# HIERARCHICAL REINFORCEMENT LEARNING FOR AERIAL VEHICLES

Harsh Goel

A DISSERTATION

in

Robotics

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Masters of Science in Engineering in Robotics

2023

Supervisor of Dissertation

Dr. Vijay Kumar, Professor and Nemirovsky Family Dean

Graduate Group Chairperson

Dr. M. Ani Hsieh, Deputy Director, GRASP Lab and Associate Professor, MEAM

HIERARCHICAL REINFORCEMENT LEARNING FOR AERIAL VEHICLES

# ACKNOWLEDGEMENT

I want to express my heartfelt gratitude to the many individuals who have supported and contributed to the completion of this thesis.

First and foremost, I am deeply grateful to Dr. Vijay Kumar for his invaluable guidance, encouragement, and support throughout my master's thesis. I would also like to sincerely thank my Ph.D. mentors, Edward Hu, Pratik Kunapuli, and Laura Jarin Lipschitz, whose insights have enriched my research experience. Their willingness to share their expertise has been immensely beneficial.

My appreciation goes out to my friends and family for their unwavering encouragement, understanding, and love. Their constant support and belief in me have been a source of strength and motivation during challenging times.

I am also grateful to the faculty members and staff, especially Dr. Pratik Chaudhari and Dr. Ani Hsieh, who have contributed to my academic and personal development.

To all those who have been a part of my academic journey, thank you for your unwavering support and encouragement. Your contributions have played a significant role in the successful completion of this thesis.

# ABSTRACT


HIERARCHICAL REINFORCEMENT LEARNING FOR AERIAL VEHICLES

Harsh Goel

Dr. Vijay Kumar

Aerial vehicles, including drones and unmanned aerial vehicles (UAVs), have witnessed remarkable advancements in recent years, particularly in agriculture, delivery services, surveillance, and disaster relief. However, achieving effective integrated trajectory planning and control of aerial vehicles remains a significant challenge. While end-to-end control of aerial vehicles for specific tasks is computationally intractable, hierarchical formulations, such as high-level task planners using motion primitives, offer promising solutions.

We focus on the problem of hierarchical control and planning for aerial vehicles. We first address the planning problem, where an aerial vehicle learns to compose a sequence of primitives for long-horizon planning tasks. Subsequently, we tackle the problem of learning primitives or skills for aerial vehicles from offline collected data, eliminating the reliance on handcrafted skills with inherent design limitations.

In this thesis, we contribute to the advancement of hierarchical reinforcement learning for aerial vehicles, offering an improved approach to enhance planning and control. The combination of high-level policies with motion primitives and goal-conditioned policies paves the way for more efficient, adaptable, and real-time task execution in complex environments.

TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF ILLUSTRATIONS

# CHAPTER 1

# INTRODUCTION

## 1.1. **Introduction**

In recent years, aerial vehicles, including drones and unmanned aerial vehicles (UAVs), have undergone rapid advancements, transforming various industries and expanding their applications to fields such as agriculture, delivery services, surveillance, and disaster relief. A critical challenge in maximizing the utility of aerial vehicles lies in task-specific integrated trajectory planning and control. While achieving end-to-end control for task completion is computationally intractable, prior research has focused on hierarchical formulations, such as motion primitives, to enable feasible planning.

This is in some ways similar to human intelligence whereby complicated general sets of tasks are accomplished by humans using abstracted low-level skills, goals, or primitives. For instance, a person walking doesn't plan and control each muscle movement. These skills and primitives are often learned through trial and error, or through demonstrations, and humans have a remarkable capability to compose these primitives in sequence to complete tasks. Drawing inspiration from human intelligence, this thesis centers on the problem of hierarchical control and planning for aerial vehicles. Specifically, it explores two key aspects: 1) addressing the planning problem by enabling an aerial vehicle to compose a sequence of primitives for long-horizon planning tasks, and 2) learning primitives or skills for aerial vehicles from offline data, moving away from handcrafted variants that are limited by design choices.

Conventional planning and control pipelines for quadrotors in various tasks are often decoupled for computational efficiency, particularly during the thinking phase of the robot. For example, in simple point-to-point locomotion in free space or constrained environments, waypoints are first selected, followed by the optimization of a reference trajectory, and ultimately, the development of a controller. However, this hierarchical construction becomes problematic in scenarios where

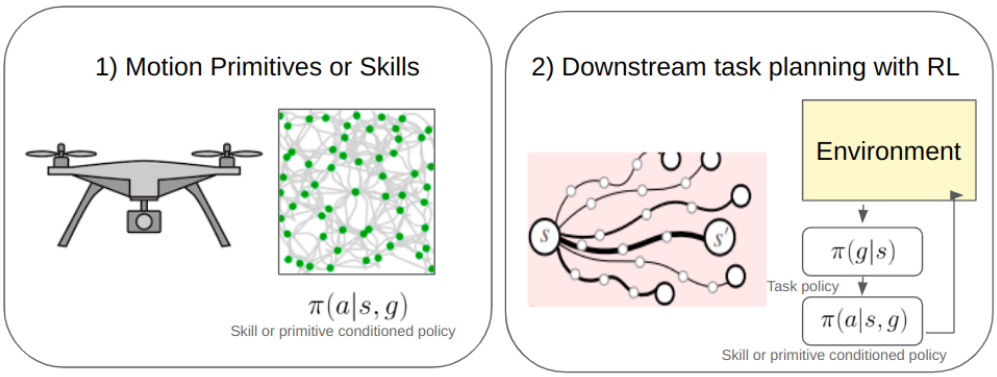


Figure 1.1: **Overview of Methods.** We are interested in the problem of hierarchical reinforcement learning (RL) for aerial robots. In this approach, the agent learns a task-specific high-level policy that selects goals, skills, or primitives. The agent then uses a low-level skill or goal-conditioned policy to interact with the environment. This thesis proposes learning a high-level task policy for an informative search problem for aerial vehicles, assuming access to engineered primitives. Additionally, we aim to learn a lower-level goal-conditioned policy from offline data to distill motion primitives or skills.

aerial vehicles must navigate narrow corridors or tunnels for inspection, mapping, or environmental monitoring. In such tasks, a decoupled planning scheme as above would have to go through multiple iterations to give good trajectories. Hence, the need for a tractable, real-time, and adaptable solution arises especially in un-modelled emergent situations.

Motion primitives for aerial vehicles do enable such adaptability whereby we can embed low-level actuator controls to a set of *robot motions or skills*. These motions or skills are embedded with the kino-dynamic constraints of a robot, allowing us to bridge the gap between higher-level task planning and actual control in a computationally tractable manner. Optimal task plans can initially be computed using search-based planning with domain-specific heuristics for both online and offline planning.

However, engineering heuristics for many multi-objective problems, such as informative search, can be very challenging. Focusing on the problem of informative search, we present a method to compose informative trajectories fully online for targets and semantic features. In this task, we pre-compute a set of motion primitives offline based on optimally dispersed states to form a planning graph, which

can be spatially extended for long planning tasks. We then devise a framework to learn a policy using reinforcement learning (RL) with these motion primitives as the action space for computing informative paths. Unlike prior heuristic-based informative search algorithms, our approach is trained, performant, and robust in scenarios where the prior over the search targets or semantics doesn't match the actual placement of the targets or semantics (exploration vs. exploitation). We show that the RL-based planner learns to balance between exploiting the prior and exploring the environment to search and identify targets or semantic features.

However, the motion primitives compiled for the above informative search problem suffer from limitations and are restricted to fixed goal states sampled during offline compilation. To address these limitations, we further propose a method to acquire lower-level locomotion skills or primitives for quadrotors using goal-conditioned deep reinforcement learning. We present a method to distill expert data consisting of motor thrust commands or desired thrust and angular velocity commands, to a single goal-conditioned policy. This policy takes the current state and desired goal state of the quadrotor as input and produces low-level control commands to reach the goal state with high probability. We devise and propose experiments to demonstrate the quality of the learned skills. Furthermore, we show an application of the learned skills especially when applied in trajectory composition for an untrained out-of-distribution goal through planning a single intermediate controller-feasible goal.

In summary, this thesis focuses on hierarchical control and planning for aerial vehicles. We first address the planning problem through the learning of high-level planning policies on motion primitives for an informative path planning task. Then we learn to improve over these primitives by learning low-level goal-conditioned policies. Extensive experiments demonstrate the efficacy and quality of the learned goal-conditioned skills and learned high-level policies showcasing their potential for trajectory composition for complex tasks such as informative search.

## 1.2. Background

In this section, we outline the basic concepts in the literature on hierarchical reinforcement learning (HRL). HRL aims to resolve long-horizon tasks into a sequence of sub-goals or skills that are easier to achieve. In this framework, a high-level controller learns to choose sub-goals or skills, which then provide goal or skill commands to a lower-level controller that selects atomic control actions. Typically, high-level controllers operate at larger timescales compared to lower-level controllers. The low-level controller or skills can be learned offline through demonstration or online through interactions, while the high-level controller is learned online. This setting is also called the options framework [75] where an option can be defined as a skill that temporally abstracts a lower-level control policy $\pi : \mathbf{S} \times \mathbf{A} \rightarrow [0, 1]$ where $\mathbf{S}$ and $\mathbf{A}$ are low-level states and actions for a given agent pose and motor thrusts for UAVs. We formalize the framework as follows:

### 1.2.1. Partially Observable Semi Markov Decision Process

A Partially Observable Semi Markov Decision Process can be described as a tuple $(\mathcal{S}, \mathcal{Z}, \mathcal{O}, T, R, O, \gamma, d_0)$, where $\mathcal{S}$ and $\mathcal{O}$ denote the state space, and observation space respectively. $R$ and $O$ are the reward and observation models, $\mathcal{T}$ is a transition dynamics model, $d_0$ is the initial state distribution, and $\gamma$ is the discount factor. $\mathcal{Z}$ represents a finite set of temporally extended skills or primitives that act in the environment for a maximum horizon of $\tau$ timesteps. The task reward is denoted by $R(s_t, z_t)$ for a skill $z_t \in \mathcal{Z}$ at state $s_t \in \mathcal{S}$. The agent transitions to the next state $s_{t+1}$ after executing a skill $z_t$ through a transition function $T(\cdot \mid z_t, s_t)$.

The high-level policy $\pi(z_t|s_t)$ being learned does not have access to the true state $s_t$ and instead has access to an observation $o_t \sim O(\cdot \mid s_t)$. The policy can be stochastic or deterministic (one-hot) distribution over the skill space $\mathcal{Z}$. The objective is to maximize the cumulative expected return from a trajectory sampled by unrolling the policy $\pi$ from an initial state $s_0 \sim d_0$ as follows:

$$J(\pi) = \mathop{\mathbb{E}}_{\substack{s_0 \sim d_0 \; z_t \sim \pi(\cdot|o_t) \\ o_t \sim \mathcal{O}(\cdot|s_t) \; s_{t+1} \sim \mathcal{T}(\cdot|z_t, s_t)}} \left[ \sum_{t=0}^{T} \gamma^t \, R(s_t, z_t) \right] \tag{1.1}$$

### 1.2.2. Deep Reinforcement Learning

Deep reinforcement learning aims to maximize the cumulative return $J(\pi)$ under a policy is parameterized by a deep neural network $\theta$ as follows $\pi_\theta$.

$$\theta^* = \underset{\theta}{\text{argmax}}\ J(\pi_\theta) \tag{1.2}$$

The policy is optimized by collecting data from many interactions in the environment. Many algorithms exist for optimizing the returns over the policy parameters and we refer the reader to Arulkumaran et. al [2]. In our setting, we use the Asynchronous Actor-Critic (A3C) [44] framework to optimize a policy over primitives as described in the next section.

### 1.2.3. Skill Learning

While motion primitives have been quite successfully deployed on real robotic platforms such as quadrotors, many of such constructions are often limited to carefully engineered discretization [36, 37] or sampling [10, 26, 27] of state or control spaces. However, these methods lack the flexibility required for most real-life settings due to their inherent fixed time durations, discretizations, or sampling constraints.

Recent methods have been devised that acquire these primitives for other robotic systems with complex dynamics such as robotic arms whereby complex behaviors or skills are distilled from expert play data [38], expert demonstrations [58, 69, 50, 79, 71, 78, 66, 40, 6] or online robot interactions [67, 68, 12, 46, 21]. Here, the skills are typically represented in a normalized latent space [58, 66, 67, 12, 69, 50, 84, 87], or through desired states or goals [40, 21, 46, 7, 92]. Typically, latent skills or goals are used to condition a separate policy network during the learning process. Such skills can then further be composed sequentially using a high-level planner over skill space such as MPPI [86], CEM [59] or another RL policy [69, 21, 46, 8] to hierarchically complete tasks. Here, skills serve as temporally extended actions that encode low-level control behaviors to reach short-term goals or states, and higher-level planners plan on such skills to identify intermediate goals for an agent to complete to accomplish these tasks.

Traditional planning and control stacks for quadrotors have typically relied on handcrafted skills or motion primitives and in this chapter, we attempt to devise such primitives from quadrotor data. The data is assumed to be unimodal and is collected from a single setting of an expert behavioral policy, and this data is distilled into a single goal-reaching policy.

### 1.2.4. Goal Conditioned Reinforcement Learning

Standard RL involves a policy optimizing for cumulative task-specific rewards. However, for some tasks specifying a reward structure can be quite challenging. Goal-conditioned RL on the other hand augments the underlying MDP of the RL problem by specifying tasks as desired goals which are then parsed with the observation while making a decision [1, 64]. In GCRL, the policy optimizes a sparse reward obtained on reaching the goal within a specified margin. This creates additional problems where *distant* to reach goals from a task perspective might not be achievable if the policy fails to collect data over the necessary states required to reach those goals during its exploration phase [56, 32, 35]. To alleviate this problem, researchers have proposed a set of tools to ensure that the goal-conditioned policy learns to reach the necessary subgoals to achieve the task-specific goal. The underlying training procedure for GCRL usually commands desired task-specific goals for a policy to collect initial data on the achieved states and goals by the policy. The task-specific goals are then relabelled with these achieved goals to sample additional trajectories and desired goals [1, 15, 89, 51, 52, 9, 56, 25, 43]. Typical goal selection mechanisms include random relabelling of achieved goals from future trajectories through online interactions [1, 15] or imagination [89]. Alternatively, rarely visited goals can be commanded to reach diverse and unexplored regions of the state space in a model free[52, 51] or model-based [25, 43]. In GCRL, the goal-conditioned policy can also be iteratively trained with supervised learning (behavior cloning) [11, 19, 90], an off-the-shelf online RL algorithm [1, 15, 89, 51, 52, 9, 25, 43] or contrastive learning [14, 13]. Succinctly, the goal-conditioned RL objective can be written as:

$$\pi^* = \underset{\pi}{argmax} \underset{s_0 \sim \mu(s)\ s_t \sim \mathcal{T}(.|s_t,a_t)\ g \sim \rho(g)\ a_t \sim \pi(.|s_t,g)}{\mathbb{E}} \left[ \sum_{t=0}^{T} \gamma^t\ r_g(s_t, a_t, g) \right]$$

where $\mu(s)$ is the initial state distribution, $\mathcal{T}$ is the transition function, $\rho(g)$ is the distribution of

goals, $\pi$ is the goal conditioned policy and $r_g(.\,,.\,,.)$ is the goal conditioned reward.

