Aerospace Science and Technology, Volume 119, December 2021, Article number 107187
DOI: 10.1016/j.ast.2021.107187

Learning Prediction-Correction Guidance for Impact Time Control

Zichao Liu[a], Jiang Wang[a], Shaoming He[a,*], Hyo-Sang Shin[b], Antonios Tsourdos[b]

[a]*School of Aerospace Engineering, Beijing Institute of Technology, Beijing 100081, China*
[b]*School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield MK430AL, UK*

Abstract

This paper investigates the problem of impact-time-control and proposes a learning-based computational guidance algorithm to solve this problem. The proposed guidance algorithm is developed based on a general prediction-correction concept: the exact time-to-go under proportional navigation guidance with realistic aerodynamic characteristics is estimated by a deep neural network and a biased command to nullify the impact time error is developed by utilizing the emerging reinforcement learning techniques. To deal with the problem of insufficient training data, a transfer-ensemble learning approach is proposed to train the deep neural network. The deep neural network is augmented into the reinforcement learning block to resolve the issue of sparse reward that has been observed in typical reinforcement learning formulation. Extensive numerical simulations are conducted to support the proposed algorithm.

Keywords: Missile guidance, impact-time-control guidance, prediction-correction, transfer learning, reinforcement learning

1. Introduction

The primary objective of missile guidance law is to guide the vehicle to intercept the target with zero miss distance [1, 2, 3, 4, 5]. The most widely-used proportional navigation guidance (PNG) law enjoys the merit of easy implementation and provides the possibility of energy minimization under certain circumstances [6, 7]. The PNG has also been proved to maximize the terminal velocity in a recent work [8]. However, conventional PNG cannot handle additional constraints, e.g., impact angle, impact time, and needs further adjustments. Among these constraints, the impact-time-control guidance attracts extensive interests in recent years since this strategy helps to improve the probability of successful penetration against close-in weapon systems equipped on land or sea platforms. Generally, impact time control can be achieved by coordinating the predicted time-to-go through communication among the missile network or control the predicted time-to-go for each missile individually by a proper guidance law.

*Corresponding author
Email address: shaoming.he@bit.edu.cn (Shaoming He)

The impact-time-control guidance (ITCG) law was first introduced in [9] by standard optimal control theory. Based on the concept of biased PNG, the authors in [10] developed a polynomial biased term to cater for the impact time constraint. A generalized method that can be leveraged in extending existing guidance laws to impact time control was proposed in [11] by utilizing a specific error dynamics. This concept was further extended to a three-dimensional engagement scenario in [12]. The work in [13] derived the analytic time-to-go estimation without any small angle assumption and hence can improve the performance of biased PNG for impact time control with large heading error scenarios. Except for PNG and its variants, geometric rules [14, 15, 16] and nonlinear control theories [17, 18, 19**?**], are also utilized in impact time control or coordination in recent works. The main idea behind these guidance algorithms is to adjust the remaining flight time, i.e., length of the trajectory, to control the intercept time. However, exact time-to-go estimation is intractable for real implementations and hence these algorithms leveraged approximate estimations by using small angle assumptions. Although guidance laws without explicit time-to-go estimation were also proposed in the literature [20, 21, 22, 23, 24], most of them require constant-speed assumption in guidance command derivation. This means that the performance in impact time control will degrade in practical scenarios. Therefore, classical closed-form guidance laws that rely on approximated models with small angle or other idealistic assumptions, are no longer appealing to solve future real-world guidance problems.

Thanks to the rapid development on embedded computational capability, there has been an increasing attention on the development of computational guidance algorithms in recent years [25, 26, 27]. Unlike classical optimal guidance laws, computational guidance algorithms generate the guidance command relies extensively on onboard computation and therefore dose not require analytic solution of specific guidance laws. Generally, computational guidance can be classified into two main categories: (1) model-based ; and (2) data-based. One of the most widely-used model-based computational missile guidance algorithms, termed as model predictive static programming (MPSP) [28, 29, 30, 31], converts a dynamic programming problem into a static programming problem, thereby providing appealing characteristics in terms of computational efficiency [32]. However, the major limitation of MPSP-based computational guidance algorithms is that they require a good initial solution guess to guarantee the convergence [33, 34].

Notice that impact time control requires accurate estimation of the remaining flight time. This requires finding the relationship between the predicted impact time and the nonlinear dynamic model. Hence, model-based impact-time-control guidance algorithms inevitably require computationally-expensive numerical integration in implementation. Another bottleneck of developing model-based impact-time-control algorithms is that the analytic dynamics model of the time-to-go (i.e., in terms of the lateral acceleration) is unknown. For this reason, the data-based model-free concept is more suitable for impact time control. Motivated by this observation, this paper aims to propose a computational impact-time-control guidance algorithm by leveraging the emerging deep learning techniques. The proposed guidance algorithm is developed based on a general prediction-correction concept: the exact time-to-go under PNG considering aerodynamic forces

is estimated by a deep neural network (DNN) based supervised learning functional block; and a biased command for impact time control is developed by utilizing the state-of-the-art reinforcement learning (RL) approaches. Extensive numerical simulations with comparisons are also carried out to verify the effectiveness of the proposed approach.

The main contributions of this paper are threefolds. First, we propose a general prediction-correction-based framework for computational guidance design. This concept can be easily extended to other guidance application scenarios, e.g., impact angle control, terminal velocity control. Up to the best of our knowledge, no similar results have been published in the existing literature. Second, unlike most data-based guidance algorithms that rely on the assumption that enough training samples are available [35, 36, 37, 38, 39, 40, 41], we propose a transfer-ensemble learning to improve the generalization performance of the supervised learning functional block. Third, the supervised learning technique is augmented into the RL block to resolve the issue of sparse reward in RL training. This concept is demonstrated to significantly improve the learning efficiency.

The remainder of this paper is organized as follows. The backgrounds and preliminaries of this paper are stated in Sec. 2. Section 3 presents the concept of the proposed computational guidance algorithm. Sec. 4 provides prediction of time-to-go, followed by the correction of impact time error in Sec. 5. Finally, some simulation results and conclusions are offered. The source code of this paper is available at `https://github.com/LutterWa/TEDNN-ITCG`.

2. Backgrounds and Preliminaries

In this section, we first present the dynamics and kinematics models of the interceptor. Then, the problem formulation of this paper is stated. Before presenting the mathematical models, we make the following general assumptions that have been widely-accepted in impact-time-control guidance law design.

Assumption 1. *Since the control loop is generally much faster than that of the guidance loop, we assume that the missile's autopilot is ideal, i.e., there is no control delay.*

Assumption 2. *The target is assumed to be stationary due to the fact the concept of simultaneous attack is generally leveraged to intercept high-value ships or ground-based targets.*

Assumption 3. *The missile provides roll stabilization and therefore the three-dimensional guidance problem can be decoupled into horizontal and vertical channels using the well-known separation concept. Hence, a three-dimensional problem can be decoupled into two two-dimensional problems and the training results and conclusions of this paper can be utilized.*

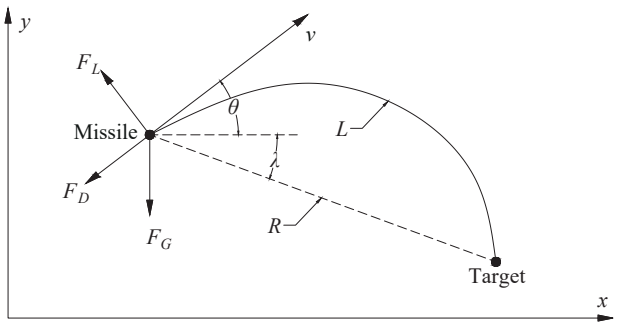

Figure 1: Definition of notations and symbols.

2.1. Nonlinear Mathematical Models

To simplify the problem, this paper only considers the vertical engagement geometry, as shown in Fig. 1. The symbol v stands for the missile's velocity and θ represents the flight path angle. The lift, drag, and gravity forces are denoted by F_L, F_D, and F_G. The notation λ is the line-of-sight (LOS) angle, and L stands for the length of the flight trajectory. The relative distance is denoted by R. The aerodynamic forces can be expressed as follows

$$F_L = C_L Q S \tag{1}$$

$$F_D = C_D Q S \tag{2}$$

$$F_G = mg \tag{3}$$

where C_L denotes the lift coefficient and C_D stands for the drag coefficient. The symbol S is the reference area of the missile and m represents the mass of the missile. The notations g stands for the gravitational acceleration and the variable Q denotes the dynamic pressure, which can be determined by

$$Q = \frac{1}{2}\rho v^2 \tag{4}$$

where ρ stands for the air density.

The differential equations of the dynamics and kinematics models are given by

$$\dot{v} = \frac{F_D - F_G \sin\theta}{m} \tag{5}$$

$$\dot{\theta} = \frac{F_L - F_G \cos\theta}{mv} \tag{6}$$

$$\dot{x} = v\cos\theta \tag{7}$$

$$\dot{y} = v\sin\theta \tag{8}$$

where (x, y) denotes the inertial position of the missile.

