrating its effectiveness for learning from pixels and planning for complex, long-horizon

tasks. More recent works have extended these ideas to real-world robotic manipulation [19, 59], showing that planning in a learned latent space can be more sample-efficient and robust than planning directly in the observation space. Our work builds directly on this foundation, using an RSSM-based world model as the backbone. However, we introduce a key modification, a *contrastive objective*, to explicitly structure the latent space for long-horizon manipulation.

2.2 Generative Models as Planners

Recent advancements in generative models, particularly denoising diffusion probabilistic models [53, 26, 55, 29], have significantly impacted planning in robotics by framing it as a conditional generation problem. Early works such as `Diffuser` [27] leverage iterative denoising to generate flexible trajectories conditioned on objectives like rewards or constraints. Building on this, Decision Diffuser (`DD`)[2] further demonstrated that return-conditional diffusion models can outperform traditional offline reinforcement learning by enabling conditioning on various factors like constraints and skills. While powerful, the iterative nature of diffusion models can be computationally intensive [11]. To tackle long-horizon tasks, hierarchical planning with diffusion models has emerged, as seen in [34, 7, 21]. Simple Hierarchical Diffuser (`SHD`)[7] employs a high-level diffusion model for sparse subgoal generation and a low-level one for dense trajectory refinement, aiming for improved efficiency and generalization. However, their reliance on diffusion models for both high-level and low-level planning increases the maintenance burden and makes them computationally expensive.

3 Preliminaries

Notation disambiguation. We use t for environment time indices, $\ell \in \{1, \dots, L\}$ for diffusion timesteps, and $u \in [0, 1]$ for the continuous flow time in rectified flow. We also denote a clean (noise-free) latent subgoal sequence by z^{clean} to avoid confusion with the initial environment state z_0.

3.1 Denoising Diffusion Models

Denoising diffusion models [53, 26] are generative models that learn a data distribution $p(\mathbf{x})$ by reversing a fixed forward noising process.

Forward Process. The forward process gradually adds Gaussian noise to a data sample $\mathbf{x}_0$ over L discrete timesteps, according to a variance schedule β_ℓ:

$$q(\mathbf{x}_\ell|\mathbf{x}_{\ell-1}) = \mathcal{N}(\mathbf{x}_\ell; \sqrt{1-\beta_\ell}\mathbf{x}_{\ell-1}, \beta_\ell \mathbf{I}) \tag{1}$$

A key property is that we can sample $\mathbf{x}_\ell$ at any timestep ℓ in closed form: $q(\mathbf{x}_\ell|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_\ell; \sqrt{\bar{\alpha}_\ell}\mathbf{x}_0, (1-\bar{\alpha}_\ell)\mathbf{I})$, where $\alpha_\ell = 1 - \beta_\ell$ and $\bar{\alpha}_\ell = \prod_{i=1}^{\ell} \alpha_i$. As $\ell \to L$, $\mathbf{x}_L$ approaches an isotropic Gaussian distribution $\mathcal{N}(0, \mathbf{I})$.

Reverse Process. The reverse process is a learned generative model that starts from noise $\mathbf{x}_L \sim \mathcal{N}(0, \mathbf{I})$ and iteratively denoises it to produce a sample. This process is modeled by a neural network $\epsilon_\theta(\mathbf{x}_\ell, \ell)$ trained to predict the noise ϵ that was added to the original sample $\mathbf{x}_0$ to produce $\mathbf{x}_\ell$. The training objective is to minimize the mean squared error between the true and predicted noise:

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{\ell, \mathbf{x}_0, \epsilon}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_\ell}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_\ell}\epsilon, \ell)\right\|^2\right] \tag{2}$$

For conditional generation (e.g., on a context c), classifier-free guidance (CFG) [25] is commonly used. The model is trained on both conditional and unconditional inputs, and the noise prediction during inference is modified to steer generation towards the context:

$$\hat{\epsilon}_\theta(\mathbf{x}_\ell, \ell, c) = \epsilon_\theta(\mathbf{x}_\ell, \ell, \emptyset) + w \cdot (\epsilon_\theta(\mathbf{x}_\ell, \ell, c) - \epsilon_\theta(\mathbf{x}_\ell, \ell, \emptyset)) \tag{3}$$

where w is the guidance scale and $\emptyset$ denotes the unconditional case. While powerful, the iterative sampling process can be computationally intensive.

3.2 Rectified Flow

Rectified Flow [36, 3, 37] is a generative modeling approach based on ordinary differential equations (ODEs) that offers a more efficient alternative to diffusion models. It learns a deterministic mapping from a simple prior distribution to a data distribution.

Let p_0 be a prior distribution (e.g., $\mathcal{N}(0, \mathbf{I})$) and p_1 be the data distribution. Rectified Flow constructs straight-line paths between pairs of samples $(\mathbf{x}_0, \mathbf{x}_1)$ drawn from these distributions. The path is defined by the linear interpolation $\mathbf{x}_u = (1-u)\mathbf{x}_0 + u\mathbf{x}_1$ for $u \in [0, 1]$. The corresponding velocity vector field is simply $\mathbf{x}_1 - \mathbf{x}_0$. The model trains a neural network $v_\theta(\mathbf{x}, u)$ to approximate this vector field by minimizing the flow-matching objective:

$$\mathcal{L}_{RF} = \mathbb{E}_{u,\mathbf{x}_0,\mathbf{x}_1}\left[\|v_\theta((1-u)\mathbf{x}_0 + u\mathbf{x}_1, u) - (\mathbf{x}_1 - \mathbf{x}_0)\|^2\right] \tag{4}$$

Once trained, generation is performed by starting with a sample from the prior, $\mathbf{x}_0 \sim p_0$, and solving the initial value problem for the learned ODE from $u = 0$ to $u = 1$:

$$\frac{d\mathbf{x}_u}{du} = v_\theta(\mathbf{x}_u, u) \tag{5}$$

This is done using a numerical ODE solver. Because it follows a deterministic, straight-line path, Rectified Flow can often generate high-quality samples in significantly fewer function evaluations than required by iterative diffusion models, making it ideal for applications requiring fast synthesis, such as real-time trajectory generation.

4 Method

In this section, we introduce `HDFlow`, a hierarchical planning framework that tackles long-horizon manipulation tasks using a two-stage training process. First, we train a world model with a contrastive objective to learn a semantically structured latent space that embeds a notion of progress. Second, we train a hierarchical planner on top of the frozen, pre-computed latent representations from this world model. This planner consists of a high-level diffusion model for strategic subgoal generation and a low-level rectified flow model for efficient trajectory synthesis.

4.1 Stage 1: World Model Learning

Our framework operates within the latent space of a world model, which is trained to model the environment's dynamics from multi-modal, high-dimensional observations (denoted as o). We adopt a Recurrent State Space Model (RSSM) architecture [20] with an encoder that leverages a pretrained DINOv2 model [44]. The RSSM learns to encode observations into a latent space $\mathcal{Z}$, and is trained to accurately reconstruct observations and predict future states. For a detailed explanation of the architecture and training objective, please see Appendix B.

$$\mathcal{L}_{\text{WM}} = \mathbb{E}_{q_\phi(z_{1:T}|o_{1:T})}\left[\sum_{t=1}^{T}\left(\log p_\phi(\hat{o}_t|z_t, h_t) - D_{KL}[q_\phi(z_t|h_t, e_t)||p_\phi(\hat{z}_t|h_t)]\right)\right] \tag{6}$$

While the standard world model objective, $\mathcal{L}_{\text{WM}}$, encourages predictable dynamics, it does not explicitly structure the latent space to reflect a notion of progress towards a goal, making long-horizon planning challenging. To address this, we introduce a contrastive learning objective designed to provide a dense learning signal and organize the latent space for more effective downstream planning. The goal is to pull representations of intermediate latent states from successful trajectories closer to their final goal representation, while pushing them away from intermediate latent states from failed trajectories.

Let $f(\cdot)$ be a projection head that maps latent states z to a new embedding space. For a given intermediate latent state z_k from a successful trajectory with final goal z_G, we form a positive pair $(f(z_k), f(z_G))$. Negative pairs are formed with intermediate latent states from failed trajectories. The contrastive loss is then given by the modified InfoNCE objective [43] as suggested by [51, 54]:

$$\mathcal{L}_{\text{contrastive}} = -\mathbb{E}\left[\log \frac{\exp(\text{sim}(f(z_k), f(z_G))/\tau)}{\sum_{j=1}^{N}\exp(\text{sim}(f(z_k), f(z_j))/\tau)}\right] \tag{7}$$

where $\text{sim}(\cdot,\cdot)$ is the cosine similarity and τ is a temperature hyperparameter [58].