### 1.2.5. Offline Reinforcement Learning

In this work, we use offline RL algorithms to distill goal-conditioned skills from a fixed dataset. Offline RL algorithms are a recent development in the field of robotics where online data collection on a robot can be expensive and difficult. This necessitates pre-training from fixed datasets that are possibly obtained from a mixture of expert policies (multiple human demonstration data). Hence, this paradigm generally penalizes out-of-distribution actions through a supervised learning loss [88, 17] or minimizing the optimality of the out-of-distribution actions through their Q functions [31]. On the other hand, implicit Q-learning [29] modifies the Bellman optimality update towards a SARSA-like update, maximizing only over actions in the dataset. Most offline RL algorithms have similar rationales of ensuring that the state action occupancy of the policy is equivalent to that of the dataset. Goal-conditioned extensions to offline RL follow similar rationales [39, 91].

## 1.3. Thesis Organisation

The thesis is structured into three chapters, each addressing different aspects of hierarchical planning and control for quadrotors. Here's an improved version of the organization description with enhanced clarity:

1. The first chapter provides a high-level overview of the problem of hierarchical planning and control for quadrotors. It lays the foundation for the research by introducing the challenges associated with task-specific integrated trajectory planning and control in aerial vehicles. The chapter also presents the background of hierarchical reinforcement learning, focusing on how skills or goals can be utilized for task planning, and explores goal-conditioned reinforcement learning for skill acquisition.

2. In Chapter 2, the focus is on addressing the problem of informative search in aerial vehicles using motion primitives. The chapter delves into the methodological approach of planning informative trajectories fully online for targets and semantic features. It explains the reinforcement learning framework used to compute informative paths using pre-computed motion primitives. The chapter highlights the advantages of this approach over existing heuristic-based algorithms for informative search.

3. Chapter 3 proposes a learning-based framework for acquiring primitives or skills in the form of a goal-conditioned policy. It presents a method to distill expert data, consisting of motor thrust commands or desired thrust and angular velocity commands, into a single goal-conditioned policy. This policy takes the current state and desired goal state of the quadrotor as input and produces low-level control commands to reach the goal state with high probability. The chapter showcases the quality of the learned skills through extensive experiments conducted on the goal-conditioned policy.

# CHAPTER 2

# ADAPTIVE INFORMATIVE PATH PLANNING WITH REINFORCEMENT LEARNING ON MOTION PRIMITIVES

## 2.1. **Abstract**

Adaptive informative path planning is the task of computing informative trajectories for an agent to discover semantic features of interest in an environment while exploiting its current belief of the feature states. This paper develops a deep reinforcement learning approach to plan informative trajectories that increase the likelihood for an uncrewed aerial vehicle (UAV) to discover features. Our approach efficiently (1) explores the environment to discover new semantic features, (2) exploits its current belief of the feature states and incorporates inaccurate sensor models for high-fidelity classification, and (3) generates dynamically feasible UAV trajectories using a motion primitive library. We perform extensive simulations on a randomly generated dataset for feature detection and a second dataset of real-world semantic maps. The results show that our DRL approach is more efficient in discovering features than several other baselines. A unique characteristic of our approach is that, in contrast to heuristic informative path planning approaches, it is robust to varying amounts of deviations of the prior belief from the true feature distribution, thereby alleviating the challenge of designing heuristics specific to the application conditions.

## 2.2. Introduction

Uncrewed aerial vehicles (UAVs) equipped with sensors play a vital role in inspection and monitoring tasks, particularly in the aftermath of natural disasters. One compelling application involves deploying UAVs in search and rescue scenarios after earthquakes to locate trapped individuals and collapsed structures. In such situations, an operator might possess a prior belief regarding important "semantic features of interest" based on past data, albeit incomplete and imprecise. This raises a crucial question: How can we plan UAV trajectories to gather essential information while considering the prior belief of semantic features and respecting the UAV's dynamics and energy budget? This problem belongs to the broad class of NP-hard informative path planning (IPP) problems [30], and in this chapter, we tackle it using a reinforcement learning approach.

This paper focuses on two critical challenges in the IPP problems. First, existing online IPP methods used for target detection [70, 53, 41] or more broadly semantic mapping [62, 72, 33] relax the constraint of planning dynamically feasible trajectories in real-time. It is essential to ensure that the UAV's trajectory adheres to its dynamics and energy budget for a feasible, smooth, and efficient traversal in the environment.

Secondly, our proposed approach incorporates the prior belief of the semantic features of interest, which sets it apart from many existing methods that assume a uniform distribution. Furthermore, the prior belief is generally incomplete and imprecise. Hence, trajectory planning requires a balance between classifying highly uncertain regions and improving estimates to confirm the existence of features of interest. This leads to the classical trade-off between *exploration* and *exploitation* [70] problem that is compounded with sensing uncertainty for which designing good heuristics for informative path planning algorithms is challenging.

To address these challenges, this paper develops a hierarchical reinforcement learning (HRL) framework for the IPP problem. We augment the framework with dynamically feasible motion primitives that abstract the lower-level control and trajectory optimization for quadrotors [42, 36]. This hierarchical structure draws inspiration from the options framework [55] enabling the learning of

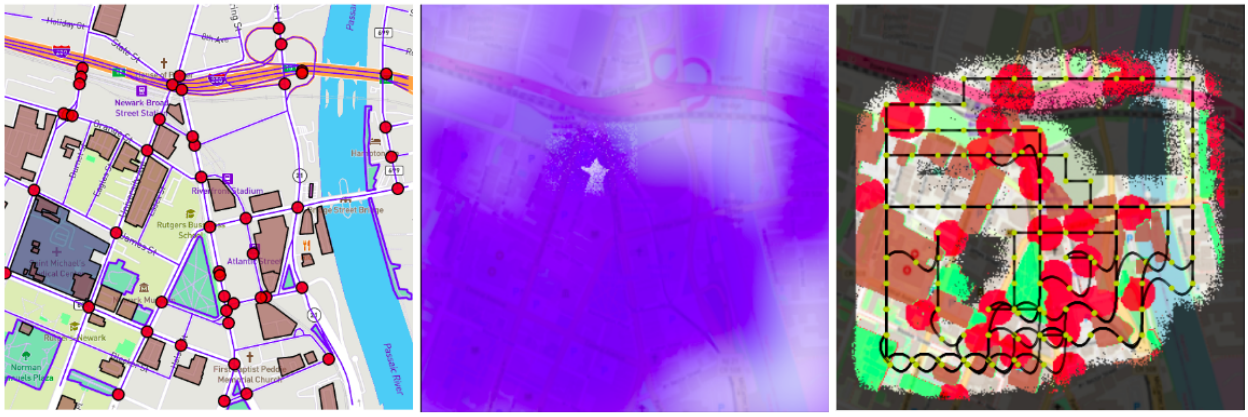

Figure 2.1: **Sample Task Setting and resulting trajectory.** Quadrotor in a semantic detection task, the image on the left shows a site with ground truth semantic features. The image in the middle is the prior belief with darker blue regions corresponding to the probability of occurrence of semantic feature, and the image on the right is the path taken and the corresponding detected semantics after the mission.

policies that plan feasible trajectories in real-time. This is in contrast to computationally prohibitive end-to-end RL methods that predict and learn the control actions for the agent. In our approach, we utilize a semi-Markov decision process (SMDP) to effectively model the environment and its evolving dynamics. The SMDP framework allows us to incorporate the temporal aspects of the environment, which is crucial for informative path planning in complex scenarios.

In our approach, we utilize a semi-Markov decision process (SMDP) [77] to effectively model the environment and its evolving dynamics. The SMDP framework allows us to incorporate the temporal abstraction aspects of the environment where the UAV executes its motion primitives and continuously updates and refines the beliefs over semantics features or targets in the environment. Sensing information obtained during the execution of each motion primitive contributes to this dynamic model update. We learn a task-level policy on these primitives and through extensive evaluation, we show that our learned policies permit robust decision-making that trades off between exploration and exploitation arising due to incomplete and imprecise priors and measurement updates.

## 2.3. Related Work

Informative path planning algorithms have been broadly applied for many active sensing problems such as target search [53, 41, 81], environmental monitoring [45, 60], inspection [24] and mapping [62, 72, 54, 65]. We present a brief background on literature for informative path planning and its applications to target search and semantic detection.

IPP algorithms for these applications can be broadly classified into five categories: coverage, combinatorial, sampling, optimization, and learning-based. Coverage planners [72, 18] are widely used IPP algorithms that follow pre-specified paths to cover the entire environment. Coverage planners follow pre-specified paths to cover the entire environment but often lack optimization for information gain, leading to suboptimal performance[62, 60, 4]. On the other hand, combinatorial methods [70, 73, 3] provably plan non-myopic informative paths. However, they can potentially query an exponentially large search space to obtain feasible trajectories and are computationally inefficient for online re-planning.

Sampling and optimization-based approaches improve over combinatorial-based planners. These methods typically begin by sampling waypoints greedily [53, 41, 62, 54], or sampling waypoints to build a tree [45, 65, 23, 82] or control actions [82] to compute candidate paths. These paths are then optimized over an information-theoretic measure, such as entropy, and have shown to be probabilistically optimal. However, the optimization routine which forward propagates many simulated measurements across the sampled candidate paths to compute this information-theoretic utility is unsuitable for online planning during agile flights. To make online planning feasible, greedily sampling candidate waypoints by computing expected information gain [4, 63] is a commonly adapted strategy to compute informative trajectories at the cost of performance [62].

Learning-based methods commonly inspired by Reinforcement Learning (RL) [2] have been proposed to address the problem of adaptively planning trajectories computationally efficiently without performance degradation. Many RL-based IPP solutions have been formulated for exploration [81], environmental monitoring [60], search and rescue [47] and localization of sources [85]. However,

these methods have been shown to be restricted to action spaces of the next candidate measurement sites and therefore planned trajectories are dynamically infeasible.

In contrast to existing work, our approach plans dynamically feasible trajectories by adapting motion primitives, which are computed offline. This enables online informative trajectory generation for agile UAVs. Leveraging the advantages of learning-based methods over classical informative path planning approaches, our planning approach algorithm leverages the A3C method to train a policy over an offline set of primitives to plan dynamically feasible informative paths.

## 2.4. **Problem Definition**

We model the semantic features of interest (FoI) detection task as an informative path-planning problem where we describe the mapping approach and the simulated sensor model. The environment can either have a single semantic FoI (target) which we refer to as target search or multiple semantic FoI which we refer to as semantic feature detection. In both settings, a UAV operates at a fixed altitude and executes a motion primitive. Along the path undertaken by the primitive, the UAV collects measurements in the form of semantic labels from a simulated sensing module which is used to update its beliefs over semantic features. The UAV would plan another primitive to execute using the updated belief map.

### 2.4.1. Informative Path Planning

The problem is cast as an informative trajectory planning problem where the overall objective is to choose an optimal trajectory $\psi^*$ from a collection of admissible trajectories $\mathbf{\Psi}$ that maximizes the information-theoretic utility $I$ on the target estimate. The IPP objective is defined as [53]

$$\psi^* = \underset{\psi \in \mathbf{\Psi}}{\operatorname{argmax}} \, I[\textsc{Measure}(\psi)] \tag{2.1}$$

$$s.t \; \textsc{Energy}(\psi) \leq B.$$

Where $B$ denotes the energy budget available for the flight of the drone, $\textsc{Measure}(\cdot)$ denotes the information gathered along the path $\psi$ through a sensor. $\textsc{Energy}(\cdot)$ estimates the energy requirement to execute the given path.

### 2.4.2. Environment Model

The probability of a semantic feature existing in the environment at a given location is modeled as discrete occupancy grids for each semantic category that also includes background semantics. Thus, a measurement $z_t$ at a given time $t$ updates the observed cell $m_{xy}$ via the log updates [80] as follows:

$$L(m_{xy} \mid z_{1:t}, x_{1:t}) = \sum_{t=1}^{N} \log \frac{p(z_t \mid m_{xy})}{p(z_t \mid \tilde{m}_{xy})} + L(m_{xy}) \tag{2.2}$$

where $p(z_t|m_{xy})$ denotes the forward sensor model, $m_{xy}$ refers to an occupied cell and $\tilde{m}_{xy}$ refers to a free cell, and $L(m_{xy})$ denotes the log odds of the belief prior at location $(x, y)$ in the grid. A semantic feature is recognized if its occupancy probability is above 95%.

### 2.4.3. Sensor Model

We assume that the UAV is equipped with a downward-looking camera that provides a semantic label $z_t \in [0, n]$ at time $t$ where $n$ is the number of semantic categories. Since the sensor is at a fixed altitude, we assume that the sensor has a fixed rectangular footprint on the environment. Here, we also assume that the likelihood of the sensor providing the correct semantic label $p(z_t \mid m_{xy})$ given that it observes the environment varies with the distance of the camera from the ground.

## 2.5. **Planning Approach**

Our planning approach has two key features. First, this method exploits the strengths of motion primitives constructed offline that respect the vehicle dynamics and can be extended spatially to plan in unbounded configuration spaces. Second, this approach learns a policy network over motion primitives conditioned on scaled-up egocentric views of feature maps. The policy can be executed during run time for fast re-planning based on the current state of the occupancy map. Our approach is suitable for both binary target mapping and semantic feature detection, and we briefly describe the observation space, state space, and respective action spaces.

### 2.5.1. Trajectory Parameterization via Dispersion Minimizing Motion Primitive Graphs

By incorporating dynamics information, UAVs can execute agile paths for a monitoring or target search task which is a primary limitation of prior work. As such we parameterize trajectories by generating a motion primitive graph oversampled states that minimize the asymmetric dispersion costs $\max(J(\mathbf{x}, \mathbf{v}), J(\mathbf{v}, \mathbf{x}))$ between quadrotor states $\mathbf{x} \in X$ and sampled graph nodes $\mathbf{v} \in V$ [26] evaluated via trajectory optimization on quadrotor dynamics[36].

$$d(V) = \sup_{\mathbf{x} \in X} \min_{\mathbf{v} \in V} \left[\max(J(\mathbf{x}, \mathbf{v}), J(\mathbf{v}, \mathbf{x}))\right] \tag{2.3}$$

Hence, our approach defines these trajectory segments offline and can be spatially extended by leveraging the translational invariance property in UAV dynamics for planning in unbounded configuration spaces [27].