Since the angle-of-attack (AoA), denoted by α, is a small variable, the aerodynamic coefficients can be approximated as

$$C_L = C_L^{\alpha}\alpha \tag{9}$$

$$C_D = C_{D0} + C_{D0}^{\alpha^2}\alpha^2 \tag{10}$$

where C_L^{α} denotes the derivative of lift coefficient with respect to AoA and C_{D0} represents the parasite drag coefficient. The notation $C_{D0}^{\alpha^2}$ stands for the induced drag coefficient. These aerodynamic coefficients can be obtained through ground wind tunnel experiment. In evaluating the lift and drag forces, we leverage the standard atmosphere model to calculate the air density.

Since tactical missiles are usually controlled by lateral acceleration command a_M, we require the auxiliary relationship between AoA and lateral acceleration, i.e.,

$$\alpha = \frac{ma_M}{C_L^{\alpha}QS} \tag{11}$$

2.2. Impact Time Control Problem

The objective of impact time control is to guide the missile to intercept the target at a prescribed time. Define t_d as the desired interception time, the impact time constraint can then be formulated as

$$t_f = t_d \tag{12}$$

where t_f denotes the terminal time.

Since

$$t_f = t_{go} + t \tag{13}$$

the problem of impact time control can be converted into an equivalent problem of adjusting the remaining flight time. Mathematically, the time-to-go can be readily formulated as

$$t_{go} = \int_t^{t_f} \frac{L}{v(\tau)} d\tau \tag{14}$$

where the length of the trajectory is determined as

$$L = \int_x^{x_f} \sqrt{1+\theta^2(x)}\mathrm{d}x \tag{15}$$

A perfect interception requires

$$x\left(t_f\right) = x_T, \quad y\left(t_f\right) = y_T \tag{16}$$

where (x_T, y_T) denotes the inertial position of the stationary target.

In summary, the main objective if this paper is to propose a computational guidance algorithm to satisfy constraints (12) and (16).

Remark 1. *Notice that the aerodynamic forces are time-varying and hence finding analytic solutions for time-to-go is generally intractable except for rare cases. This means that solving the impact-time-control problem requires computationally-expensive numerical integration to find t_{go}. To this end, most existing guidance laws utilized some assumptions to approximate the time-to-go and hence the performance will degrade in realistic scenarios. From extensive numerical simulations, we also demonstrate that conventional ITCG algorithms with approximate analytic time-to-go cannot guide the missile to intercept the target in some scenarios.*

3. Computational Impact-Time-Control Guidance Algorithm

To solve the impact time control problem, we propose a learning-based prediction-correction framework to design a computational guidance algorithm, which is formulated as a composite command, i.e.,

$$a_c = a_0 + a_b \tag{17}$$

where the baseline command a_0 is utilized to provide zero miss distance for target interception and the biased command a_b is developed to nullify the impact time error.

Without loss of generality, we choose the gravity-compensated energy-optimal PNG as the baseline command, i.e.,

$$a_0 = 3v\dot{\lambda} + g\cos\theta \tag{18}$$

The proposed algorithm to design the biased command a_b is composed of two functional blocks: transfer-ensemble DNN-based real-time predictor and RL-based online corrector. The relationship between these two functional blocks is shown in Fig. 2.

The predictor leverages DNN to learn the unknown nonlinear mapping from current states to time-to-go under the baseline PNG a_0 and can be trained offline. A transfer-ensemble DNN (TEDNN) is developed to accommodate the issue of sample insufficiency, i.e., improving the performance of generalization to support practical applications. Once the TEDNN is trained properly, we can then utilize the trained TEDNN to obtain accurate estimation of time-to-go in real-time, thereby avoiding computationally-expensive numerical integration.

The corrector leverages the state-of-the-art RL algorithm, i.e., Proximal Policy Optimization (PPO), to train a guidance agent that directly outputs the biased command a_b to regulate the impact time error. The rationale behind using RL in the correction step is that the explicit dynamics model of time-to-go is unknown. In RL training, we leverage the trained DNN to predict the time-to-go and use this information as one input to the RL agent. This augmentation transforms the terminal constraint into a state regulation problem and hence resolves the issue of the sparse reward that has been observed in typical RL formulation.

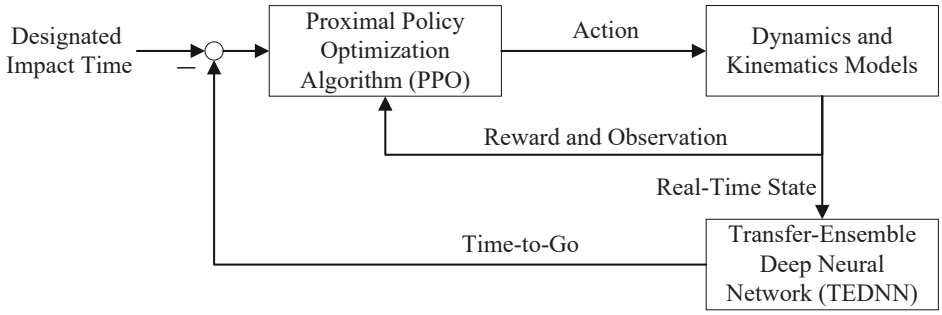

Figure 2: Computational Impact-Time-Control Guidance

Remark 2. *Notice that the proposed approach is a general prediction-correction framework for computational guidance law design that can be easily extended to other guidance applications, e.g., impact angle control, terminal velocity control.*

Remark 3. *Instead of learning from scratch, we utilize a domain-knowledge-aided approach [42] in RL training, i.e., we formulate the guidance command as a biased PNG. With this formulation, we only need to train the biased term for impact time error regulation and hence greatly improves the learning effectiveness during the training process.*

4. The Prediction of Time-to-Go

DNN leverages multiple hidden layers to process features that have been extracted from the input layer and hence provides the possibility of learning complicated nonlinear mappings. For this reason, the supervised DNN learning provides us a promising way to learn the unknown relationship between flight states and time-to-go under PNG. However, it is well understood that conventional supervised learning requires large amounts of labeled data during the training process. However, it is intractable to collect enough labeled data for aerospace applications using real flight experiments. Hence, the training samples are usually generated from the simulated environment, which inevitably cannot cover different aerodynamic models that would appear in practice. Therefore, it is necessary to reuse and transfer the knowledge learned from the source tasks, i.e., the simulated environment, to the target task with small amounts of data that can be collected from experiment. For this purpose, we propose a new TEDNN that enables knowledge generalization to accurately predict the time-to-go in real time in this section.

4.1. Architecture of TEDNN

The proposed TEDNN is composed of two different parts: N specific DNNs and one transfer-ensemble process. Each specific DNN is trained for one fixed source task, e.g., with one aerodynamic model from the

simulated environment. Once all DNN are trained properly, we froze the hidden layers of all DNNs and leverage one ensemble neuron to integrate the information from all DNNs for generalization. During the transfer learning process, we only utilize small amounts of data that can be acquired from real experimental tests. The rational behind this formulation is that the prediction error of one single DNN will likely be compensated by using the information from other DNNs. Hence, the overall prediction and generalization performance can be improved with the help of an ensemble neuron. Fig. 3 presents the architecture of the proposed TEDNN.

Remark 4. *Since the environmental disturbances can be introduced in simulation models, the predictor could adapt the parameter changes problem by re-training every DNNs in the TEDNN model with simulation models closer to the actual environment.*