To further encourage the latent space to be informative for control, we also include an inverse dynamics model [1, 45]. This model, $a_t \sim p_\phi(a_t|z_t, z_{t+1})$, is trained to predict the action that was taken to transition between two consecutive latent states. It is trained with a mean squared error loss:

$$\mathcal{L}_{IDM} = \mathbb{E}\left[\|a_t - \text{MLP}(z_t, z_{t+1})\|^2\right] \tag{8}$$

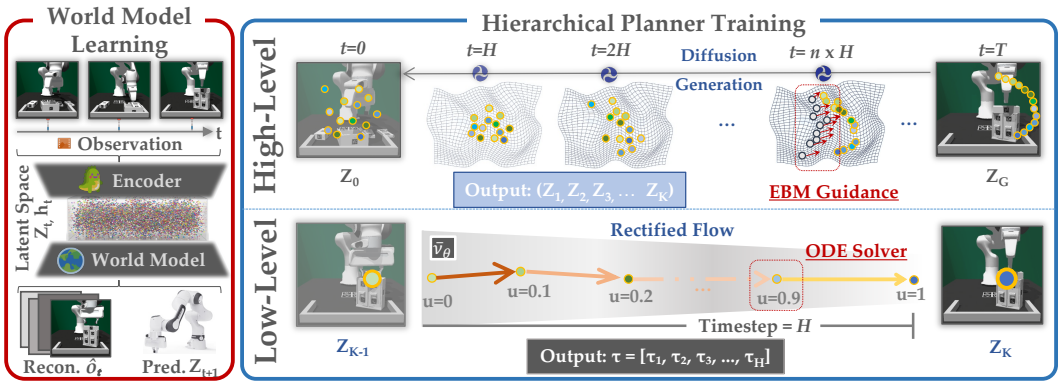

Figure 1: `HDFlow` **pipeline.** The framework consists of two main stages: **World Model Learning** (left), where observations are encoded into a structured latent space, and **Hierarchical Planner Training** (right). The latter involves a *High-Level* diffusion planner generating sparse strategic subgoals $(z_1, \ldots, z_K)$ with EBM guidance, and a *Low-Level* rectified flow planner synthesizing dense trajectories $\tau = [\tau_1, \ldots, \tau_H]$ between subgoals using an ODE solver.

This objective ensures that the latent states encode action-relevant information, which is beneficial for the downstream planner. The full training objective combines the world model loss, the inverse dynamics loss, and the contrastive loss:

$$\mathcal{L}_{\text{WM-total}} = \lambda_{WM}\mathcal{L}_{WM} + \lambda_{IDM}\mathcal{L}_{IDM} + \lambda_{\text{contrastive}}\mathcal{L}_{\text{contrastive}} \tag{9}$$

After this stage, the world model's weights are frozen, and its encoder is used to generate a static dataset of structured latent representations for the next stage.

4.2 Stage 2: Hierarchical Planner Training

With the structured latent space from Stage 4.1 fixed, we frame the long-horizon planning problem as a conditional generative modeling task. We decompose the problem by defining a temporal abstraction, a common and effective strategy in hierarchical planning [34, 7]. For each full-length trajectory in our latent dataset, we define subgoals by subsampling the trajectory at a fixed interval of H timesteps. For a trajectory of length T (environment timesteps), this yields $K = \lfloor T/H \rfloor$ subgoals, which can vary across trajectories. This process creates two distinct datasets for our planners:

- For the **high-level planner**, we create sparse sequences of K latent subgoals, $z = (z_1, \ldots, z_K)$, where each subgoal z_k corresponds to the latent state at timestep $k \cdot H$.
- For the **low-level planner**, we create a dataset of dense, fixed-length trajectory segments. Each segment τ_k contains the H latent states and actions between consecutive subgoals z_{k-1} and z_k.

This decomposition allows us to train a high-level planner π_{HL} to generate a sequence of latent subgoals, and a low-level planner π_{LL} to generate the dense trajectory segment to reach each subgoal.

4.2.1 High-Level Planner: Manifold-Aware EBM-Guided Diffusion

The high-level planner is a conditional diffusion model, trained to generate a sequence of K latent subgoals, $z = (z_1, \ldots, z_K)$, conditioned on the context $c = (z_0, z_G)$, where z_0 is the current latent state and z_G is the goal latent state. It is trained by minimizing the standard noise prediction error from Eq. 2:

$$\mathcal{L}_{HL} = \mathbb{E}_{\ell, z^{\text{clean}}, \epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_\ell}\, z^{\text{clean}} + \sqrt{1-\bar{\alpha}_\ell}\,\epsilon, \ell, c\right)\right\|^2\right] \tag{10}$$

While this provides a strong prior for generating plausible plans, it does not guarantee that all generated plans will be successful, especially in long-horizon scenarios where small errors can compound. To address this and actively steer the planner towards high-quality solutions, we introduce an Energy-Based Model (EBM) for explicit guidance at inference time. The EBM, $E_\phi(z|z_0, z_G)$, is a separate network trained to predict a low energy for high-quality latent subgoal sequences and a high

energy for poor ones. It is trained with a contrastive loss that pushes down the energy of plans from successful trajectories (z_{pos}) and pushes up the energy of plans from failed trajectories (z_{neg}):

$$\mathcal{L}_{EBM} = \log(1 + \exp(E_\phi(z_{\text{pos}}) - E_\phi(z_{\text{neg}}))) \tag{11}$$

However, in high-dimensional latent spaces, inexact guidance can cause *manifold deviation* [23], a phenomenon where guided samples drift away from the feasible latent subgoal manifold $\mathcal{M}_t$. We formalize this issue through the guidance gap:

Definition 1 *Let $\nabla_{z_t} E_{true}(z_t|c)$ denote the true optimal energy guidance and $\nabla_{z_t} E_\phi(z_t|c)$ be our learned EBM guidance. The guidance gap at z_t is:*

$$\Delta_{EBM}(z_t) = \|\nabla_{z_t} E_{true}(z_t|c) - \nabla_{z_t} E_\phi(z_t|c)\|_2 \tag{12}$$

Proposition 4.1 *(Proof in Appendix) The EBM guidance gap $\Delta_{EBM}(z_t)$ has a lower bound scaling as $\frac{c}{\sqrt{1-\bar{\alpha}_t}}\sqrt{d}$ in high-dimensional latent spaces, where $c > 0$ is a constant independent of dimensionality d.*

Definition 1 and Proposition 4.1 show that inaccuracies in energy guidance grow with scenarios involving long planning horizons and high-dimensional latent spaces, leading sampled trajectories to deviate away. Check Appendix D.3 for a detailed proof.

Manifold-Aware Guidance. High-dimensional latent representations often exhibit intrinsic low-dimensional structure. Under our contrastive training, successful latent subgoal sequences concentrate on a k-dimensional submanifold $\mathcal{M}_0 \subset \mathbb{R}^d$ with $k \ll d$. To mitigate this manifold deviation, we enhance the standard EBM guidance with a two-step manifold-aware process as suggested in [32]. Rather than applying guidance directly to the noise prediction, we perform:

Step 1: Guided Sampling.

$$z_{\ell-1}^{\text{temp}} \sim \mathcal{N}\big(\mu_\theta(z_\ell) + w_{ebm}\,\Sigma^\ell\, g, \Sigma^\ell\big) \tag{13}$$

where $g = \nabla_{z_\ell} E_\phi(z_\ell|c)$ is the EBM guidance and μ_θ, Σ^ℓ are the mean and covariance of the reverse diffusion transition (scheduler-dependent, e.g., DDPM variance-preserving).

Step 2: Manifold Projection.

$$z_{\ell-1} = \mathcal{P}_{\mathcal{T}_{z_{\ell-1}}\mathcal{M}_{\ell-1}}\big(z_{\ell-1}^{\text{temp}}\big) \tag{14}$$

where $\mathcal{P}_{\mathcal{T}_{z_{\ell-1}}\mathcal{M}_{\ell-1}}$ projects onto the local tangent space of the latent manifold $\mathcal{M}_{\ell-1}$.

The manifold projection is computed using local low-rank approximation: we first obtain a denoised estimate using Tweedie's formula [47, 9, 10]:

$$\hat{z}^{0|\ell-1} = \frac{1}{\sqrt{\bar{\alpha}_{\ell-1}}}\Big(z_{\ell-1}^{\text{temp}} - \sqrt{1-\bar{\alpha}_{\ell-1}}\,\epsilon_\theta(z_{\ell-1}^{\text{temp}}, \ell-1, c)\Big) \tag{15}$$

We then retrieve k nearest neighbors from successful latent subgoal sequences using cosine similarity [13], forward diffuse them to timestep $\ell - 1$, and perform rank-r PCA to obtain the projection basis $\boldsymbol{U} \in \mathbb{R}^{d\times r}$. Let $\boldsymbol{\mu}$ be the local mean of these neighbors. The mean-centered projection is

$$\mathcal{P}(\mathbf{z}) = \boldsymbol{\mu} + \boldsymbol{U}\boldsymbol{U}^T(\mathbf{z} - \boldsymbol{\mu}).$$

Proposition 4.2 *Given a base diffusion planner ϵ_θ trained for classifier-free guidance, a learned energy function $E_\phi(z|c)$, and a manifold projection operator $\mathcal{P}_\mathcal{M}$, the manifold-aware guided planner, which combines EBM guidance with manifold projection, corresponds to sampling from a posterior distribution $p(z|y = 1, z \in \mathcal{M}, c)$ that maximizes the likelihood of generating a successful and feasible goal-conditioned plan.*

4.2.2 Low-Level Planner: Rectified Flow for Trajectory Generation

The low-level planner's role is to generate a dense, short-horizon latent trajectory τ_z to a given subgoal z_k. We can frame this subproblem through the lens of optimal transport, which seeks the most efficient way to transform one probability distribution into another. Here, the task is to find the optimal mapping from the distribution of initial states around z_{k-1} to the distribution of target

states around z_k. The *cost* of transport is minimized by trajectories that are as straight as possible in the latent space. We use a conditional rectified flow model, $v_\theta(\tau_u, u, c_k)$, for its speed. It is trained to generate a trajectory segment τ conditioned on the context $c_k = (z_{k-1}, z_k)$ by minimizing the standard flow-matching objective from Eq. 4:

$$\mathcal{L}_{LL} = \mathbb{E}_{u,\tau_0,\tau_1}\left[\left\|v_\theta\big((1-u)\tau_0 + u\tau_1, u, c_k\big) - (\tau_1 - \tau_0)\right\|^2\right] \tag{16}$$

Construction of training pairs. For each consecutive latent subgoal pair (z_{k-1}, z_k), we extract the dense H-step latent segment from successful demonstrations. We set τ_1 to be the observed segment that ends at z_k and τ_0 to be the segment that starts at z_{k-1} from the same demonstration. We optionally apply small Gaussian perturbations in latent space and mild time-warping for robustness; see Appendix B for details.

