Extremum Flow Matching for Offline Goal Conditioned Reinforcement Learning

Quentin Rouxel, Clemente Donoso, Fei Chen, Serena Ivaldi, and Jean-Baptiste Mouret

***Abstract*—Imitation learning is a promising approach for enabling generalist capabilities in humanoid robots, but its scaling is fundamentally constrained by the scarcity of high-quality expert demonstrations. This limitation can be mitigated by leveraging suboptimal, open-ended play data, often easier to collect and offering greater diversity. This work builds upon recent advances in generative modeling, specifically Flow Matching, an alternative to Diffusion models. We introduce a method for estimating the extremum of the learned distribution by leveraging the unique properties of Flow Matching, namely, deterministic transport and support for arbitrary source distributions. We apply this method to develop several goal-conditioned imitation and reinforcement learning algorithms based on Flow Matching, where policies are conditioned on both current and goal observations. We explore and compare different architectural configurations by combining core components, such as critic, planner, actor, or world model, in various ways. We evaluated our agents on the OGBench benchmark and analyzed how different demonstration behaviors during data collection affect performance in a 2D non-prehensile pushing task. Furthermore, we validated our approach on real hardware by deploying it on the Talos humanoid robot to perform complex manipulation tasks based on high-dimensional image observations, featuring a sequence of pick-and-place and articulated object manipulation in a realistic kitchen environment. Experimental videos and code are available at: https://hucebot.github.io/extremum_flow_matching_website/**

I. Introduction

In recent years, imitation learning has reemerged as a powerful approach to solving complex manipulation tasks on real robots [1]–[3]. Unlike reinforcement learning, which typically relies on simulation and sim-to-real transfer, imitation learning bypasses the need for simulation, reward engineering, and domain adaptation by directly learning behaviors from teleoperated demonstrations (i.e., behavior cloning). This allows policies to handle environments that are difficult to simulate, such as dexterous or nonprehensile manipulation involving complex contacts, friction, and interactions with articulated, soft, or deformable objects. Moreover, end-to-end training on pixel-based inputs makes it possible to train policies for a wide range of tasks without any feature engineering. However, imitation learning comes at the cost of requiring large and diverse demonstration datasets, a process that is slow and resource intensive to collect on real hardware.

Q. Rouxel, C. Donoso, S. Ivaldi, and J. B. Mouret are with Inria, CNRS, Université de Lorraine, France (firstname.lastname@inria.fr). Q. Rouxel, and F. Chen are with the Department of Mechanical and Automation Engineering, T-Stone Robotics Institute, The Chinese University of Hong Kong, Hong Kong (quentinrouxel@cuhk.edu.hk, f.chen@ieee.org).

This research was supported by the CPER CyberEntreprises, the Creativ'Lab platform of Inria/LORIA, the EU Horizon project euROBIN (GA n.101070596), the France 2030 program through projects PEPR O2R AS3 and PI3 (ANR-22-EXOD-007, ANR-22-EXOD-004).

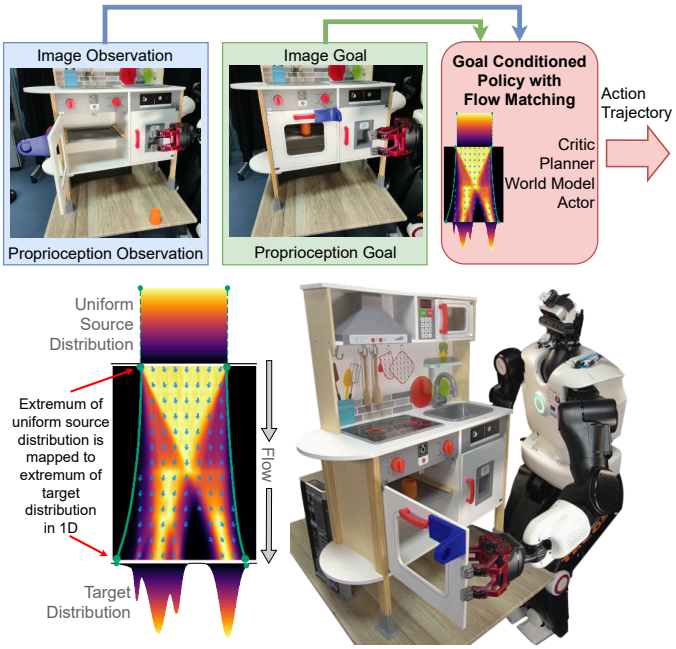


Figure 1. Goal conditioned policy using Extremum Flow Matching for manipulation from play demonstrations on the Talos humanoid robot (bottom right). The policy takes the current and goal images as input and outputs the robot's command trajectory (top). With Flow Matching, the flow vector field (bottom left) continuously and deterministically transforms a 1D source distribution into a target distribution. Since flow paths do not intersect, the boundaries of a uniform source distribution are mapped to the minimum and maximum of the target distribution's support. Using this idea, Extremum Flow Matching addresses the problem of optimality by selecting the shortest path to the goal.

As the aim of this line of research is to build fully generalist policies capable of solving various tasks, it becomes essential for users to specify desired goals. This need has driven the rise of goal-conditioned imitation learning [4], [5]. Goals can be provided in various forms, through discrete labels (which require costly dataset labeling and segmentation), goal images [6], [7], or, more recently, natural language instructions [8] via vision-language-action models [3].

These recent advances in machine learning coincide with the rapid development of next-generation humanoid robot hardware [9], [10], driven by a similar long-standing ambition: to create generalist robotic platforms that combine versatile locomotion and dexterous manipulation skills, multi-contact capabilities, compact footprints suitable for constrained environments, and human-like morphology adapted to operate within human-centric spaces. The convergence of such hardware, traditional model-based control, and scalable, multimodal

learning frameworks, acting as high-level controllers and planners, opens exciting new frontiers in humanoid robotics.

In this work, we learn challenging long-horizon manipulation tasks in a realistic kitchen setting with a humanoid robot. As illustrated in Fig. 1 (bottom right) and attached video, our learning algorithms can generate motions combining grasping, door opening/closing and tray pulling/pushing using the Talos humanoid robot, all based on high-dimensional visual input.

Imitation learning typically relies on near-optimal demonstrations focused on specific tasks to produce high-quality policies. However, scaling data collection under these constraints is costly and impractical. To overcome this, recent work has focused on learning from play datasets [6], [11]–[13], in which demonstrations freely explore the environment through open-ended interactions without pursuing specific goals. This approach provides broader state-action coverage, greater variability, and makes data collection far more scalable and convenient, opening the door to large-scale, unstructured datasets collected from diverse sources. By conditioning on goals, it becomes possible to train from such play data, removing the need for carefully curated demonstrations. However, two major challenges remain (see Fig. 2): actions in play data are often suboptimal for specific tasks, and full trajectories from an initial state to a distant goal are rarely demonstrated. Although fragments of optimal behavior exist across episodes, they are not stitched into a complete solution.

Optimality can be addressed with purely supervised learning techniques by conditioning on returns [14]. While such methods have shown interesting empirical performance, they lack a principled mechanism for stitching partial trajectories toward long-horizon goals. In contrast, offline reinforcement learning [15]–[19] explicitly tackles both optimality and stitching by learning value functions via dynamic programming.

These recent advances in imitation learning have been largely driven by progress in generative modeling, particularly Diffusion models [20], [21] and Flow Matching [22]. Generative methods have proven decisive because they enable learning and sampling from the entire distribution, effectively handling multi-modal distributions. In contrast, traditional supervised learning often averages over non-convex distributions, leading to inaccurate predictions. Additionally, generative models can produce high-dimensional outputs, making them well-suited for generating full trajectories in planning. Addressing stitching capabilities, the application of Diffusion for long-term trajectory planning is an actively studied research direction [18], [19], [23]–[26].