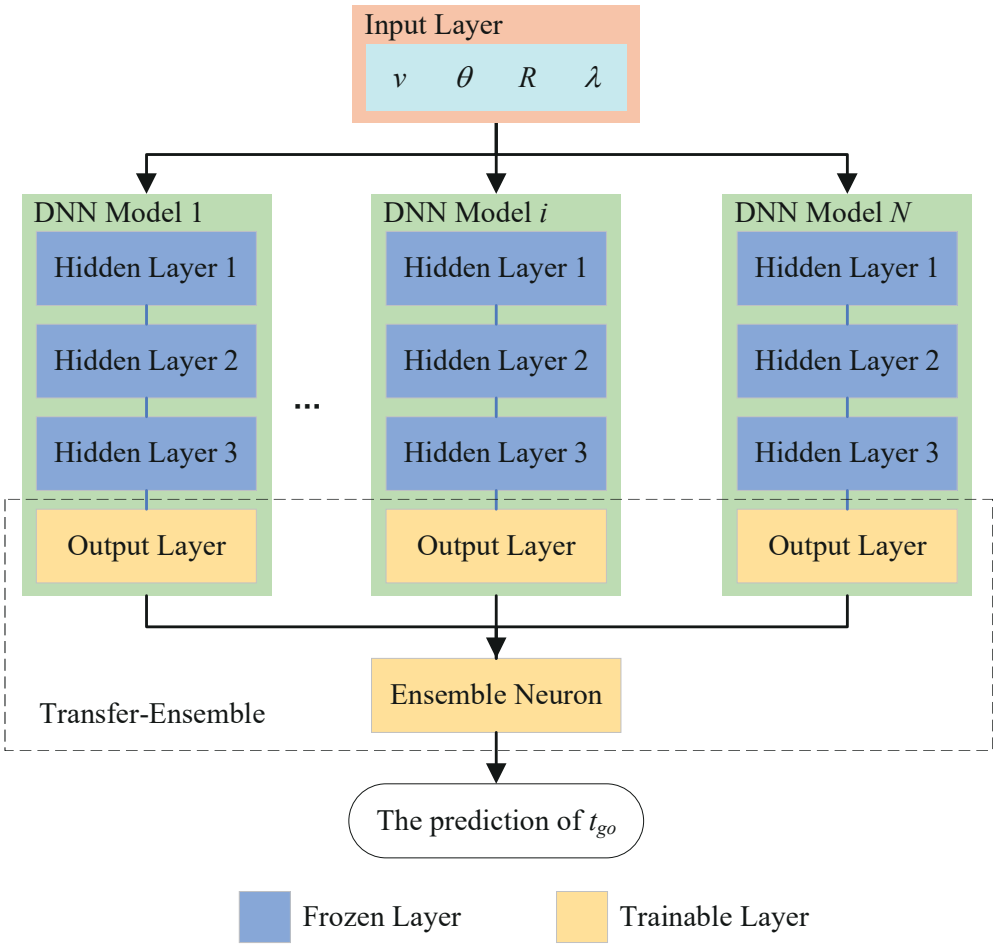

Figure 3: The architecture of TEDNN.

4.2. Training Specific DNNs

To predict the time-to-go using DNN, it is natural to choose the output of the DNN as t_{go}. According to Eq. (14), t_{go} is a function of moving speed v and trajectory length L. It is clear that the shape of the interception trajectory is influenced by the heading direction θ and therefore this information should also be considered in the time-to-go estimation. Notice that the future velocity depends on the aerodynamic characteristics of the airframe. Since the air density changes with the variation of height, the remaining flight time under PNG is also indirectly affected by the interceptor's inertial position or the relative geometry (R, λ). In conclusion, the time-to-go t_{go} can be formulated as a function of moving speed v, flight path angle θ, relative distance R and LOS angle λ as

$$t_{go} = f_t\left(v, \theta, R, \lambda\right) \tag{19}$$

Due to the time-varying and nonlinear properties of the aerodynamic model, finding analytic expression of f_t is intractable. For this reason, we utilize a DNN to learn the unknown mapping from (v, θ, R, λ) to t_{go}. The DNN that has been used in this paper leverages 3 fully-connected hidden layers and each hidden layer is composed of 100 neurons. We leverage the well-known rectified linear units (ReLU) function as the activation function for every neuron due to its fast convergence during the training process [43]. The ReLU function is defined as

$$f_a(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{20}$$

The training data of one DNN is collected by the simulated flight experiment with one realistic aerodynamic model. At every simulation run, we randomly initialize the initial conditions to cover the entire application scenarios. To predict the time-to-go under PNG, we assume that the interceptor is guided by the energy-optimal PNG in each simulation run. Due to the randomness of the initialization, the missile might not be able to intercept the target if the sampled scenario is beyond the physical limits of the interceptor. For this reason, we terminate each simulation run when the relative distance between the missile and the target is less than a threshold or the vertical position of the missile is no longer bigger than zero, i.e., $y \leq 0$. Once the simulation is terminated, the missile's states and its flight paths with time information are collected. Then, the time to reach the target from any position in the flight path can be readily obtained by

$$t_{go} = t_f - t \tag{21}$$

Define the DNN is parameterized by β and the network parameters are optimized by leveraging the ADAM optimizer [44] with the following loss function

$$J\left(\beta\right) = \frac{1}{N_D} \sum_{i=1}^{N_D} \left(\hat{t}_{go,i} - t_{go,i}\right)^2 \tag{22}$$

where $\hat{t}_{go,i}$ is the predicted time-to-go by the DNN from the ith sample, and $t_{go,i}$ is the true time-to-go of the ith sample. The symbol N_D denotes the number of samples that have been randomly drawn from the training set to train the network.

With objective function (25), the network parameter β is then updated by the gradient method in a recursive way as

$$\beta_{new} = \beta_{old} + \alpha_\beta \nabla_\beta J(\beta) \tag{23}$$

where α_β is the learning rate.

4.3. Transfer-Ensemble Process

The main purpose of the transfer-ensemble process is to transfer the learned knowledge from the source task, i.e., several aerodynamic models, to the target task. Once all DNNs are trained dedicated to different aerodynamic models, we leverage the fine-tune concept [45, 46] to froze the input and hidden layers of all DNNs and utilize one single ensemble neuron to fuse the information from all DNNs. This simple ensemble neuron is constructed without any bias and activation function and hence can be considered as a weighted sum of all DNNs' outputs. The initial weights of the single ensemble neuron are set equally for all DNNs as $1/N$.

Consider the output layers of all DNNs and the ensemble neuron as a new transfer-ensemble network, which is parameterized by ξ. Assume that we have N_E ($N_E \ll N_D$) samples that can be collected from real experiment flight tests. We simultaneously adjust the ensemble neuron and DNNs' output layers by a gradient method as

$$\xi_{new} = \xi_{old} + \alpha_\xi \nabla_\xi J(\xi) \tag{24}$$

where α_ξ is the learning rate and $J(\xi)$ denotes the loss function, which is defined as

$$J(\xi) = \frac{1}{N_E} \sum_{i=1}^{N_E} \left(\bar{t}_{go,i} - t_{go,i}\right)^2 \tag{25}$$

where $\bar{t}_{go,i}$ is the predicted time-to-go by the TEDNN from the ith sample

5. The Correction of Impact Time Error

Since the dynamics model of the predicted impact time by DNN is unknown, this section introduces the utilization of the state-of-the-art RL algorithm, i.e., PPO, to develop a computational biased command a_b to satisfy the impact time constraint. We first briefly review the PPO algorithm for the completeness of this paper and then presents the RL formulation of the computational guidance agent.

5.1. Proximal Policy Optimization Algorithm

The RL problem is often formalized as a Markov Decision Process (MDP) or a partially observable MDP (POMDP). A MDP is described by a five-tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ refers to the set of states, $\mathcal{O}$ the set of observations, $\mathcal{A}$ the set of actions, $\mathcal{P}$ the state transition probability and $\mathcal{R}$ the reward function. If the process is fully observable, we have $\mathcal{S} = \mathcal{O}$. Otherwise, $\mathcal{S} \neq \mathcal{O}$. At each time step t, an observation $o_t \in \mathcal{O}$ is generated from the internal state $s_t \in \mathcal{S}$ given to the agent. The agent utilizes this state to generate an action $a_t \in \mathcal{A}$ that is sent to the environment based on a specific action policy $\pi(a_t \mid s_t)$. The action policy is a function that maps the state to a probability distribution over the actions. The environment then leverages the action and the current state to generate the next state s_{t+1} with conditional probability $\mathcal{P}(s_{t+1}|s_t, a_t)$ and a scalar reward signal $r_t \sim \mathcal{R}(s_t, a_t)$.

The goal of RL is to seek a policy for an agent to interact with an unknown environment while maximizing the expected total reward it received over a sequence of time steps. The total reward in RL is defined as the summation of discounted reward to facilitate temporal credit assignment as

$$R_t = \sum_{i=t}^{N} \gamma^{i-t} r_i \tag{26}$$

where $\gamma \in (0, 1]$ denotes the discount factor.

Given current state s_t, the expected total reward is known as the value function

$$V_\pi(s_t) = \mathbb{E}_\pi[R_t|s_t] \tag{27}$$

Many approaches in reinforcement learning also make use of the action-value function

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi[R_t|s_t, a_t] \tag{28}$$

The PPO algorithm, proposed by OpenAI [47, 48], is one of the state-of-the-art policy gradient algorithms that learn a nonlinear function that directly maps the states to the actions, rather than taking the action that globally maximizes the value function. The action function is updated by following the gradient direction of the value function with respect to the action, thus termed as policy gradient. Thanks to this property, the policy gradient algorithms are applicable to guidance problems. PPO utilizes a typical actor-critic structure, as is shown in Fig. 4, where the actor generates the control action based on the current states and the critic approximates the value function to evaluate the performance of the action. Both the actor and the critic leverage the gradient method with a batch of N_s samples to update the network parameter.