4.2.3 Planner Training Objective

The components of the hierarchical planner are trained jointly with a composite loss function that combines the objectives for the high-level planner, the low-level planner, the EBM, and manifold consistency:

$$\mathcal{L}_{\text{planner}} = \lambda_{HL}\mathcal{L}_{HL} + \lambda_{LL}\mathcal{L}_{LL} + \lambda_{\text{EBM}}\mathcal{L}_{\text{EBM}} + \lambda_{\text{proj}}\mathcal{L}_{\text{projection}} \tag{17}$$

where the λ terms are hyperparameters and $\mathcal{L}_{\text{projection}}$ encourages the generated latent subgoals to remain close to the learned latent manifold:

$$\mathcal{L}_{\text{projection}} = \mathbb{E}_{z\sim\pi_{HL}}\left[\|z - \mathcal{P}_{\mathcal{M}}(z)\|^2\right] \tag{18}$$

This multi-faceted loss function enables HDFlow to learn to identify successful plans (via the EBM), generate them effectively (via the guided planners), and maintain feasibility (via manifold projection) within the well-structured latent space provided by the contrastive world model.

4.2.4 Inference and Online Deployment

During online deployment in an MPC framework, the high-level planner is invoked iteratively. At each replanning step, it takes the robot's *current latent state* (which updates as the low-level planner executes segments) and the *goal state* as input to generate a new sequence of subgoals for the remaining portion of the task. The low-level planner takes the first subgoal from this sequence, and generates a dense latent trajectory. Then consecutive latent pairs (z_t, z_{t+1}) along the generated latent trajectory are mapped to control actions via the inverse dynamics model $p_\phi(a_t \mid z_t, z_{t+1})$ introduced in Stage 4.1. Actions are executed for H steps within an MPC loop before replanning with the updated state.

5 Experiments

To evaluate the efficacy of our proposed framework, we design a set of experiments to answer the following key questions: (**1**) Does `HDFlow` achieve state-of-the-art **performance** on complex, long-horizon robotic assembly tasks compared to existing methods? (**2**) How does our **hybrid** architecture compare against non-hybrid hierarchical planner as well as single planner approaches? (**3**) What are the contributions of the **core components** of `HDFlow`? We will conduct ablation studies to analyze the impact of our hierarchical structure and planner choices. (**4**) Does `HDFlow` offer superior **computational efficiency** during inference, a critical factor for real-time robotic control?

5.1 Simulation Experiments

Tasks and Environment. We evaluate our method on FurnitureBench [24], a challenging benchmark for long-horizon, contact-rich robotic assembly. We chose 4 tasks from the benchmark: `one_leg`, `lamp`, `round_table`, and `cabinet` (see Figure 2 for start and goal positions), with the default initial randomization protocol: `Low`, `Med`, and `High`. We define task success as assembling all the furniture parts in their goal poses. We report the success rate calculated over 100 episodes for each task. Further details about the tasks and dataset collection are provided in Appendix A

Implementation Details. We adopt Diffusion Transformer (DiT) [46] as the backbone for the high-level diffusion planner and Rectified Flow Transformer [12] for the low-level rectified flow

	one_leg			lamp			round_table			cabinet	
Method	Low	Med	High	Low	Med	High	Low	Med	High	Low	Med
BC	0	0	0	0	0	0	0	0	0	0	0
DP	51	19	3	18	7	1	6	2	0	4	1
JUICER	68	22	3	27	12	2	23	8	2	11	5
Diffuser	56	22	4	22	9	1	21	7	1	6	2
DD	60	22	4	24	11	1	22	8	2	9	3
HDMI	66	26	11	37	16	11	33	15	9	17	8
SHD	71	31	15	43	22	16	41	21	12	21	11
Ours	**92**	**71**	**39**	**68**	**49**	**34**	**61**	**43**	**27**	**55**	**36**

Table 1: Main results on FurnitureBench tasks in simulation. Success rates (%) are reported for different initial randomization levels (Low, Med, High).

Method	one_leg	lamp
FD	60	24
HF	63	24
HD	71	43
Ours	**92**	**68**

Table 2: Ablation study on the choice of generative models for the high-level and low-level planners. Success rates (%) are reported for the one_leg and lamp tasks under Low randomization.

Method	one_leg	lamp
Ours	**92**	**68**
w/o Manifold Projection	84	57
w/o Manifold-aware EBM	61	33
w/o Contrastive WM	58	27

Table 3: Ablation study on the core components of HDFlow. Success rates (%) are reported for the one_leg and lamp tasks under Low randomization.

planner. Detailed descriptions of our experimental setup, including model architectures, training procedures, and hyperparameter settings, are provided in Appendix B

Baselines. We compare HDFlow against a comprehensive set of state-of-the-art methods that cover different paradigms for long-horizon manipulation. We provide more details about each baseline in the Appendix C.

- **Imitation Learning (IL) Baselines**: These methods learn directly from successful demonstrations without the use of explicit reward signals.(a) Vanilla behaviour cloning (BC) (b) Diffusion Policy (DP) [8] (c) JUICER [4]
- **Diffusion-based Planners**: These methods use diffusion-based planning methods from a mixed dataset of successful and failure demonstrations. (a) Diffuser [27]: A diffusion probabilistic model that plans by iteratively denoising trajectories. It treats planning as a conditional generation problem and can produce diverse and high-quality trajectories. (b) Decision Diffuser (DD) [2] (c) HDMI [34] (d) Simple Hierarchical Diffuser (SHD) [7]

Analysis. The results presented in Table 1 demonstrate HDFlow's superior performance across all four challenging furniture assembly tasks and varying levels of initial randomization. Our hybrid, hierarchical planning framework consistently achieves the highest success rates, significantly outperforming imitation learning, non-hierarchical, and other hierarchical diffusion planners. This performance gap widens with task complexity and randomization, particularly on the cabinet task, where HDFlow achieved a 36% success rate compared to less than 11% for other methods. This success stems from its ability to generate diverse, high-quality subgoals and execute them with fast, precise trajectories, offering a robust and efficient solution for complex, long-horizon robotic manipulation.

5.2 Ablation Studies

To systematically analyze the contribution of each component within HDFlow, we conduct extensive ablation studies. In this section, we consider one_leg and lamp tasks under Low randomization. For more ablation experiments, please see Appendix E

Choice of Generative Model. We compare HDFlow against variants that use alternative generative models. Specifically, we consider FD (flat diffusion), HF (hierarchical flow), and HD (hierarchical diffusion). Table 2 shows that our planner significantly outperforms single-paradigm or flat planning approaches.

Method	SR	Inference Time
FD	24	197
HF	24	53
HD	43	142
Ours	**68**	**88**

Table 4: Ablation study on computational efficiency for the lamp task (Low randomization). This table presents both success rates (SR, %) and average inference time (milliseconds per planning step).

Comparison of Wall-clock times. To assess the computational efficiency of HDFlow and its variants, we conducted an ablation study measuring the average inference time per planning step. Table 4 presents these results alongside the success rates for the lamp task (Low randomization), directly building upon the analysis from Table 2.

HDFlow achieves an optimal balance, combining robust exploration from its high-level diffusion planner with efficient, fast trajectory synthesis from its low-level rectified flow planner, making it suitable for real-time robotic deployment without compromising planning quality.

Table 4 reveals the trade-off between planning performance and computational efficiency. FD has the highest inference time due to its iterative denoising. HD improves by decomposing the problem but still incurs significant cost. HF is fastest but sacrifices success rates, indicating limitations for high-level strategic planning.

Contribution of Core Components. We investigate the impact of the contrastive world model, EBM guidance, and manifold projection by progressively removing them from HDFlow. Table 3 demonstrates that each novel component measurably contributes to HDFlow's overall success, highlighting the importance of our multi-faceted approach to long-horizon robotic assembly.

6 Conclusion

In this work, we introduced HDFlow, a novel hierarchical planning framework that effectively addresses long-horizon, contact-rich robotic assembly tasks by synergistically combining the strengths of diffusion and rectified flow models. Our approach leverages a contrastively-trained world model to learn a semantically structured latent space, a manifold-aware EBM-guided high-level diffusion planner for strategic subgoal generation, and a fast low-level rectified flow planner for efficient trajectory synthesis. We demonstrated state-of-the-art performance on challenging FurnitureBench tasks, showcasing the robustness and efficiency of HDFlow.