### 2.5.2. IPP with primitives as a HRL problem

We cast IPP problem as an HRL problem where the objective is to maximize the total information gain for semantic detection over a trajectory sampled from the spatially extended motion primitive graph. The POSMDP for the IPP is defined as follows:

**States**

In our setup, the agent's global state consists of three crucial maps - belief maps, coverage maps, and obstacle maps. For the target search application, our agents maintain a single belief map

solely focused on tracking targets. The belief maps are sized at 30 meters by 30 meters with a fine resolution of 1 meter by 1 meter per pixel, without any padding.

For the semantic detection application, our agents manage multiple belief maps, each corresponding to a semantic category, including an additional map for the background. Initially, the environment size for semantic detection is a 1024 meters by 1024 meters satellite view, down-scaled to produce a semantic map of 256 by 256 pixels at a resolution of 4 meters by 4 meters per pixel.

Throughout both applications, we dynamically update the belief maps based on sensor measurements collected during the execution of motion primitives. These updates continuously refine the agent's understanding of the environment and the presence of targets or semantic features. To further enhance the learning of valid motion primitives, particularly near the environment's boundaries, we maintain an obstacle map. This map indicates whether a specific space is free or occupied by an obstacle. Since our focus lies in free-space environments, the obstacle map represents the boundaries of the environment.

Additionally, our agents keep a coverage map, which gets updated as they execute motion primitives and their sensors interact with the environment. This coverage map serves to identify the areas explored and covered by the agent. This helps in discovering new areas in the environment for further investigation. Together, the combination of these three maps and the quadrotor's global state provides a comprehensive representation of the environment and its dynamics.

**Observations**

For informative search applications, the size of the environment may vary by application. Hence, using the above state representations as inputs would require a change in the architecture of the policy network and the network would have to be re-trained. To prevent this we adopt an observation model that takes in an egocentric view of the belief, obstacle, and coverage maps of the agent (see Fig. 2.5.

Using local information may make the learned policies myopic, hence we also add egocentric observations at different scales [1,2,4] see Fig. 2.5. The egocentric observations at higher scales are

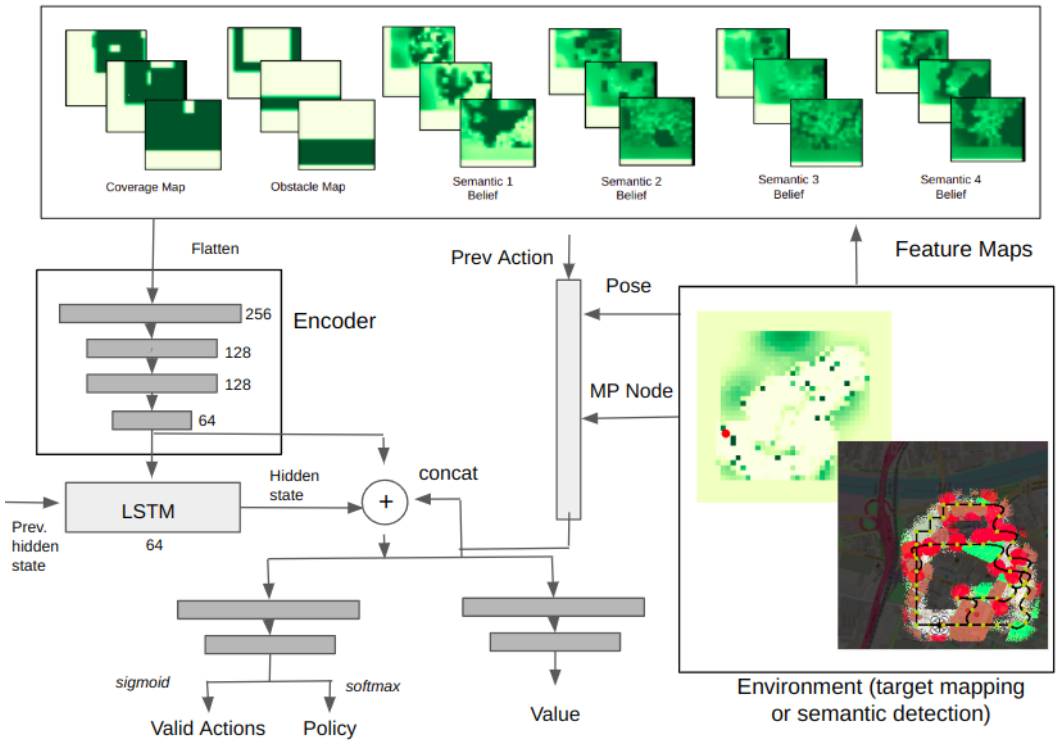


Figure 2.2: **Model Architecture**

down-sampled (reduced resolution). In the target mapping application, [8 x 8, 16 x 16, 32 x 32] pixel patches around the agent for each map are taken and down-sampled to [8 x 8] patches each. In the semantic feature detection application, we consider [48 x 48, 96 x 96, 192 x 192] pixel patches around the agent for each map taken and down-sampled to [24 x 24] patches as inputs to the policy network. Observations to the policy network also include the position of the UAV normalized to the size of the map, the one-hot encoding of the prior motion primitive taken, and the one-hot encoding of the current state node $V$ on the motion primitive graph $G$.

### Actions

Given a motion primitive lattice ($G = (V, E)$), the skill space of the UAV is composed of primitives $E(\mathbf{v})$ at the node $\mathbf{v}$ of the motion primitive graph. A stochastic policy is outputted over the set of valid motion primitives at the given state. These valid primitives are learned for collision avoidance at environment boundaries and the dynamic feasibility of the path through a valid loss specified in 2.5.3.

**Rewards**

Rewards are designed for the agents to reduce the uncertainty of the target occupancy map, prioritize covering high-target occupancy estimate regions, and find targets as quickly as possible while penalizing high-cost primitives to transition across states. Here, the reward for the uncertainty reduced is characterized by the reduction in the entropy of the map as follows:

$$\mathcal{H}(M_t) = \sum_{xy \in \mathbb{R}^2} \mathcal{H}(m_{xy} \mid z_{1:t}) \tag{2.4}$$

Thereby the rewards due to the reduction in the uncertainty of the map are given by:

$$r_t = \mathcal{H}(M_{t+1}) - \mathcal{H}(M_t) \tag{2.5}$$

where $M_{t+1}$ is the state of the map at time $t+1$ after primitive action $a_t$ is executed. The total reward to reduce the total entropy of the map to 0 is +15. In addition, we define the coverage reward as the proportion of unvisited areas covered with respect to the environment size when a primitive $z_t$ is executed. This reward is essential for the UAV to increasingly explore the environment and the maximum reward is +10.0 for covering the entire map. In addition, agents are rewarded for detecting semantics or targets once their occupancy probability exceeds 0.95. This reward is much higher than uncertainty reduction and coverage (+20 for each semantic category or target) as this would motivate agents to discover better behaviors that balance between exploration and exploitation, instead of over-fitting to the coverage and uncertainty reduction reward. Additionally, a penalty is in proportion to the cost of the selected primitive. This cost is quantified by the total energy cost and time taken to execute the primitive [26] and is cumulatively set to (-20) for an episode in order for the agent to take longer informative paths.

### 2.5.3. Learning

We leverage the A3C framework to train a policy over the motion primitive graph where environments are generated across many local worker threads and a global policy network is updated via gradients from local experience buffers in individual workers.[44].

**Environment Generation**

We train our policy across various iterations of generated environments to prevent over-fitting for both the target mapping and semantic detection environments.

For the target mapping scenario, ground truth targets are generated from a ground truth distribution (GT) (referred to as the world model) modeled as a GMM. An agent's prior belief over targets is initialized by shifting the ground truth Gaussian centers and sampling additional Gaussian centers as noise. Through this modification of the ground truth distribution, policies learned would not directly exploit the agents' belief to find targets as agents' beliefs in a real-world setting are expected to be inaccurate to an unknown degree.

In the semantic detection setting, we obtain the ground truth distribution of the semantic targets through the open-source software OpenStreetMap over various University locations. A GMM is approximately fit over each semantic feature ground truth distribution to model an agent's belief over each semantic feature. Additionally, random noise is injected into the weights of the Gaussian mixture to ensure that the semantic feature priors are not directly exploitable and the policy learns to balance between exploration and exploitation to efficiently detect semantic features in the presence of inaccurate semantic priors.

**Network Architecture**

The actor-critic network outputs a policy $\boldsymbol{\pi}(\cdot \mid \mathbf{o}_t; \theta_c)$, a value estimate $V_\delta(\mathbf{o}_t; \theta_c)$ and an estimate of valid actions $\boldsymbol{\psi}(\cdot \mid \mathbf{o}_t; \theta_c)$ from the observation inputs, as shown in Fig 2.5. Here the value estimates are computed separately for each reward contribution as specified in the reward structure. The value are predicted separately predicted for the coverage ($\delta = cov$), map entropy reduction ($\delta = e$), cost ($\delta = c$), and semantic feature search portions($\delta = r$) of the reward. Here $\mathbf{o}_t = \{o_i\}_{i=0}^{t}$ refers to the history of observed spatial map inputs, which is maintained through an additional LSTM module.

**Training Losses**

During a training episode, the agent samples an action (primitive) from the policy distribution at a given observation and an episode terminates once the agent exhausts its budget.

The critic is regressed against the discounted lambda returns $G_t^\lambda$[76]. Empirically $\lambda = 0.8$ gave us the best results. The critic loss is given by

$$L_{\theta_c} = \sum_{t=0}^{T} \Big( G_t^\lambda - V(\mathbf{o}_t; \theta_c) \Big)^2 \tag{2.6}$$

To improve the policy $\pi(\cdot \mid \mathbf{o}_t; \theta_p)$, the advantage $A(\mathbf{o}_t, z_t; \theta) = r(s_t, z_t) + \gamma V(\mathbf{o}_{t+1}; \theta_c) - V(\mathbf{o}_t; \theta_c)$ is computed for a given primitive [44]. These advantage terms are discounted from a given time step to compute the generalized advantage estimate $A_t^{GAE}$ [76] for the actor loss $L_{actor} = \sum_{t=1}^{T} \log\left(\pi(z_t \mid s_t; \theta_p)\right) A_t^{GAE}$.

In addition, we add a one-step entropy loss $L_{ent} = \sum_{t=1}^{T} \mathcal{H}(\boldsymbol{\pi}(s_t); \theta_p))$ to encourage exploration and a supervised asymmetric binary cross entropy loss $L_{val}$ to ensure that the policy network outputs a valid distribution over primitives. The supervised loss heavily penalizes invalid primitives. While action masking is an alternative strategy, in practice we found it to be brittle to use to train policies due to the asynchronous nature of the gradient updates made to the policy network.

$$L_{val} = \sum_{t=1}^{T} \beta_1 (1 - \mathbf{y_t}) \log(1 - \boldsymbol{\phi}(s_t; \theta_p)) + \beta_2 \mathbf{y_t} \log(\boldsymbol{\phi}(s_t; \theta_p)) \tag{2.7}$$

where $\mathbf{y_t}$ refers to the set of valid actions/primitives. In summary, the policy loss can be given by:

$$L(\theta_p) = \alpha_1 L_{ent} - \alpha_2 \, L_{actor} + L_{valid} \tag{2.8}$$

## 2.6. **Results**

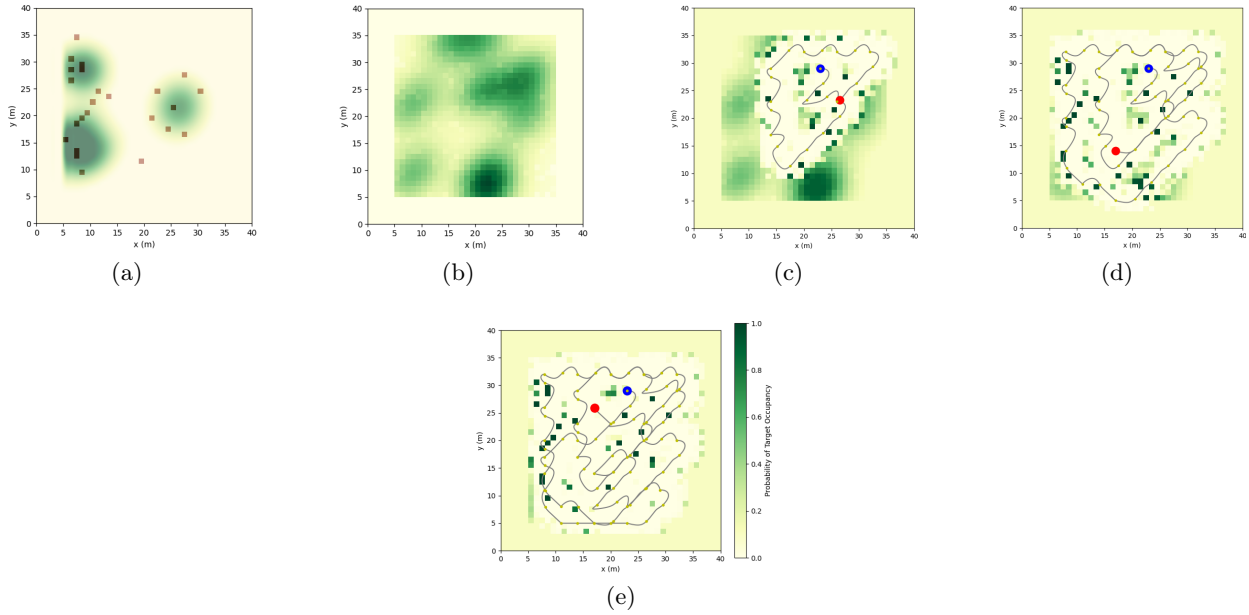


Figure 2.3: **Visualization of the traveled trajectory at varying remaining budgets.** Agent's starting position is depicted via a green dot and the ending position is depicted via a red dot. From right to left, the plots show the (a) True target distribution with targets marked in red; (b) Prior belief of the map; the following plots show the traveled trajectory at c) 70% budget; d) 30% budget; (e) 0% budget.

We evaluate our policy with respect to the state-of-the-art IPP heuristics over a fixed suite of test environments for both target mapping and semantic mapping. We generate 100 test environments for target search and 30 test environments for semantic feature detection following the environment generation strategy described in Section 2.5.3. Note that the ground truth semantic features are fixed and obtained from OpenStreetMap for the semantic detection environments, unlike the target search environments. We evaluate the search performance of the agent based on a fixed starting location of the robot in the environment.

### 2.6.1. Benchmark Algorithms

To our knowledge, existing methods do not use dynamically feasible motion primitives for IPP. Hence, our method is evaluated against some of the classical well-tuned IPP heuristics which have been adapted to our motion primitive framework to ensure a fair comparison.