Diffusion and flow-based methods have been unified within a common theoretical framework based on optimal transport theory [27], [28], where Diffusion corresponds to the stochastic formulation and Flow Matching to the deterministic one. Flow Matching has emerged as an alternative to Diffusion and has been applied to imitation and reinforcement learning [29]–[32] as well as Vision-Language-Action models [3], primarily due to its faster inference times, a critical advantage for real-time robotic applications.

Specific to offline reinforcement learning, recent work exploited the unique properties of Flow Matching to address the key challenge of evaluating the critic value V and action-value Q functions. [33] focused on improving the guidance of the generative process, while Flow Q-Learning [34] introduced a distillation mechanism that integrates Flow Matching into reinforcement learning algorithms without requiring simulating the flow during training.

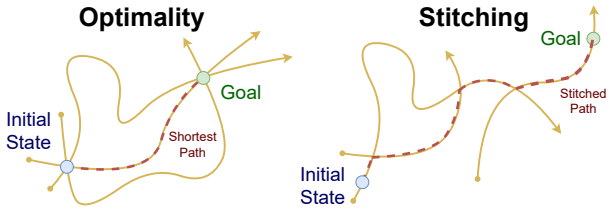


Figure 2. Imitation and offline reinforcement learning from play data face two key challenges: Optimality (left) – selecting actions that lead to the shortest possible paths, and Stitching (right) – combining sub-trajectories that originate from distinct demonstrated episodes to reach long-horizon tasks.

Concurrently with this line of work, we exploit the unique advantages of Flow Matching over Diffusion – namely, its deterministic inference and its ability to handle arbitrary source distributions – to address the reward maximization problem in learning the critic for reinforcement learning.

Contributions: Our key contributions are threefold:

- To address the challenge of optimality and learn critic in offline reinforcement learning algorithms, we propose a method called *Extremum Flow Matching*, which estimates distributional bounds using Flow Matching and conditioning on returns.
- We apply this method to design several goal conditioned imitation and offline reinforcement learning agents based on Flow Matching.
- We conduct extensive evaluations, comparing these agents against the state of the art using the OGBench benchmark, analyzing the impact of dataset collection behaviors, and validating performance through real-world experiments on the Talos humanoid robot (see attached video).

II. Background

A. Flow Matching

Flow Matching is a generative method that enables to model and then sample from complex, potentially multi-modal probability distributions. It does so by learning a deterministic and continuous transformation from a simple source distribution $\boldsymbol{x}^{\text{src}} \sim \mathcal{P}^{\text{src}}$ to a target distribution $\boldsymbol{x}^{\text{dst}} \sim \mathcal{P}^{\text{dst}}$, where both $\boldsymbol{x}^{\text{src}}$ and $\boldsymbol{x}^{\text{dst}}$ lie in the same space $\mathbb{R}^n$. The flow's vector field f, typically parameterized by a neural network, is trained using supervised learning via the Flow Matching loss $\mathcal{L}_{\text{flow}}$ without requiring simulation of the full integration process of the flow:

$$\begin{aligned} \boldsymbol{x}_t &= (1-t)\boldsymbol{x}^{\text{src}} + t\boldsymbol{x}^{\text{dst}},\ t \sim \mathcal{U}(0,1) \\ \mathcal{L}_{\text{flow}} &= \left\| f(\boldsymbol{x}_t, t, \boldsymbol{c}) - (\boldsymbol{x}^{\text{dst}} - \boldsymbol{x}^{\text{src}}) \right\|_2^2 \end{aligned} \tag{1}$$

where $t \in \mathbb{R}$ represents the interpolation, i.e. the progress of the transport from the source to the target distribution, and is

sampled uniformly from the interval $[0,1]$. The vector $\boldsymbol{c} \in \mathbb{R}^m$ is an optional conditioning variable that allows modeling a conditional target distribution $P^{\text{dst}}(\boldsymbol{x}|\boldsymbol{c})$.

During inference, novel samples from the target distribution $\tilde{\boldsymbol{x}}^{\text{dst}}$ are generated by first sampling a point $\boldsymbol{x}_0$ from the source distribution $\mathcal{P}^{\text{src}}$, and then integrating the learned flow from $t=0$ to $t=1$. This integration is typically performed using Euler integration:

$$\boldsymbol{x}_0 \sim \mathcal{P}^{\text{src}},\ \boldsymbol{x}_{t+\Delta t} = \boldsymbol{x}_t + \Delta t f(\boldsymbol{x}_t, t, \boldsymbol{c}),\ \tilde{\boldsymbol{x}}^{\text{dst}} = \boldsymbol{x}_1 \tag{2}$$

In the remainder of this paper, we denote the generative process F conditioned by $\boldsymbol{c}$ as follows: the model definition and training process is expressed as $F : \mathcal{P}^{\text{src}}|\boldsymbol{c} \mapsto \boldsymbol{x}^{\text{dst}}$. Sampling-based inference is written as $\tilde{\boldsymbol{x}}^{\text{dst}} = F(\mathcal{P}^{\text{src}}|\boldsymbol{c})$, where a point is sampled from the source distribution $\mathcal{P}^{\text{src}}$ and then transformed through the flow. Deterministic inference is denoted by $\tilde{\boldsymbol{x}}^{\text{dst}} = F(\boldsymbol{x}_0|\boldsymbol{c})$, where given a specific point $\boldsymbol{x}_0$, only the flow's integration from source to target is computed.

B. Trajectory Dataset Formalism

In this work, we leverage the generative method Flow Matching to devise novel goal conditioned imitation and offline reinforcement learning algorithms. Rather than using the classical Markov Decision Process framework, where transitions are represented as $(\boldsymbol{o}_k, \boldsymbol{a}_k, \boldsymbol{o}_{k+1})$ and only consider immediate actions and subsequent observations, we adopt a trajectory-based formalism that captures sequences of observations and actions across episodes. Demonstrations are collected through teleoperation of the robot across multiple episodes, with each episode comprising a full trajectory of observations and corresponding actions. The set of demonstrated episodes is fixed and used in an offline reinforcement learning setting, where no further interaction with the environment occurs during training.

From the demonstrated episodes, we construct a trajectory-based training dataset composed of tuples of the form $(\boldsymbol{o}_k, \boldsymbol{\tau}^o_k, \boldsymbol{\tau}^a_k, d, \boldsymbol{g})$, where $k \in \mathbb{N}$ denotes the time step within an episode, $\boldsymbol{o}_k$ is the observation at time step k, $\boldsymbol{\tau}^o_k = \boldsymbol{o}_{k:k+L_o S_o : S_o}$ is the sub-sampled trajectory of future observations starting from k of length L_o, $\boldsymbol{\tau}^a_k = \boldsymbol{a}_{k:k+L_a S_a : S_a}$ is the sub-sampled trajectory of future actions starting from time step k of length L_a, L_o and $L_a \in \mathbb{N}$ specify respectively the lengths of the future observation and action trajectories, S_o and $S_a \in \mathbb{N}$ specify respectively the sub-sampling strides of future observation and action trajectories. Akin to Hindsight Experience Replay [35], the goal observation $\boldsymbol{g} = \boldsymbol{o}_{k+d}$ is sampled from the same episode, d time steps after the current step k. The distance offset $d \in \mathbb{N}$ is drawn uniformly as $d \sim \mathcal{U}(0, L_g)$ where $L_g \in \mathbb{N}$ is the maximum goal horizon. Note that, when there is no ambiguity, the time step k is omitted for clarity.

In this formalism, the return of an episode starting from observation $\boldsymbol{o}_k$ and reaching the goal $\boldsymbol{g}$ is defined as the time-step distance d, which the agent aims to minimize. In this work, we also assume that observations and goals lie in the same space, although prior work [3], [6]–[8] has demonstrated that goals can be encoded in alternative latent spaces or modalities, such as natural language.