Limitations. Despite its significant performance, HDFlow has several limitations that suggest avenues for future research. Currently, our framework relies on a dataset of successful and failed demonstrations for training the world model and the EBM. While effective, collecting such data can be resource-intensive. Another area for improvement lies in the computational efficiency of the high-level diffusion planner. Although rectified flow handles low-level trajectory generation efficiently, the iterative nature of diffusion can still pose a bottleneck for tasks that require rapid online replanning. Furthermore, the fixed subgoal interval (H) in our current hierarchical decomposition might not be optimal for all tasks; adaptive subgoal generation that dynamically adjusts H based on task complexity or progress could lead to more flexible and efficient planning. Lastly, while HDFlow excels at furniture assembly, exploring its generalization capabilities to a broader range of complex, open-ended manipulation tasks would be a valuable direction.

References

[1] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

[2] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023.

[3] Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=li7qeBbCR1t`.

[4] Lars Ankile, Anthony Simeonov, Idan Shenfeld, and Pulkit Agrawal. Juicer: Data-efficient imitation learning for robotic assembly. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5096–5103. IEEE, 2024.

[5] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement–residual rl for precise assembly. *arXiv preprint arXiv:2407.16677*, 2024.

[6] Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In *International Conference on Machine Learning*, pages 1276–1301. PMLR, 2023.

[7] Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion. In *The Twelfth International Conference on Learning Representations*, 2024.

[8] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[9] Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022.

[10] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. The International Conference on Learning Representations, 2023.

[11] Zibin Dong, Jianye Hao, Yifu Yuan, Fei Ni, Yitian Wang, Pengyi Li, and Yan Zheng. Diffuserlite: Towards real-time diffusion planning. *Advances in Neural Information Processing Systems*, 37: 122556–122583, 2024.

[12] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[13] Lang Feng, Pengjie Gu, Bo An, and Gang Pan. Resisting stochastic risks in diffusion planners with the trajectory aggregation tree. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=NbYAmsFJrc`.

[14] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[15] Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Fei-Fei Li, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models. In *European Conference on Computer Vision*, pages 393–411. Springer, 2024.

[16] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018.

[17] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

[18] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

[19] Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=wZk69kjy9_d.

[20] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[21] Ce Hao, Anxing Xiao, Zhiwei Xue, and Harold Soh. Chd: Coupled hierarchical diffusion for long-horizon tasks. *arXiv preprint arXiv:2505.07261*, 2025.

[22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[23] Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, and Stefano Ermon. Manifold preserving guided diffusion. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=o3BxOLoxm1.

[24] Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. *The International Journal of Robotics Research*, 2025.

[25] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[26] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[27] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022.

[28] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE international conference on robotics and automation*, pages 1470–1477. IEEE, 2011.

[29] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35: 26565–26577, 2022.

[30] Kenneth Kimble, Karl Van Wyk, Joe Falco, Elena Messina, Yu Sun, Mizuho Shibata, Wataru Uemura, and Yasuyoshi Yokokohji. Benchmarking protocols for evaluating small parts robotic assembly systems. *IEEE robotics and automation letters*, 5(2):883–889, 2020.

[31] Craig A. Knoblock. Learning abstraction hierarchies for problem solving. In *Proceedings of the Eighth National Conference on Artificial Intelligence - Volume 2*, AAAI'90, page 923–928. AAAI Press, 1990. ISBN 026251057X.

[32] Kyowoon Lee and Jaesik Choi. Local manifold approximation and projection for manifold-aware diffusion planning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=EHG5Iv1mmb.

[33] Youngwoon Lee, Edward S Hu, and Joseph J Lim. Ikea furniture assembly environment for long-horizon complex manipulation tasks. In *2021 ieee international conference on robotics and automation (icra)*, pages 6343–6349. IEEE, 2021.

[34] Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, pages 20035–20064. PMLR, 2023.

[35] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-LM improves controllable text generation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=3s9IrEsjLyk.

[36] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[37] Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

[38] Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? In *The Thirteenth International Conference on Learning Representations*, 2025.

[39] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2836–2844, 2021. doi: 10.1109/CVPR46437.2021.00286.

[40] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=YJ7o2wetJ2.

[41] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=tGbpgz6yOrI.

[42] Alexander Quinn 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. In *International Conference on Machine Learning*, pages 16784–16804. PMLR, 2022.

[43] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[44] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

[45] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 2050–2053, 2018.

[46] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[47] Herbert E Robbins. An empirical bayes approach to statistics. In *Breakthroughs in Statistics: Foundations and basic theory*, pages 388–394. Springer, 1992.

[48] Earl D Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial intelligence*, 5(2): 115–135, 1974.

[49] J. Schmidhuber. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 253–258 vol.2, 1990. doi: 10.1109/IJCNN.1990.137723.

[50] Jürgen Schmidhuber. Reinforcement learning in markovian and non-markovian environments. In R.P. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3. Morgan-Kaufmann, 1990. URL https://proceedings.neurips.cc/paper_files/paper/1990/file/70c639df5e30bdee440e4cdf599fec2b-Paper.pdf.

[51] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[52] Satinder P. Singh. Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, page 202–207. AAAI Press, 1992. ISBN 0262510634.

[53] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[54] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

[55] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

[56] Francisco Suárez-Ruiz and Quang-Cuong Pham. A framework for fine robotic assembly. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 421–426. IEEE, 2016.

[57] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Bruce Porter and Raymond Mooney, editors, *Machine Learning Proceedings 1990*, pages 216–224. Morgan Kaufmann, San Francisco (CA), 1990. ISBN 978-1-55860-141-3. doi: https://doi.org/10.1016/B978-1-55860-141-3.50030-4. URL https://www.sciencedirect.com/science/article/pii/B9781558601413500304.

[58] Feng Wang and Huaping Liu. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2495–2504, 2021.

[59] Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1exf64KwH.

A Tasks and Dataset

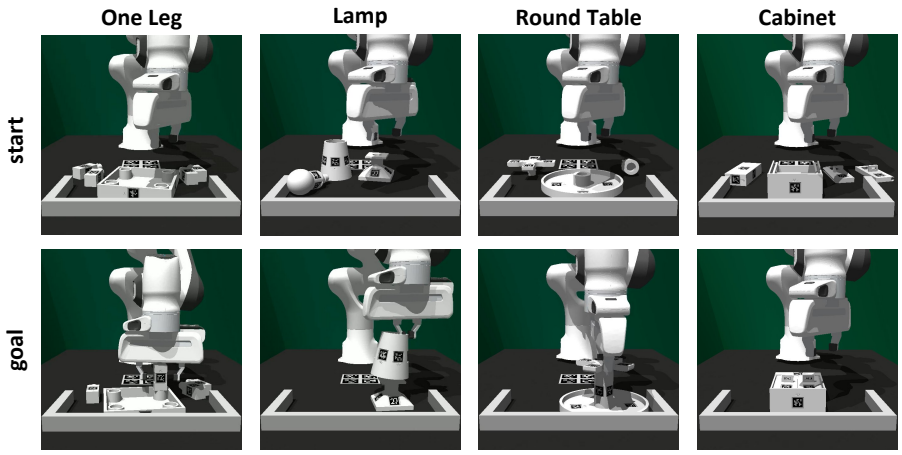

Figure 2: Overview of tasks from FurnitureBench in simulation.

FurnitureBench [24] is a novel furniture assembly benchmark for testing complex, long-horizon manipulation tasks. We choose a subset of 4 tasks from the available 9 tasks in the benchmark. The tasks involve assembling various pieces of furniture from individual parts using a simulated Franka Emika Panda robot in a IsaacGym environment. The robot receives multi-modal observations, including front and wrist RGBD images and its proprioceptive state. The goal is to assemble the furniture correctly, which requires a sequence of precise manipulations over a long time horizon as shown in Fig 2. Each task comes with three different levels with respect to the randomness in the initial furniture part configuration, making the manipulation task more challenging.

- **Low**: Furniture parts are randomly offset from their original positions by $[-1.5, 1.5]$ cm in the horizontal plane
- **Med**: Furniture parts are randomly offset from their original positions by $[-5, 5]$ cm and from their original rotations by $[-45°, 45°]$ in the horizontal plane.
- **High**: Furniture parts are randomly initialized on the workspace.

Below we describe each task in detail:

1. **One Leg**
   - **Task**: The task is to assemble one leg of a table. First, it has to stabilize the tabletop in one corner of a U-shaped wall, then it has to grasp, insert and screw the leg in its goal position.
   - **Phases**: 5 phases in total.
   - **Success Metric**: Assemble 2 parts in their goal positions.
   - **Max. Episode Length**: 700
2. **Lamp**
   - **Task**: The task is to assemble one lamp base, bulb, and lamp hood. First, it has to stabilize the lamp base in one corner of a U-shaped wall. Then, it has to grasp, insert and screw the bulb into the lamp base. Finally, it has to grasp, and insert the lamp hood into the bulb.
   - **Phases**: 7 phases in total.
   - **Success Metric**: Assemble 3 parts in their goal positions.
   - **Max. Episode Length**: 1100
3. **Round Table**
   - **Task**: The task is to assemble one round tabletop, leg, and table base. First, it has to stabilize the round tabletop in one corner of a U-shaped wall. Then, it has to grasp, insert and screw the leg into the tabletop. Finally, it has to grasp, insert and screw the table base into the leg.

