A Dynamics and Task Decoupled Reinforcement Learning Architecture for High-Efficiency Dynamic Target Intercept

Dora D. Liu[*1,3], **Liang Hu**[*†2,1], **Qi Zhang**[†4,1], **Tangwei Ye**[1], **Usman Naseem**[5], **Zhong Yuan Lai**[1]

[1]DeepBlue Academy of Sciences
[2]Tongji University
[3]BirenTech Research
[4]University of Technology Sydney
[5]University of Sydney
liudongmei_0506@163.com, lianghu@tongji.edu.cn, qi.zhang-13@student.uts.edu.au
ytw_email@163.com, usman.naseem@sydney.edu.au, abrikosoff@yahoo.com

Abstract

Due to the flexibility and ease of control, unmanned aerial vehicles (UAVs) have been increasingly used in various scenarios and applications in recent years. Training UAVs with reinforcement learning (RL) for a specific task is often expensive in terms of time and computation. However, it is known that the main effort of the learning process is made to fit the low-level physical dynamics systems instead of the high-level task itself. In this paper, we study to apply UAVs in the dynamic target intercept (DTI) task, where the dynamics systems equipped by different UAV models are correspondingly distinct. To this end, we propose a dynamics and task decoupled RL architecture to address the inefficient learning procedure, where the RL module focuses on modeling the DTI task without involving physical dynamics, and the design of states, actions, and rewards are completely task-oriented while the dynamics control module can adaptively convert actions from the RL module to dynamics signals to control different UAVs without retraining the RL module. We show the efficiency and efficacy of our results in comparison and ablation experiments against state-of-the-art methods.

Introduction

Unmanned aerial vehicles (UAVs) have seen increasingly widespread use in various fields of activity in recent years, mainly due to their flexibility and ease of control. Nowadays, most use cases for UAVs are recreational in nature, e.g., photography and video-making, but this situation is changing rapidly to include usage in tasks, e.g., industrial maintenance. Increasingly, they are being deployed in more mission-critical tasks, e.g., path following (Rysdyk 2003), aerial refueling (Nalepka and Hinchman 2005), and aerial interception (Beard et al. 2002). These applications are often tightly dependent on flight control systems (FCSs), which need to preset different navigation, tracking, and interception rules for a specific task with clear boundaries, e.g., fixed rendezvouses and trajectories.

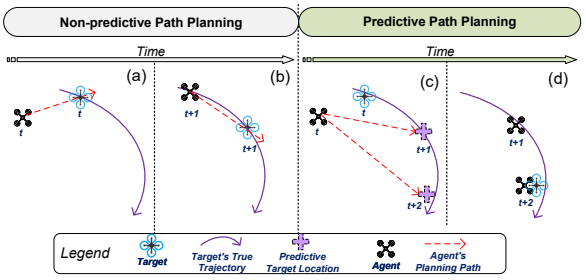


Figure 1: The demonstration of dynamic target intercept (DTI) process without predictive path planning (a, b) and with predictive path planning (c, d).

However, environments of many real-life tasks, e.g., aerial interception, are generally open and complex with limited information available to preset a complete rule base. As a result, rule-based FCSs may fail to accomplish missions with highly uncertain configurations. For example, Figure 1 demonstrates a case of intercepting a moving target UAV without knowing its flying trajectory. Figure 1 (a) and (b) depict an agent (i.e., the UAV serves as the interceptor), equipped with traditional FCSs, attempting to intercept a flying UAV by taking its current position as the target. Intuitively, this maneuver will fail to intercept the target for every subsequent time step, since the target has moved to the next waypoint whenever the agent arrives at the target position. This leads to the modern version of Zeno's paradox — "*Achilles (the interceptor) will never catch up with a tortoise (the target UAV)*". To successfully intercept the dynamic target, a feasible maneuver is to predict the trajectory of the target in advance. Figure 1 (c) and (d) depict the predictive path planning strategy for the agent to move, so it is possible to intercept the target at the predicted waypoints at future time steps. In this paper, we aim to find a general method that can effectively address the challenge of dynamic target intercept (DTI).

In the traditional FCSs, proportional-integral-derivative (PID) controllers are the most widely adopted. The most sig-

[*]These authors contributed equally.
[†]Corresponding authors

nificant weakness of PIDs is that they only function in closed environments where control behaviors are well-defined according to the UAVs' dynamical models. However, PID control is unable to work in a relatively open environment because it is almost impossible to design a huge set of omniscient PIDs that can deal with every uncertainty in this environment. Therefore, the DTI challenge addressed in this paper cannot be tackled by PIDs due to its dynamic and uncertain nature. To explore an open environment, deep reinforcement learning (DRL), in recent years, has become the best choice due to its high adaptability and flexibility in the context of dynamic uncertainty (Becker-Ehmck et al. 2020a; Azar et al. 2021). In this paper, DRL is also employed as a core component to implement DTI tasks, which enables the agents to learn and adapt their behaviors to accommodate the uncertain environmental context.

Apart from the DTI task studied in this paper, DRL has been equipped to carry out many other UAV tasks, such as autonomous navigation (Song et al. 2021) and optimum drone delivery (Munoz et al. 2019). In these tasks, DRL replaces controllers based on physical dynamical systems of UAVs. As a result, the design of DRL control models is tightly coupled with the dynamics of UAVs. More specifically, actions output by the DRL models is directly applied to control UAV models based on physical control signals, e.g. thrust and/or PWM (pulse-width modulation). Hence, the states of conventional DRL have to accommodate both task-specific variables (high-level states), e.g., positions and/or speed, and the physical variables (low-level states), e.g., RPM (revolutions per minute) and/or weight to learn physical controls. Accordingly, the reward functions are also dependent on the low-level physical states and actions. Unfortunately, such DRL models generally take a long time to train due to environmental uncertainty. If the underlying physical dynamics systems are too complex, DRL may fail to train to complete the high-level tasks because most of the effort would go towards learning how to control the complex dynamics models. Furthermore, the physical dynamics systems of different UAV types, e.g., fixed-wing UAVs and quadrotors, are generally different. As a result, we always need to design specialized states, actions, and rewards for different UAVs and train each of them.