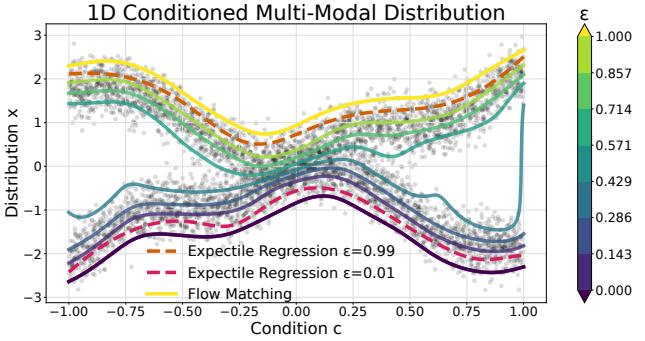


Figure 3. Comparison between Expectile Regression and Extremum Flow Matching for estimating the minimum and maximum of a one-dimensional multi-modal conditional distribution. Extremum Flow Matching allows selecting the parameter ϵ at inference and provides tighter estimates.

III. Method

A. Problem Formulation

Given a dataset of unstructured play demonstrations $\{(\boldsymbol{o}, \boldsymbol{\tau}^o, \boldsymbol{\tau}^a, d, \boldsymbol{g})_i\}$, our objective is to learn a policy $\pi : \boldsymbol{o}, \boldsymbol{g} \mapsto \boldsymbol{\tau}^a$ that maps the current observation $\boldsymbol{o}$ and goal observation $\boldsymbol{g}$ to the next actions $\boldsymbol{\tau}^a$, such that the agent progresses toward the goal. However, achieving optimality (see Fig. 2) is challenging because the dataset contains both efficient and inefficient trajectories: some action sequences lead directly to the goal, while others involve long detours. The distribution $\mathcal{P}(\boldsymbol{\tau}^a, d \,|\, \boldsymbol{o}, \boldsymbol{g})$ thus includes both optimal actions (associated with low d values, representing more direct paths) and suboptimal ones (associated with higher d values). The minimal distance between $\boldsymbol{o}$ and $\boldsymbol{g}$ within the dataset is given by $\min \mathcal{P}(d \,|\, \boldsymbol{o}, \boldsymbol{g})$. To follow such shortest paths, the policy should condition on this minimal distance and select actions from the corresponding conditional distribution: $\mathcal{P}(\boldsymbol{\tau}^a | \boldsymbol{o}, \boldsymbol{g}, d = \min \mathcal{P}(d | \boldsymbol{o}, \boldsymbol{g}))$. In this sense, the problem of optimality reduces to learning and approximating the minimum (or maximum, in a reward-based formulation) of a conditional distribution.

Recent offline reinforcement learning methods [15]–[17] have addressed this challenge of minimizing or maximizing conditional distributions using Expectile Regression [15], [36]. In this work, we propose an alternative approach based on Flow Matching, which enables the estimation of the lower and upper bounds of a conditional or unconditional distribution learned from offline data. Our method, along with recent work such as [18], [19], is inspired by the integration of reinforcement learning principles with conditioning on returns framework.

B. Extremum Flow Matching

One-Dimensional Distributions: Fig. 1 (bottom left) illustrates the core concept of our approach. Flow Matching learns a transformation that is both continuous and deterministic, governed by a vector field whose integrated paths do not intersect. As a result, in the one-dimensional case, the minimum and maximum of the source distribution's support are mapped to the minimum and maximum of the target distribution. Moreover, Flow Matching offers a key

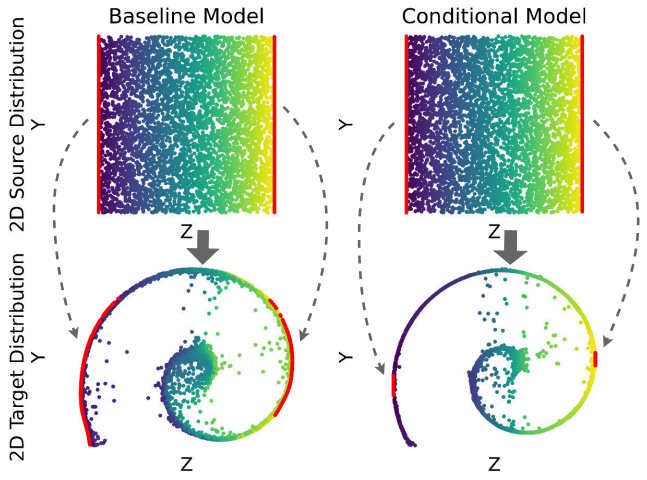


Figure 4. Extrema of unconditioned 2D distributions. Red samples illustrate how extremal values along the z axis are mapped from the uniform source distribution (top) to the target distribution (bottom). The baseline approach using a single Flow Matching model (left) is compared to the proposed conditional model (right) described in Equations (3) and (4). Proposed Extremum Flow Matching using a conditional model enables sampling the target distribution at the minimum and maximum of the z axis.

advantage over Diffusion-based methods: it allows the use of arbitrary source distributions, not just Gaussian ones. For this reason, we opt for a uniform distribution as the source which is not only easy to sample from but also provides a closed support with well-defined minimum and maximum bounds.

More formally, using the notation introduced earlier, a one-dimensional $x(\boldsymbol{c}) \in \mathbb{R}$ distribution conditioned on $\boldsymbol{c} \in \mathbb{R}^m$ is learned by training the Flow Matching model $F : \epsilon \sim \mathcal{U}(0,1) | \boldsymbol{c} \mapsto x(\boldsymbol{c})$, where $\mathcal{U}(0,1)$ is the one-dimensional uniform distribution between 0 and 1. The minimum and maximum bounds are then inferred as $F(\epsilon = 0|\boldsymbol{c})$ and $F(\epsilon = 1|\boldsymbol{c})$, respectively.

Fig. 3 shows a comparison between our Flow Matching approach and Expectile Regression on a one-dimensional conditional and multi-modal distribution. Expectile Regression estimates the bounds of the distribution by training the model g using an asymmetric supervised loss, defined as $\mathcal{L}^{\epsilon}_{\text{expectile}} = L^{\epsilon}_2(x - g(\boldsymbol{c}))$ where $L^{\epsilon}_2(u) = |\epsilon - \mathbb{1}(u < 0)|u^2$. Expectile Regression approximates the minimum and maximum of the target distribution, respectively by setting $\epsilon = 0.01$ and $\epsilon = 0.99$.

While both methods perform well, Extremum Flow Matching, in contrast to Expectile Regression, learns to model the entire distribution, allowing the choice of ϵ at inference time rather than fixing it during training. We observed that Flow Matching tends to provide tighter estimates of the distributional bounds. Expectile Regression, on the other hand, generally yield more conservative approximations of the minimum and maximum, especially on multi-modal distributions.

Multi-Dimensional Distributions: In the multi-dimensional setting where $\boldsymbol{x} \in \mathbb{R}^n$, we consider the problem of minimizing or maximizing the target distribution along a specific dimension $z \in \mathbb{R}$. We decompose the space as $\boldsymbol{x} = (z, \boldsymbol{y})$ with $z \in \mathbb{R}$ and $\boldsymbol{y} \in \mathbb{R}^{n-1}$. Formally, the objective is to sample from the conditional distribution $\mathcal{P}(z, \boldsymbol{y} | z = \max \mathcal{P}(z))$.