- **Phases**: 8 phases in total.
- **Success Metric**: Assemble 3 parts in their goal positions.
- **Max. Episode Length**: 1500

4. **Cabinet**
    - **Task**: The task is to assemble one cabinet body, left door, right door, and cabinet top. First, it has to stabilize the cabinet body in one corner of a U-shaped wall, then it has to grasp, insert and slide each door in its goal position. Then, it has to flip the cabinet body along with the assembled doors orthogonally. Finally, it has to grasp, insert and screw the cabinet top.
    - **Phases**: 11 phases in total.
    - **Success Metric**: Assemble 4 parts in their goal positions.
    - **Max. Episode Length**: 1500

Simulation. We use a scripted policy provided by the original benchmark to collect 100 successful and 50 failure demonstrations for each task and randomness type. Unfortunately, we were unable to collect any successful trajectories with high randomness initialization for cabinet task, so we only considered low and med initial randmoness.

Observation Space. The observation space for our world model consists of two RGBD images from the front and wrist camera and robot proprioceptive states. The front and wrist camera RGB images are first resized from 1280×720 to 320×240, and then center cropped to 224×224. The robot proprioceptive state consists of current end-effector (EE) state and gripper width. In particular, 3 dimensional EE position, 4 dimensional EE orientation, 3 dimensional linear velocity, 3 dimensional rotational velocity, and 1 dimensional gripper width.

Action Space. We use an 8D action space, which consists of 3 dimensional delta EE position, 3 dimensional delta EE orientation (quaternion), and 1 dimensional gripper action. The action space is bounded between -1 and $+1$.

Goal Specification. We initialize the furniture in its final assembled state and capture the front and wrist RGBD images as well as the robot's proprioceptive state. These are then passed through the pre-trained world model's encoder to produce a fixed latent goal vector z_G. This vector is then used as the conditioning for the planner in all subsequent experiments for that task.

B Implementation Details

B.1 World Model

World models learn a compressed representation of the environment's state and a model of its dynamics within this latent space. We use a Recurrent State-Space Model (RSSM) [17], which has demonstrated strong performance in modeling complex dynamics from high-dimensional observations. We use observations from both front and wrist RGB-D cameras. The RSSM consists of the following components:

- **RGB Encoder:** A visual encoder leveraging a pretrained DINOv2 model [44] for RGB images, mapping them to a lower-dimensional embedding.
- **Depth Encoder:** A separate CNN that encodes depth images into an embedding.
- **State Encoder:** An MLP that encodes the robot's proprioceptive state into an embedding.
- **Combined Observation Embedding:** The embeddings from the RGB, Depth, and State encoders are concatenated to form the high-dimensional observation embedding $e_t = \text{Enc}_\phi(o_t)$.
- **Dynamics Model:** The dynamics are modeled in two parts. A deterministic RNN, $h_{t+1} = f_\phi(h_t, z_t)$, updates its hidden state to summarize the history. This hidden state is then used to predict a prior distribution over the current latent state, $\hat{z}_t \sim p_\phi(\hat{z}_t|h_t)$.
- **Representation Model:** A posterior distribution over the latent state $z_t \sim q_\phi(z_t|h_t, e_t)$ is inferred from the combined observation embedding e_t and the deterministic hidden state h_t of the RNN.
- **Decoder:** A decoder $\hat{o}_t \sim p_\phi(\hat{o}_t|h_t, z_t)$ reconstructs the original observation from the latent state.

The model is trained by maximizing the Evidence Lower Bound (ELBO) on the data log-likelihood, which encourages accurate reconstruction and prediction while regularizing the latent space. The objective function is:

$$\mathcal{L}_{\text{WM}} = \mathbb{E}_{q_\phi(z_{1:T}|o_{1:T})}\left[\sum_{t=1}^{T}\left(\log p_\phi(\hat{o}_t|z_t, h_t) - D_{KL}[q_\phi(z_t|h_t, e_t)||p_\phi(\hat{z}_t|h_t)]\right)\right] \tag{19}$$

This objective trains the model to form a compressed and predictive latent space $\mathcal{Z}$, where planning can be performed efficiently.

Parameter	Value
Visual Encoder	DINOv2
RGB Images	2
Depth Encoder	Yes
RSSM Stochastic Latent State Size	32
RSSM Deterministic State Size	1024
Combined Latent State Size	2048
Training Epochs	100
Batch Size	64
Learning Rate	1e-4
Loss Weight λ_{WM}	1.0
Loss Weight λ_{IDM}	0.1
Loss Weight $\lambda_{\text{contrastive}}$	0.1

Table 5: World Model Hyperparameters

B.2 Hierarchical Planners

Parameter	Value
Model Architecture	Diffusion Transformer (DiT)
Layers	4
Attention Heads	8
Hidden Dimension	512
Training Epochs	10,000
Batch Size	64
Learning Rate	1e-4
Diffusion Timesteps	1000 (linear beta schedule)
EBM Architecture	4-layer Transformer
Subgoal Interval (H)	Task-dependent
Projection Steps	$t \in [T/3, 2T/3]$
Nearest Neighbors (k)	10
Projection Variance Retention	$\lambda = 0.99$
Loss Weight λ_{HL}	1.0
Loss Weight λ_{EBM}	0.1
Loss Weight λ_{proj}	0.05
Inference Steps	100
Classifier-Free Guidance Scale	2.0
EBM Guidance Scale	0.1
Context Conditioning	z_0, z_G

Table 6: High-Level Planner Hyperparameters

Task	`one_leg`	`lamp`	`round_table`	`cabinet`
Subgoal interval (H)	35	55	75	75

Table 7: Task-dependent high-level planning subgoal intervals (H)

Parameter	Value
Model Architecture	Conditional Rectified Flow
Layers	4
Attention Heads	8
Hidden Dimension	512
Training Epochs	10,000
Batch Size	64
Learning Rate	1e-4
Loss Weight λ_{LL}	1.0
ODE Solver	Dormand-Prince
Number of Integration Steps	20
Context Conditioning	z_{k-1}, z_k

Table 8: Low-Level Planner Hyperparameters

C Baselines

In this section, we provide a brief description of each baseline method.

- `DP` [8]: A policy learning method that leverages diffusion models to directly model the distribution of actions conditioned on observations, enabling sample-efficient learning from offline data.
- `JUICER` [4]: A data-efficient imitation learning framework for robotic assembly that leverages expressive policy architectures, dataset expansion, and simulation-based data augmentation to learn multi-part, long-horizon assembly directly from RGB images.
- `Diffuser` [27]: A diffusion probabilistic model that plans by iteratively denoising trajectories. It treats planning as a conditional generation problem and can produce diverse and high-quality trajectories.
- `DD` [2]: A conditional diffusion model that views decision-making as a conditional generative modeling problem, leveraging classifier-free guidance with low-temperature sampling to extract high-likelihood, return-maximizing trajectories from offline datasets without relying on dynamic programming.
- `HDMI` [34]: Hierarchical Diffusion for Offline Decision Making (HDMI) proposes a hierarchical trajectory-level diffusion probabilistic model to tackle challenges in offline reinforcement learning, especially for long-horizon tasks. It employs a cascade framework with a Reward-Conditional Goal Diffuser for discovering subgoals and a Goal-Conditional Trajectory Diffuser for generating action sequences.
- `SHD` [7]: A hierarchical diffusion-based planning method that uses a "jumpy" planning strategy at the high level for subgoal generation and a low-level diffuser for subgoal achievement, aiming for improved efficiency and generalization in long-horizon tasks.

D Theoretical Framework

This section provides the detailed mathematical proofs for the propositions made in the previous sections.

D.1 Proof for Proposition on Manifold-Aware Guided Planning

Statement. *Given a base diffusion planner ϵ_θ trained for classifier-free guidance, a learned energy function $E_\phi(z|c)$, and a manifold projection operator $\mathcal{P}_\mathcal{M}$, the manifold-aware guided planner, which combines EBM guidance with manifold projection, corresponds to sampling from a posterior distribution $p(z|y = 1, z \in \mathcal{M}, c)$ that maximizes the likelihood of generating a successful and feasible goal-conditioned plan.*

Proof. The manifold-aware guidance process can be viewed as implementing constrained Bayesian inference. We seek to sample from the posterior:

$$p(z|y = 1, z \in \mathcal{M}, c) \propto p(y = 1|z, c)p(z|c)\mathbf{1}[z \in \mathcal{M}]$$

where $\mathbf{1}[z \in \mathcal{M}]$ is the indicator function for the feasible manifold.

The two-step process approximates this constrained posterior:

1. The guided sampling step samples from $p(y = 1|z, c)p(z|c)$, implementing the unconstrained Bayesian posterior. This is achieved by combining classifier-free guidance for the conditional term $p(z|c)$ and EBM guidance for the success term $p(y = 1|z, c)$, as detailed in Proof D.2
2. The projection step enforces the manifold constraint $z \in \mathcal{M}$ by mapping to the closest point on the approximated manifold.