(1) Actor update. Define the actor network is parameterized by w. The original PPO, also termed as Trust Region Policy Optimization (TRPO) [47], utilizes the following objective function to optimize the

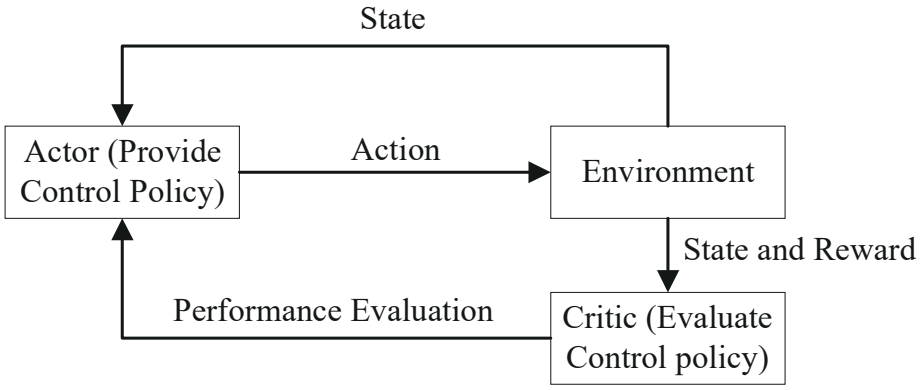

Figure 4: The structure of typical PPO.

actor

$$J(w) = \frac{1}{N_s}\sum_{t=1}^{N_s}\left[\frac{\pi(a_t \mid s_t)}{\pi_{old}(a_t \mid s_t)} A_{\pi_{\text{old}}}(s_t, a_t)\right] \tag{29}$$

$$\text{s.t.} \quad \bar{D}_{KL}(\pi_{\text{old}}, \pi) \leqslant \delta \tag{30}$$

where $\pi_{old}(a_t \mid s_t)$ denotes the policy that has been optimized in the previous time step; $\bar{D}_{KL}(\pi_{\text{old}}, \pi)$ is the average Kullback-Leibler divergence (KLD) between π_{old} and π; δ is a small constant to constrain the average KLD; $A_\pi(s_t, a_t)$ is the advantage function, which is defined as the difference between the action-value function and value function, i.e.,

$$A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t) \tag{31}$$

Notice the average KLD constraint is leveraged to limit the update speed of the policy. This constraint is demonstrated to improve the learning stability during the training process [47]. However, calculating the average KLD is time-consuming and hence the learning speed is constrained by a simple `clip` function in [48] as

$$J(w) = \frac{1}{N_s}\sum_{t=1}^{N_s}\left[\min\left(r_t(w) A_{\pi_{old}}(s_t, a_t), \text{clip}\left(r_t(w), 1-\epsilon, 1+\epsilon\right) A_{\pi_{old}}(s_t, a_t)\right)\right] \tag{32}$$

where

$$r_t(w) = \frac{\pi(a_t \mid s_t)}{\pi_{old}(a_t \mid s_t)} \tag{33}$$

$$\text{clip}\left[r_t(w), 1-\epsilon, 1+\epsilon\right] = \begin{cases} 1-\epsilon, & r_t(w) < 1-\epsilon \\ 1+\epsilon, & r_t(w) > 1+\epsilon \\ r_t(w), & 1-\epsilon < r_t(w) < 1+\epsilon \end{cases} \tag{34}$$

and ϵ is a small constant to constrain the learning speed.

From Eq. (32), it can be readily observed that the *clip* function constrains the policy probability ratio with in the range $(1-\epsilon, 1+\epsilon)$, thereby indirectly limits the update rate of the policy distribution. With objective function (32), the network parameter w is then updated by moving the policy in the direction of the gradient of $J(w)$ in a recursive way as

$$w_{new} = w_{old} + \alpha_w \nabla_w J(w) \tag{35}$$

where α_w is the learning rate.

(2) Critic update. Define the critic network is parameterized by ρ. PPO utilizes the square of the advantage function as objective function in optimizing the critic, i.e.,

$$J(\rho) = \frac{1}{N_s} \sum_{t=1}^{N_s} \|A_\pi(s_t, a_t)\|^2 \tag{36}$$

With objective function (36), the network parameter ρ is then updated by the gradient method in a recursive way as

$$\rho_{new} = \rho_{old} + \alpha_\rho \nabla_\rho J(\rho) \tag{37}$$

where α_ρ is the learning rate.

5.2. RL Formulation of the Impact-Time-Control Guidance Problem

To develop a computational impact-time-control guidance algorithm by using PPO, we formulate the problem into the RL framework in this subsection. This includes action section, state formulation, reward shaping and network architecture design.

5.2.1. Action Selection

As we stated before, the proposed guidance command is based on the general prediction-correction concept. We propose a TEDNN predictor to obtain accurate time-to-go estimation under PNG and leverage the PPO to develop a RL guidance agent that directly generates the biased command a_b to nullify the impact time error. For this reason, the agent action is naturally defined as

$$\text{action} = a_b, \quad |a_b| \leq a_{\max} \tag{38}$$

Notice that the magnitude of the biased command should be constrained, ensuring that the PNG command plays the dominant role to guarantee target capture. Since the impact-time-control guidance algorithm is normally developed for anti-ship or other air/surface-to-surface missiles, the maximum permissible value of the biased term is limited to $3g$, i.e., $a_{\max} = 3g$, to cater for physical constraint.

5.2.2. State Formulation

The environmental states are utilized as the inputs to both the actor network and the critic network of PPO. The first consideration in state formulation for our problem is the impact time error since the main objective of the biased command a_b is to nullify the impact time error to satisfy the time-of-arrival constraint. Let ε_t be the impact time error as

$$\varepsilon_t = t_d - \left(t + \hat{t}_{go}\right) \tag{39}$$

where $\hat{t}_{go}$ denotes that predicted time-to-go by the proposed TEDNN.

As the main purpose of the corrector is to minimize the impact time error, we define the state of the tth training sample as

$$s_t = \frac{\varepsilon_t}{t_{go}} \tag{40}$$

Another advantage of the preceding state definition is that it only focuses on the impact time error, and hence we can safely predict that the trained PPO is insensitive to different scenarios. This will be empirically evaluated in the following section.

Remark 5. *Notice that the input of PPO is only related to the time-to-go. When the aerodynamic model changes, PPO remains good generalization performance, and hence does not require transfer.*

5.2.3. Reward Shaping

The most important and challenging part in the RL formulation is the design of a proper reward function because this function determines the learning efficiency and guarantees the convergence of the training process. To consider N_o different objectives in reward shaping, we can utilize the multi-objective optimization method to formulate the reward function, i.e.,

$$r_t = \sum_{i=1}^{N_o} a_i r_i \tag{41}$$

where r_i denotes the ith reward; a_i is the weight coefficient of the ith reward and satisfies the following condition

$$\sum_{i=1}^{N_o} a_i = 1 \tag{42}$$

In reward shaping, we consider the following two different objectives to cater for impact time constraint and ensure target interception.

(1) Impact time error. The essence of impact-time-control guidance is a finite-time tracking control problem, in which the impact time error is the tracking error. One recent work [49] discussed the optimal

convergence pattern of the tracking error for general guidance law design. The optimal error dynamics is defined as

$$\dot{\varepsilon_t} + \frac{k}{t_{go}}\varepsilon_t = 0, \quad k > 0 \tag{43}$$

which gives the following analytic solution

$$\varepsilon_t = \varepsilon_{t,0}\left(\frac{t_{go}}{t_f}\right)^k \tag{44}$$

where $\varepsilon_{t,0}$ denotes the initial value of the tracking error ε_t.

From Eq. (44), it is clear that the tracking error ε_t converges to zero at the time of impact. According to [49], error dynamics (43) minimizes a meaningful performance index, which helps to tune the guidance gain k. To mimic this optimal error dynamics, we consider the following reward for the impact time error

$$r_1 = e^{-\varepsilon_r^2} \tag{45}$$

where the auxiliary variable ε_r is defined as

$$\varepsilon_r = \frac{\varepsilon_t}{t_{go}} \tag{46}$$

(2) Altitude. Notice that there are different trajectories that can cater for the constraint of target interception with desired impact time. Hence, the missile might approach the ground before intercepting the target, see Fig. 5 for an illustration example, where the dashed line stands for the interception trajectory with PNG and the two solid lines represents the interception trajectories that have the same impact time. From this figure, we can note that if the missile approaches the ground before intercepting the target, the mission will fail and hence we penalize the flight altitude in the third reward as

$$r_2 = e^{\frac{-(y-R)^2}{\sigma^2}} \tag{47}$$

where σ denotes the normalization constant of $(y - R)$.

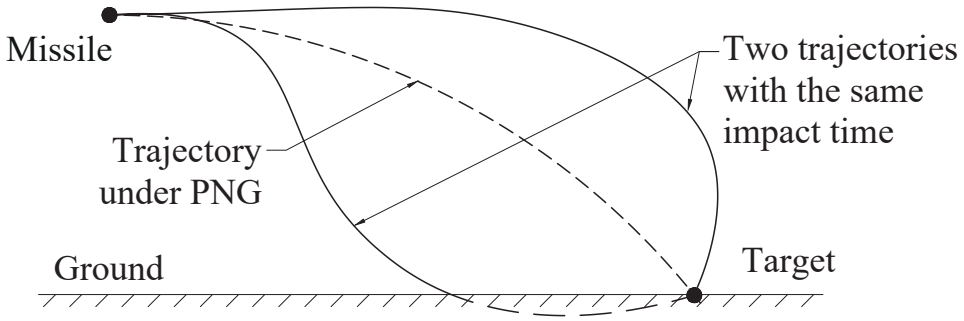

Figure 5: Illustration of different trajectories with the same impact time.