Since a distribution $\mathcal{P}$ can be decomposed as $\mathcal{P}(\boldsymbol{x}) = \mathcal{P}(z, \boldsymbol{y}) = \mathcal{P}(z)\mathcal{P}(\boldsymbol{y}|z)$, we propose to address the objective by training the following two models using Flow Matching:

$$F_1 : \mathcal{U}(0,1) \mapsto z, \quad F_2 : \mathcal{P}^{\text{src}} | z \mapsto \boldsymbol{y}, \tag{3}$$

where $\mathcal{P}^{\text{src}}$ is an arbitrary source distribution that is easy to sample from for $\boldsymbol{y} \in \mathbb{R}^{n-1}$.

At inference time using these two generative processes, a sample $\tilde{\boldsymbol{x}} = (\tilde{z}, \tilde{\boldsymbol{y}})$ corresponding to the minimization or maximization of the distribution along the z axis is obtained with:

$$\tilde{z} = F_1(0) \text{ or } F_1(1), \quad \tilde{\boldsymbol{y}} = F_2(\mathcal{P}^{\text{src}} | \tilde{z}). \tag{4}$$

This approach is illustrated in Fig. 4 (right). In contrast, the baseline with the single model $F(\mathcal{U}(0,1), \mathcal{U}(0,1)) \mapsto (z, y)$, shown on the left, demonstrates that without decomposing the generative process into two separate models, the extremal values along the z axis in the source distribution are not reliably mapped to the corresponding extrema in the target distribution.

C. Goal Conditioned Agents using Extremum Flow Matching

We propose to use Extremum Flow Matching to devise a family of goal conditioned imitation learning and offline reinforcement learning algorithms. As pointed out in [19], imitation learning and reinforcement learning agents are typically composed of multiple interacting components, which can generally be grouped into four main model types: Critic : $\bullet \mapsto d$ estimates the expected return (e.g., distance in time-step to goal) given an observation, goal, and/or action, Planner : $\bullet \mapsto \boldsymbol{\tau}^o$ [23]–[26] generates a single sub-goal or a trajectory of future observations that aims to reach the goal, Actor : $\bullet \mapsto \boldsymbol{\tau}^a$ produces a single action or a trajectory of actions, World : $\boldsymbol{\tau}^a, \bullet \mapsto \boldsymbol{\tau}^o$ (world model) [37], [38] predicts the environment's dynamics by generating future observations from a trajectory of actions and current observation.

Different algorithms make use of some, but not necessarily all, of these components. As shown in recent works [17], [19], [38], there are numerous possible combinations of these modules. However, the best performing configuration appears to be highly task and dataset dependent, and understanding the effect of these dependencies is still an open question. As noted by [17], a key advantage of decomposing the agent into modular components is that certain models, such as the Planner, can be trained without action labels, potentially enabling the use of large-scale datasets.

To further investigate and compare the influence of these different components, we introduce the set of agents described in Table I. Here, $\mathcal{P}^{\text{src}}_{\tau_a}$ and $\mathcal{P}^{\text{src}}_{\tau_o}$ denote arbitrary source distributions for action and observation trajectories, and $\boldsymbol{\tau}^o_{-1}$ refers to the last observation of the trajectory $\boldsymbol{\tau}^o$. Note that by setting the trajectory lengths L_a or L_o to 1, this formalism includes and generalizes a wide range of previously proposed algorithms that either predict only the immediate next action or plan for a single sub-goal.

The agent **FM-GC** is a simple imitation learning policy baseline trained using Flow Matching. While it is conditioned

Table I
PROPOSED FLOW MATCHING AGENTS

Name	Training	Inference	Comment
FM-GC	Actor: $\mathcal{P}^{\text{src}}_{\tau_a}\vert\boldsymbol{o},\boldsymbol{g}\mapsto\boldsymbol{\tau}^a$	$\tilde{\boldsymbol{\tau}}^a=\text{Actor}(\mathcal{P}^{\text{src}}_{\tau_a}\vert\boldsymbol{o},\boldsymbol{g})$	Baseline goal conditioned with Flow Matching
FM-AC	Critic: $\mathcal{U}(0,1)\vert\boldsymbol{o},\boldsymbol{g}\mapsto d$ Actor: $\mathcal{P}^{\text{src}}_{\tau_a}\vert\boldsymbol{o},\boldsymbol{g},d\mapsto\boldsymbol{\tau}^a$	$\tilde{d}=\text{Critic}(0\vert\boldsymbol{o},\boldsymbol{g})$ $\tilde{\boldsymbol{\tau}}^a=\text{Actor}(\mathcal{P}^{\text{src}}_{\tau_a}\vert\boldsymbol{o},\boldsymbol{g},\tilde{d})$	Actor conditioned, inspired by GCIQL [15], [17]
FM-PC	Critic: $\mathcal{U}(0,1)\vert\boldsymbol{o},\boldsymbol{g}\mapsto d$ Planner: $\mathcal{P}^{\text{src}}_{\tau_o}\vert\boldsymbol{o},\boldsymbol{g},d\mapsto\boldsymbol{\tau}^o$ Actor: $\mathcal{P}^{\text{src}}_{\tau_a}\vert\boldsymbol{o},\boldsymbol{\tau}^o\mapsto\boldsymbol{\tau}^a$	$\tilde{d}=\text{Critic}(0\vert\boldsymbol{o},\boldsymbol{g})$ $\tilde{\boldsymbol{\tau}}^o=\text{Planner}(\mathcal{P}^{\text{src}}_{\tau_o}\vert\boldsymbol{o},\boldsymbol{g},\tilde{d})$ $\tilde{\boldsymbol{\tau}}^a=\text{Actor}(\mathcal{P}^{\text{src}}_{\tau_a}\vert\boldsymbol{o},\tilde{\boldsymbol{\tau}}^o)$	Planner conditioned, inspired by HIQL [17]
FM-PS	Critic: $\mathcal{U}(0,1)\vert\boldsymbol{o},\boldsymbol{g}\mapsto d$ Planner: $\mathcal{P}^{\text{src}}_{\tau_o}\vert\boldsymbol{o}\mapsto\boldsymbol{\tau}^o$ Actor: $\mathcal{P}^{\text{src}}_{\tau_a}\vert\boldsymbol{o},\boldsymbol{\tau}^o\mapsto\boldsymbol{\tau}^a$	$T^o=\{\boldsymbol{\tau}^o\vert\boldsymbol{\tau}^o\sim\text{Planner}(\mathcal{P}^{\text{src}}_{\tau_o}\vert\boldsymbol{o})\}$ $\tilde{\boldsymbol{\tau}}^o=\underset{\boldsymbol{\tau}^o\in T^o}{\text{argmin}}\ \text{Critic}(0\vert\boldsymbol{\tau}^o_{-1},\boldsymbol{g})$ $\tilde{\boldsymbol{\tau}}^a=\text{Actor}(\mathcal{P}^{\text{src}}_{\tau_a}\vert\boldsymbol{o},\tilde{\boldsymbol{\tau}}^o)$	Planner rejection sampling, inspired by Diffusion Veteran [19]
FM-AS	Critic: $\mathcal{U}(0,1)\vert\boldsymbol{o},\boldsymbol{g}\mapsto d$ Actor: $\mathcal{P}^{\text{src}}_{\tau_a}\vert\boldsymbol{o}\mapsto\boldsymbol{\tau}^a$ World: $\mathcal{P}^{\text{src}}_{\tau_o}\vert\boldsymbol{o},\boldsymbol{\tau}^a\mapsto\boldsymbol{\tau}^o$	$T^a=\{\boldsymbol{\tau}^a\vert\boldsymbol{\tau}^a\sim\text{Actor}(\mathcal{P}^{\text{src}}_{\tau_a}\vert\boldsymbol{o})\}$ $\tilde{\boldsymbol{\tau}}^a=\underset{\boldsymbol{\tau}^a\in T^a}{\text{argmin}}\ \text{Critic}(0\vert\boldsymbol{\tau}^o_{-1},\boldsymbol{g})$ where $\boldsymbol{\tau}^o=\text{World}(\mathcal{P}^{\text{src}}_{\tau_o}\vert\boldsymbol{o},\boldsymbol{\tau}^a)$	Actor rejection sampling with world model

on the goal to generate action trajectories, it does not take the return d into account. As a result, it lacks the notion of optimality and is theoretically incapable of stitching together partial trajectories across episodes to reach long-horizon goals.