To this end, we propose a Decoupling Trajectory prediction, Task learning, and Dynamics control (DTTD) architecture that decomposes the DTI task into three modules: (1) *Target's Trajectory Prediction Module* to learn and predict the trajectories of the dynamic target to intercept; (2) *Task's Interception Path Planning Module* to learn high-level interception path planning tasks with DRL; (3) *Agent's Dynamics Control Module* to convert high-level task-specific actions to low-level dynamics-specific signals to control the physical UAV models.

The main contributions of our paper are given as follows:

- We propose a DTTD interceptor that effectively reduces model complexity for DRL of UAV control by dividing high-level task-specific learners and low-level physical dynamics controllers into separative modules;
- Inspired by energy-based models, we propose *energy-based reward functions* that are appropriate for optimizing the high-level movement behaviors for the DTI task;
- Extensive experiments and demonstrations are conducted on the simulation environments. All results show that our approach achieves considerably better performance against the other comparison methods.

Related Work

Automated target intercept with UAVs is a relatively recent field of research (Triharminto, Adji, and Setiawan 2011; Himawan Triharminto et al. 2013). Target intercept requires both path planning and interception. Both these steps can be carried out independently and several previous works have combined different approaches.

Path Planning

Traditional Methods Chaudhuri and Konar (2007) propose a target tracking scheme based on the extended Kalman filters (EKF) where a series of dynamic images of the target are fed to the EKF in order to obtain predictions of the target path. The reviews (Montazeri, Can, and Imran 2021; Gasparetto et al. 2015; Danancier et al. 2019; Radmanesh et al. 2018) have conducted a comprehensive investigation on the traditional path planning algorithms for UAVs. However, all these algorithms have obvious weaknesses: they are hard to generalize to a dynamic setting with a moving target, and they also require knowledge of the complete map before being able to perform planning.

Reinforcement Learning Methods RL has become more popular in path planning for UAVs in recent years. Hwangbo et al. (2017) propose an early framework based on a deterministic on-policy algorithm which results in lower variance for the value/policy function estimate and hence more stable UAV control. Song et al. (2021) present a model-free on-policy RL framework for learning a minimum-time trajectory for drone racing. The authors faced problems dealing with the high-dimensional search space and complexity of the maneuvers due to the on-policy condition. In (Becker-Ehmck et al. 2020b), a model-based RL architecture is implemented for deep UAV control. To solve the problem of poor generalizability, the authors integrate a latent state space model (LSSM) into a traditional actor-critic framework. Finally, Belkhale et al. (2021) utilized a similar framework as in (Becker-Ehmck et al. 2020b) to deal with uncertainty in the environment, in this case coming from irregular weights of suspended payloads.

Target Intercept

Traditional Methods Target intercept is often achieved by state estimation and prediction methods, where Kalman filters (KF) (Himawan Triharminto et al. 2013) / EKFs (Montazeri, Can, and Imran 2021) are often applied. Nath, Sudheesh, and Jayakumar (2016) propose the use of an EKF to track and intercept inbound missiles, where robust and accurate interception performance from the EKF is obtained. Pan et al. (2010) apply the EKF to the task of 2D target interception by extending a traditional interception algorithm (the

augmented proportional navigational, APN) scheme and obtain robust results. Another variant of the conventional KF, the optimal KF, was also used in (Xiong and Zheng 2015) for variance minimization and robust state estimation.

Reinforcement Learning Methods RL techniques have also been applied to target intercept tasks, especially via the works of Gaudet (Gaudet, Furfaro, and Linares 2020; Gaudet et al. 2021; Gaudet, Linares, and Furfaro 2020). In (Gaudet, Furfaro, and Linares 2020) is proposed a meta-RL framework that performs target intercept tasks with only angle information as input to the learner. Gaudet et al. (2021) propose a meta-RL framework for adaptive guidance, navigation, and control of vehicles for exoatmospheric interception of maneuvering targets. Their framework can deal with commonly occurring parasitic effects. Gaudet, Linares, and Furfaro (2020) presents a meta-RL algorithm for the precision landing of a lunar/Mars lander module with multiple degrees of freedom. In this work, the authors introduced a meta-learning framework via recurrent layers in their policy networks which captures unobserved information in their persistent hidden states.

DTTD Training and Execution for DTI

As presented in the introduction, the physical dynamics systems of UAVs are generally complex, which presents difficulties when training a DRL-based controller with both task-specific and physical dynamics components. This is because the effort to learn realistic physical dynamics overwhelms that required to learn high-level strategies for carrying out the DTI task. In this section, we present details of our DTTD framework, which can significantly reduce the difficulties of learning high-level control while improving efficiency.

As illustrated in Figure 2, DTTD decompose the DTI task into three cohesively functional modules: (1) *Target's Trajectory Prediction Module*; (2) *Task's Interception Path Planning Module*; (3) *Agent's Dynamics Control Module*. These three modules can be trained or derived independently offline and work together to complete the DTI task during the online phase.

Target's Trajectory Prediction Module

This module aims to learn and predict trajectories of the dynamic target for interception. The trajectory of a target is composed of a sequence of waypoints for all time steps. Intuitively, multivariate time series (MTS) models can be employed to learn historical trajectories and predict the waypoints at future time stamps.