In summary, the reward function is defined by combining Eqs. (45) and (47) as the following weighted sum

$$r_t = a_1 e^{-\varepsilon_r^2} + a_2 e^{-\frac{(y-R)^2}{\sigma^2}} \tag{48}$$

5.2.4. Network Architecture

Inspired by the original PPO algorithm [48], both the actor and the critic are represented by four-layer fully-connected neural networks. Note that this four-layer network architecture is commonly utilized in deep reinforcement learning applications [50]. The layer sizes of these two networks are summarized in Table 1. Except for the actor output layer, each neuron in other layers is activated by a ReLU function, which provides faster processing speed than other nonlinear activation functions due to the linear relationship property. The output layer of the actor network is activated by the tanh function, which is given by

$$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{49}$$

Table 1: Network layer size.

Layer	Actor network	Critic network
Input layer	1 (Size of states)	1 (Size of states)
Hidden layer 1	64	64
Hidden layer 2	64	64
Output layer	1 (Size of action)	1 (Scalar value function)

The benefit of the utilization of tanh activation function in actor network is that it can prevent the control input from saturation as the actor output is constrained by $[-1, 1]$. The output layer of the actor network is scaled by a constant $a_{\max}$ to constrain the biased command within the range $[-a_{\max}, a_{\max}]$.

5.2.5. Network Training

In the training process, we use a buffer to store a batch of N_s transition experience samples. A transition experience e_t is defined as

$$e_t = (s_t, a_t, r_t, s_{t+1}) \tag{50}$$

We assume that the action policy is subject to a Gaussian distribution, i.e.,

$$a_t \sim \mathcal{N}(\mu_a, \sigma_a) \tag{51}$$

where $\mathcal{N}(\mu_a, \sigma_a)$ denotes a Gaussian distribution with its mean and standard deviation as μ_a and σ_a, respectively.

For simplicity, the standard deviation σ_a is assumed as constant and this parameter is trained with the actor parameter w in an integrated manner by the gradient method. Hence, the output of the actor network is the mean of action, i.e., μ_a. Both the critic and the actor networks are optimized by the ADAM algorithm with an episodic manner. At the beginning of each episode, the states of the environment are initialized randomly and the experience buff is initialized as a zero set. Once N_s samples are stored in the

experience buffer, the network parameters are then updated by using these N_s samples and we empty the experience buffer. If the number of time steps reaches the maximum value $T_{\max}$ or the missile approaches the ground/target, the episode is terminated.

6. Simulation Results

In this section, the performance of the proposed computational ITCG algorithm is evaluated by numerical simulations. We first investigate the accuracy of the TEDNN time-to-go estimator and then analyze the performance of the PPO-based impact time error corrector using Monte-Carlo simulations.

6.1. Performance of TEDNN Predictor

6.1.1. Data Collection

The samples from source tasks and target task is collected by using different interception trajectories under energy-optimal PNG. The initial conditions of these simulated scenarios are summarized in Table 2. Notice that the missile's initial position, initial flight path angle and initial moving speed are randomly sampled from a uniform distribution within the ranges that have been specified in Table 2. The reference area of the interceptor is $S = 0.0572556m^2$ and the gravitational acceleration is constant as $g = 9.81m/s^2$. To meet the physical constraints of the missile, the AoA command is limited by a maximum value of 20°. The basic aerodynamic characteristics of the considered airframe with respect to different Mach numbers are presented in Table 3 and the aerodynamic coefficients of other operational points are obtained by interpolation. We consider 5 different source tasks with their corresponding aerodynamic coefficients, respectively, being 0.1, 0.5, 1.0, 2.0 and 5.0 times of the basic one. In the target task, aerodynamic coefficients C_L^{α}, C_{D0} and $C_D^{\alpha^2}$ are, respectively, scaled by 3.0, 1.5 and 1.0 times of the basic one.

Table 2: Initial conditions.

Parameter	Description	Value or Interval	Units
x_0	Initial missile position in the x direction	(-30,-10)	km
y_0	Initial missile position in the y direction	(10,30)	km
x_T	Target position in the x direction	0	km
y_T	Target position in the y direction	0	km
v_0	Initial missile velocity	(200,300)	m/s
θ_0	Initial flight path angle	(0,45)	deg
m	The mass of the missile	200	kg

In generating the training samples for one DNN, we simulate 1000 different interception flight paths and collect 5000000 samples with each sample contains the information of network input (v, θ, R, λ) and

Table 3: Basic aerodynamic coefficients.

Mach Number	C_L^{α}/rad^{-1}	C_{D0}	$C_D^{\alpha^2}/rad^{-2}$
0.4	39.056	0.4604	39.072
0.6	40.801	0.4682	39.735
0.8	41.372	0.4635	39.242
0.9	42.468	0.4776	40.531

output t_{go} pair. For the target task, we only collect one single interception trajectory with 5000 samples. These samples are divided into two categories according to the 80/20 principle: the training set is composed of 80% samples and the other 20% samples belong to the test set. This principle was demonstrated to outperform other empirical principles in [51]. Since the input states have different units and scales, we use their corresponding mean values obtained from all samples to normalize the input states during the training process. The learning rates in training TEDNN are set as $\alpha_\beta = 0.001$ and $\alpha_\xi = 0.0001$.

6.1.2. Performance Evaluation

The performance of the proposed TEDNN is evaluated by three metrics: root mean square error (RMSE), mean absolute error (MAE), and coefficient of determination (CR). These metrics are mathematically defined as

$$\text{RMSE} = \sqrt{\frac{1}{N_t}\sum_{i=1}^{N_t}\left(t_{go,i} - \bar{t}_{go,i}\right)^2} \tag{52}$$

$$\text{MAE} = \frac{1}{N_t}\sum_{i=1}^{N_t}\left|t_{go,i} - \bar{t}_{go,i}\right| \tag{53}$$

$$\text{CR} = \frac{\left(N_t\sum\limits_{i=1}^{N_t}\hat{t}_{go,i}t_{go,i} - \sum\limits_{i=1}^{N_t}\hat{t}_{go,i}\sum\limits_{i=1}^{N_t}t_{go,i}\right)^2}{\left[N_t\sum\limits_{i=1}^{N_t}\hat{t}_{go,i}^2 - \left(\sum\limits_{i=1}^{N_t}\hat{t}_{go,i}\right)^2\right]\left[N_t\sum\limits_{i=1}^{N_t}t_{go,i}^2 - \left(\sum\limits_{i=1}^{N_t}t_{go,i}\right)^2\right]} \tag{54}$$

where N_t denotes the number of test samples.

Table 4 presents the performance evaluation results of different predictors with different test samples. From this table, we can readily observe that the specific DNNs provide accurate impact time prediction for their own source tasks and the maximum prediction error is smaller than $0.5s$. However, their performance degrades drastically when evaluated with the target task. As a comparison, the proposed TEDNN provides significant performance improvement for the target task by introducing an additional transfer-ensemble process. The CR result also indicates the proposed TEDNN is reliable and hence can be utilized in real-time prediction of time-to-go.

Table 4: Performance comparison with different tasks.

	Source tasks			Target tasks		
Model	RMSE	MAE	CR	RMSE	MAE	CR
DNN Model 1	0.2024	0.1687	0.9999	8.4276	6.7864	0.6104
DNN Model 2	0.1355	0.1064	0.9999	7.4257	5.8434	0.6975
DNN Model 3	0.3888	0.3119	0.9998	5.4756	4.2266	0.8355
DNN Model 4	0.2970	0.2373	0.9999	3.4618	2.4942	0.9343
DNN Model 5	0.2793	0.2212	0.9999	2.8206	2.3725	0.9564
TEDNN Model				1.0670	0.7573	0.9938

6.2. The Performance of PPO Corrector

6.2.1. Training the PPO Corrector

The initial conditions of the scenarios that have been utilized in PPO training are the same as described in Table 2. We choose the basic aerodynamic coefficients, as described in Table 3, to train the PPO. To make the impact-time-control problem feasible, the desired impact time is randomly set as 1.1 to 1.2 times of the predicted result for every episode. The hyper parameters that are utilized in PPO training for our problem are summarized in Table 5. Notice that the tuning of hyper parameters imposes great effects on the performance of PPO and this tuning process is not consistent across different ranges of applications, i.e., different works utilized different set of hyper parameters for their own problems. For this reason, we tune these hyper parameters for our guidance problem based on several trial and error tests.

Table 5: Hyper parameter settings in training PPO.