By the principle of alternating projections and the contraction property of projection operators, this two-step process converges to a point that balances optimality (high success probability) with feasibility (remaining on the manifold). The approximation error depends on the quality of the local manifold approximation, which improves with the number of neighbors k and the intrinsic dimensionality of the subgoal space. This shows that our combined guidance approach is a principled implementation of Bayes-optimal sampling, steering the generative process towards plans that are both relevant to the goal and likely to succeed, while remaining on the feasible manifold. □

D.2 Proof for EBM-based Guidance

Statement. *Given a base diffusion planner ϵ_θ trained for classifier-free guidance and a learned energy function $E_\phi(z|c)$ that estimates the probability of success conditioned on context c, the EBM-guided component of the planner corresponds to sampling from a posterior distribution that maximizes the likelihood of generating a successful, goal-conditioned plan.*

Proof. The proof proceeds in two steps. First, we derive the theoretically optimal score function for sampling successful, goal-conditioned plans using Bayes' rule. Second, we show how our combined guidance mechanism implements an effective approximation of this optimal score.

1. **Deriving the Optimal Score Function.** Our goal is to sample from the posterior distribution $p(z|y = 1, c)$, which is the distribution of plans z that are successful ($y = 1$) given the context $c = (z_0, z_G)$. Using Bayes' rule, we can express this posterior as:

$$p(z|y = 1, c) = \frac{p(y = 1|z, c)p(z|c)}{p(y = 1|c)} \propto p(y = 1|z, c)p(z|c)$$

Taking the logarithm, we get:

$$\log p(z|y = 1, c) = \log p(y = 1|z, c) + \log p(z|c) + \text{const.}$$

The score function is the gradient of the log-probability with respect to the plan z. The optimal score for our desired posterior is therefore:

$$\nabla_z \log p(z|y = 1, c) = \nabla_z \log p(y = 1|z, c) + \nabla_z \log p(z|c)$$

This equation tells us that the optimal guided score is the sum of two terms:

- $\nabla_z \log p(z|c)$: The score of the original conditional planner. This term ensures the plan is **relevant** to the context c.
- $\nabla_z \log p(y = 1|z, c)$: The gradient of the log-probability of success. This term ensures the plan is **viable** and likely to succeed.

2. **Implementing the Score with Combined Guidance.** Our framework approximates each of these two terms:

- **EBM Guidance for Success:** The Energy-Based Model $E_\phi(z|c)$ is trained to model the success probability. We define $p(y = 1|z, c) \propto \exp(-E_\phi(z|c))$, where low energy corresponds to high success probability. The gradient of the log-probability of success is therefore directly related to the gradient of the energy function:

  $$\nabla_z \log p(y = 1|z, c) = -\nabla_z E_\phi(z|c)$$

  This is the **EBM Guidance** term in our equation.
- **CFG for Conditional Relevance:** The base diffusion model, ϵ_θ, is trained to predict the noise, which is proportional to the score. The term $\nabla_z \log p(z|c)$ is the score of the conditional model. Classifier-Free Guidance (CFG) is a technique to strengthen this conditioning at inference time. The CFG-adjusted score is:

  $$\hat{\nabla}_z \log p(z|c) \approx \nabla_z \log p(z|\emptyset) + w_{cfg}(\nabla_z \log p(z|c) - \nabla_z \log p(z|\emptyset))$$

  This is the **Classifier-Free Guidance** term in our equation, expressed in terms of the noise predictions ϵ_θ.

By combining these two approximations, our final guided noise prediction implements the theoretically optimal score:

$$\hat{\epsilon}_\theta(z_\ell, \ell, c) = \underbrace{\epsilon_\theta(z_\ell, \ell, \emptyset) + w_{cfg}(\epsilon_\theta(z_\ell, \ell, c) - \epsilon_\theta(z_\ell, \ell, \emptyset))}_{\text{Approximates } \nabla_z \log p(z|c) \text{ via CFG}} - \underbrace{w_{ebm}\sqrt{1 - \bar{\alpha}_\ell}\nabla_{z_\ell} E_\phi(z_\ell|c)}_{\text{Approximates } \nabla_z \log p(y=1|z,c) \text{ via EBM}}$$

This shows that our combined guidance approach is a principled implementation of Bayes-optimal sampling, steering the generative process towards plans that are both relevant to the goal and likely to succeed. □

D.3 Proof for Proposition on EBM Guidance Gap

Statement. *The EBM guidance gap $\Delta_{EBM}(z_\ell)$ has a lower bound scaling as $\frac{c}{\sqrt{1-\bar{\alpha}_\ell}}\sqrt{d}$ in high-dimensional latent spaces, where $c > 0$ is a constant independent of dimensionality d.*

Proof. We aim to provide a more detailed derivation for the lower bound of the EBM guidance gap in high-dimensional latent spaces. The core of this issue stems from the discrepancy between the true optimal energy guidance and our learned EBM approximation, particularly in how they weight successful plans.

1. **Forward Process and Conditional Score.** The forward diffusion process defines how a clean latent state z_0 is noised to z_ℓ at timestep ℓ:

   $$z_\ell = \sqrt{\bar{\alpha}_\ell} z_0 + \sqrt{1 - \bar{\alpha}_\ell}\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

   From this, the conditional distribution $q(z_0|z_\ell)$ can be expressed as a Gaussian with mean $\mu(z_\ell, \ell) = \frac{1}{\sqrt{\bar{\alpha}_\ell}}(z_\ell - \sqrt{1 - \bar{\alpha}_\ell}\epsilon)$ and variance $\Sigma(\ell) = (1 - \bar{\alpha}_\ell)\mathbf{I}$. The gradient of the log-probability of this conditional distribution with respect to z_ℓ is crucial for relating scores in z_0 space to z_ℓ space:

   $$\nabla_{z_\ell} \log q(z_0|z_\ell) = \nabla_{z_\ell} \log \mathcal{N}\left(z_0; \frac{1}{\sqrt{\bar{\alpha}_\ell}}(z_\ell - \sqrt{1 - \bar{\alpha}_\ell}\epsilon), (1 - \bar{\alpha}_\ell)\mathbf{I}\right)$$

   This simplifies to:

   $$\nabla_{z_\ell} \log q(z_0|z_\ell) = -\frac{1}{\sqrt{1 - \bar{\alpha}_\ell}}\epsilon$$

2. **True Optimal Energy Guidance.** The true optimal energy guidance, $\nabla_{z_\ell} E_{\text{true}}(z_\ell|c)$, aims to steer the diffusion process towards regions of low energy (high success probability) in the z_0 space. This gradient is given by the expectation of the score of $q(z_0|z_\ell)$ weighted by the exponential of the negative energy function, effectively performing importance sampling towards more successful z_0 configurations:

$$\nabla_{z_\ell} E_{\text{true}}(z_\ell|c) = \frac{\mathbb{E}_{q(z_0|z_\ell)}[e^{-E(z_0|c)} \nabla_{z_\ell} \log q(z_0|z_\ell)]}{\mathbb{E}_{q(z_0|z_\ell)}[e^{-E(z_0|c)}]}$$

Substituting the expression for $\nabla_{z_\ell} \log q(z_0|z_\ell)$ and assuming $\mathbb{E}_{q(z_0|z_\ell)}[e^{-E(z_0|c)}]$ is a normalizing constant, we get:

$$\nabla_{z_\ell} E_{\text{true}}(z_\ell|c) = \frac{1}{\sqrt{1-\bar{\alpha}_\ell}} \frac{\mathbb{E}_{q(z_0|z_\ell)}[e^{-E(z_0|c)}(-\epsilon)]}{\mathbb{E}_{q(z_0|z_\ell)}[e^{-E(z_0|c)}]} = -\frac{1}{\sqrt{1-\bar{\alpha}_\ell}} \mathbb{E}_{q(z_0|z_\ell)}[\epsilon|e^{-E(z_0|c)}]$$

where $\mathbb{E}[\epsilon|e^{-E(z_0|c)}]$ denotes the expectation of ϵ conditioned on z_0 being sampled with a probability proportional to $e^{-E(z_0|c)}$.

3. **Learned EBM Guidance.** Our learned EBM guidance, $\nabla_{z_\ell} E_\phi(z_\ell|c)$, typically approximates the gradient of the energy function at z_ℓ. In many practical implementations, this effectively corresponds to a linear weighting of the score of $q(z_0|z_\ell)$ by the energy function itself, rather than its exponential:

$$\nabla_{z_\ell} E_\phi(z_\ell|c) \approx \mathbb{E}_{q(z_0|z_\ell)}[E(z_0|c) \nabla_{z_\ell} \log q(z_0|z_\ell)] = -\frac{1}{\sqrt{1-\bar{\alpha}_\ell}} \mathbb{E}_{q(z_0|z_\ell)}[E(z_0|c)\epsilon]$$

This approximation introduces a discrepancy because the relationship between the energy $E(z_0|c)$ and the success probability $p(y=1|z_0,c)$ is exponential ($p(y=1|z_0,c) \propto e^{-E(z_0|c)}$), not linear.