Both **FM-AC** and **FM-PC** use the conditional scheme introduced in Section III-B to estimate distribution extrema and address multi-dimensional distributions. They first train a Critic to model the distribution of returns d given the current observation and goal. The optimal return (i.e., shortest time-step distance to goal) is inferred via $\text{Critic}(0\vert\boldsymbol{o},\boldsymbol{g})$ and used to condition the Actor in **FM-AC**, or the Planner in **FM-PC**. In **FM-PC**, the Actor plays the role of an inverse dynamics model, generating actions that realize the Planner's predicted short-horizon observation trajectory.

These two agents represent Flow Matching counterparts to Return Conditioned Reinforcement Learning methods [14], [18], with a key distinction: prior approaches typically use hand-tuned, extreme return values (e.g., very low d or high reward) as conditioning inputs, without estimating true optimal return based on observation and goal. This often pushes the model into out-of-distribution regions, relying on the network's ability to extrapolate. For instance, in 2D maze tasks, overly aggressive return values can lead to unrealistic plans, such as crossing walls. In contrast, our method estimates returns directly from the distribution seen in the training dataset, ensuring that the values used for conditioning are in-distribution.

Instead of conditioning on the returns, agents **FM-PS** and **FM-AS** follow the rejection sampling strategy benchmarked in Diffusion Veteran [19]. In **FM-PS**, the Planner, not conditioned on the goal, generates a set T^o of candidate future observation trajectories from the current observation. At inference, the Critic evaluates the final observation $\boldsymbol{\tau}^o_{-1}$ of each candidate trajectory, and the one closest to the goal is selected. The Actor, trained as an inverse dynamics model, then produces the corresponding actions. Conversely in **FM-AS**, the Actor is used to samples a set T^a of candidate action trajectories conditioned only on the current observation. These are passed through a learned World Model to predict resulting observations, and the Critic selects the best plan based on proximity to the goal.

D. Reinforcement Learning Recursive Bootstrap

While conditioning on returns or using rejection sampling can theoretically address action optimality, these methods are insufficient for enabling agents to stitch together trajectory segments across episodes. An ability that is essential for solving long-horizon tasks that are not fully demonstrated in the dataset. For each Critic-based agent in Table I, we define two variants: **no-RL**, and **use-RL**. The **use-RL** variant augments the training batch using the following procedure: for each tuple $(\boldsymbol{o},\boldsymbol{\tau}^o,\boldsymbol{\tau}^a,d,\boldsymbol{g})$ from the batch, an observation $\boldsymbol{g'}$ is uniformly sampled from the dataset, and the augmented tuple is added to the batch:

$$\begin{aligned}&(\boldsymbol{o},\boldsymbol{\tau}^o,\boldsymbol{\tau}^a,d+\text{Critic}(\epsilon_g\vert\boldsymbol{g},\boldsymbol{g'}),\boldsymbol{g'}),\\&\text{with } \epsilon_g\sim\mathcal{U}(0,r_g),r_g\in[0,1],\end{aligned}\tag{5}$$

using the Critic model being trained. The term $d+\text{Critic}(\epsilon_g\vert\boldsymbol{g},\boldsymbol{g'}),\boldsymbol{g'})$ implements a Bellman-style backup by stitching together the returns of two trajectory segments: one from $\boldsymbol{o}$ to $\boldsymbol{g}$ with cost d, and another from $\boldsymbol{g}$ to $\boldsymbol{g'}$ with estimated return provided by the Critic. This effectively augments the return distribution with compositional trajectories, a structure that Flow Matching handles naturally. The additional goal $\boldsymbol{g'}$ can originate from a different episode [18] than $\boldsymbol{o}$ and $\boldsymbol{g}$, such that the Critic is trained to estimate the return between any pair of observations in the dataset. The scaling factor $r_g\in[0,1]$ serves as a regularization hyperparameter to mitigate underestimation bias (overestimation of rewards) by introducing suboptimal return estimates into the distribution preventing collapse. The training and inference pseudocode for agent **FM-AC-use-RL** using high-dimensional image observations is detailed in Algorithm 1. To further stabilize training, we also use the double networks trick [39] on the Critic model.

IV. EXPERIMENTAL RESULTS

A. Comparison with OGBench Benchmark

We first evaluated and compared our proposed agents across a diverse suite of simulated locomotion and manipulation envi-

		no-RL				use-RL				OGBench					
OGBench Dataset	**FM-GC**	**FM-AC**	**FM-PC**	**FM-PS**	**FM-AS**	**FM-AC**	**FM-PC**	**FM-PS**	**FM-AS**	**GCBC**	**GCIVL**	**GCIQL**	**QRL**	**CRL**	**HIQL**
pointmaze-large-navigate-v0	66	60	60	31	29	**89**	**89**	67	64	29	45	34	86	39	58
pointmaze-large-stitch-v0	39	37	23	15	14	40	40	42	44	7	12	31	**84**	0	13
antmaze-large-navigate-v0	7	5	5	15	1	7	22	34	15	24	16	34	75	83	**91**
antmaze-large-stitch-v0	1	0	0	6	3	0	3	18	7	3	18	7	18	11	**67**
cube-double-play-v0	**69**	32	13	22	14	2	1	12	16	1	36	40	1	10	6
scene-play-v0	53	52	32	42	43	7	16	40	**55**	5	42	51	5	19	38
puzzle-4x4-play-v0	1	0	3	22	**48**	0	1	14	38	0	13	26	0	0	7

Figure 5. Comparison of our proposed agents on the OGBench benchmark [40]. We report the average of binary success rates (%) across state-based observation datasets of intermediate difficulty. For each task, agents are evaluated on 5 goals, and replicated 20 times. We include for reference results from state-of-the-art algorithms reported in [40].

Algorithm 1 FM-AC-use-RL with image observations

Training:
  Input: Training dataset
  Initialize Encoder, Critic and Actor models
  while not converged **do**
    for $(\boldsymbol{o},\boldsymbol{\tau}^a,d,\boldsymbol{g})$ sampled from batch **do**
      Sample observation $\boldsymbol{g'}$ from the whole dataset
      Encode observations into latent spaces: $\boldsymbol{l}_o=\text{Encoder}(\boldsymbol{o})$, $\boldsymbol{l}_g=\text{Encoder}(\boldsymbol{g})$, $\boldsymbol{l}_{g'}=\text{Encoder}(\boldsymbol{g'})$
      Augment batch with $(\boldsymbol{l}_o,\boldsymbol{\tau}^a,d+\text{Critic}(\epsilon_g|\boldsymbol{l}_g,\boldsymbol{l}_{g'}),\boldsymbol{l}_{g'})$ (5)
    end for
    Update Encoder and Critic model to minimize $\mathcal{L}_{\text{flow}}$ (1) with $x^{\text{src}}=\mathcal{U}(0,1),x^{\text{dst}}=d,c=(\boldsymbol{l}_o,\boldsymbol{l}_g)$
    Update Encoder and Actor model to minimize $\mathcal{L}_{\text{flow}}$ (1) with $x^{\text{src}}=\mathcal{P}^{\text{src}},x^{\text{dst}}=\boldsymbol{\tau}^a,c=(\boldsymbol{l}_o,\boldsymbol{l}_g,d)$
  end while
  Return: Encoder, Critic and Actor models
Inference:
  Input: current $\boldsymbol{o}$ and goal $\boldsymbol{g}$ observations
  Encode observations into latent spaces: $\boldsymbol{l}_o=\text{Encoder}(\boldsymbol{o})$, $\boldsymbol{l}_g=\text{Encoder}(\boldsymbol{g})$
  $\tilde{d}=\text{Critic}(0|\boldsymbol{l}_o,\boldsymbol{l}_g)$ using flow integration (2)
  $\tilde{\boldsymbol{\tau}}^a=\text{Actor}(\mathcal{P}^{\text{src}}_{\tau_a}|\boldsymbol{l}_o,\boldsymbol{l}_g,\tilde{d})$ using flow integration (2)
  Return: action trajectory $\tilde{\boldsymbol{\tau}}^a$