Parameter	Description	Value
ϵ	Ratio clipping	0.2
α_w	Actor learning rate	0.0001
α_ρ	Critic learning rate	0.0002
γ	Discounting factor	0.995
N_s	Size of the experience buffer	256
$T_{\max}$	Maximum permissible time steps of one episode	400
$E_{\max}$	Maximum permissible episodes	500
a_1	The weight of the 1st reward	0.99
a_2	The weight of the 2nd reward	0.01
σ	The normalization parameter for the 2nd reward	1×10^4

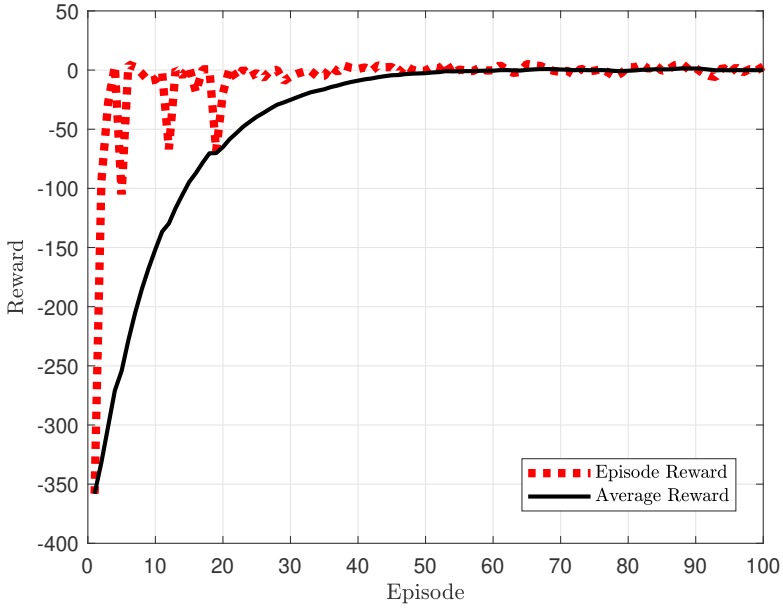

Figure 6: Learning curve of the proposed PPO.

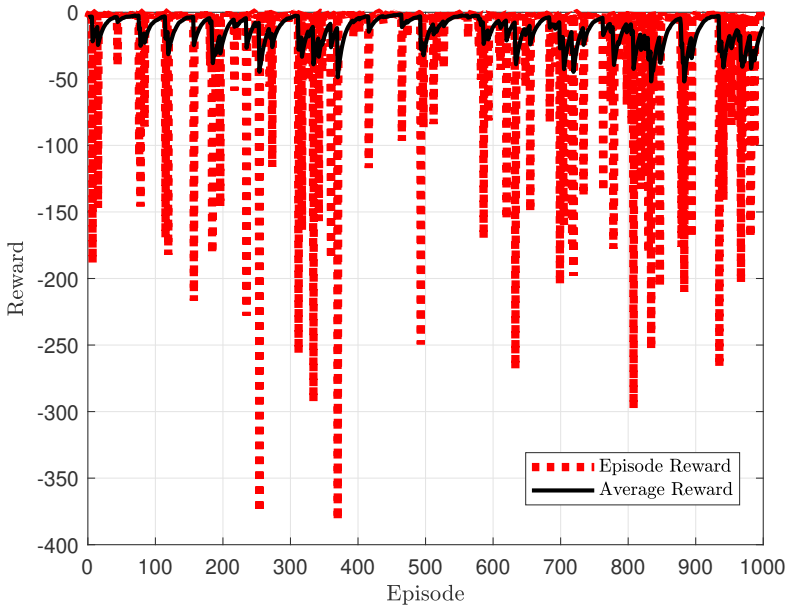

Figure 7: Learning curve without using TEDNN.

The convergence pattern of the reward function in training the PPO is shown in Fig. 6. From this figure, it can be clearly observed that the average reward of the proposed computational ITCG algorithm converges to the steady-state within 100 episodes. To show the effectiveness of the proposed TEDDN+PPO

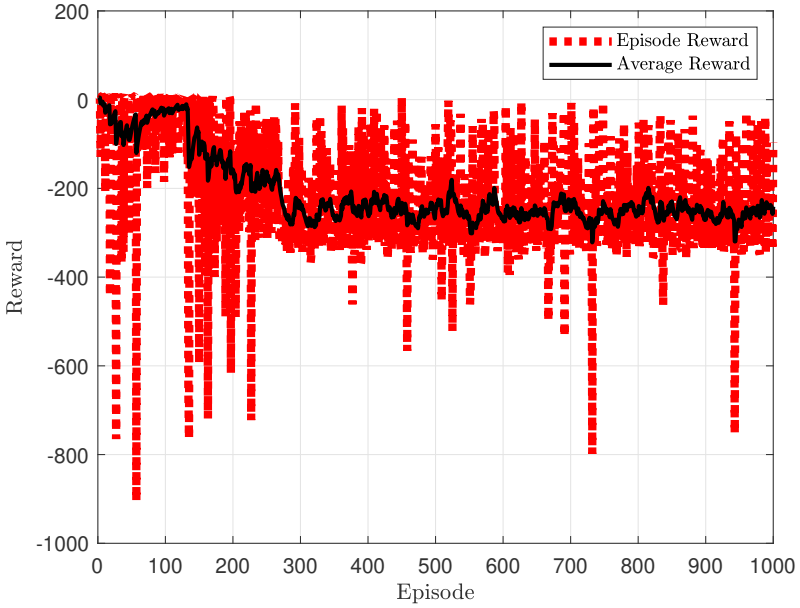

Figure 8: Learning curve of PPO with approximate PNG time-to-go.

concept, we also conduct comparison simulations with training PPO from scratch and training PPO with approximate PNG time-to-go. Training PPO from scratch refers to the concept without using TEDNN to predict the time-to-go and we give a positive reward once the missile intercept the target with desired time. The learning curve of this concept is plotted in Fig. 7, which demonstrates that the learning process is not stable due to the effect of sparse reward. This can be attributed the fact that the probability of intercepting the target with a desired impact time under random initial conditions is very low. Figure 8 presents the learning curve of training PPO with approximate PNG time-to-go, which is determined by [9]

$$\hat{t}_{go} = \frac{\left[1 + \frac{(\theta-\lambda)^2}{10}\right] R}{v} \tag{55}$$

The results in Fig. 8 show that the average reward cannot be maximized if we use approximate time-to-go estimations for realistic scenarios. The reason is that time-to-go estimation (55) assumes that the moving speed of the interceptor is constant and ignores the effect of gravity. This demonstrates the importance of augmenting the TEDNN predictor into the PPO training process to ensure stable learning.

6.2.2. Performance Analysis of the Proposed Computational ITCG Algorithm

In order to show the robustness of the proposed PPO against different scenarios, the simulation is conducted for the target task. We consider a fixed initial condition to investigate the performance of the proposed computational ITCG algorithm. The initial condition is set as

$$x_0 = -20km, \quad y_0 = 20km, \quad v_0 = 200m/s, \quad \theta_0 = 0^\circ \tag{56}$$

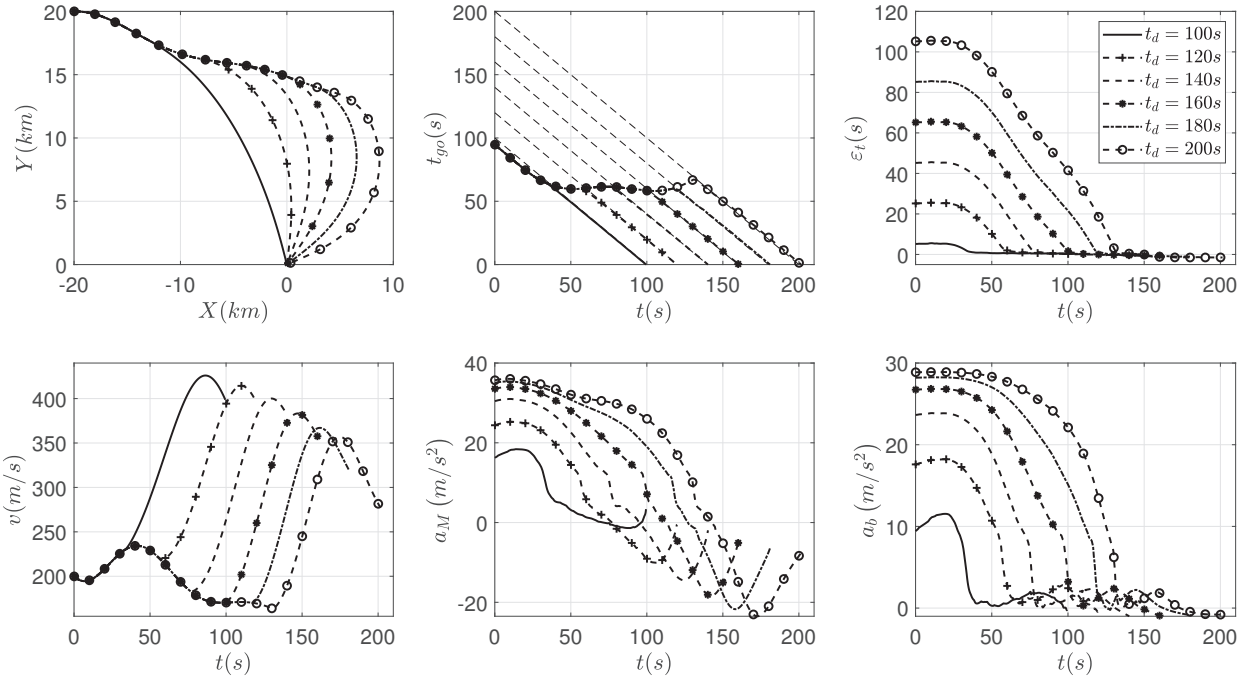

Figure 9: Performance of the proposed ITCG algorithm with different desired impact time.