Table 2.1: Search Performance Comparisons of Baselines over a fixed budget

| **Methods** | **Coverage (%)** | **Entropy Reduction (%)** | **Search Efficiency (%)** |
|---|---|---|---|
| Greedy-IPP | 40.94±23.71 | 41.76 ±24.54 | 54.37±33.18 |
| DP-IPP | 57.84±11.73 | 60.23±12.42 | 80.24±20.84 |
| CMAES-IPP | 52.10±7.03 | 55.81±7.43 | 75.52±15.95 |
| Coverage | **62.24 ± 0.28** | 60.48 ±1.48 | 82.07±10.53 |
| Prioritised Coverage | 51.99±10.10 | 56.10±10.56 | 82.49±18.11 |
| **Ours-noLSTM** | 57.72±3.22 | 61.26±3.37 | 85.73±10.02 |
| **Ours-LSTM** | 60.20±3.31 | **63.75±2.77** | **88.52±8.71** |

.

1. Greedy: The greedy algorithm selects the primitive with the highest one-step utility. Here the utility is the informativeness of a certain location (x,y) on the map and is defined as:

$$I(q) = f_1 m_{xy} + f_2 H(m_{xy})$$

2. Dynamic Programming: Dynamic Programming maximizes information gain over a sequence of primitives in a receding horizon manner. Here, the agent executes a single primitive and replans.

3. Covariance Matrix Adaptation Evolutionary Strategy(CMAES): CMAES[48] samples a sequence of primitives and refines these paths over a fixed horizon (5) to maximize information gain. The sequence of primitives over the fixed horizon is then executed for the duration of that horizon.

4. Online Coverage: Agents maximize the uniform coverage utility to determine the next motion primitive to execute at a given state.

5. Online Prioritized Coverage: Agents prioritize coverage utility towards regions with higher

target or semantic occupancies when planning for the next motion primitive.

### 2.6.2. Results on Target Search

Table 1 shows the performance of the algorithms averaged out over a set of 100 test environments with varied ground truth target distributions and we conclude that the performance of our method outperforms the baseline heuristics implemented in terms of search efficiency. Fig 2.3 shows an example of the traveled path taken by the quad-rotor for target search over a 2D environment through our learned policy over motion primitives. The left plot confirms that the agent explores the space while re-visiting some of the sites to improve estimates that would result in finding targets.

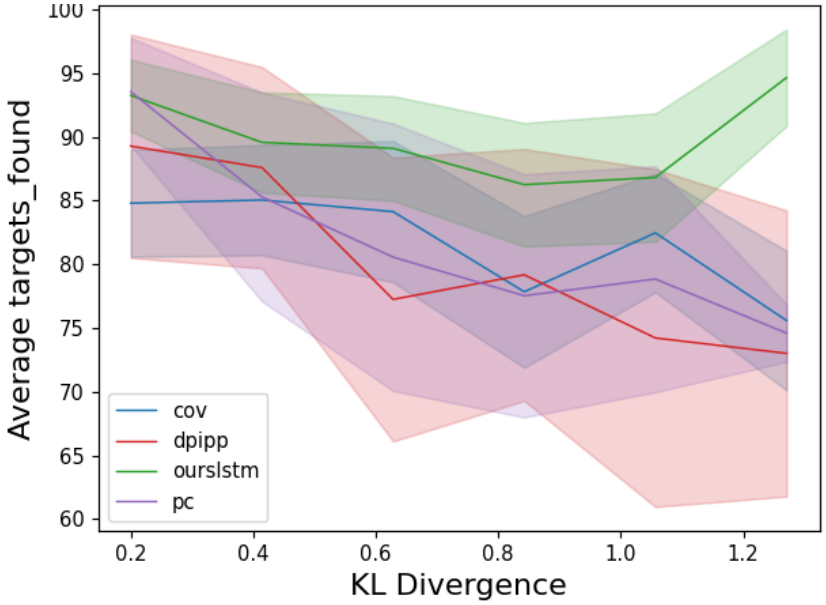


Figure 2.4: **Robustness of our approach with varying distributions of the prior.** The prior over the environment can be inaccurate when compared to the true target distribution, as indicated by the KL divergence ($x$-axis). Our LSTM approach (*green* generally performs better than the prioritized coverage (PC) heuristic (*purple*), DP-IPP method (*red*), and coverage heuristic (*blue* in terms of search efficiency ($y$-axis). Furthermore, our approach has smaller deviations from the mean, indicating that the performance of our approach is robust to varying divergence of the prior from the true target distribution.

Through extensive tests, we conclude that our approach outperforms the baseline heuristics by learning a policy over motion primitives that balances exploration and exploitation. A pure ex-

Table 2.2: Search Performance Comparisons of Baselines over a fixed budget for semantic mapping

| **Methods** | **Coverage** (%) | **Entropy Reduction** (%) | **Search Efficiency** (%) |
|---|---|---|---|
| Greedy-IPP | 14.21±2.27 | 2.57±1.48 | 6.53±1.98 |
| DP-IPP | 67.83±7.21 | 30.08±3.73 | 53.03±5.38 |
| CMAES-IPP | 33.64±19.95 | 8.72±8.42 | 20.54±15.31 |
| Coverage | **88.67 ±0.0** | 30.77 ± 4.43 | 60.07±2.47 |
| Prioritised Coverage | 69.13±11.75 | 15.24±4.91 | 34.51±8.27 |
| **Ours-noLSTM** | 74.88± 2.59 | 32.16±4.91 | 59.56 ±2.74 |
| **Ours-LSTM** | 83.78 ± 3.43 | **34.24±6.10** | **63.61 ±3.08** |

ploration strategy such as the coverage heuristic seeks out primitives that visit more unobserved locations in the environment, hence, having the highest coverage performance but poorer target search performance. On the other hand, prioritized coverage is an exploitation-based strategy, hence, primitives which would most likely confirm the existence of the target are selected, thereby, sacrificing coverage for improved target search efficiency. However, targets in the unexplored space of the environment weren't found. To account for this, the information-theoretic utility described above for IPP-based algorithms was tuned to balance exploration vs exploitation. Hence, these methods especially DP-IPP-based were observed to have better coverage performance and uncertainty reduction as compared to prioritized coverage, however, target search efficiency was slightly worse as the agent failed to revisit areas to better confirm the existence of targets. CMAES-IPP had poor trajectory quality due to limited online re-planning. In contrast, our learned policy is able to balance both these requirements. The policy maintains high levels of environmental coverage(2nd to coverage-based heuristic) while improving target search efficiency (better than prioritized coverage), indicating that our planning approach learns a policy to balance between exploration and exploitation as compared to IPP-based methods.

This result is also substantiated by Fig 2.4, where we contrast the performance of our approach, prioritized coverage, and DP-IPP with respect to the KL divergence between the ground truth distribution and the agent's prior. We show that our learned policy not only outperforms but also

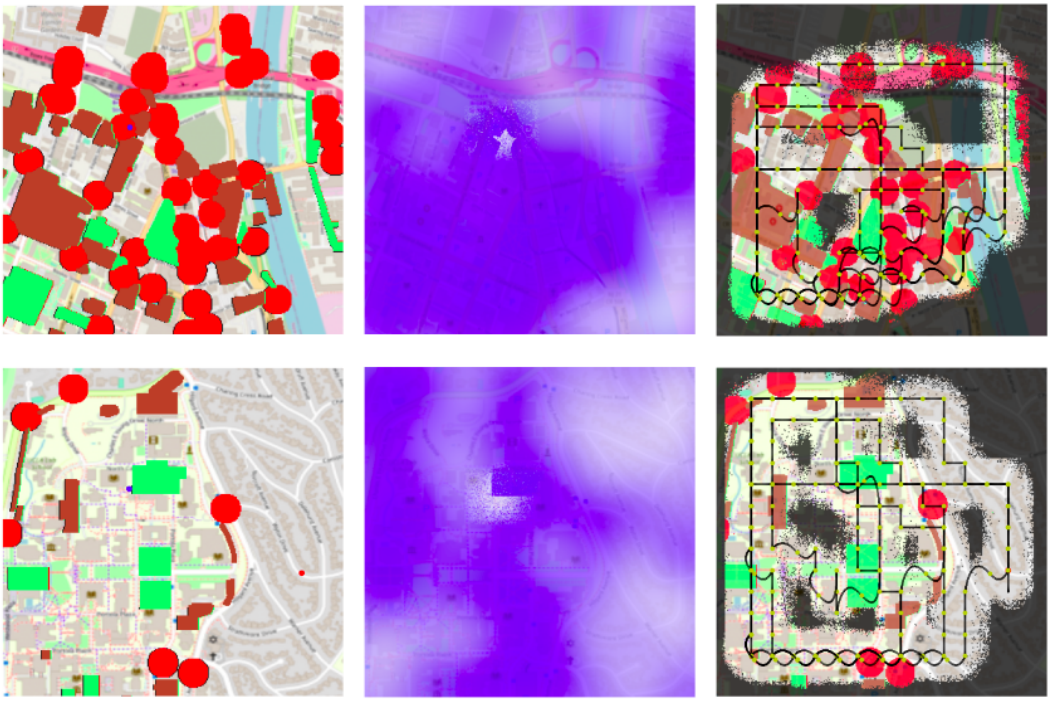

Figure 2.5: **Resulting trajectories with our learned planner for sample tasks.** Final path (right images) taken by the UAV for two semantic detection tasks with their ground truth semantic targets (left-hand side), prior over semantic features (middle)

is more robust than both DP-IPP and Prioritised coverage in terms of search efficiency when the prior beliefs diverge significantly from the ground truth target distribution from which targets are sampled.

### 2.6.3. Results on Semantic Feature Detection Environments

In addition to target search, we also validate our learning framework over 30 semantic feature detection tasks with fixed priors over semantic features across all tests. Policies learned by our method outperform the baseline heuristics implemented in terms of semantic detection efficiency (see Table 2).

For semantic feature detection, we observe that the coverage algorithm outperforms all the baseline heuristics. Agents with greedy-based IPP planners were often found to get stuck at local minima and thereby their search performance was observed to be poorer. In the case of CMAES-IPP, limited online planning coupled with the stochasticity in the path sampling procedure resulted in

trajectories that often went beyond the environment bounds and the lack of adaptation to the current state typically resulted in invalid episodes. In contrast, prioritized coverage was observed to have degraded performance compared to DP-IPP and coverage. Due to multiple semantic features, the algorithm failed to plan trajectories that would reliably improve the detection of each category. DP-IPP-based methods had poor solution quality due to insufficient coverage. In contrast to the baselines, our learned policy maintains a competitive coverage over the environment while improving semantic search efficiency as compared to the coverage-based heuristic.

#### 2.6.4. Simulation Verification

We demonstrate our planning approach in a realistic Gazebo simulation for both target mapping and semantic mapping. For target mapping, the targets (boxes) are randomly placed and a prior for the target occupancy map is initialized over the search space. For semantic mapping, we use the ground truth semantic map generated from OpenStreetMap over the University of Pennsylvania and assign a prior over each semantic feature as discussed earlier.

The policy learned with our proposed approach is merged with the Kumar autonomy stack for real-time flight planning over a simulated in-house Falcon 4 platform. The UAV executes the primitive outputted via the policy network during run time, collects measurements using a simulated camera, and updates the semantic map.

## 2.7. Conclusion

This paper proposes a novel RL-based adaptive Informative Path Planning with policies defined over dynamic motion primitives of a fast-moving robot to achieve target search. The proposed algorithm enables fast information gathering for active classification tasks by planning dynamically feasible agile paths. A key component in our approach is the use of motion primitives to parameterize trajectories that respect UAV's dynamics. The learned policy is validated by comparing its performance to multiple IPP-based benchmarks adapted over motion primitives and improves target search efficiency without compromising on runtime constraints, and adaptive replanning to balance between exploration and exploitation. Future work would investigate this planning approach for 3D UAV planning with real-world experiments for experimental validation, and UAV teams.

# CHAPTER 3

# OFFLINE GOAL CONDITIONED SKILL ACQUISITION FOR QUADROTORS

## 3.1. **Abstract**

In this chapter, we present a learning-based approach to address the formulation of motion primitives or locomotion skills for an aerial vehicle. By employing goal-conditioned deep reinforcement learning, we acquire reusable locomotion skills from expert data by learning a single goal-conditioned policy. This chapter experiments with different offline goal-conditioned RL algorithms for skill learning and identifies a viable choice for learning goal-conditioned skills from expert data. We further demonstrate the efficacy of the learned skills by proposing a planner that plans intermediate goals to reach an out-of-distribution task goal. By utilizing skill learning frameworks on offline quadrotor data and the proposed planner, we aim to close the gap between motion planning and control.

## 3.2. Introduction

A hallmark of human intelligence has always been the ability to accomplish a general set of tasks using low-level skills goals or primitives. These skills and primitives are often learned through trial and error, or through demonstrations, and humans have a remarkable capability to compose these primitives in sequence to complete tasks.

Motion primitives have also been popularly used for real-time tasks such as mapping [10] and navigation [26] [27] and informative target search [20] using aerial vehicles. These primitives can be queried online during task completion [10] or can be constructed offline [26]. A primary advantage for motion primitives generated online by sampling from the control space and forward propagated based on the robot's current state is that it produces dynamically feasible and continuous task-specific paths. However, given that the planning query is often posed in the state space, the selection and generation of a satisfactory set of primitives from the input space in an online manner is not obvious for nontrivial system dynamics [26]. Furthermore, these primitives are constructed from aerial vehicles' current states, hence these are often not reusable [10].

On the other hand, motion primitives that are constructed offline enable reusability while preserving generalization to various tasks by first forming a planning graph over the environment [26][27]. These planning graphs can be further queried downstream either online or offline to compose trajectories for the robot to complete such tasks. A typical planning pipeline involves computing an optimal plan prior to task completion or execution; during what is colloquially called the 'thinking' phase of the robot. Alternatively, search heuristics [20] [27] can be employed to approximate the optimal motion plans for some tasks to make online planning feasible.

However, motion primitives constructed offline are limited to either fixed durations of time or fixed sampled states [26, 27] and thereby constrain the quality of trajectory planning. This is corroborated in [27] work where straight paths are often composed of loopy motion primitive segments.