In this paper, we employ the following three MTS models:

- *LSTM* (Hochreiter and Schmidhuber 1997): The LSTM model uses the most recent N waypoints at time T as the input context to predict the future waypoint $\mathbf{p}_{T+1}$ at the next time step, and then $\mathbf{p}_{T+1}$ is added to the context to predict $\mathbf{p}_{T+2}$ at time $T+2$. This procedure is iterated to obtain M predictive waypoints $\{\mathbf{p}_{T+m}\}_{m=1}^{M}$;
- *Seq2seq* (Sutskever, Vinyals, and Le 2014): The sequence-to-sequence model consists of a encoder-decoder architecture, where the encoder is a LSTM network to encode N input waypoints into a persistent context representation $\mathbf{h}$, and then $\mathbf{h}$ is input into the decoder, which is also a LSTM network, to output the predicted waypoint sequence $\{\mathbf{p}_{T+m}\}_{m=1}^{M}$;
- *CNN-Seq2seq* (Donahue et al. 2016): This is an extension of the above Seq2seq model. First, the N-waypoint sequence is fed into a CNN, where a set of filters with different lengths are used to capture correlations at different timescales to generate the convolution features. Then, the convolutional feature sequence is input into Seq2seq to predict the future waypoint sequence $\{\mathbf{p}_{T+m}\}_{m=1}^{M}$.

Offline Phase In this phase, we perform offline trajectory learning for the above three MTS models over the historical trajectory database, as shown in the left part of Figure 2(a). These training trajectories are predefined by experts and/or collected from historical DTI tasks.

Online Phase In this phase, we first load the pretrained parameters obtained from the offline phase onto an agent to initialize its MTS network, as shown in the right part of Figure 2(a). During the interception process, the agent updates its MTS networks via online learning with the newly observed waypoints to better capture the target's recent behaviors. For each time step, the MTS models will output M predictive waypoint$\{\mathbf{p}_{T+m}\}_{m=1}^{M}$ for the *Task's Interception Path Planning Module*.

Task's Interception Path Planning Module

This module aims to learn high-level interception path planning tasks with DRL. In particular, we implement our DRL algorithm with the soft actor-critic (SAC) (Haarnoja et al. 2019) framework because SAC is an off-policy DRL algorithm that enables to generate actions in continuous spaces. Given the Markov Decision Process (MDP) $\langle S, A, R, P, \rho_\pi \rangle$ where the notation refers to states S, actions A, reward function R, and probability $P(s'|s,a)$ of transitioning into state $s' \in S$ from state $s \in S$ with action $a \in A$; $\rho_\pi(s)$ and $\rho_\pi(s,a)$ denote the state and the state-action marginals of the state distribution induced by the policy $\pi(a|s)$. Then, the optimal policy π^* for SAC is to find parameters θ to maximize the objective:

$$\pi_\theta^* = \arg\max_\theta \mathbb{E}_{(s_t,a_t)\sim\rho_{\pi_\theta}} \sum_{t=0}^{\infty} \gamma^t [R(s_t,a_t) + \alpha H(\pi_\theta(\cdot|s_t))] \quad (1)$$

where γ is the discount factor and α is a hyperparameter which fixes the contribution coming from the entropy regularization term $H(\pi_\theta(\cdot|s_t))$.

States and Actions As stated previously, states and actions should preferably be decoupled from the physical dynamics. In this paper, we only consider position related states because other critical variables related to UAV control can be easily derived from the position sequence, include the velocity $\mathbf{v}_t$ and acceleration $\mathbf{a}_t$ by using positions $\mathbf{p}_t = (x_t, y_t, z_t)$ and $\mathbf{p}_{t-1} = (x_{t-1}, y_{t-1}, z_{t-1})$ of two successive time steps.

$$\mathbf{v}_t = \dot{\mathbf{p}}_t = \mathbf{p}_t - \mathbf{p}_{t-1}, \quad \mathbf{a}_t = \dot{\mathbf{v}}_t = \mathbf{v}_t - \mathbf{v}_{t-1} \quad (2)$$

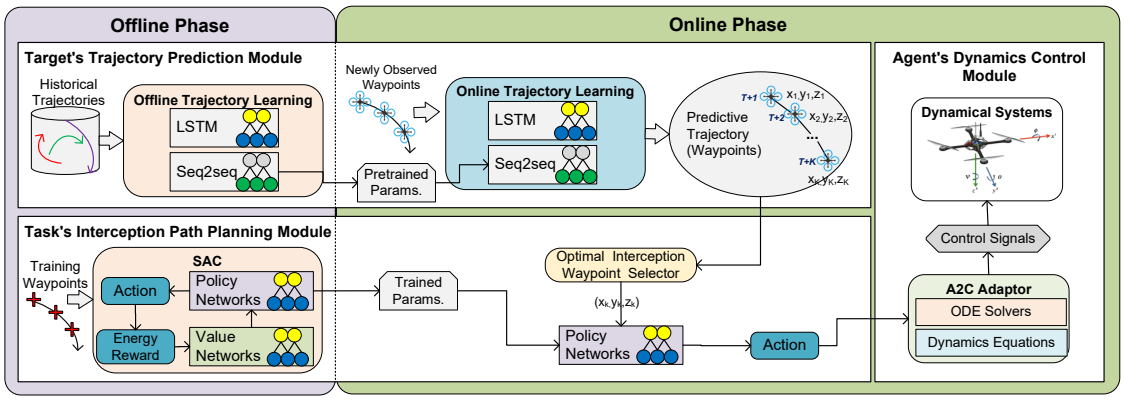


Figure 2: The Architecture of DTTD for High-efficiency DTI, which consists of three cohesively functional modules: (1) *Target's Trajectory Prediction Module*; (2) *Task's Interception Path Planning Module*; (3) *Agent's Dynamics Control Module*.