The simulation results, including interception trajectory, time-to-go history, convergence of impact time error, moving speed, guidance command and biased command, under the proposed computational ITCG algorithm with different desired impact time $t_d = 100s, 120s, 140s, 160s, 180s, 200s$ are presented in Fig. 9. From this figure, it can be clearly observed that the missile can successfully intercept the target at the desired time under the proposed computational ITCG algorithm. The interception trajectory becomes more curved with the increase of the desired impact time and hence requires large biased acceleration command to nullify the impact time error. The results also indicate that the guidance command of the proposed algorithm converges to around zero at the time of impact, thereby providing enough operational margins to cope with undesired external disturbances. The velocity profile shown in Fig. 9 demonstrates that the moving speed significantly changes with different interception trajectories. This indirectly means that the ITCG algorithms under constant-speed assumption cannot cater for practical scenarios. To see this, we further conduct numerical comparisons with PNG and existing analytic ITCG algorithms [9, 11] with desired impact time set as $t_d = 120s$. The guidance commands of these two analytic guidance laws are formulated as

$$\text{ITCG1 in [9]} : a_M = 3v\dot{\lambda} + \frac{-120v^5}{3v\dot{\lambda}R^3}\left(t_d - t - \hat{t}_{go}\right) + g\cos\theta \tag{57}$$

$$\text{ITCG2 in [11]} : a_M = -\frac{3v^2}{R}\left(\theta - \lambda\right) + \frac{100v^2}{R\left(\theta - \lambda\right)}\frac{t_d - t - \hat{t}_{go}}{t_d - t} + g\cos\theta \tag{58}$$

The simulation results, obtained from different guidance laws, are presented in Fig. 10, which demon-

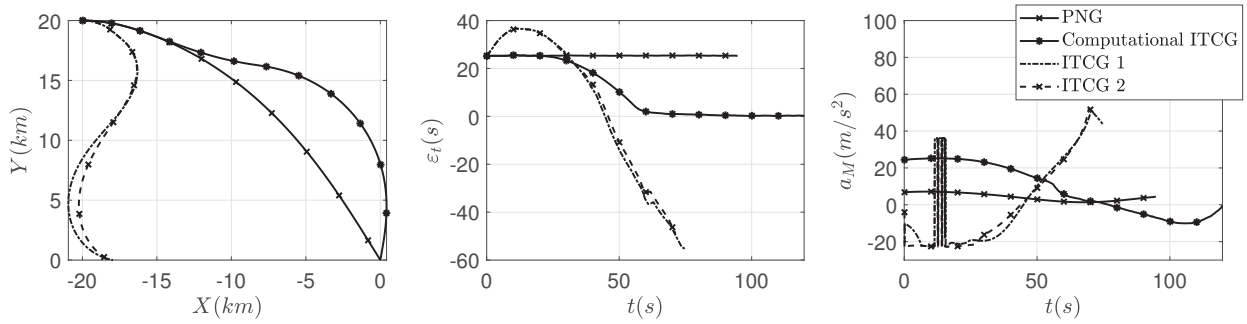

Figure 10: Comparison results with existing analytic ITCG algorithms.

strate that both PNG and the proposed computational ITCG algorithm can successfully guide the missile to intercept the target. Due to the introduced biased term, the proposed computational ITCG algorithm can be leveraged to satisfy the impact time constraint and the recorded interception time is $t_f = 119.8s$. This requires additional control effort to make detour maneuvers to increase the flight time compared to PNG, as confirmed by the profile of the acceleration command. As a comparison, both analytic ITCG laws [9, 11] fail to intercept the target since these two algorithms are derived under ideal conditions.

6.2.3. Monte-Carlo Analysis of the Proposed Computational ITCG Algorithm

To test the proposed computational ITCG algorithm under various conditions, Monte-Carlo simulations are performed with random initial conditions and random desired impact time. The Monte-Carlo simulation results, including interception trajectory and acceleration command, are shown in Fig. 11. To show the benefit of the proposed transfer-ensemble learning, we also compare the proposed algorithm with different specific DNN models + PPO architecture. The statistical comparison results of the impact time error for 100 Monte-Carlo runs are summarized in Table 6. The results reveal that the specific DNN + PPO architectures provide good performance for their own source tasks. However, their performance degrades drastically when the test scenario is different from the source task. As a comparison, the proposed TEDNN + PPO provides significant performance improvement induced by the transfer and ensemble learning.

7. Conclusion

This paper proposes a computational impact-time-control guidance algorithm based on a general prediction-correction concept. A transfer-ensemble deep neural network architecture is proposed as a real-time predictor to estimate the time-to-go under PNG with realistic aerodynamic models. The biased command to nullify the impact time error is developed by utilizing the emerging reinforcement learning techniques. Extensive numerical simulations reveal that the proposed approach provides promising performance in implementation under realistic scenarios.

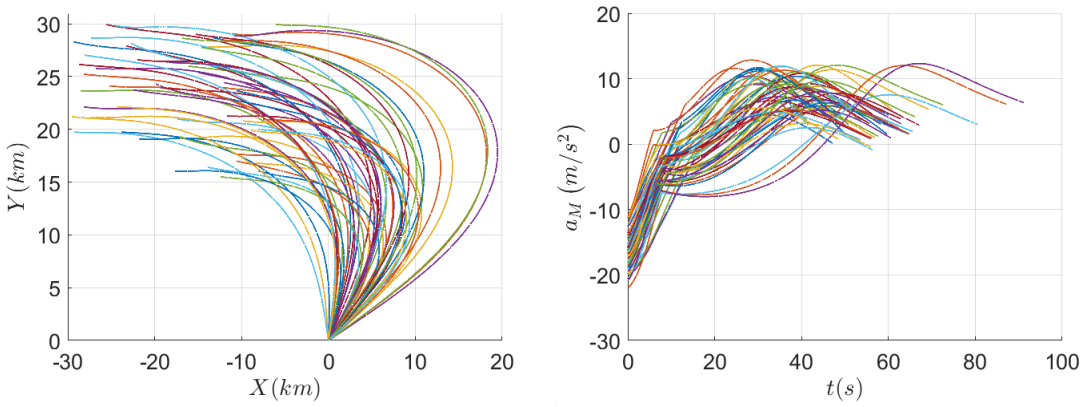

Figure 11: Monte-Carlo simulation results with random initial conditions.

Table 6: Statistical characteristics of the impact time error.

	Source tasks			Target tasks		
Algorithm	Mean	RMSE	MAE	Mean	RMSE	MAE
DNN Model 1 + PPO	0.8206	1.1954	0.8692	-17.4236	18.3347	17.4236
DNN Model 2 + PPO	-0.0756	0.7854	0.4610	-15.2798	16.4749	15.2798
DNN Model 3 + PPO	-0.6489	2.8576	1.2532	-11.2372	12.9349	11.2372
DNN Model 4 + PPO	-1.1295	3.2762	1.3884	-8.7018	9.8613	8.7018
DNN Model 5 + PPO	-4.2617	9.9751	5.6486	-8.9161	14.0909	10.7904
TEDNN Model + PPO				-2.5081	3.7754	2.5081

References

[1] I. Rusnak, H. Weiss, G. Hexner, Optimal guidance laws with prescribed degree of stability, Aerospace Science and Technology 99 (2020) 105780.

[2] T. Han, Q. Hu, M. Xin, Analytical solution of field-of-view limited guidance with constrained impact and capturability analysis, Aerospace Science and Technology 97 (2020) 105586.

[3] B. Kim, Y.-W. Kim, N. Cho, C.-H. Lee, Collision-geometry-based optimal guidance for high-speed target, Aerospace Science and Technology 115 (2021) 106766.

[4] Q. Hu, R. Cao, T. Han, M. Xin, Field-of-view limited guidance with impact angle constraint and feasibility analysis, Aerospace Science and Technology 114 (2021) 106753.

[5] C. Li, J. Wang, S. He, C.-H. Lee, Collision-geometry-based generalized optimal impact angle guidance for various missile and target motions, Aerospace Science and Technology 106 (2020) 106204.

[6] I.-S. Jeon, J.-I. Lee, Optimality of proportional navigation based on nonlinear formulation, IEEE Transactions on Aerospace and Electronic Systems 46 (4) (2010) 2051–2055.

[7] N. Cho, Y. Kim, Optimality of augmented ideal proportional navigation for maneuvering target interception, IEEE Transactions on Aerospace and Electronic Systems 52 (2) (2016) 948–954.