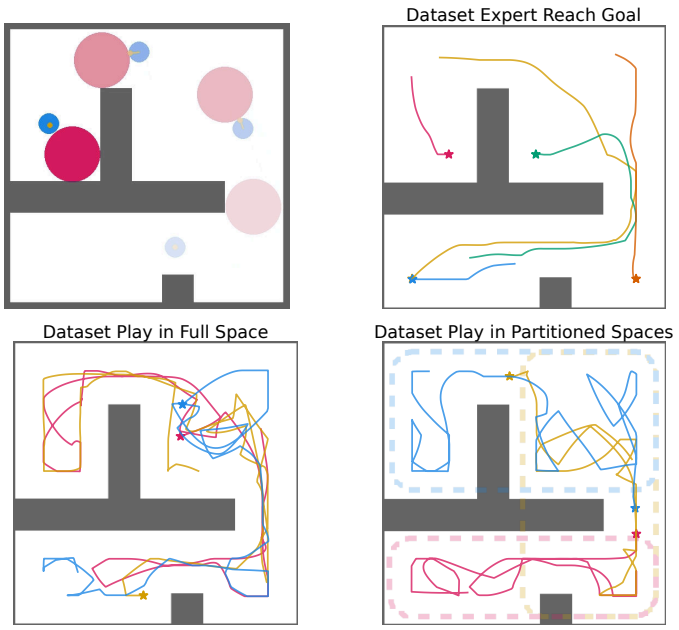


Figure 6. Planar pushing task in a maze (top left): the agent controls the small blue circle via position commands to move and push the larger passive red circle toward a corner of the maze, navigating around gray obstacle walls. Three datasets are recorded from human demonstrations exhibiting different behaviors. For each dataset, example episodes are shown, illustrating the motion trajectories of the red circle, with the star marker indicating the final position.

ronments using low-dimensional, state-based observations from the OGBench benchmark [40]. OGBench is a recent project specifically targeting offline reinforcement learning, where training and evaluation datasets highlight multi-goal, stitching and combinatorial challenges. OGBench provides reference implementation and comparison of several state-of-the art algorithms.

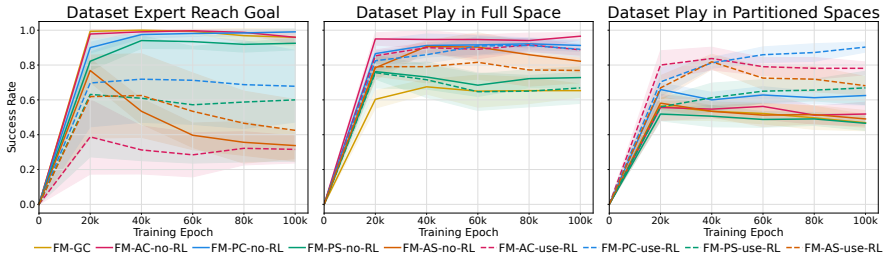
Fig. 5 presents the performance of our agents alongside several recent offline goal-conditioned reinforcement learning algorithms, comparing GCBC [41], GCIVL and GCIQL [15], [17], QRL [42], CRL [43], and HIQL [17]. Unlike our approach, these baselines do not leverage generative methods nor the trajectory-based formalism. All of our agents were trained for 200k epochs using the same set of hyperparameters across all tasks and agents. In contrast, the results reported from OGBench were obtained with hyperparameters being tuned for each task and baseline.

As observed in [40], no single method consistently outperforms all others across every task. Our results reflect this observation: while some of our agents achieve strong performance on manipulation tasks, such as *cube* and *puzzle* requiring combinatorial stitching, they are comparatively less effective in certain locomotion tasks, especially *antmaze* given fixed hypermarameters.

B. Impact of Demonstration Behaviors

To further investigate these benchmark results, we studied the impact on agent's performance of demonstration behavior, i.e. the strategy, style, or consistency with which humans performed the task during data collection. We compared our agents on the planar pushing task illustrated in Fig. 6 (top left), using three datasets collected from human demonstrations with distinct behaviors. All agents were trained with identical hyperparameters and evaluated on the same initial-goal observation pairs.

As shown in Fig. 6, each dataset reflect a different behavior: In *Expert Reach Goal* dataset (46793 samples), each episode demonstrates an optimal trajectory where the red circle is pushed from a random initial state to one of the maze corners, emphasizing goal directed expert behavior. In *Play in Full Space* dataset (36661 samples), longer, exploratory play episodes move freely the red circle throughout the maze without targeting specific goals, capturing diverse but suboptimal be-



Agent	Expert	Full	Partitioned
FM-GC	**96**±5	65±4	47±5
FM-AC-no-RL	**96**±4	**97**±2	52±7
FM-PC-no-RL	**99**±2	91±3	62±5
FM-PS-no-RL	92±4	73±6	47±4
FM-AS-no-RL	34±8	82±6	49±5
FM-AC-use-RL	32±8	89±3	78±4
FM-PC-use-RL	68±21	89±4	**90**±3
FM-PS-use-RL	60±35	67±9	67±10
FM-AS-use-RL	42±18	77±7	68±5

Figure 7. Success rates of proposed agents across three datasets with varying demonstration behaviors. Performance is evaluated on the same set of 10 initial-goal observation pairs, averaged over 8 random training seeds and 4 runs per evaluation pair. The plots (left) show the evolution of success rates during training, while the table (right) presents the final success rates with standard deviation.

haviors. In *Play in Partitioned Spaces* dataset (29087 samples), episodes also consist of exploratory play but are confined to one of the three maze regions, shown as dashed lines in Fig. 6 (bottom right), with no single trajectory spanning all regions. This dataset highlights agents' trajectory stitching abilities, as reaching distant evaluation goals requires crossing all three regions, a path that was never demonstrated in the dataset.

Evaluation results are shown in Fig 7. As expected, *Expert Reach Goal* is the easiest dataset, followed by *Play in Full Space* with sub-optimal demonstrations, and *Play in Partitioned Spaces*, which is the most difficult due to the need for trajectory stitching. No single agent consistently outperforms others across all settings, aligning with findings in [40]. Notably, using RL backups (Eq. 5) significantly degrades performance on the expert dataset, slightly reduces it on full-space play, but is crucial for solving partitioned-space play, where stitching is required. The underlying cause of this trade-off remains unclear and is a key direction for future research. On the expert dataset, simple goal-conditioned imitation with Flow Matching baseline performs comparably to more complex agents (without RL backups). However, in the more challenging play datasets, agents with Critics considering optimality outperform the baseline. The agent **FM-AS** using a world model performed poorly on the expert dataset regardless of whether RL backups were used. Its performance consistently degraded over the course of training, a phenomenon we plan to investigate further.

C. Vision-Based Manipulation with Talos Humanoid Robot

We further validated our method with high-dimensional image observations on real hardware using the Talos humanoid robot, featuring 24 degrees of freedom plus a gripper-equipped hand. Talos is controlled via SEIKO whole-body admittance controller[1] based on Quadratic Programming optimization [44]–[46], which receives Cartesian hand pose commands in the robot's world frame and computes joint position references while maintaining balance constraints.