The state space for the DTI task consists of three position-derived spatial variables:

$$\mathbf{s}_t^{S2T} = \mathbf{p}_0 - \mathbf{p}_T, \mathbf{s}_t^{C2T} = \mathbf{p}_t - \mathbf{p}_T, \mathbf{s}_t^{P2T} = \mathbf{p}_{t-1} - \mathbf{p}_T \quad (3)$$

where $\mathbf{s}_t^{S2T} \in \mathbb{R}^3$ is the state to provide the spatial information from agent's start position to the target position; $\mathbf{s}_t^{C2T} \in \mathbb{R}^3$ describes the spatial relation from agent's position at the current step to the target position; and $\mathbf{s}_t^{P2T} \in \mathbb{R}^3$ describes the spatial relation from agent's position at the previous step to the target position; As a result, we obtain the state vector at time t, $\mathbf{s}_t = [\mathbf{s}_t^{S2T}, \mathbf{s}_t^{C2T}, \mathbf{s}_t^{P2T}] \in \mathbb{R}^9$.

In the DTI task, the actions guide the movement of the agent to the next position on the planning path. Therefore, we set the action vector as the amount of movement along each axis, i.e., $\mathbf{a}_t = [\Delta x_t, \Delta y_t, \Delta z_t] \in [-v_{\max}, v_{\max}]^3$, where $v_{\max}$ denotes the maximum speed of the agent w.r.t. a time step. Given action $\mathbf{a}_t$, we immediately obtain the agent's next position $\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{a}_t$ and next states $\mathbf{s}_{t+1}$ by Eq. 3.

Energy-based Reward Function The reward function is critical in RL, which directly determines the learning performance. For the DTI task, the design of reward functions needs to address three challenges: *C1. Sparsity*, *C2. High-dimensional search space*, and *C3. Physical dynamics dependency*. First, *C1* arises due to the nature of intercept tasks: normally, the agent is considered to have a positive reward only if it successfully intercepts the target within an episode. Second, the 3D continuous spatial space needs huge amounts of explorations for training, which results in *C2*. Moreover, the reward function is often coupled with physical dynamics states instead of task-oriented only (*C3*).

An energy-based model (EBM) (LeCun et al. 2006) is governed by an energy function that describes a certain state, where the energy function is borrowed from physical systems to describe the movement inside this system. Well-designed energy functions give lower energy to the correct values and higher energy to incorrect values (LeCun et al. 2006). Inspired by EBMs, we direct our efforts to design energy-based reward functions (ERF) since our task aims to model and control the movement of an agent. Following this idea, we design a system assigning lower energies (higher rewards) to a strategy with more efficient movement. According to state (Eq. 3), the following energies are defined:

$$E_c(\mathbf{p}_t|\mathbf{p}_0, \mathbf{p}_T) = -\tanh\left(E_{\text{closeness}}/\tau\right) \quad (4)$$
$$E_m(\mathbf{p}_t|\mathbf{p}_{t-1}) = -\tanh\left(E_{\text{movement}}/\tau\right) \quad (5)$$
$$E_s(t|\mathbf{p}_0, \mathbf{p}_T) = -\left(tol - E_{\text{step}}\right) \quad (6)$$

Here, temperature parameter $\tau = 2$ is set in this paper, tol is the tolerance rate allowed to exceed E_{step}, and

$$E_{\text{closeness}} = (\|\mathbf{s}_t^{S2T}\| - \|\mathbf{s}_t^{C2T}\|)/\|\mathbf{s}_t^{S2T}\| \quad (7)$$
$$E_{\text{movement}} = (\|\mathbf{s}_t^{P2T}\| - \|\mathbf{s}_t^{C2T}\|)/v_{\max} \quad (8)$$
$$E_{\text{step}} = (\lceil\|\mathbf{s}_t^{C2T}\|/v_{\max}\rceil \quad (9)$$

Obviously, E_c (Eq. 4) has lower energy when the agent is closer to the target position. E_m (Eq. 5) is assigned with lower energy for larger movement towards the target position. E_s (Eq. 6) has lower energy for fewer time steps.

Accordingly, we define the reward based on the energy functions Eq. 4-6.

$$R(\mathbf{s}_t, \mathbf{a}_t) = r_c + r_v + r_s \quad (10)$$

where
$$r_c = \exp\left(-\alpha_c E_c + \beta_c\right) \quad (11)$$
$$r_m = \exp\left(-\alpha_m E_m + \beta_m\right) \quad (12)$$
$$r_s = \begin{cases} \alpha_s(1 - \exp\left(E_s\right)) & \text{if } E_s < 0 \\ -\lambda_s E_s & \text{otherwise} \end{cases} \quad (13)$$

where $\alpha_c, \alpha_m, \alpha_s, \lambda_s$ are the scale parameters and β_c, β_m are the positive bias. We set $\alpha_c = 2, \alpha_m = 2, \alpha_s = 10, \lambda_s = 2, \beta_c = 2, \beta_m = 1$ in this paper by empirical evaluations.

- r_c tends to higher values with lesser energy E_c (Eq. 4). As a result, it forces the agent to approach the target. The exponential function assigns increasingly larger rewards as the agent approaches the target;
- r_m tends to higher values with lesser energy E_m (Eq. 5). As a result, it encourages faster movement of the agent towards the target position;

- r_s tends to higher values with the lesser energy E_s (Eq. 6). It assigns a higher reward to lesser moving steps below some tolerance tol while penalizing too many steps.

The ERF effectively tackles the challenges *C1-C3*. The reward components r_c, r_m, r_s address *C1* by providing progressive rewards for each step. To address *C2*, r_c encourages efficient movement and r_s penalizes the spatial search with too many steps. Moreover, our state design is free of physical dynamics while our ERFs are based on these states (see Eq. 4-6), so it naturally avoids *C3*.

Offline Phase As shown in the left part of Figure 2(b), we offline train path planning for the agent using SAC with the states, actions, and rewards presented above. Here, we sequentially give a set of (predefined or randomly generated) target positions to intercept, which simulates the sequentially updated target waypoint predicted by the *Target's Trajectory Prediction Module* at each time step.