4. **Analysis of the Guidance Gap.** The EBM guidance gap is defined as $\Delta_{\text{EBM}}(z_\ell) = \|\nabla_{z_\ell} E_{\text{true}}(z_\ell|c) - \nabla_{z_\ell} E_\phi(z_\ell|c)\|_2$. Substituting the expressions, we get:

$$\Delta_{\text{EBM}}(z_\ell) = \left\| -\frac{1}{\sqrt{1-\bar{\alpha}_\ell}} \left(\mathbb{E}_{q(z_0|z_\ell)}[\epsilon|e^{-E(z_0|c)}] - \mathbb{E}_{q(z_0|z_\ell)}[E(z_0|c)\epsilon] \right) \right\|_2$$

Let $\delta(z_0) = \frac{e^{-E(z_0|c)}}{\mathbb{E}_{q(z_0|z_\ell)}[e^{-E(z_0|c)}]} - E(z_0|c)$ represent the difference between the ideal exponential weighting (normalized) and the approximate linear weighting. The term in the parenthesis can be viewed as $\mathbb{E}_{q(z_0|z_\ell)}[\delta(z_0)\epsilon]$. In high-dimensional latent spaces (i.e., when d is large), the noise vector ϵ typically has a magnitude of approximately $\sqrt{d}$ (i.e., $\|\epsilon\|_2 \approx \sqrt{d}$). Furthermore, the components of ϵ are largely independent. Due to the Central Limit Theorem and concentration inequalities, even small biases in the weighting function $\delta(z_0)$ can lead to a significant accumulated error when multiplied by a high-dimensional random vector. Specifically, if $\mathbb{E}_{q(z_0|z_\ell)}[\delta(z_0)] \neq 0$, the term $\|\mathbb{E}_{q(z_0|z_\ell)}[\delta(z_0)\epsilon]\|_2$ will tend to scale with $\sqrt{d}$ in expectation, as the contributions from different dimensions accumulate.

Therefore, there exists a constant $c > 0$ (which depends on the magnitude of the mismatch $\delta(z_0)$ and the properties of the noise distribution) such that:

$$\Delta_{\text{EBM}}(z_\ell) \geq \frac{c}{\sqrt{1-\bar{\alpha}_\ell}} \sqrt{d}$$

This lower bound shows that the inaccuracies in energy guidance are exacerbated in high-dimensional latent spaces and as the diffusion process approaches z_0 (i.e., as $\ell \to 0$, $1-\bar{\alpha}_\ell \to 0$, making the term $\frac{1}{\sqrt{1-\bar{\alpha}_\ell}}$ large). This leads sampled trajectories to deviate significantly from the true data manifold. □

E More Ablation Studies

E.1 Choice of Vision Encoder

To evaluate the impact of the vision encoder on the performance of our world model and the overall HDFlow framework, we conducted an ablation study comparing different visual backbones. We trained the world model with various encoders: a simple CNN, R3M [41], VIP [40], MAE [22], and our chosen DINOv2 [44]. The results are summarized in Table 9.

Backbone	one_leg			lamp			round_table			cabinet	
	Low	Med	High	Low	Med	High	Low	Med	High	Low	Med
CNN	55	18	2	28	8	0	20	6	0	3	0
R3M	68	25	5	39	15	4	29	10	2	7	1
VIP	75	32	9	45	20	7	36	14	5	10	3
MAE	83	45	18	57	33	10	48	25	8	15	6
DINOv2	**92**	**71**	**39**	**68**	**49**	**34**	**61**	**43**	**27**	**55**	**36**

Table 9: Ablation study on the choice of vision encoder for the world model. This table reports HDFlow success rates (%) on all tasks with respective randomization levels.

We found out that DINOv2 leads to significantly more accurate and detailed reconstructions of observations, which translates to a richer and more semantically structured latent space. This improved latent representation is crucial for both the high-level diffusion planner and the low-level rectified flow planner, enabling them to generate more effective and feasible plans. The quantitative results in Table 9 further confirm that DINOv2 consistently yields the highest HDFlow success rates, underscoring its importance as a foundational component of our framework.

E.2 Performance under different total subgoals

We investigate the impact of the total number of subgoals K on the performance of HDFlow. The total number of subgoals dictates the temporal abstraction of our hierarchical planner, influencing both the complexity of high-level subgoals and the length of low-level trajectories. We conduct experiments on all tasks with varying randomness, varying K and reporting the success rates in Table 10.

Subgoals	one_leg			lamp			round_table			cabinet	
K	Low	Med	High	Low	Med	High	Low	Med	High	Low	Med
10	61	21	2	33	13	2	23	9	0	6	0
15	78	35	10	52	25	8	40	18	5	25	10
20	**92**	**71**	**39**	**68**	**49**	**34**	**61**	**43**	**27**	**55**	**36**
25	89	60	30	63	45	25	58	40	20	50	30
30	85	45	15	60	35	12	50	28	10	35	18

Table 10: Ablation study on the choice of total subgoals K. Success rates (%) of HDFlow on all tasks with respective randomization levels for different K values, highlighting the importance of temporal abstraction for optimal performance.

The results indicate that an optimal total number of subgoals K exists for each task. A very small K (e.g., 10) leads to an overly coarse high-level plan, requiring the low-level rectified flow model to bridge larger gaps in latent space, which can be challenging. Conversely, a very large K (e.g., 30) makes the high-level diffusion model generate too many subgoals, which can accumulate errors. A value of $K = 20$ consistently yields the highest success rates, representing a balance where the high-level planner provides meaningful strategic guidance, and the low-level planner can efficiently and accurately execute the dense trajectories. This ablation highlights the importance of carefully tuning the temporal abstraction level for optimal performance in hierarchical planning.

F Simulation Rollouts

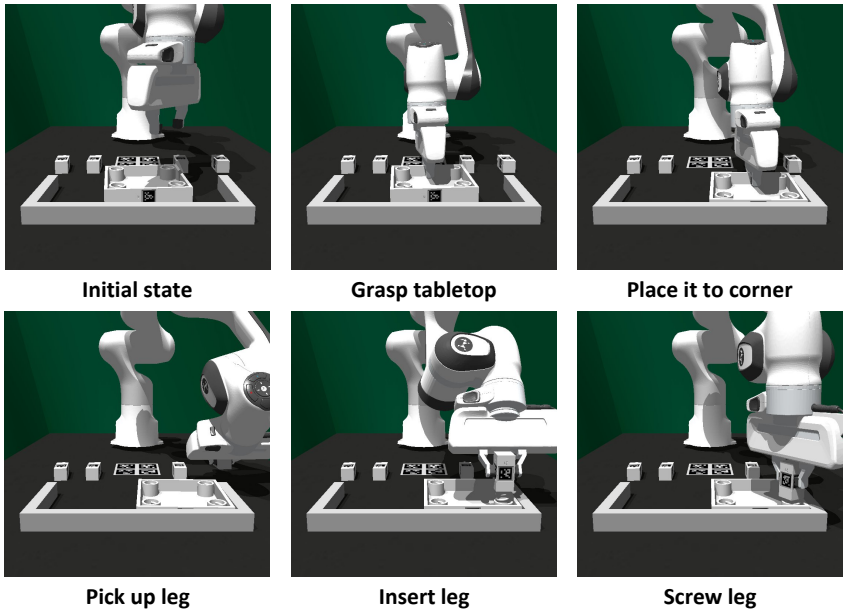

Figure 3: A successful rollout of `HDFlow` planner on the `one_leg` assembly task initialized with `Low` randomness.

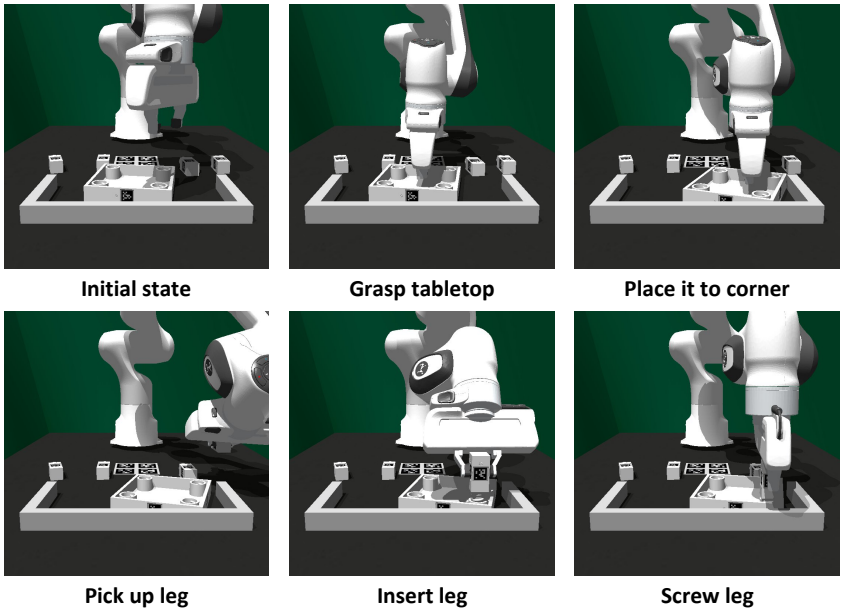

Figure 4: A successful rollout of `HDFlow` planner on the `one_leg` assembly task initialized with `Med` randomness.