The robot interacts with a realistic kitchen environment as shown in Fig 1 (bottom right), and Fig. 8 (left). Using one hand, the robot picks up and places a red pot, opens and closes an articulated door, and operates a pull-out tray. The pot is placed either on the table or on the tray. Observations include RGB images from a fixed external camera and proprioceptive

[1]SEIKO implementation: https://github.com/hucebot/seiko_controller_code

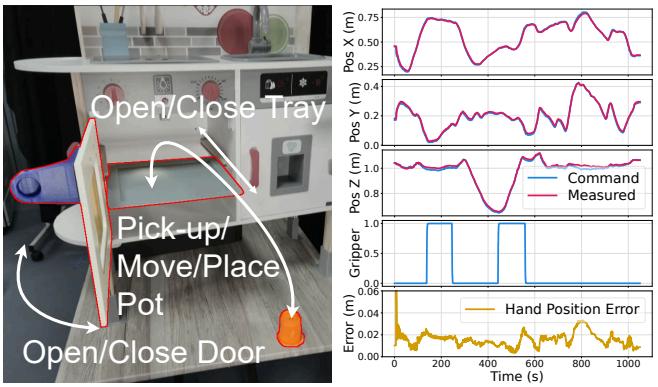


Figure 8. Kitchen task setup for image-based experiments on the Talos humanoid robot (left), and example of autonomous policy execution (right): left-hand position (commanded and measured), gripper command (open=0, close=1) and hand position error in world frame are displayed. The robot grasps and pulls open the tray, lowers its hand to grasp the pot on the table, places it on the tray and closes the tray by pushing it (see attached video).

data: measured and commanded hand positions and orientations. The pot remains visible from the camera even when inside the kitchen's cabinet and when the door is closed. Images are encoded using a ResNet-based small IMPALA encoder [47] trained from scratch. Hand orientations are represented using the 6D continuous representation [48]. The goal conditioned policy takes the current and goal observations as input, defining the desired state of the environment, and outputs a trajectory of future actions, including hand position/orientation and gripper commands. Demonstrations were collected via teleoperation using a Vive pose tracking device, covering 32 minutes over multiple episodes at 33Hz. Action trajectories are sub-sampled at 11Hz ($S_a = 3$) with a length of $L_a = 16$ steps, resulting in trajectories of 1.45s long. The policy runs on an external machine with an NVIDIA GTX 1080 GPU, achieving about 0.14ms inference time and recomputing trajectory updates every 0.8s.

During demonstrations, the human operator performed unstructured play, manipulating the environment without a fixed task or sequence, opening/closing the door and tray, and moving the pot freely. After training, the policy is able to reach specific visual goal observations from an initial observation. One such rollout is shown in Fig. 8 (right) and Fig. 9: the robot opens the tray, places the pot inside the kitchen, closes the tray, and then closes the door to match the provided goal

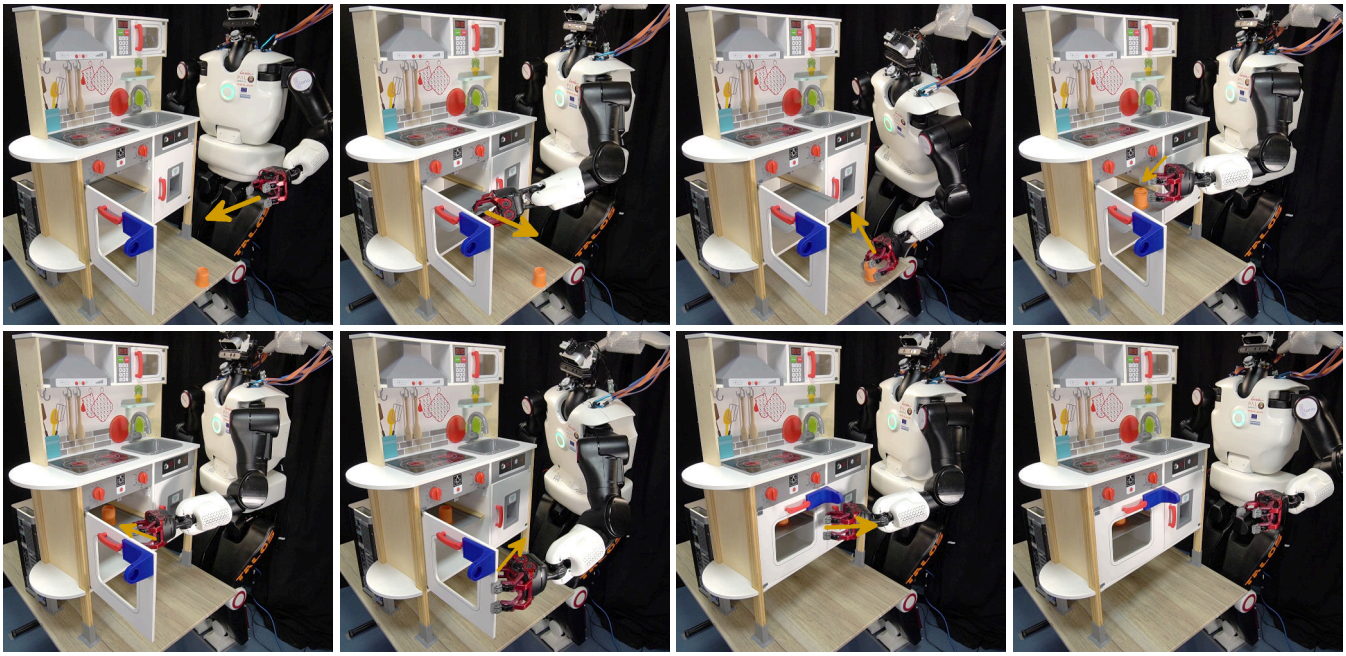


Figure 9. When conditioned on the current and desired goal images shown in Fig. 1 (top), the **FM-AC-no-RL** policy commands the Talos humanoid robot to sequentially open the tray, pick up the pot from the table, place it on the tray, close the tray, and shut the door to reach the goal.

image. We compared qualitatively the three agents **FM-GC**, **FM-AC-no-RL**, and **FM-AC-use-RL**. Consistent with the results in Section IV-B, **FM-AC-no-RL** demonstrated the best performance and was used to generate the accompanying videos[2]. Real-world success rate reaches 30%, primarily due to hand motion inaccuracies during critical phases like door manipulation and object grasping, which require sub-centimeter precision. The whole-body controller introduces slight hand positioning errors (see Fig. 8 right bottom row) due to closed-loop admittance control based on force-torque sensor feedback. As the foot force-torque sensors tend to drift over time, these errors are not always repeatable between demonstrations and evaluations. Although these effects are measurable and included in the proprioceptive observation, the policy does not generalize well and adapt given the limited demonstration data.

V. Limitations

Compared to algorithms such as GCBC, GCIVL, GCIQL, QRL, CRL, and HIQL implemented in OGBench, which do not rely on generative models, our agents require significantly more training time. This overhead is further increased when using the RL backup variants, as the batch augmentation procedure (Eq. 5) involves integrating the flow model to infer the Critic, adding substantial computational cost.

In our comparison, we evaluated all proposed agents using the same set of hyperparameters to enable a fair assessment of their robustness to parameter finetuning. However, we did not conduct a comprehensive exploration of hyperparameter sensitivity, which may influence the relative performance rankings on both the OGBench benchmark and the planar pushing task.

Regarding real-world deployment on the Talos humanoid, the policy was not able to generalize well to out of distribution cases, for instance, grasping the pot when it was positioned differently from the training demonstrations. This limitation is likely due in part to the small size of the training dataset, which constrained the policy's ability to learn robust generalization.