Online Phase After offline training, the optimized policy network produces optimal actions to intercept the given target positions, as shown in the right part of 2. The optimal target position to intercept is calculated by the *Optimal Interception Waypoint Selector*. Given the predictive waypoints $\{\mathbf{p}_{T+t}\}_{t=1}^{M}$ by the *Trajectory Prediction Module*, we select the earliest waypoint $\mathbf{p}_{T+\hat{t}}$ that is possible for the agent to intercept

$$\hat{t} = \min\{t \mid \|\mathbf{p}_{T+t} - \mathbf{q}_T\| \leq v_{\max}\}_{t=1}^{M}$$

where $\mathbf{q}_T$ is the current position of the agent at time T. There are two reasons to use this strategy: (1) a successful interception is generally the earlier the better; (2) the prediction of earlier waypoints is generally more accurate than that of later ones.

Agent's Dynamics Control Module

The actions output from *Task's Interception Path Planning Module* are physical dynamics independent. This module aims to convert high-level actions to low-level control signals for the UAV's dynamics system.

Dynamics Systems Different UAV types often have different dynamics systems. We take a part of dynamics system equations used in our experiments as an example to illustrate the working mechanism of this module.

$$a_x = (\cos\phi\cos\theta\cos\psi + \sin\phi\sin\psi)\frac{f_z}{m} \tag{14}$$

$$a_y = (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\frac{f_z}{m} \tag{15}$$

$$a_z = \cos\phi\cos\theta\frac{f_z}{m} - g \tag{16}$$

where (a_x, a_y, a_z) are accelerations along each axis, m and g stands for the mass of UAV and gravitational acceleration, roll ϕ, pitch θ, yaw ψ are Euler angles, and f_z is the thrust.

A2C Adaptor . This component aims to convert the actions to control signals (A2C) by solving dynamics equations. Taking the above dynamics Eq. 14-16 as the example, we need to find the solution of f_z, which serves as one of the control signals to the UAV. In Eq. 14-16, the left-hand values (a_x, a_y, a_z) can be obtained by Eq 2. If we fix $\phi = 0$, it is easy to obtain the analytic solution:

$$f_z = m\sqrt{\frac{a_x^2}{cos^2\psi} + \frac{a_y^2}{\sin^2\psi}} \tag{17}$$

For more complex cases with no analytic solutions, we turn to ODE (ordinary difference equation) solvers (Atkinson, Han, and Stewart 2011) to find numerical solutions, as shown in the rightmost part of Figure 2.

Experiments

In this section, we show the overall efficiency and efficacy of our approach by comparing it with other SOTA methods via extensive simulation experiments. Moreover, ablation studies are conducted to evaluate performance with different configurations.

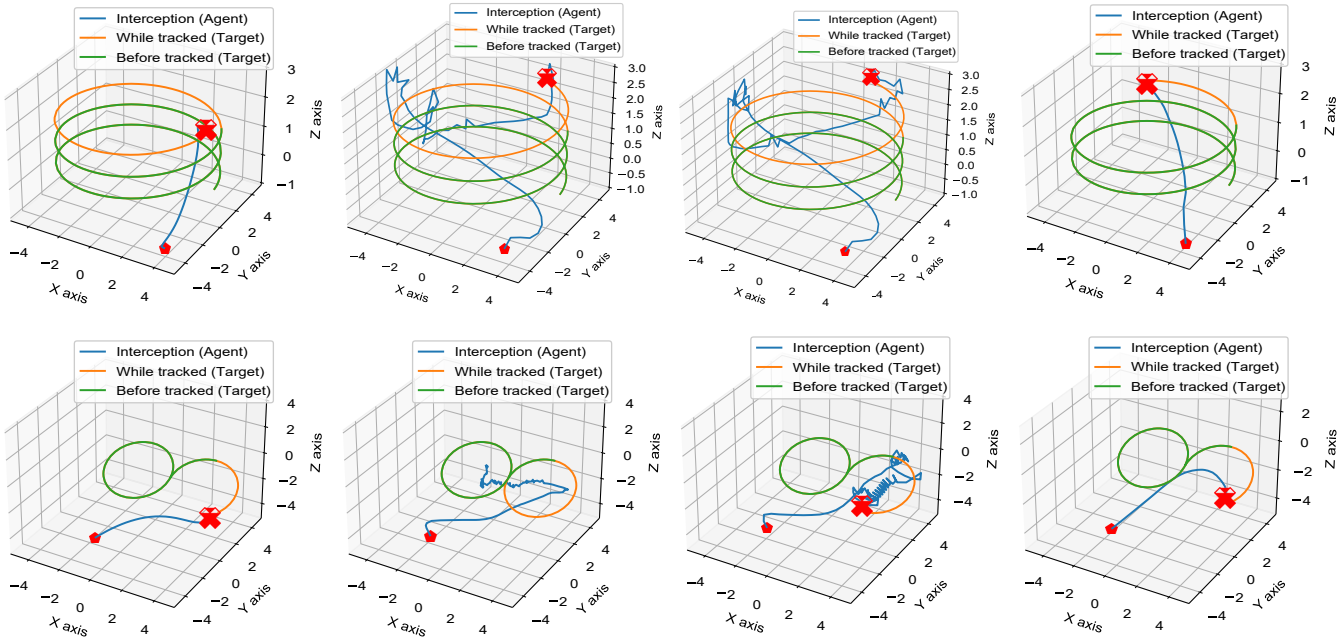
Comparison Methods for DTI

As presented in related work, there is limited work that addresses the DTI task with RL whereas most traditional methods are incompetent to complete relatively dynamic tasks in an open environment. In the experiments, the following SOTA methods are used for comparison.