[8] I.-S. Jeon, M. Karpenko, J.-I. Lee, Connections between proportional navigation and terminal velocity maximization guidance, Journal of Guidance, Control, and Dynamics 43 (2) (2020) 383–388. doi:10.2514/1.G004672.

[9] I.-S. Jeon, J.-I. Lee, M.-J. Tahk, Impact-time-control guidance law for anti-ship missiles, IEEE Transactions on Control Systems Technology 14 (2) (2006) 260–266. doi:10.1109/TCST.2005.863655.

[10] T.-H. Kim, C.-H. Lee, M.-J. Tahk, I.-S. Jeon, Biased png law for impact-time control, Transactions of the Japan Society for Aeronautical and Space Sciences 56 (4) (2013) 205–214. doi:10.2322/tjsass.56.205.

[11] M.-J. Tahk, S.-W. Shim, S.-M. Hong, H.-L. Choi, C.-H. Lee, Impact time control based on time-to-go prediction for sea-skimming antiship missiles, IEEE Transactions on Aerospace and Electronic Systems 54 (4) (2018) 2043–2052. doi:10.1109/TAES.2018.2803538.

[12] S. He, D. Lin, Three-dimensional optimal impact time guidance for antiship missiles, Journal of Guidance, Control, and Dynamics 42 (4) (2019) 941–948. doi:10.2514/1.G003971.

[13] N. Cho, Y. Kim, Modified pure proportional navigation guidance law for impact time control, Journal of Guidance, Control, and Dynamics 39 (4) (2016) 852–872. doi:10.2514/1.G001618.

[14] R. Tsalik, T. Shima, Circular impact-time guidance, Journal of Guidance, Control, and Dynamics 42 (8) (2019) 1836–1847. doi:10.2514/1.G004074.

[15] B. Zadka, T. Tripathy, R. Tsalik, T. Shima, Consensus-based cooperative geometrical rules for simultaneous target interception, Journal of Guidance, Control, and Dynamics (2020) 1–8doi:10.2514/1.G005065.

[16] P. Wang, Y. Guo, G. Ma, B. Wie, New differential geometric guidance strategies for impact-time control problem, Journal of Guidance, Control, and Dynamics 42 (9) (2019) 1982–1992. doi:10.2514/1.G004229.

[17] A. Sinha, S. R. Kumar, D. Mukherjee, Three-dimensional guidance with terminal time constraints for wide launch envelops, Journal of Guidance, Control, and Dynamics (2020) 1–17doi:10.2514/1.G005180.

[18] X. Chen, J. Wang, Sliding-mode guidance for simultaneous control of impact time and angle, Journal of Guidance, Control, and Dynamics 42 (2) (2019) 394–401. doi:10.2514/1.G003893.

[19] S. R. Kumar, D. Ghose, Impact time guidance for large heading errors using sliding mode control, IEEE Transactions on Aerospace and Electronic Systems 51 (4) (2015) 3123–3138. doi:10.1109/TAES.2015.140137.

[20] S. He, W. Wang, D. Lin, H. Lei, Consensus-based two-stage salvo attack guidance, IEEE Transactions on Aerospace and Electronic Systems 54 (3) (2017) 1555–1566. doi:10.1109/TAES.2017.2773272.

[21] H.-G. Kim, H. J. Kim, Backstepping-based impact time control guidance law for missiles with reduced seeker field-of-view, IEEE Transactions on Aerospace and Electronic Systems 55 (1) (2018) 82–94. doi:10.1109/TAES.2018.2848319.

[22] H.-G. Kim, D. Cho, H. J. Kim, Sliding mode guidance law for impact time control without explicit time-to-go estimation, IEEE Transactions on Aerospace and Electronic Systems 55 (1) (2018) 236–250. doi:10.1109/TAES.2018.2850208.

[23] R. Tekin, K. S. Erer, F. Holzapfel, Control of impact time with increased robustness via feedback linearization, Journal of Guidance, Control, and Dynamics 39 (7) (2016) 1682–1689. doi:10.2514/1.G001719.

[24] R. Tekin, K. S. Erer, F. Holzapfel, Polynomial shaping of the look angle for impact-time control, Journal of Guidance, Control, and Dynamics 40 (10) (2017) 2668–2673. doi:10.2514/1.G002751.

[25] P. Lu, Introducing computational guidance and control, Journal of Guidance, Control, and Dynamics 40 (2) (2017) 193–193. doi:10.2514/1.G002745.

[26] S. Kang, J. Wang, G. Li, J. Shan, I. R. Petersen, Optimal cooperative guidance law for salvo attack: An mpc-based consensus perspective, IEEE Transactions on Aerospace and Electronic Systems 54 (5) (2018) 2397–2410. doi:10.1109/TAES.2018.2816880.

[27] X. Yan, J. Zhu, M. Kuang, X. Yuan, A computational-geometry-based 3-dimensional guidance law to control impact time and angle, Aerospace Science and Technology 98 (2020) 105672.

[28] P. N. Dwivedi, A. Bhattacharya, R. Padhi, Suboptimal midcourse guidance of interceptors for high-speed targets with

alignment angle constraint, Journal of Guidance, Control, and Dynamics 34 (3) (2011) 860–877. doi:10.2514/1.50821.

[29] H. B. Oza, R. Padhi, Impact-angle-constrained suboptimal model predictive static programming guidance of air-to-ground missiles, Journal of Guidance, Control, and Dynamics 35 (1) (2012) 153–164. doi:10.2514/1.53647.

[30] H. Hong, A. Maity, F. Holzapfel, S. Tang, Model predictive convex programming for constrained vehicle guidance, IEEE Transactions on Aerospace and Electronic Systems 55 (5) (2019) 2487–2500. doi:10.1109/TAES.2018.2890375.

[31] H. Hong, A. Maity, F. Holzapfel, S. Tang, M. Wang, Smooth interpolation-based fixed-final-time command generation, IEEE Transactions on Aerospace and Electronic Systems 55 (6) (2019) 3039–3049.

[32] R. Padhi, M. Kothari, Model predictive static programming: a computationally efficient technique for suboptimal control design, International Journal of Innovative Computing, Information and Control 5 (2) (2009) 399–411.

[33] B. Pan, Y. Ma, R. Yan, Newton-type methods in computational guidance, Journal of Guidance, Control, and Dynamics 42 (2) (2019) 377–383. doi:10.2514/1.G003931.

[34] Y. Ma, B. Pan, Parallel-structured newton-type guidance by using modified chebyshev–picard iteration, Journal of Spacecraft and Rockets (2020) 1–12doi:10.2514/1.A34676.

[35] B. Gaudet, R. Linares, R. Furfaro, Deep reinforcement learning for six degree-of-freedom planetary landing, Advances in Space Research 65 (7) (2020) 1723–1741.

[36] K. Hovell, S. Ulrich, Deep reinforcement learning for spacecraft proximity operations guidance, Journal of Spacecraft and Rockets 58 (2) (2021) 254–264.

[37] B. Gaudet, R. Furfaro, R. Linares, Reinforcement learning for angle-only intercept guidance of maneuvering targets, Aerospace Science and Technology 99 (2020) 105746.

[38] C. E. Oestreich, R. Linares, R. Gondhalekar, Autonomous six-degree-of-freedom spacecraft docking with rotating targets via reinforcement learning, Journal of Aerospace Information Systems (2021) 1–12.

[39] S. He, H.-S. Shin, A. Tsourdos, Computational missile guidance: A deep reinforcement learning approach, Journal of Aerospace Information Systems (2021) 1–12.

[40] V. Shalumov, Cooperative online guide-launch-guide policy in a target-missile-defender engagement using deep reinforcement learning, Aerospace Science and Technology 104 (2020) 105996.

[41] J. T. English, J. P. Wilhelm, Defender-aware attacking guidance policy for the target–attacker–defender differential game, Journal of Aerospace Information Systems 18 (6) (2021) 366–376.

[42] H.-S. Shin, S. He, A. Tsourdos, Computational flight control: A domain-knowledge-aided deep reinforcement learning approach, arXiv preprint arXiv:1908.06884 (2019).

[43] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444. doi:10.1038/nature14539.

[44] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[45] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, arXiv preprint arXiv:1411.1792 (2014).

[46] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, Proceedings of the IEEE 109 (1) (2020) 43–76.

[47] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, 2015, pp. 1889–1897.

[48] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[49] S. He, C.-H. Lee, Optimality of error dynamics in missile guidance problems, Journal of Guidance, Control, and Dynamics 41 (7) (2018) 1624–1633. doi:10.2514/1.G003343.

[50] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[51] R. Patgiri, H. Katari, R. Kumar, D. Sharma, Empirical study on malicious url detection using machine learning, in: International Conference on Distributed Computing and Internet Technology, Springer, 2019, pp. 380–388.

Learning prediction-correction guidance for impact time control

Liu, Zichao

2021-10-28

Liu Z, Wang J, He S, et al., (2021) Learning prediction-correction guidance for impact time control. Aerospace Science and Technology, Volume 119, December 2021, Article number 107187 https://doi.org/10.1016/j.ast.2021.107187