VI. Conclusion

We introduced a novel method, Extremum Flow Matching, designed to estimate the extrema of a multivariate probability distribution along a specific axis. This is made possible by the unique property of Flow Matching to utilize a uniform distribution as the source, combined with a conditioning scheme analogous to conditioning on returns. Building on this foundation, we proposed several imitation learning and offline reinforcement learning agents based on Flow Matching, varying in their internal architecture and module combinations.

Through simulation experiments, some of our agents outperformed state-of-the-art baselines on specific tasks within the OGBench benchmark, while underperforming on others, mirroring the task-specific variability seen in existing methods. Our findings reinforce that algorithm performance is highly sensitive to the data collection policy, as datasets can exhibit markedly different properties depending on the behavior used to generate them. Understanding and mitigating this sensitivity is crucial for developing more robust and generalizable agents capable of performing reliably across a diverse range of tasks.

We demonstrated the real-world applicability of our approach by deploying the proposed goal-conditioned agents on the

[2]Project webpage: https://hucebot.github.io/extremum_flow_matching_website/

full-size Talos humanoid robot. The learned vision-based policy interfaces with a low-level, model-based whole-body controller, enabling the robot to perform complex manipulations in a kitchen environment. Notably, the policy was trained using unstructured, teleoperated play demonstrations. This showcases the potential of leveraging large-scale, suboptimal datasets to train more generalist humanoid robots.

REFERENCES

[1] P. Florence *et al.*, "Implicit behavioral cloning," in *CoRL*, 2022.

[2] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *IJRR*, 2023.

[3] K. Black *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[4] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, "Goal-conditioned imitation learning," *NeurIPS*, 2019.

[5] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-conditioned imitation learning using score-based diffusion policies," *RSS*, 2023.

[6] Z. J. Cui *et al.*, "From play to policy: Conditional behavior generation from uncurated robot data," *ICLR*, 2023.

[7] A. Sridhar, D. Shah, C. Glossop, and S. Levine, "Nomad: Goal masked diffusion policies for navigation and exploration," in *IEEE ICRA*, 2024.

[8] M. Reuss *et al.*, "Multimodal diffusion transformer: Learning versatile behavior from multimodal goals," *RSS*, 2024.

[9] L. Cao, "Ai robots and humanoid ai: Review, perspectives and directions," *arXiv preprint arXiv:2405.15775*, 2024.

[10] Y. Tong, H. Liu, and Z. Zhang, "Advancements in humanoid robots: A comprehensive review and future prospects," *IEEE/CAA Journal of Automatica Sinica*, vol. 11, no. 2, pp. 301–328, 2024.

[11] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *CoRL*, 2020.

[12] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE RA-L*, 2022.

[13] C. Wang *et al.*, "Mimicplay: Long-horizon imitation learning by watching human play," *CoRL*, 2023.

[14] A. Ajay *et al.*, "Is conditional generative modeling all you need for decision-making?" *ICLR*, 2023.

[15] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *ICLR*, 2021.

[16] Z. Liu *et al.*, "Enhancing decision transformer with diffusion-based trajectory branch generation," *arXiv preprint arXiv:2411.11327*, 2024.

[17] S. Park, D. Ghosh, B. Eysenbach, and S. Levine, "Hiql: Offline goal-conditioned rl with latent states as actions," *NeurIPS*, 2023.

[18] S. Kim *et al.*, "Stitching sub-trajectories with conditional diffusion model for goal-conditioned offline rl," in *AAAI*, 2024.

[19] H. Lu, D. Han, Y. Shen, and D. Li, "What makes a good diffusion planner for decision making?" in *ICLR*, 2025.

[20] J. Ho *et al.*, "Denoising diffusion probabilistic models," in *NeurIPS*, 2020.

[21] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *ICLR*, 2021.

[22] X. Liu, C. Gong, and qiang liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *ICLR*, 2023.

[23] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *ICML*, 2022.

[24] U. A. Mishra, S. Xue, Y. Chen, and D. Xu, "Generative skill chaining: Long-horizon skill planning with diffusion models," in *CoRL*, 2023.

[25] C. Chen, F. Deng, K. Kawaguchi, C. Gulcehre, and S. Ahn, "Simple hierarchical planning with diffusion," *ICLR*, 2024.

[26] Y. Luo, U. A. Mishra, Y. Du, and D. Xu, "Generative trajectory stitching through diffusion composition," *arXiv preprint arXiv:2503.05153*, 2025.

[27] M. S. Albergo *et al.*, "Stochastic interpolants: A unifying framework for flows and diffusions," *arXiv preprint arXiv:2303.08797*, 2023.

[28] M. S. Albergo and E. Vanden-Eijnden, "Building normalizing flows with stochastic interpolants," in *ICLR*, 2023.

[29] Q. Rouxel *et al.*, "Flow matching imitation learning for multi-support manipulation," in *IEEE-RAS Humanoids*, 2024.

[30] X. Hu, Q. Liu, X. Liu, and B. Liu, "Adaflow: Imitation learning with variance-adaptive flow-based policies," *NeurIPS*, 2024.

[31] M. Braun, N. Jaquier, L. Rozo, and T. Asfour, "Riemannian flow matching policy for robot motion learning," in *IEEE/RSJ IROS*, 2024.

[32] Q. Zhang, Z. Liu, H. Fan, G. Liu, B. Zeng, and S. Liu, "Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation," in *AAAI*, 2025.

[33] S. Zhang, W. Zhang, and Q. Gu, "Energy-weighted flow matching for offline reinforcement learning," *arXiv preprint arXiv:2503.04975*, 2025.

[34] S. Park, Q. Li, and S. Levine, "Flow q-learning," *arXiv preprint arXiv:2502.02538*, 2025.

[35] M. Andrychowicz *et al.*, "Hindsight experience replay," *NeurIPS*, 2017.

[36] W. K. Newey and J. L. Powell, "Asymmetric least squares estimation and testing," *Econometrica: Journal of the Econometric Society*, 1987.

[37] Z. Ding, A. Zhang, Y. Tian, and Q. Zheng, "Diffusion world model: Future modeling beyond step-by-step rollout for offline reinforcement learning," *arXiv preprint arXiv:2402.03570*, 2024.

[38] V. Sobal, W. Zhang, K. Cho, R. Balestriero, T. G. Rudner, and Y. LeCun, "Learning from reward-free offline data: A case for planning with latent dynamics models," *arXiv preprint arXiv:2502.14819*, 2025.

[39] H. Hasselt, "Double q-learning," *NeurIPS*, 2010.

[40] S. Park, K. Frans, B. Eysenbach, and S. Levine, "Ogbench: Benchmarking offline goal-conditioned rl," in *ICLR*, 2025.

[41] D. Ghosh *et al.*, "Learning to reach goals via iterated supervised learning," *ICLR*, 2021.

[42] T. Wang, A. Torralba, P. Isola, and A. Zhang, "Optimal goal-reaching reinforcement learning via quasimetric learning," in *ICML*, 2023.

[43] B. Eysenbach *et al.*, "Contrastive learning as goal-conditioned reinforcement learning," *NeurIPS*, 2022.

[44] Q. Rouxel, S. Ivaldi, and J.-B. Mouret, "Multi-contact whole-body force control for position-controlled robots," *IEEE RA-L*, 2024.

[45] Q. Rouxel, K. Yuan, R. Wen, and Z. Li, "Multicontact motion retargeting using whole-body optimization of full kinematics and sequential force equilibrium," *Trans. on Mechatronics*, vol. 27, no. 5, 2022.

[46] R. Wen, Q. Rouxel, M. Mistry, Z. Li, and C. Tiseo, "Collaborative bimanual manipulation using optimal motion adaptation and interaction control," *IEEE RAM*, 2023.

[47] L. Espeholt *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *ICML*, 2018.

[48] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *IEEE/CVF CVPR*, 2019.