- *EKF* (Nath, Sudheesh, and Jayakumar 2016) denotes the conventional extended Kalman filtering (EKF) method used for tracking and intercepting the target.
- *ZEM* (Lin 1983) is a proportional guidance method that aims to minimize the miss distance between the interceptor and target, assuming no further maneuvers from the current position.
- *AZEM* (Gaudet, Linares, and Furfaro 2020) uses an augment ZEM policy with the full engagement states, including velocity and target acceleration.
- *EARL* (Gaudet, Furfaro, and Linares 2020) is a meta RL framework that performs DTI tasks by learning a simplified Euler-angle dynamics model.
- *EARL-PC* (Gaudet, Furfaro, and Linares 2020; Gaudet 2020) extends EARL with an action conditional predictive coding model.
- *DTTD* is our DRL framework which decouples the trajectory prediction, path planning, and dynamics control.

Path Planning for Interception Tests

The performance of path planning for interception is directly associated with the success of DTI tasks. In this experiment, we evaluate the capability of interception path planning for each comparison method mentioned above with two nonlinear flying trajectories of the target, the spiral, and the lemniscate (see Figure 3). To evaluate the efficacy of interception path planning, we define the following weighted rank-sum ratio (*WRSR*) (Wang et al. 2015), where $\#^{1st}$, $\#^{2nd}$ and $\#^{other}$ stand for the number of times a method achieves 1st, 2nd and other places (ranked by the number of steps for a successful intercept, the smaller the better), respectively, in a comparison between all methods over N_T test cases. $\#^{fail}$



(a) AZEM (b) EARL (c) EARL-PC (d) DTTD

Figure 3: The demonstration of targets' trajectories and agents' planning paths for DTI maneuvers via different methods. The first row shows a spiral target maneuver, while the second row shows a lemniscate target maneuver.

is the number of fails over all test cases. In this experiment, we set $w_1 = 3, w_2 = 2, w_o = 1, w_f = 2$ to assign the weighted score for *Rank 1*, *Rank 2*, *Other Rank* and *Fail*.

$$WRSR = \frac{w_1 \#^{1st} + w_2 \#^{2nd} + w_o \#^{other} - w_f \#^{fail}}{\max\{w_1, w_2, w_o, w_f\} \times N_T}$$

Table 1 reports the results from interception path planning test over ten different start positions for the agent to intercept, i.e., $N_T = 10$. In all cases, DTTD consistently achieves the best path planning (the fewest interception time steps) without fail. We emphasize consistency since the performance of all the other methods in our comparison was found to be strongly dependent on the start points of their respective trajectories; for non-adaptive methods such as ZEM and augmented ZEM, this is to be expected since they are solutions of dynamics equations and hence largely dependent on start conditions. The EKF was found to perform the best among the conventional baselines; KF-guided interceptions easily suffer from a spiral into failure if there is a small initial noise (i.e. different initial positions in this case) in the learned model, the EKF either succeeds or fails completely due to the rapid error propagation (lack of robustness). As to other RL methods, EARL has no predictive path planning component, so it tends to fail when the target is rapidly moving along nonlinear trajectories (e.g., the case of lemniscate); this is a typical Archilles situation. On the other hand, although the RL for EARL-PC learns to jointly optimize target prediction, path planning, and dynamics control, such training complexity heavily degrades its adaptability on different trajectories since EARL-PC does not possess the modularity of DTTD.

	Sprial Trajectory				Lemniscate Trajectory			
Method	#1st	#2nd	#fail	WRSR	# 1st	#2nd	#fail	WRSR
EKF	0	3	2	0.23	0	2	6	-0.20
ZEM	0	1	7	-0.33	0	1	9	-0.53
AZEM	0	1	**8**	-0.43	0	4	6	-0.23
EARL	0	2	2	0.20	0	0	**10**	-0.67
EARL-PC	0	3	0	0.43	0	2	7	-0.30
DTTD	**10**	0	0	**1.00**	**10**	0	0	**1.00**

Table 1: Comparison of Interception Path Planning

Figure 3 illustrates the interception planning path for each comparison method, where we set the trajectories of the target before being tracked in blue and while being tracked by the agent in orange. We find that DTTD (see Figure 3(d)) achieves the best-planned path for successful interception of the dynamic target using the least time steps (see the blue and the orange trajectories). This can be seen from the continuous nature of the blue trajectories and their relatively short lengths.

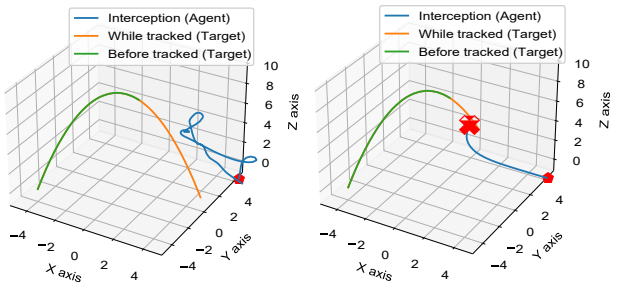
Interception Efficiency Tests

This experiment simulates the case of missile interception before hitting a target on the ground. The results in Table 2 show interception efficiency comparison. In this case, we count the number of successful and failed interceptions for all comparison methods for the parabolic target maneuver, as demonstrated in Figure 4. $\delta = 0.3$ and $\delta = 0.5$ denote the *maximum effective hit distance* to check if the agent hits the target within distance δ. We see that out of a total of

	$\delta = 0.5$			$\delta = 0.3$		
Method	#hit	#miss	accuracy	#hit	#miss	accuracy
EKF	3	3	0.50	1	5	0.17
ZEM	0	**6**	0.00	0	**6**	0.00
AZEM	0	**6**	0.00	0	**6**	0.00
EARL	4	2	0.67	1	5	0.17
EARL-PC	4	2	0.67	2	4	0.33
DTTD	**6**	0	**1.00**	**6**	0	**1.00**

Table 2: Comparison of Interception Rates

six tests from different start positions; no other baseline was able to achieve comparable accuracies with DTTD. Conventional methods (ZEM, AZEM) were completely unable to perform successful interceptions in all tests, due to their heavy dependence on start positions. The EKF was also unable to achieve robustness in interception due to its dependence on the learned model, i.e., resulting in the propensity of a small error in its state estimation to propagate rapidly along with the learning process. EARL and EARL-PC achieved performances of around 33% ($\delta = 0.3$), which is consistent with the numbers obtained for the WRSR in the previous section due to the inefficacy of physical dynamics coupled learning.