In summary, existing techniques for devising motion primitives have two major shortcomings. First, composing lower-level skills for smooth online trajectories can be computationally expensive and

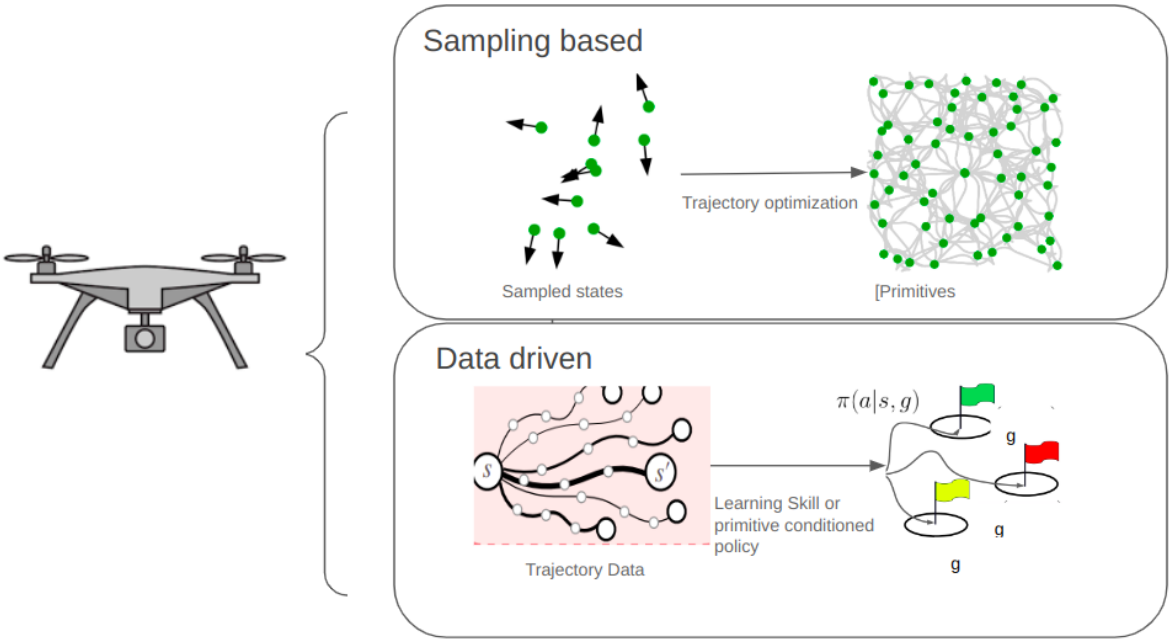


Figure 3.1: **Proposed Approach.** This image depicts the difference between sampling-based motion primitives and data-driven goal-conditioned policies for trajectory planning. The former involves stochastic sampling of waypoints or control actions to build candidate paths, while the latter uses offline collected data and learning to directly map states and achieved goals to actions, providing primitives in the form of fine-grained goal-conditioned policies for smoother trajectories.

non-reusable if done online. Second, these primitives if learned offline are often restricted to fixed durations of time or fixed goal states if done offline, thereby constraining downstream planning quality.

We aim to propose a learning-based method whereby robots particularly quadrotors can acquire such lower-level locomotion skills or primitives that permit good quality planning solutions online. We would look to learn such primitives to complete a simple navigation task of moving a quadrotor to a specified goal state without having any explicit expert data to reach this goal. Just as motion primitives can be formulated to be reused [26], we learn to capture and acquire reusable skills that could be chained downstream to accomplish complex tasks.

These locomotion skills or primitives can be conditioned on image, language, or robot state [34]. In light of its recent successes in the robotics domain, goal-conditioned deep reinforcement learning

(RL) has the potential to learn such skills or primitives. Particularly, recent work has seen the development of methods to distill a goal-conditioned policy parameterized by a neural network from expert data [84, 40]. For a quadrotor, this expert data would consist of either lower-level motor thrust commands or desired thrust and angular velocity commands that reach some desired goal states in the form of orientation, velocity, and position from raw aerial vehicle states in accordance with the dynamics of the vehicle. We aim to distill this expert data into a single goal-conditioned policy that captures continuous state-based locomotion skills for quadrotors whereby the policy takes the aerial vehicle's current state and desired goal state as input and produces low-level control commands to reach the desired goal state with high probability. This is in contrast to [26] where fixed goal states are sampled within a handcrafted reachability set for aerial vehicles. Our approach would result in low-level goal-conditioned skills or primitives for the quadrotor that would not be restricted to a fixed time duration or fixed states. Having access to this controller, the agent would have to optimize for a sequence of controller *feasible* goals to complete the task.

Thus the contributions of this chapter are two-fold. We first propose a method to distill a base set of skills completely offline from data collected from an expert controller to accomplish a set of proximal goals. The goal-conditioned skills trained offline are then used by a proposed planning module to reach goal states not present in the training data. Through this planning module, we also show that we can learn to reach a large proportion of distant goals which are out-of-the-distribution of the training goals as compared to a goal-conditioned policy trained on expert data.

## 3.3. Background and Related Work

### 3.3.1. Quadrotor Dynamics

The quadrotor is a 6-degree of freedom robot with a mass $m$ and inertia matrix $\mathbf{J} = diag(J_x, J_y, J_z)$. The full state $\mathbf{x}$ of the quadrotor can be defined by a vector $[\mathbf{p}, \mathbf{q}, \mathbf{v}, \omega, \mathbf{\Omega}]$ where $\mathbf{p}$ is the position, $\mathbf{q}$ is the orientation, $\mathbf{v}$ is the velocity, $\omega$ is the body rate and $\mathbf{\Omega}$ is the rotor speed of the quadrotor.

Assuming no friction or rotor drag, the dynamics of the quadrotor are written as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{q}} \\ \dot{\mathbf{v}} \\ \dot{\omega} \\ \dot{\mathbf{\Omega}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{v}} \\ \mathbf{q} \cdot \begin{bmatrix} 0 \\ \omega \end{bmatrix} \\ \frac{1}{m}(\mathbf{q} \odot \mathbf{f}_{prop}) + \mathbf{g}_w \\ \mathbf{J}^{-1}(\tau_{prop} - \omega \times \mathbf{J}\omega) \\ \frac{1}{C_m}(\mathbf{\Omega}_{cmd} - \mathbf{\Omega}) \end{bmatrix}$$

where $\mathbf{g}_w = [0\ 0\ -9.81 m/s^2]$ denotes earth's gravity, $\mathbf{f}_{prop}, \tau_{prop}$ are the collective force and the torque produced by the propellers. Each propeller i produces a force $\mathbf{f}_{prop,i} = [0\ 0\ k_f \Omega_i^2]$ and moment $\mathbf{M}_{prop,i} = [0\ 0\ k_w \Omega_i^2]$ along the body frame. Thus the total force $\mathbf{f}_{prop} = [0\ 0\ \sum_i f_{prop,i}]$ in the body frame and a moment $\tau_{prop} = \sum_i \mathbf{M}_{prop,i} + \mathbf{r}_{prop,i} \times \mathbf{f}_{prop,i}$. Here $\mathbf{r}_{prop,i}$ is the distance of the rotor from the body's center of mass in the body frame.

### 3.3.2. Trajectory Planning and Control in Quadrotors

Trajectory planning and control for quadrotors can be primarily classified into three classes - differential flatness-based, optimization-based, and search/sampling-based.

Quadrotor trajectory generation via differential flatness is one of the most popular and hardware-tested paradigms for quadrotor flight [42]. Here, the planning and control are segregated, where a trajectory plan is generated to a goal state(s) by leveraging the differential flatness property of aerial vehicle platforms, and a tracking controller is designed to track the target trajectory [42].

These trajectories are typically represented by a polynomial on the flat outputs [42, 22]. Obtaining an optimal trajectory plan and associated control commands are fast to compute and generalizable to many goal states using a standard constrained quadratic program[42, 22]. This makes differential flatness-based planners and controllers suitable for the curation of an expert dataset for a few goal states.

More advanced trajectory planning and control methods solve a constrained non-linear program to obtain either obtain an optimal trajectory plan [57], control commands [74, 34], or time segments between waypoints [16]. These methods optimize over non-linear quadrotor dynamics models with complex non-linear constraints for specific navigation or tracking tasks. For instance, Romero et al. [57] optimize for a sequence of waypoints to track a fixed trajectory. Sun et al.[74] optimize for control outputs to track fixed agile trajectories and Foehn et al. [16] optimize for the time between trajectory segments indexed by waypoints that the quadrotor is supposed to visit. These methods can be computationally heavy and thereby curating an expert dataset can be computationally expensive using optimization-based methods.

The third class of methods involves searching either over a fixed discretization of the control inputs [36], sampled control inputs [10], or fixed discretization of states [37], or states that are optimally dispersed [26, 27]. Algorithms like A* [61] can be used to search over these control primitive tress or planning graphs or RRT* [83] can be used to build and search over planning graphs in primitive space to compose trajectories to goal states from a given initial state. However, search and sampling-based methods are often restricted to fixed time intervals or fixed states. Hence, curating an expert dataset from sampling-based methods can prohibit the learning of good quality skills.

## 3.4. Method

In this section, we will outline the specifics of the skill-learning pipeline for aerial vehicles. Skills in the form of goal-reaching policies are distilled through expert data and we first outline the choice of expert. Second, we outline the distillation process of how we learn a single goal-conditioned policy from the expert data. Additionally, we also outline a practical algorithm to plan a sequence of goals using the goal-conditioned policy distilled from expert data. The planner uses learned value functions distilled from the expert data to plan intermediate goals and value-based disagreement to eliminate goal candidates on which the value functions haven't been trained.

### 3.4.1. Data Collection

We first curate a dataset offline $\mathcal{D}$ for the aerial vehicle which starts at the origin with no velocity and hover orientation. This aerial vehicle is tasked to reach a dataset of 250 desired goals $\rho_{\mathcal{D}}(g)$ within a 1m to 2m position box, with random velocities up to 2.5 $m/s$ range and accelerations between 5 $m/s^2$ across all three dimensions. Given the initial conditions and the final conditions, we construct a hierarchical differential flatness-based trajectory planner that computes the minimum time needed to reach the desired goal without violating the constraints imposed on the velocities and accelerations and computes the desired smooth polynomial trajectory to reach the desired goal.

**Trajectory planning**

Following the work [42, 22], we leverage the differential flatness property of quadrotors to plan a trajectory to a specified goal in a flat space. The flat outputs of the quadrotor are $\mathbf{p}$ and $\mathbf{\Psi}$ which correspond to the positions and yaws of the aerial vehicle. We can leverage the derivatives of these flat outputs to compute the net thrust and desired angular velocities for the quadrotor.

The trajectory planner for data generation for skill learning minimizes *jerk* as opposed to *snap* as done in [42]. This is done by parameterizing the trajectory as a 5th (jerk) order polynomial in time in flat space assuming we have access to how long each trajectory would last. The planner in [42] computes the polynomial coefficients through a quadratic program (QP) where additional linear constraints in the form of obstacles, or desired accelerations can be embedded. For our data generation, we plan a single polynomial spline to reach the desired goal state with the desired

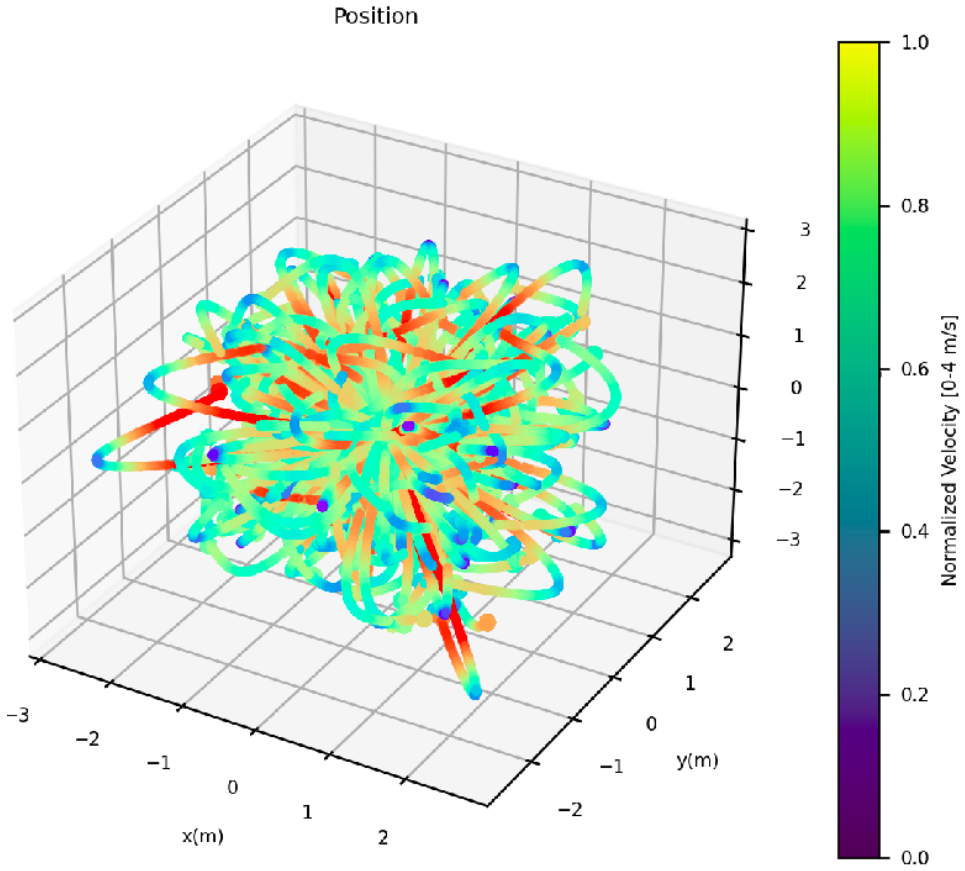


Figure 3.2: Visualization of dataset compiled for the quadrotor. A dataset of 250 desired goals $\rho_{\mathcal{D}}(g)$ within a 1m to 2m position box, with random velocities up to 2.5 $m/s$range and accelerations between 5 $m/s^2$. Reference trajectories are planned and we plot the resulting trajectories of the aerial vehicle while tracking the reference trajectory using the controller.

acceleration to be capped at 5 $m/s^2$. This is to ensure we have good-quality trajectories by not saturating the controller.

However, unlike [42] which pre-specifies the time needed for each polynomial segment, we also optimize the time required to reach the specified goal. Hence, we devise a search-based optimizer to compute the minimal time for a trajectory plan optimized by the lower-level QP without violating any constraints. We run a binary search on the time variable starting from a fixed $4s$ range. If the optimizer finds a valid solution, we decrease the time needed and recompute a solution. If the optimizer doesn't find a valid solution, we increase the time and solve the QP optimization. This is done till the time range reduces to $0.02s$. The algorithm is outlined in Alg. 1

**Controller**

We use the non-linear geometric controller proposed by [42] to track the given reference trajectory from the planner given above. Here, the geometric controller computes the feed-forward desired thrusts based on the desired acceleration profile along the optimized trajectory polynomial using a PD controller. The desired thrust acts from the center of mass along the aerial vehicle's body frame $z_b$ axis. Note that the quadrotor must be oriented such that the desired thrust produces the necessary acceleration. Hence, the desired moments to control the attitude of the aerial vehicle are computed analogously to [42] to correct for orientation errors through a PD controller. This attitude control loop converges at a much higher frequency than the desired acceleration by setting the gains following [42].

The exact parameters of the planner and the controller can be referenced in the attached codebase. The visualization of the collected expert dataset is presented in 3.2.