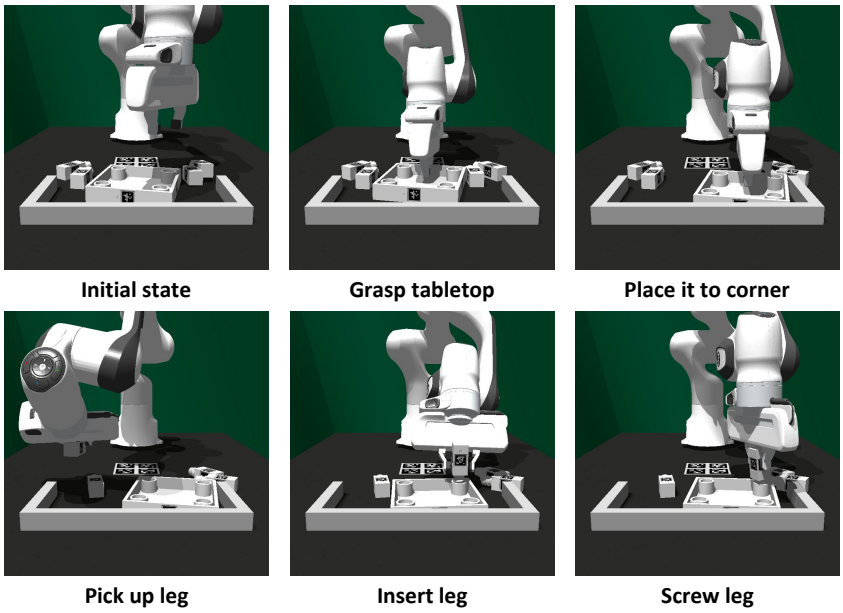

Figure 5: A successful rollout of `HDFlow` planner on the `one_leg` assembly task initialized with `High` randomness.

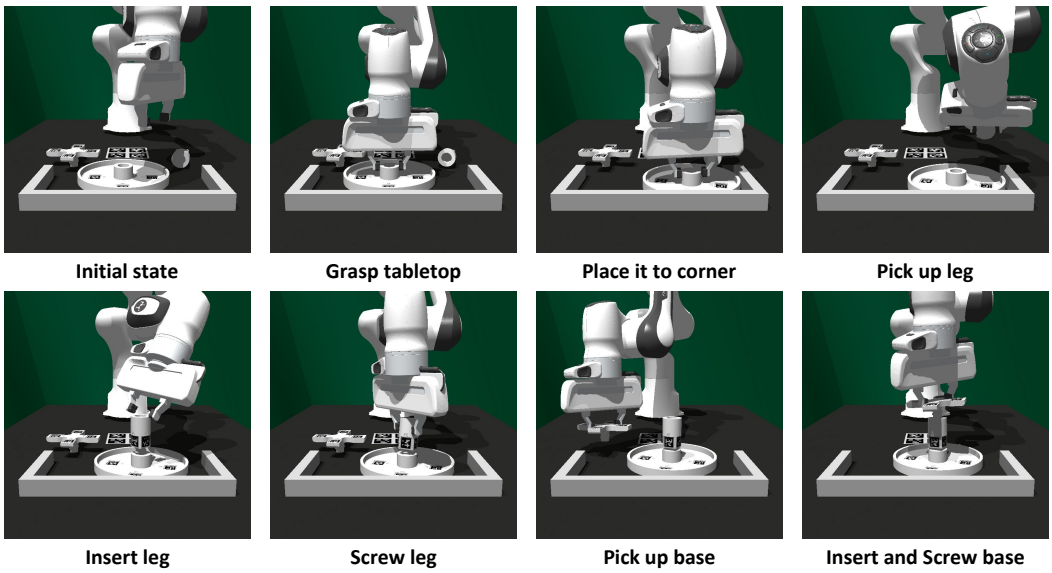

Figure 6: A successful rollout of `HDFlow` planner on the `round_table` assembly task initialized with `Low` randomness.

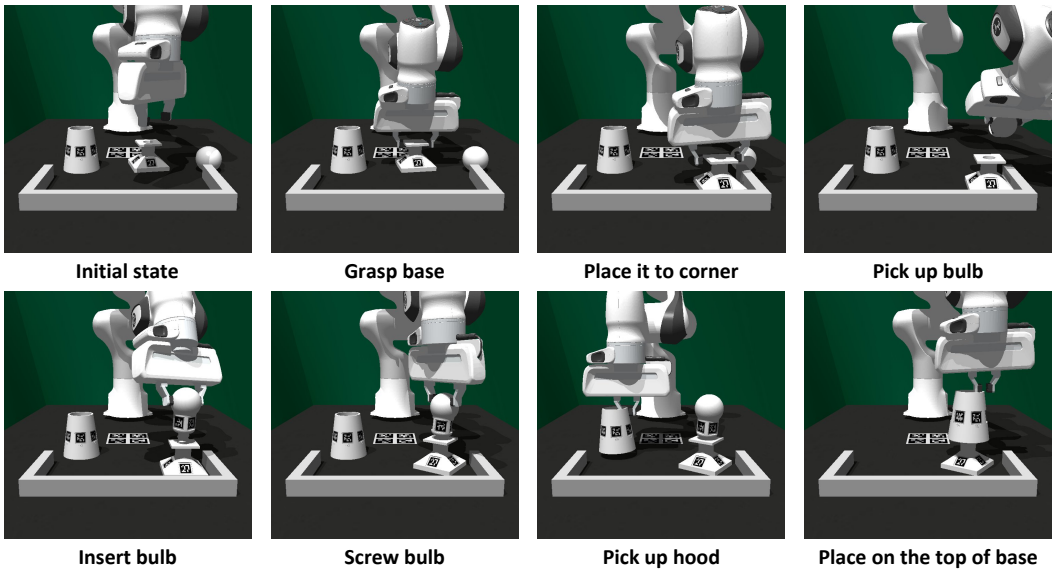

Figure 7: A successful rollout of `HDFlow` planner on the `lamp` assembly task initialized with `Low` randomness.

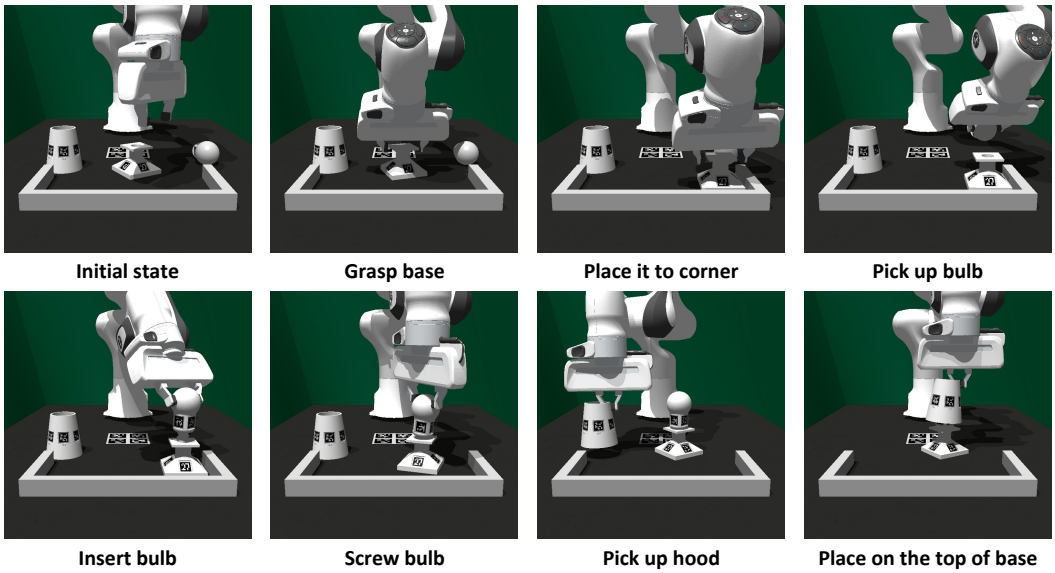

Figure 8: A successful rollout of `HDFlow` planner on the `lamp` assembly task initialized with `Med` randomness.

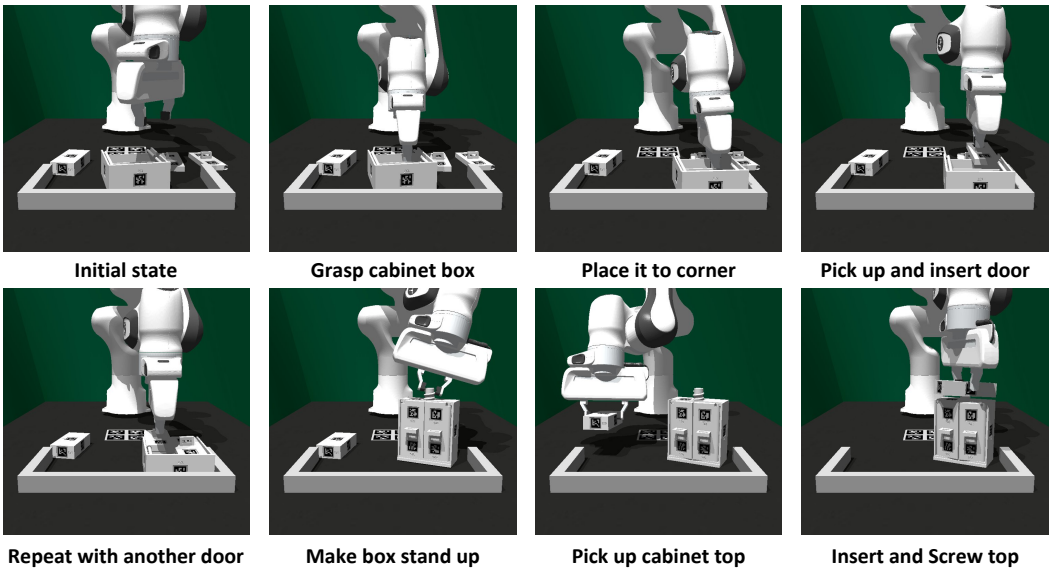

Figure 9: A successful rollout of `HDFlow` planner on the `cabinet` assembly task initialized with `Low` randomness.