(a) EARL-PC (b) DTTD

Figure 4: Visualizations of Interception Paths

Figure 4 shows the efficacy of interception of a target executing a parabolic maneuver w.r.t. EARL-PC and DTTD. For the same target maneuver and start positions, EARL-PC fails to intercepting the target while DTTD executes a smooth and optimized intercepting trajectory in a short time.

Ablation Studies

We ablate for two functional aspects of our model: in the first section, we discuss the effects of different trajectory prediction models; and in the second section, we look at how different reward components (Eq. 11-13) affects the performance. For both studies, we collect the time steps required for a successful interception from different start positions (the coordinates are labeled in Table 3 and 4).

Effect of Different Trajectory Prediction Models We consider different prediction models that are presented in

	$\delta = 0.5$			$\delta = 0.3$			
Model	(4,5,0)	(4,3,0)	(6.3,0)	(6.5,0)	(5,4,0)	(5,4,4)	#1st
LSTM	78	78	58	86	88	Fail	0
Seq2seq	**47**	**48**	**47**	**49**	**52**	**41**	**6**
CNN-Seq2seq	60	55	48	62	61	46	0

Table 3: #ITS of Different Trajectory Prediction Models

	$\delta = 0.5$			$\delta = 0.3$			
Reward	(4,5,0)	(4,3,0)	(6.3,0)	(6.5,0)	(5,4,0)	(5,4,4)	#1st
r_c	60	55	49	61	60	45	0
$r_c + r_m$	56	52	51	63	59	53	0
$r_c + r_m + r_s$	**47**	**48**	**47**	**49**	**52**	**41**	**6**

Table 4: #ITS of Different Reward Functions

Trajectory Prediction Module. For each model, we evaluate the interception time steps (ITS) for a parabolic target maneuver while fixing the full reward function (see Eq. 10). Table 3 gives the statistics of time steps for different start positions and different δ values. We find that in all cases, the Seq2seq module consistently achieves the best prediction score. We attribute this to the efficiency of the encoder-decoder architecture to learn and predict the target's multiple future positions at once, compared to the LSTM, which only learns one-step forward positions when training. CNN-Seq2seq does not achieve the best performance, which may be attributed to the loss of position transition information due to the overly complex convolution operations.

Effect of Superposition of Reward Components The details of three reward components r_c, r_m, r_s (Eq. 11-13) have been introduced in the reward function section, where each component is responsible for a feature of the path planning and movement to encourage the agent to successfully achieve the DTI task. In this evaluation, we take single r_c as the most basic reward function. Based on r_c, we test the performance two superposition reward functions, $r_c + r_m$ and $r_c + r_m + r_s$ (the full version given in Eq. 10). We compute the ITS, again for two values of the δ distance and different start positions. The results are shown in Table 4. We find conclusive evidence that the superposition version of our reward function, $r_c + r_m + r_s$, is the most efficient for DTI tasks, although the basic reward r_c can also complete the tasks, albeit in a less efficient manner.

Conclusion

In this paper, we propose DTTD, a decoupling task learning and dynamics control architecture for DTI based on reinforcement learning. DTTD is highly modular and is adaptive enough to detach hard-to-train physical dynamics models from simpler to train but essential high-level task-oriented strategy frameworks. We show, via extensive experiments and comparisons with SOTA methods, that DTTD significantly improves the efficiency and efficacy of DTI tasks. We also point out that DTTD can be seen as a general learning paradigm for decoupled RL, which can be easily applied to other tasks apart from DTI.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (Granted No. 62276190, 62076184, 61976158, 61976160, 62076182) and in part by the Shanghai Innovation Action Project of Science and Technology (20511100700) and Shanghai Natural Science Foundation (22ZR1466700). This work was initiated by a joint research project with the DeepBlue Academy of Sciences.

References

Atkinson, K.; Han, W.; and Stewart, D. E. 2011. *Numerical solution of ordinary differential equations*, volume 108. John Wiley & Sons.

Azar, A. T.; Koubaa, A.; Ali Mohamed, N.; Ibrahim, H. A.; Ibrahim, Z. F.; Kazim, M.; Ammar, A.; Benjdira, B.; Khamis, A. M.; Hameed, I. A.; et al. 2021. Drone Deep Reinforcement Learning: A Review. *Electronics*, 10(9): 999.

Beard, R. W.; McLain, T. W.; Goodrich, M. A.; and Anderson, E. P. 2002. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE transactions on robotics and automation*, 18(6): 911–922.

Becker-Ehmck, P.; Karl, M.; Peters, J.; and van der Smagt, P. 2020a. Learning to Fly via Deep Model-Based Reinforcement Learning. arXiv:2003.08876.

Becker-Ehmck, P.; Karl, M.; Peters, J.; and van der Smagt, P. 2020b. Learning to Fly via Deep Model-Based Reinforcement Learning. arXiv:2003.08876.

Belkhale, S.; Li, R.; Kahn, G.; McAllister, R.; Calandra, R.; and Levine, S. 2021. Model-Based Meta-Reinforcement Learning for Flight With Suspended Payloads. *IEEE Robotics and Automation Letters*, 6(2): 1471–1478.

Chaudhuri, S. S.; and Konar, A. 2007. Vision Based Target-Tracking Realized with Mobile Robots using Extended Kalman Filter. *Eng. Lett.*, 14(1): 176–184.