**Algorithm 1** Computing minimum time single polynomial reference trajectories to specified goals from an initial state

**Initialize:**
    Goal state $g$, initial state $s_0$, $t_{min}$, $t_{max}$, $a_{max}$ max acceleration, $a_{min}$ min acceleration and threshold
**while** $t_{max} - t_{min} \geq \delta$ **do**
    Set $x_0 = s_0$, $x_f = g$ and $t = \frac{t_{max}+t_{min}}{2}$
    Set initial trajectory coefficients $\mathbf{p} = 0$
    Set Linear constraints $\mathcal{C}$ second order derivative of polynomial trajectory to acceleration constraints $a_{max}$ and $a_{min}$
    Compute valid optimal trajectory coefficients with min-jerk optimization [42] with initialization of coefficients as $\mathbf{p}$, constraints $\mathcal{C}$, initial state $x_0$, final state $x_f$ and time $t$
    **if** Optimization times out **then**
        Set $t_{min} = t$
    **else**
        Set $t_{max} =$ t
    **end if**
**end while**

### 3.4.2. Distilling expert data into a single goal conditioned policies

We outline the pre-training phase for learning the goal-conditioned locomotion skills. We train the goal-conditioned policy that maps state observations and goals in the form of control actions. We detail the corresponding state, goals, and action spaces for the pre-training phase. We then leverage the goal-conditioned variant of TD3-BC for offline reinforcement learning to train goal-conditioned control policies on a sparse reward state.

**State, Goals, Actions, and Rewards**

The state is represented by position, velocity, orientation, and body rates. The goals for the aerial vehicle are represented by position, velocity, and orientation. Given that the quadrotor is translationally invariant and is navigating in free space, the state and the goals for the aerial vehicle are made relative to its absolute position in space. Hence, the position vector for the state is always 0. We specify the orientation for both the state and the goals as rotation matrices in the global coordinate frame. Furthermore, the velocities and body rates are also specified in the global coordinate frame. Note that we don't have body rate vectors specified as goals as this was observed to result in poor-quality expert data.

Many action spaces have been proposed for quadrotor control [28] such as Linear Velocities and Yaw rate (LV), Collective Thrust and Body Rate (CTBR),and Single Rotor Thrusts(SRT). As control policies in the form of CTBR were been shown to be stable to train, and robust to dynamics mismatches and controller delays, we choose this particular abstraction of the action space for our application. The CTBR action space commands the aerial vehicle body thrusts and angular velocities. This action space computes the thrust commands for each propeller using an internal PD controller.

The reward functions in a simple binary single indicating whether the desired goal has been reached i.e.

$$r_g(s_t, a_t, g) = 1 \ (goal\ reached)$$

The criterion for reaching the goal is the proximity of the aerial vehicle to its goal along all the

position, velocity, and orientation goal dimensionalities. Hence, the reward is received if the distances are smaller than a small number $\epsilon$ i.e $||s_{t+1} - g|| \leq \epsilon$. We use the following conditions for the position, orientation and velocity dimensionality of the goals $\epsilon_p = 0.15, \epsilon_r = 0.15, \epsilon_v = 0.35$.

**Policy Learning**

The policy objective is to maximize the cumulative goal-conditioned reward introduced in Eq 1.1. However, unlike online GCRL we don't have access to the trajectories that would be generated by the goal-directed agent. Hence, we command the agent with goals drawn from the dataset $\rho_{\mathcal{D}}(g)$ instead of $\rho(g)$(See Sec. 1.2.4). We leverage offline reinforcement learning, more specifically GCSL [19], that optimizes the behavioral goal-conditioned policy $\pi(a|s, g)$. Compared to other goal-conditioned offline RL methods [91, 39], TD3BC has much lesser complexity and fewer number of hyperparameters. While TD3BC [17] is observed to have better performance on high-quality expert data on standard RL benchmark tasks, we observe that GCSL exhibits better performance for policy learning.

The policy is optimized using the following cost function.

$$\pi^* = \underset{\pi}{argmax}\, \mathbb{E}_{s,a\sim\mathcal{D},g\sim\rho_{\mathcal{D},ag}(g)} \left[ \; (\pi(s, g) - a)^2 \right] \tag{3.1}$$

Since the goal-conditioned reward is sparsely obtained only on reaching the goal, this would result in slow policy convergence. Hence we also relabel the desired goals from the expert dataset $\rho_{\mathcal{D}}(g)$ with goals achieved by the expert for the same trajectories in hindsight to alleviate the sparse reward signal problem. More specifically, we relabel $g \sim \rho_{\mathcal{D}}(g)$ with the goals achieved by the expert $g_{ag} \sim \rho_{\mathcal{D}}(.|g)$ when commanded with the desired goal $g$. No additional expert data is collected and the relabelling occurs with future achieved goals only over a single trajectory that reaches a goal $g$ in $\rho_{\mathcal{D}}(g)$.

An additional advantage of this offline relabelling scheme is that this permits the policy to learn to reach goals in the desired goal distribution $g \in \rho_{\mathcal{D}}(g)$ but also the intermediary goals that the expert achieves. The full algorithm is outlined in Alg 2.

**Value Learning**

The Q network is learned in standard TD3 style to prevent the Q networks from overestimating the returns on in-distribution actions since the maximum goal-conditioned reward is clipped at 1. We use five state action critic networks parametrized by $\phi_j$ in contrast to TD3 where two critic networks are trained. Additionally, we maintain the target networks parameterized by $\phi'_j$ whose weight updates are temporally delayed. These target networks are used the compute the state action critics at the future state whereby the action is sampled from the behavioral policy learned through GCSL. These targets are clipped targets to lie between 0 and 1 given that there is a sparse reward for reaching a goal to prevent the overestimation of the state action utilities. The TD3 style state action value loss is given as follows.

$$\mathbb{E}_{s,a,s' \sim \mathcal{D}, g \sim \rho_{\mathcal{D},ag}(g)} \left[ clip \left( r(s,a,g) + \gamma \min_i \mathbb{E}_{a' \sim \pi(s',g)} \left[ Q_{\phi'_j}(s',a',g) \right] \right) - Q_{\phi_j}(s,a,g) \right]^2 \quad (3.2)$$

Learning this value network is critical for the planning phase described in the next section.

**Algorithm 2** Learning the goal conditioned policy

1: **Initialize:**
   Buffer $\mathcal{B} = \{\tau\}_{i=1}^{N} = \{\{s_i^j, a_i^j, r_i^j, g_i, ag_i^j s_i^{j+1}\}_{j=1}^{T}\}_{i=1}^{N}$, policy parameters $\theta$,
   critic parameters $\phi_j$ and critic target parameters $\phi'_j$, discount factor $\gamma$ ,
   relabel proportion $\delta$, $M$ epochs , target weight update ratio $\beta$
2: **for** step = 1 to M **do**
3:  Sample Batch of transitions $B = \{s_i^j, a_i^j, r_i^j, g_i, ag_i^j s_i^{j+1}\}$ for some $j \in [1, T]$ and $i \in [1, N]$
4:  Relabel $g_i$ with $ag_i^k$ for some $j$ such that $k < T$ *and* $k > j$ with probability $\delta$
5:  Train $\theta$ with policy loss in Eq. 3.1
6:  Train each critic network $\phi_i$ with gradient of loss in Eq 3.2
7:  Update $\phi'_i = (1 - \beta) * \phi + \beta * \phi'_i$
8: **end for**

### 3.4.3. Planning with Goal conditioned policy

Given a test goal $g$ which lies beyond the distribution of the goals that the aerial vehicle was trained for, the objective of the planner is to plan a vector sequence of $n$ intermediate goals $\mathbf{g} = [g_i, i \in n]$ to reach $g$. The objective function is thereby given as follows:

$$\mathbf{g}^* = \max_{g_i \in \rho_{\mathcal{D},ag}(g)} \mathop{\mathbb{E}}_{\substack{s_{g_1} \sim p^{\pi(.|.,g_1)}(s_{g_1}|s_0,g_1) \\ s_{g_2} \sim p^{\pi(.|.,g_2)}(s_f|s_{g_1},g_2) \\ \dots \\ s_{g_n} \sim p^{\pi(.|.,g_n)}(s_f|s_{g_{n-1}},g_n)}} \left[ p^{\pi(.|s_{g_n},g)}(s_{t+} = s_g|s_{g_n}, g) \right] \tag{3.3}$$

Here each goal $g_{i+1}$ achievable through $g_i$ using the distilled goal-conditioned controller $\pi(.|s_{g_i}, g_{i+1})$ and this induces a trajectory $\tau^{i+1}$ which is conditioned on the goal $g_{i+1}$. Here $p^{\pi(.|.,g_i)}(s_f = s_{g_i}|s_{g_{i-1}}, g_i)$ is the discounted state occupancy measure [76, 14] i.e of the goal conditioned policy and $s_g$ is the terminal state of the policy when commanded with the goal $g$. The discounted state occupancy is given as $p^{\pi(.|.,g_i)}(s_f = s_{g_i}|s_{g_{i-1}}, g_i) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t p_t^{\pi(.|.,g_i)}(s_f = s_g)$.

The optimization objective can be expanded with the discounted state visitations as follows:

$$\mathbf{g}^* = \max_{g_i \in \rho_{\mathcal{D},ag}(g)} \int \int \dots \int \Big[ p^{\pi(.|s_{g_n},g)}(s_{t+} = s_g|s_{g_n}, g)\ p^{\pi(.|.,g_1)}(s_{g_1}|s_0, g_1) \\ p^{\pi(.|.,g_2)}(s_f|s_{g_1}, g_2) \dots\ p^{\pi(.|.,g_n)}(s_f|s_{g_{n-1}}, g_n) ds_{g_1} ds_{g_2} .. ds_{g_n} \tag{3.4}$$

Leveraging the equivalence between the discounted state distribution of the goal-conditioned policy and the goal-conditioned state action critic from [14] under the assumption of the goal reaching reward, the optimization can be written as:

$$\mathbf{g}^* = \max_{g_i \in \rho_{\mathcal{D},ag}(g)} \int \int \dots \int Q(s_{g_n}, \pi(.|s_{g_n}, g), g)\ Q(s_0, \pi(.|s_0, g_1), g_1) \\ Q(s_{g_1}, \pi(.|s_{g_1}, g_2), g_2) \dots\ Q(s_{g_{n-1}}, \pi(.|s_{g_{n-1}}, g_n), g_n) ds_{g_1} ds_{g_2} .. ds_{g_n} \tag{3.5}$$

### 3.4.4. Simplifying the planning objective

Solving the optimization problem directly is intractable due to the presence of an integral over the terminal state occupancy distributions of the individual goal-conditioned policies. Furthermore, optimizing the objective in Eq 3.6 is infeasible as it assumes complete data coverage to evaluate the integral accurately. Since the skills are distilled from offline data, the optimization objective with critics trained on insufficient data coverage over state goal pairs would be biased and can be overly optimistic. Second, we assume that we have a well-behaved policy such that terminal state distribution conditioned on a single goal is Dirac delta. Note that since we learn 5 critic networks, we use the disagreement between these 5 critics to be pessimistic with respect to the uncertain state goal pairs. Disagreement is a standard objective for exploring uncovered skills or goals in many skill learning and goal-conditioned RL works [43, 49, 5]. Hence, we propose an alternative objective to optimize for intermediate goals as follows:

$$\mathbf{g}^* = \underset{g_i \in \rho_{\mathcal{D},ag}(g)}{max} \Bigg[ \frac{1}{m} \sum_{j=1}^{m} \Bigg( Q_{\phi_j}(s_{g_n}, \pi(.|s_{g_n}, g), g) \prod_{i}^{n} Q_{\phi_j}(s_{g_{i-1}}, \pi(.|s_{g_{i-1}}, g_i), g_i) Q_{\phi_j}(s_0, \pi(.|s_0, g_1), g_1) \Bigg)$$
$$-\alpha \left( \sum_{1}^{n} var_j \left[ Q_{\phi_j}(s_{g_{i-1}}, \pi(.|s_{g_{i-1}}, g_i), g_i) \right] + var_j \left[ Q_{\phi_j}(s_{g_n}, \pi(.|s_{g_n}, g), g) \right] + var_j \left[ Q_{\phi_j}(s_0, \pi(.|s_0, g_1), g_1) \right] \right) \Bigg] \tag{3.6}$$

Here, $var_j$ is the variance and $m$ is the number of critics. In the experiments below, we sample a single intermediate goal $g*$ to reach an out-of-distribution goal $G$ from the initial state $s_0$, thereby the objective is

$$g* = \underset{g \in \rho_{\mathcal{D},ag}(g)}{max} \frac{1}{m} \sum_{j=1}^{m} \left( Q_{\phi_j}(s_g, \pi(.|s_g, G), G) Q_{\phi_j}(s_0, \pi(.|s_0, g), g) \right)$$
$$- \alpha \left( var_j \left[ Q_{\phi_j}(s_g, \pi(.|s_g, G), G) \right] + var_j \left[ Q_{\phi_j}(s_0, \pi(.|s_0, g), g) \right] \right). \tag{3.7}$$

## 3.5. Experiments

We aim to analyze the learned skills' quality and usefulness for a planning task by devising experiments to understand the following questions.

1. Are the quality of the skills sensitive to the choice of the offline RL algorithm used for skill distillation?

2. Can we plan to reach out-of-distribution goals by chaining the learned goal-conditioned skills?

### 3.5.1. Algorithm Sensitivity

We evaluate the skills learned by different offline RL algorithms using a fixed dataset that is procedurally generated using the algorithm outlined in Sec 3.4.1. In this evaluation, we evaluate four offline RL algorithms that have been adapted to the goal-conditioned counterparts [91]. To ensure fair and identical evaluation we use the author-specific implementations of the algorithms.

We compare and contrast our GCSL variant with CQL [31], TD3BC[17] and TD3-AWBC inspired from [91]. Note that for TD3-AWBC we only augment the behavior cloning loss introduced for policy regularisation [17] with advantage-weighted regression term. For CQL, we refer to the author specified for the critic regularisation. Across all algorithms, we maintain the standard policy and critic architectures which are MLPs with 4 layers of sizes (512,2048,2048,512).

**Evaluation Tasks**

Each algorithm is run for 2 million steps with a batch size of 1048 state action transitions. We run two different sets of evaluations in parallel. First, we evaluate the learned goal-conditioned policies from offline data with respect to the desired distribution of goals $\rho_{\mathcal{D}}(g)$. Here, the quadrotor is initialized from the origin with 0 velocities and no world frame orientation. Note that the offline data for the quadrotor is collected by setting goals from $\rho_{\mathcal{D}}(g)$ as terminal goal states. The first evaluation occurs every 5000 steps and comprises 25 episodes. Here, the success criterion is the policy's ability to efficiently reach the specific desired goals over which the original expert data was collected from the origin state.

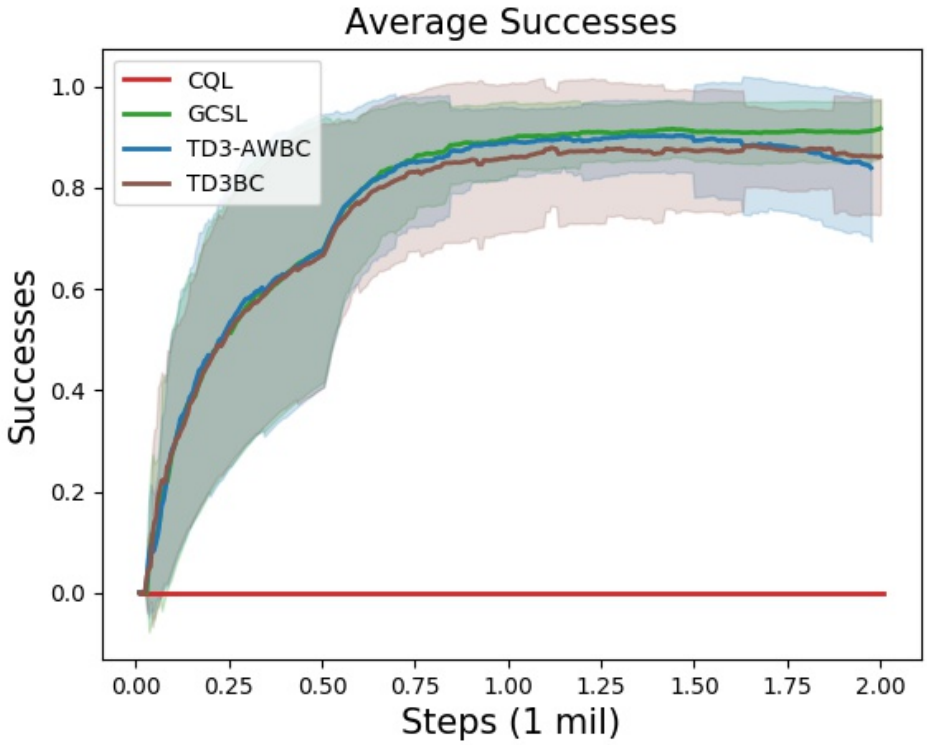


Figure 3.3: Skill learning performance of different algorithms over the desired goals $\rho_{\mathcal{D}}(g)$ over which the expert data is collected. The aerial vehicle is initialized at the origin.

Second, we also evaluate whether the goal-conditioned policy can reach intermediate achieved goals by the expert with high success from any initial state along the expert trajectory. More specifically, we initialize the quadrotor at a state along the expert trajectory for some desired goal and task it to reach a goal at most 45 actions away along the same expert trajectory.

The second set of evaluations assesses the adaptability of the goal-conditioned policy. Unlike existing methods for construction motion primitives, which may have fixed durations of time or sampled states, our evaluation demonstrates the policy's capability to reach intermediate achieved goals along the expert trajectory from an initial state. This adaptability is crucial in real-world scenarios, and by successfully reaching intermediate goals with high success rates, we demonstrate that our method learns fine-grained goal-conditioned skills. This evaluation occurs every 15000 steps and consists of 75 episodes. In this scenario, the quadrotor is initialized at different states along the expert trajectory for some desired goal. 25 of these episodes sample the goals from the original set

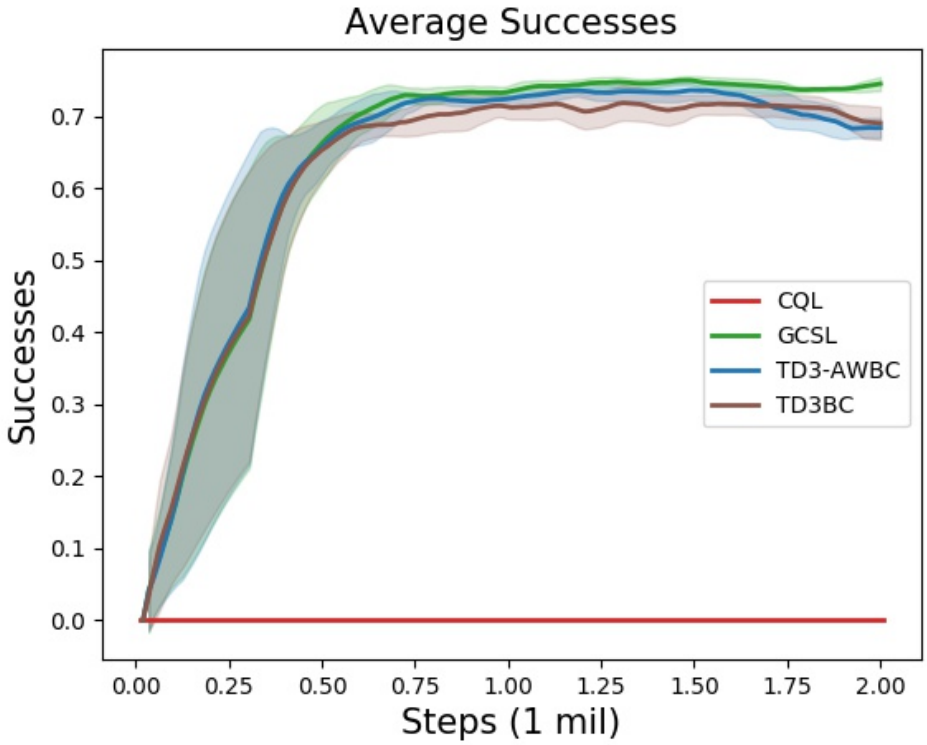


Figure 3.4: Skill learning performance of different algorithms over a mixture of intermediate achieved goals and desired goals from different initialization along the expert trajectory

of 250 desired goals using which the expert data was collected. The remaining 50 episodes test the policy's adaptability to reach any intermediate goal from a given initial state along the given expert trajectory data.

The average success rates of the learned goal-conditioned policy are summarised in Table 3.1 over the policy weights learned from the last 10 epochs. Here, the two tasks *Reach Desired* and *Reach Intermediate* are equivalent to the evaluation tasks mentioned for learning performance. In this

Table 3.1: Performance Comparisons of polices learned by different algorithms on two tasks *Reach-Desired* and *Reach-Intermediate*

| **Tasks** | **CQL (%)** | **TD3-BC (%)** | **TD3-AWBC (%)** | **GCSL (%)** |
|---|---|---|---|---|
| *Reach-Desired* | 0.00±0.00 | 85.20±4.29 | 82.80±7.28 | **93.2 ±3.78** |
| *Reach-Intermediate* | 0.00±0.00 | 67.6±2.23 | 66.8 ±3.52 | **75.6 ±3.78** |

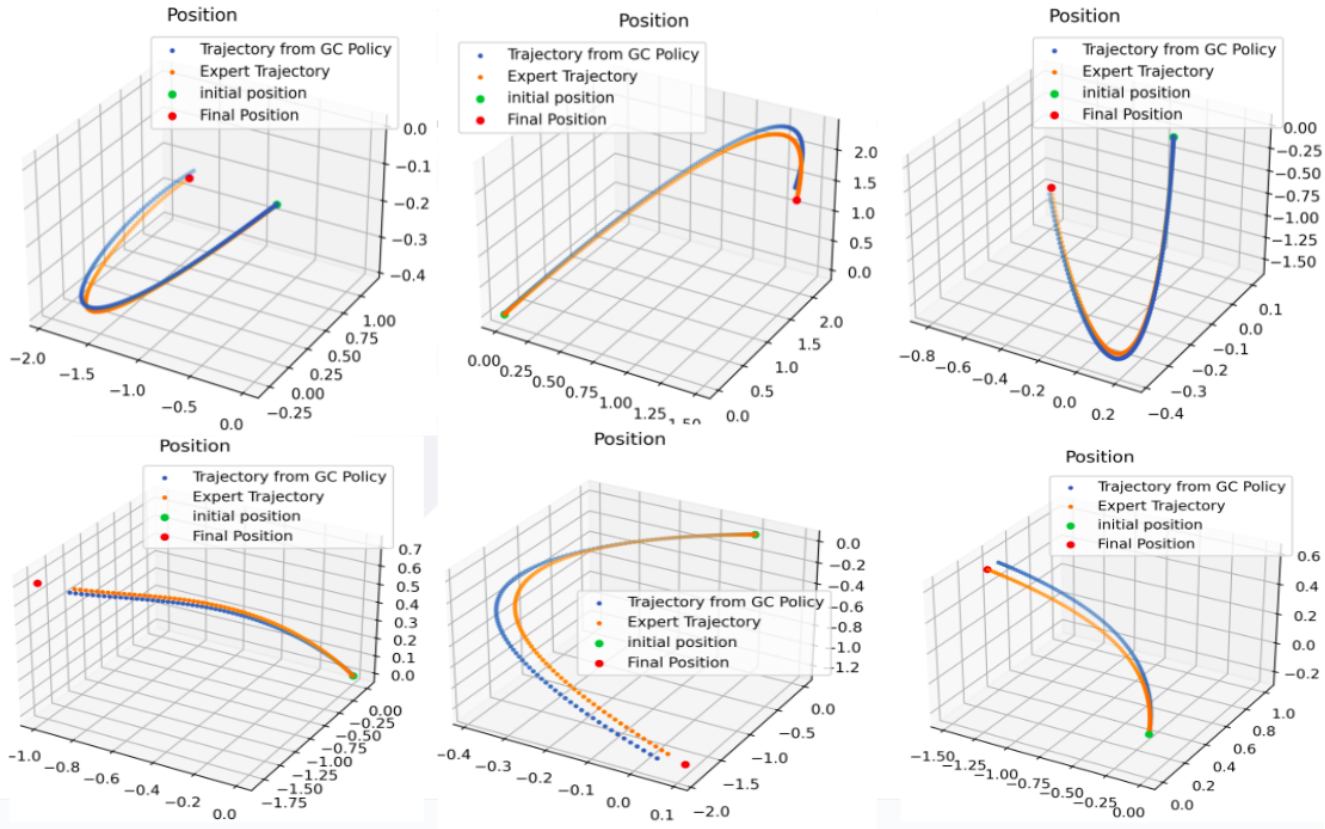


Figure 3.5: Trajectory visualizations of the goal conditioned policy over the *Reach-Desired* tasks

case, 250 episodes of tests are conducted for both tasks. We qualitatively visualize some of the trajectories resulting from the goal-conditioned policy learned through the GCSL policy loss in Fig 3.5.

Finally, we also summarize qualitatively and quantitatively the resulting trajectories obtained by planning a single intermediary goal and reaching an out-of-distribution goal. This dataset of 250 goals is different from the original desired goals $\rho_{\mathcal{D}}(g)$ over which the expert data was collected.

Table 3.2: Task performance over a set of 250 out-of-distribution goals with respect to a goal-conditioned policy with and without the planner

| **Method** | **Successes (%)** | **Near Successes (%)** |
|---|---|---|
| With Planner | 11.6 | 29.2 |
| Without Planner | 0.00 | 4.4 |

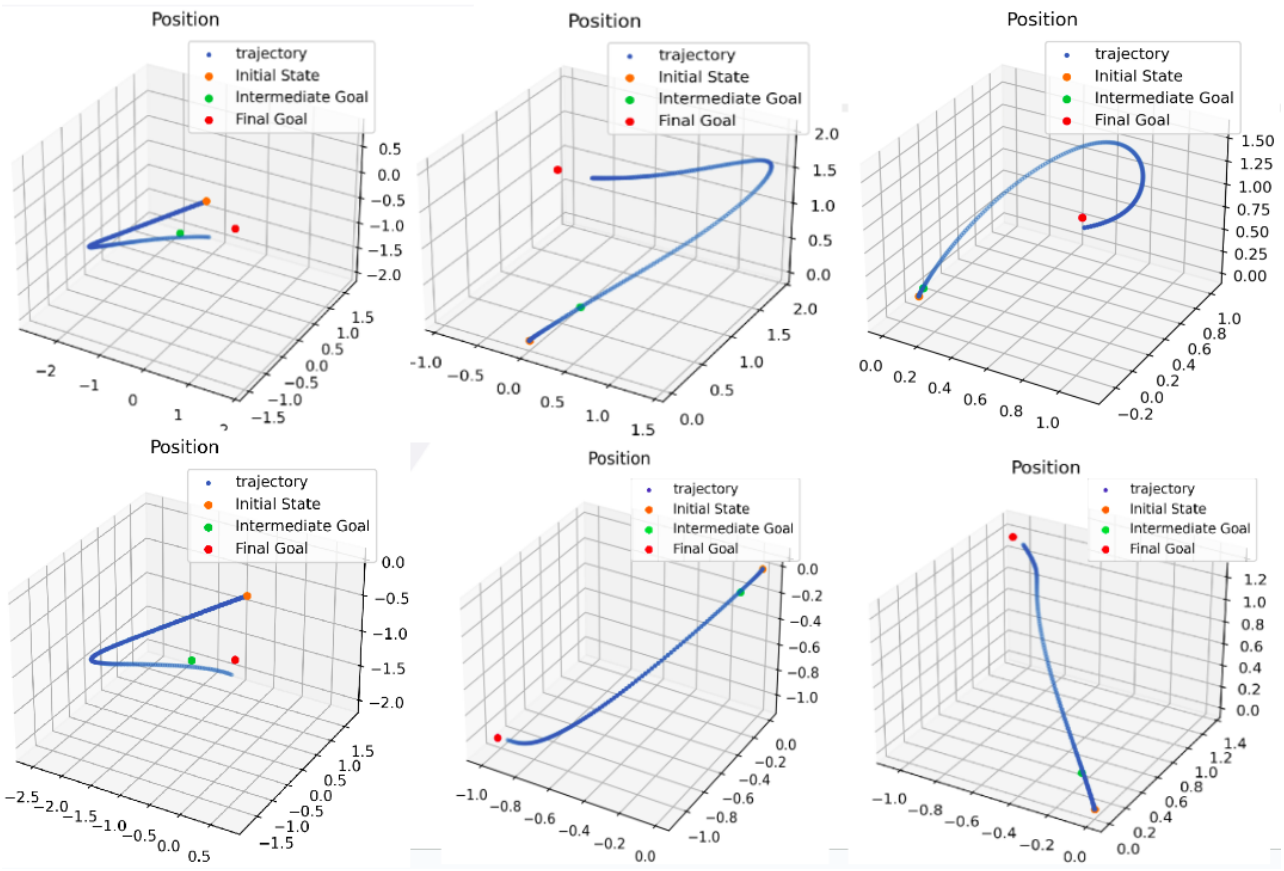


Figure 3.6: Visualization of successful trajectories where the goal-conditioned policy reaches an out-of-distribution goal (red) by planning an intermediate goal (green) from the initial state at origin (orange).