Danancier, K.; Ruvio, D.; Sung, I.; and Nielsen, P. 2019. Comparison of Path Planning Algorithms for an Unmanned Aerial Vehicle Deployment Under Threats. *IFAC-PapersOnLine*, 52(13): 1978–1983. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.

Donahue, J.; Hendricks, L. A.; Rohrbach, M.; Venugopalan, S.; Guadarrama, S.; Saenko, K.; and Darrell, T. 2016. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. arXiv:1411.4389.

Gasparetto, A.; Boscariol, P.; Lanzutti, A.; and Vidoni, R. 2015. *Path Planning and Trajectory Planning Algorithms: A General Overview*, 3–27. Cham: Springer International Publishing. ISBN 978-3-319-14705-5.

Gaudet, B. 2020. Adaptive Scale Factor Compensation for Missiles with Strapdown Seekers via Predictive Coding. arXiv:2009.00975.

Gaudet, B.; Furfaro, R.; and Linares, R. 2020. Reinforcement learning for angle-only intercept guidance of maneuvering targets. *Aerospace Science and Technology*, 99: 105746.

Gaudet, B.; Furfaro, R.; Linares, R.; and Scorsoglio, A. 2021. Reinforcement Metalearning for Interception of Maneuvering Exoatmospheric Targets with Parasitic Attitude Loop. *Journal of Spacecraft and Rockets*, 58(2): 386–399.

Gaudet, B.; Linares, R.; and Furfaro, R. 2020. Adaptive guidance and integrated navigation with reinforcement meta-learning. *Acta Astronautica*, 169: 180–190.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; and Levine, S. 2019. Soft Actor-Critic Algorithms and Applications. arXiv:1812.05905.

Himawan Triharminto, H.; Prabuwono, A. S.; Adji, T. B.; Setiawan, N. A.; and Chong, N. Y. 2013. UAV Dynamic Path Planning for Intercepting of a Moving Target: A Review. In Omar, K.; Nordin, M. J.; Vadakkepat, P.; Prabuwono, A. S.; Abdullah, S. N. H. S.; Baltes, J.; Amin, S. M.; Hassan, W. Z. W.; and Nasrudin, M. F., eds., *Intelligent Robotics Systems: Inspiring the NEXT*, 206–219. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-40409-2.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780.

Hwangbo, J.; Sa, I.; Siegwart, R.; and Hutter, M. 2017. Control of a Quadrotor With Reinforcement Learning. *IEEE Robotics and Automation Letters*, 2(4): 2096–2103.

LeCun, Y.; Chopra, S.; Hadsell, R.; Ranzato, M.; and Huang, F. 2006. A tutorial on energy-based learning. *Predicting structured data*, 1(0).

Lin, C.-F. 1983. New Missile Guidance and Control Technology [panel disc. Introduction]. In *1983 American Control Conference*, 1090–1090.

Montazeri, A.; Can, A.; and Imran, I. H. 2021. Chapter 3 - Unmanned aerial systems: autonomy, cognition, and control. In Koubaa, A.; and Azar, A. T., eds., *Unmanned Aerial Systems*, Advances in Nonlinear Dynamics and Chaos (ANDC), 47–80. Academic Press. ISBN 978-0-12-820276-0.

Munoz, G.; Barrado, C.; Cetin, E.; and Salami, E. 2019. Deep Reinforcement Learning for Drone Delivery. *Drones*, 3(3).

Nalepka, J.; and Hinchman, J. 2005. Automated aerial refueling: extending the effectiveness of UAVs. In *AIAA modeling and simulation technologies conference and exhibit*, 6005.

Nath, T. S. G.; Sudheesh, P.; and Jayakumar, M. 2016. Tracking Inbound Enemy Missile for Interception from Target Aircraft Using Extended Kalman Filter. In Müller, P.; Thampi, S. M.; Bhuiyan, M. Z. A.; Ko, R. K. L.; Doss, R.; and Calero, J. M. A., eds., *Security in Computing and Communications - 4th International Symposium, SSCC 2016, Jaipur, India, September 21-24, 2016, Proceedings*, volume 625 of *Communications in Computer and Information Science*, 269–279. Springer.

Pan, S.; Su, H.; Chu, J.; and Wang, H. 2010. Applying a novel extended Kalman filter to missile–target interception with APN guidance law: A benchmark case study. *Control Engineering Practice*, 18(2): 159–167. Special Issue of the 3rd International Symposium on Advanced Control of Industrial Processes.

Radmanesh, M.; Kumar, M.; Guentert, P. H.; and Sarim, M. 2018. Overview of Path-Planning and Obstacle Avoidance Algorithms for UAVs: A Comparative Study. *Unmanned Systems*, 06(02): 95–118.

Rysdyk, R. 2003. UAV path following for constant line-of-sight. In *2nd AIAA" Unmanned Unlimited" Conf. and Workshop & Exhibit*, 6626.

Song, Y.; Steinweg, M.; Kaufmann, E.; and Scaramuzza, D. 2021. Autonomous Drone Racing with Deep Reinforcement Learning. arXiv:2103.08624.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, 3104–3112. Cambridge, MA, USA: MIT Press.

Triharminto, H. H.; Adji, T. B.; and Setiawan, N. A. 2011. Dynamic uav path planning for moving target intercept in 3D. In *2011 2nd International Conference on Instrumentation Control and Automation*, 157–161.

Wang, Z.; Dang, S.; Xing, Y.; Li, Q.; and Yan, H. 2015. Applying rank sum ratio (RSR) to the evaluation of feeding practices behaviors, and its associations with infant health risk in Rural Lhasa, Tibet. *International Journal of Environmental Research and Public Health*, 12(12): 15173–15181.

Xiong, J.-J.; and Zheng, E.-H. 2015. Optimal Kalman Filter for state estimation of a quadrotor UAV. *Optik*, 126(21): 2862–2868.