After planning an intermediate goal, the goal-conditioned policy is first tasked with reaching the intermediate goal in an expected time frame and then tasked to reach the final desired goal once the intermediate goal is reached. We compare this with the goal-conditioned policy tasked to reach the final goal directly. We define two metrics, success, and near success. The former metric measures successful episodes where the aerial vehicle reaches within $\epsilon$ bound (in Sec 3.4.2) of the goal. Near successes refer to episodes where the aerial reaches within $2.5\epsilon$ of the goal.

## 3.6. Conclusion

In conclusion, this chapter explores goal-conditioned learning to acquire continuous state-based locomotion skills for quadrotors. By distilling expert data comprising lower-level motor thrust commands or desired thrust and angular velocity commands, we create a goal-conditioned policy that allows the quadrotor to smoothly navigate towards selected goal states without having exclusively planning reference trajectories. Unlike previous approaches that involve fixed goal states, our method provides adaptability and flexibility in trajectory planning, enabling the quadrotor to reach a diverse range of goals efficiently.

The contributions of this chapter are two-fold. Firstly, we present an offline approach to distill a fundamental set of skills that enable the quadrotor to achieve proximal goals. These skills are later used in conjunction with a planning module to reach goals that are out of distribution with respect to the training goals. We highlight that this dedicated planning module which plans an intermediate goal reduces the gap between trajectory planning and control for aerial vehicles.

Overall, by combining the advantages of data-driven policy learning with the flexibility of dynamic trajectory planning, our method showcases the potential for closing the gap between trajectory planning and control for aerial vehicles in challenging real-world scenarios. The results obtained from the extensive evaluations demonstrate the efficacy and promise of our approach in achieving high-quality and dynamically feasible trajectories.

Future work can focus on developing more sophisticated planners to plan intermediary goals in an optimized manner. This can be extended to learning-based planning methods where a neural network is used to plan intermediate goals. Lastly, the quality of expert data over which the goal-conditioned policy is distilled from the effects of the generalization capability of the higher-level planner for general goals in the environment. Future work can focus on autonomously acquiring such skills.

# BIBLIOGRAPHY


[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[3] Jonathan Binney and Gaurav S Sukhatme. Branch and bound for informative path planning. In *2012 IEEE international conference on robotics and automation*, pages 2147–2154. IEEE, 2012.

[4] Hermann Blum, Silvan Rohrbach, Marija Popovic, Luca Bartolomei, and Roland Siegwart. Active learning for uav-based semantic mapping, 2019. URL https://arxiv.org/abs/1908.11157.

[5] Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020.

[6] Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jake Varley, Alex Irpan, Benjamin Eysenbach, Ryan Julian, Chelsea Finn, et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *arXiv preprint arXiv:2104.07749*, 2021.

[7] Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of*

*Artificial Intelligence Research*, 74:1159–1199, 2022.

[8] Murtaza Dalal, Deepak Pathak, and Russ R Salakhutdinov. Accelerating robotic reinforcement learning via parameterized action primitives. *Advances in Neural Information Processing Systems*, 34:21847–21859, 2021.

[9] Todor Davchev, Oleg Sushkov, Jean-Baptiste Regli, Stefan Schaal, Yusuf Aytar, Markus Wulfmeier, and Jon Scholz. Wish you were here: Hindsight goal selection for long-horizon dexterous manipulation. *arXiv preprint arXiv:2112.00597*, 2021.

[10] Mihir Dharmadhikari, Tung Dang, Lukas Solanka, Johannes Loje, Huan Nguyen, Nikhil Khedekar, and Kostas Alexis. Motion primitives-based path planning for fast and agile exploration using aerial robots. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 179–185, 2020. doi: 10.1109/ICRA40945.2020.9196964.

[11] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

[12] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[13] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. *arXiv preprint arXiv:2011.08909*, 2020.

[14] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

[15] Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. *Advances in neural information processing systems*, 32, 2019.

[16] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 6(56), jul 2021. doi: 10.1126/scirobotics.abh1221. URL https://doi.org/10.1126%2Fscirobotics.abh1221.

[17] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[18] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous systems*, 61(12):1258–1276, 2013.

[19] Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

[20] Harsh Goel, Laura Jarin Lipschitz, Saurav Agarwal, Sandeep Manjanna, and Vijay Kumar. Reinforcement learning for agile active target sensing with a uav, 2022.

[21] Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *arXiv preprint arXiv:2206.04114*, 2022.

[22] Zhichao Han, Zhepei Wang, Neng Pan, Yi Lin, Chao Xu, and Fei Gao. Fast-racing: An open-source strong baseline for se(3) planning in autonomous drone racing, 2021.

[23] Geoffrey A. Hollinger and Gaurav S. Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014.

[24] Geoffrey A Hollinger, Brendan Englot, Franz S Hover, Urbashi Mitra, and Gaurav S Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *The International Journal of Robotics Research*, 32(1):3–18, 2013. doi: 10.1177/0278364912467485. URL https://doi.org/10.1177/0278364912467485.

[25] Edward S Hu, Richard Chang, Oleh Rybkin, and Dinesh Jayaraman. Planning goals for exploration. *arXiv preprint arXiv:2303.13002*, 2023.

[26] Laura Jarin-Lipschitz, James Paulos, Raymond Bjorkman, and Vijay Kumar. Dispersion-minimizing motion primitives for search-based motion planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12625–12631. IEEE, 2021.

[27] Laura Jarin-Lipschitz, Xu Liu, Yuezhan Tao, and Vijay Kumar. Experiments in adaptive replanning for fast autonomous flight in forests. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8185–8191, 2022. doi: 10.1109/ICRA46639.2022.9812235.

[28] Elia Kaufmann, Leonard Bauersfeld, and Davide Scaramuzza. A benchmark comparison of learned control policies for agile quadrotor flight. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10504–10510. IEEE, 2022.

[29] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[30] Andreas Krause. *Optimizing Sensing: Theory and Applications*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2008.

[31] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–

1191, 2020.

[32] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 2022.

[33] Gaston Lenczner, Adrien Chan-Hon-Tong, Bertrand Le Saux, Nicola Luminari, and Guy Le Besnerais. Dial: Deep interactive and active learning for semantic segmentation in remote sensing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:3376–3389, 2022. doi: 10.1109/JSTARS.2022.3166551.

[34] Guanrui Li, Alex Tunchez, and Giuseppe Loianno. Learning model predictive control for quadrotors. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5872–5878. IEEE, 2022.

[35] Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

[36] Sikang Liu, Nikolay Atanasov, Kartik Mohta, and Vijay Kumar. Search-based motion planning for quadrotors using linear quadratic minimum time control. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2872–2879. IEEE, 2017.

[37] Sikang Liu, Kartik Mohta, Nikolay Atanasov, and Vijay Kumar. Search-based motion planning for aggressive flight in se (3). *IEEE Robotics and Automation Letters*, 3(3):2439–2446, 2018.

[38] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on robot learning*, pages 1113–1132. PMLR, 2020.

[39] Jason Yecheng Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. Offline goal-conditioned reinforcement learning via $f$-advantage regression. *Advances in Neural Information Processing Systems*, 35:310–323, 2022.

[40] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv preprint arXiv:2003.06085*, 2020.

[41] Ajith Anil Meera, Marija Popović, Alexander Millane, and Roland Siegwart. Obstacle-aware adaptive informative path planning for uav-based target search. In *International Conference on Robotics and Automation (ICRA)*, pages 718–724, 2019.

[42] Daniel Mellinger. *Trajectory generation and control for quadrotors.* University of Pennsylvania, 2012.

[43] Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021.

[44] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[45] Brady Moon, Satrajit Chatterjee, and Sebastian Scherer. Tigris: An informed sampling-based algorithm for informative path planning. *arXiv preprint arXiv:2203.12830*, 2022.

[46] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical

reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[47] Farzad Niroui, Kaicheng Zhang, Zendai Kashino, and Goldie Nejat. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019.

[48] Mohammad Nabi Omidvar and Xiaodong Li. A comparative study of cma-es on large scale global optimisation. In *Australasian Joint Conference on Artificial Intelligence*, pages 303–312. Springer, 2010.

[49] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pages 5062–5071. PMLR, 2019.

[50] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions On Graphics (TOG)*, 41(4):1–17, 2022.

[51] Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pages 7750–7761. PMLR, 2020.

[52] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

[53] Marija Popović, Gregory Hitz, Juan Nieto, Inkyu Sa, Roland Siegwart, and Enric Galceran. Online informative path planning for active classification using UAVs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5753–5758, 2017.

[54] Marija Popović, Teresa Vidal-Calleja, Jen Jen Chung, Juan Nieto, and Roland Siegwart. Informative path planning for active field mapping under localization uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 10751–10757, 2020.

[55] Doina Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

[56] Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. *Advances in Neural Information Processing Systems*, 32, 2019.

[57] Angel Romero, Sihao Sun, Philipp Foehn, and Davide Scaramuzza. Model predictive contouring control for time-optimal quadrotor flight. *IEEE Transactions on Robotics*, 38(6):3340–3356, 2022. doi: 10.1109/TRO.2022.3173711.

[58] Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning*, pages 1838–1849. PMLR, 2023.

[59] Reuven Y Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

[60] Julius Rückin, Liren Jin, and Marija Popović. Adaptive informative path planning using deep reinforcement learning for uav-based active sensing. In *International Conference on Robotics and Automation (ICRA)*, pages 4473–4479, 2022.

[61] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[62] Julius Rückin, Liren Jin, Federico Magistri, Cyrill Stachniss, and Marija Popović. Informative

path planning for active learning in aerial semantic mapping, 2022. URL https://arxiv.org/abs/2203.01652.

[63] Inkyu Sa, Marija Popović, Raghav Khanna, Zetao Chen, Philipp Lottes, Frank Liebisch, Juan Nieto, Cyrill Stachniss, Achim Walter, and Roland Siegwart. Weedmap: A large-scale semantic weed mapping framework using aerial multispectral imaging and deep neural network for precision farming. *Remote Sensing*, 10(9), 2018. ISSN 2072-4292. doi: 10.3390/rs10091423. URL https://www.mdpi.com/2072-4292/10/9/1423.

[64] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

[65] Lukas Schmid, Michael Pantic, Raghav Khanna, Lionel Ott, Roland Siegwart, and Juan Nieto. An efficient sampling-based method for online informative path planning in unknown environments. *IEEE Robotics and Automation Letters*, 5(2):1500–1507, 2020.

[66] Tanmay Shankar, Shubham Tulsiani, Lerrel Pinto, and Abhinav Gupta. Discovering motor programs by recomposing demonstrations. In *International Conference on Learning Representations*, 2020.

[67] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

[68] Mohit Sharma, Jacky Liang, Jialiang Zhao, Alex LaGrassa, and Oliver Kroemer. Learning to compose hierarchical object-centric controllers for robotic manipulation. *arXiv preprint arXiv:2011.04627*, 2020.

[69] Lucy Xiaoyang Shi, Joseph J Lim, and Youngwoon Lee. Skill-based model-based reinforcement

learning. *arXiv preprint arXiv:2207.07560*, 2022.

[70] Amarjeet Singh, Andreas Krause, and William J. Kaiser. Nonmyopic adaptive informative path planning for multiple robots. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, page 1843–1850, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[71] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020.

[72] Felix Stache, Jonas Westheider, Federico Magistri, Cyrill Stachniss, and Marija Popović. Adaptive path planning for uavs for multi-resolution semantic segmentation. *Robotics and Autonomous Systems*, 159:104288, 2023. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2022.104288. URL https://www.sciencedirect.com/science/article/pii/S0921889022001774.

[73] Junghun Suh, Kyunghoon Cho, and Songhwai Oh. Efficient graph-based informative path planning using cross entropy. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5894–5899, 2016. doi: 10.1109/CDC.2016.7799176.

[74] Sihao Sun, Angel Romero, Philipp Foehn, Elia Kaufmann, and Davide Scaramuzza. A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight. *IEEE Transactions on Robotics*, 38(6):3357–3373, 2022.

[75] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[76] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. MIT press

Cambridge, 1998.

[77] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

[78] Daniel Tanneberg, Kai Ploeger, Elmar Rueckert, and Jan Peters. Skid raw: Skill discovery from raw trajectories. *IEEE Robotics and Automation Letters*, 6(3):4696–4703, 2021.

[79] Mehrdad Tavassoli, Sunny Katyara, Maria Pozzi, Nikhil Deshpande, Darwin G Caldwell, and Domenico Prattichizzo. Learning skills from demonstrations: A trend from motion primitives to experience abstraction. *arXiv preprint arXiv:2210.08060*, 2022.

[80] Sebastian Thrun. Learning occupancy grids with forward models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1676–1681, 2001.

[81] Alberto Viseras and Ricardo Garcia. Deepig: Multi-robot information gathering with deep reinforcement learning. *IEEE Robotics and Automation Letters*, 4(3):3059–3066, 2019.

[82] Kelen CT Vivaldini, Thiago H Martinelli, Vitor C Guizilini, Jefferson R Souza, Matheus D Oliveira, Fabio T Ramos, and Denis F Wolf. Uav route planning for active disease classification. *Autonomous robots*, 43(5):1137–1153, 2019.

[83] Dustin J Webb and Jur Van Den Berg. Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics. In *2013 IEEE international conference on robotics and automation*, pages 5054–5061. IEEE, 2013.

[84] William Whitney, Rajat Agarwal, Kyunghyun Cho, and Abhinav Gupta. Dynamics-aware

embeddings. *arXiv preprint arXiv:1908.09357*, 2019.

[85] Thomas Wiedemann, Cosmin Vlaicu, Josip Josifovski, and Alberto Viseras. Robotic information gathering with reinforcement learning assisted by domain knowledge: an application to gas source localization. *IEEE Access*, 9:13159–13172, 2021.

[86] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

[87] Kevin Xie, Homanga Bharadhwaj, Danijar Hafner, Animesh Garg, and Florian Shkurti. Latent skill planning for exploration and transfer. *arXiv preprint arXiv:2011.13897*, 2020.

[88] Haoran Xu, Li Jiang, Li Jianxiong, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 4085–4098, 2022.

[89] Rui Yang, Meng Fang, Lei Han, Yali Du, Feng Luo, and Xiu Li. Mher: Model-based hindsight experience replay. *arXiv preprint arXiv:2107.00306*, 2021.

[90] Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl. *arXiv preprint arXiv:2202.04478*, 2022.

[91] Wenyan Yang, Huiling Wang, Dingding Cai, Joni Pajarinen, and Joni-Kristen Kämäräinen. Swapped goal-conditioned offline reinforcement learning. *arXiv preprint arXiv:2302.08865*, 2023.

[92] Andrii Zadaianchuk, Georg Martius, and Fanny Yang. Self-supervised reinforcement learning

with independently controllable subgoals. In *Conference on Robot Learning*, pages 384–394. PMLR, 2022.