IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Maneuver-Conditioned Decision Transformer for Tactical In-Flight Decision-Making

Hoseong Jung, Yong-Duk Kim, and Youngjung Kim*

***Abstract*— Autonomous maneuver decision in air combat is a challenging task with high-dimensional state-action spaces and nonlinear dynamics. Existing approaches are usually based on online learning paradigms, which hinders their application to real-world scenarios where online interactions, *i.e.*, trial-and-error, are impractical or dangerous. In this paper, we explore the offline reinforcement learning framework for tactical air combat. To this end, we first construct a large-scale offline dataset of demonstrations from hand-designed planners, humans, and expert policies using an interactive simulator. A transformer-based architecture with a lightweight maneuver pool is then proposed to store and retrieve information for generating effective tactical maneuvers, while maintaining the sequential modeling ability of the decision transformers. The maneuver pool is structured in a key-value memory space, where key-value pairs are used for addressing and reading the maneuver pool. We also propose pool diversity and centralizing losses to learn our offline policy, boosting the discriminative power of learned features from the offline dataset. Our formulation allows us to learn maneuver-specific feature prototypes, and to explicitly leverage such knowledge during inference. Extensive experimental results and ablation studies demonstrate the effectiveness and flexibility of the proposed method, outperforming other offline baselines.**

I. INTRODUCTION

Autonomous agents are widely utilized for surveillance, reconnaissance, and search, requiring a high level of decision-making capabilities to respond to complex and ever-changing environments. It is thus essential to perform a wide variety of missions accompanied by highly skilled and split-second decisions. This capability has been often validated as such in Atari games [1] where states were extracted from pixel-level information. However, these environments are far from being applicable to dynamic and complex real-world tasks since 2D representations of the visual world are involved only. Here, we consider a 6-degree of freedom (DOF) flight simulation environment for playing a within visual range (WVR) air combat. The task of air combat involves flight dynamics models and high-dimensional continuous states, and is composed of multiple tactical maneuvers, such as pursuit curve, flat scissors, and defensive spiral [2]. These properties pose significant challenges for accurate air combat decision-making. To accomplish this task, the agent should be able to perceive potentially uncertain surroundings, and generalize over different engagement scenarios.

This work was supported by the Agency For Defense Development grant funded by the Korean Government in 2023.

Hoseong Jung, Yong-Duk Kim, and Youngjung Kim are with the Agency for Defense Development, Republic of Korea {ghtjdaleka, ydkim.4653}@gmail.com, read12300@naver.com

*Corresponding author

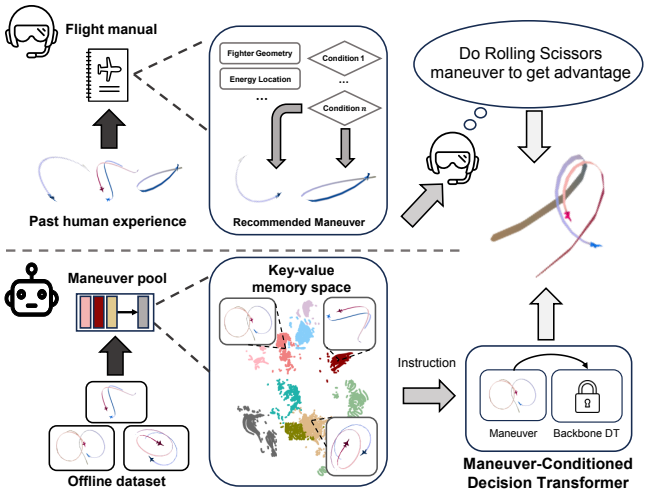


Fig. 1: The flight manual [2] is a technical manual that describes the tactics and its operating procedures. The beginning pilots are trained to learn this knowledge so that they are able to complete various missions. We propose a lightweight maneuver pool in the offline RL framework to mimic the aforementioned procedure.

Various methods have been devised to address air combat [3]–[14], which can broadly be divided into two categories: Conventional planning and control methods and learning-based methods. Conventional approaches [3]–[8] design rule-based policies, typically using the basic fighter maneuver (BFM) derived from human pilot experience [2]. For example, Yang *et al.* [8] codify a behavior tree to choose a specific maneuver in the BFM, followed by the pure pursuit algorithm for calculating future trajectories that will move an aircraft. While conventional methods permit the incorporation of prior human knowledge in decision-making, they require complex handcrafted rules for manually categorizing the type of maneuvers and adjusting parameters.

On the other hand, learning-based methods [9]–[14] with online reinforcement learning (RL) have been shown to be among the most successful ones to solve air combat. Without expert experience or supervision, online RL methods learn to maximize a specified reward through actively interacting with environments by taking actions and receiving rewards, and achieve superhuman performance [9]–[14]. These methods have become increasingly popular thanks to their ability to leverage high-capacity deep architectures, allowing agents to make tactical decisions on complex domains. They however suffer from sample inefficiency, and rely on active data collection which involves the high cost and danger of interacting with environments [15]. In addition, online methods have difficulty leveraging a static dataset of previously

collected trajectories for training [16]. Very recently, a new paradigm called offline RL has been introduced to extract an optimal policy from trajectories collected by a set of behavior policies. A number of works have illustrated the power of this paradigm for data-driven learning of policies in Atari game [17], [18] and robotic manipulation [19]. However, to our best knowledge no previous work has focused on utilizing offline RL for the task of air combat.

In this paper, we present the offline RL framework to determine appropriate maneuvers and actions in air combat. To realize this, we first collect a diverse training dataset from hand-designed controllers, humans, and online RL algorithms with an interactive simulator, making it more reflective of practical settings. We then adopt the decision transformer [17], as a baseline architecture, to obtain a sequence modeling objective for air combat. Next, we assume that it is possible to extract additional meaningful information from previously collected trajectories, similar to the flight manual [2] contextualized from past human experience. To implement this idea, we propose a lightweight maneuver pool to record prototypical representations of diverse maneuvers, reminiscent of the learned flight manual (see Fig. 1). These are structured in a key-value memory space, where individual keys in the pool are used in a query mechanism to dynamically lookup a relevant maneuver based on the input state history. Paired values act as conditioning prompts that condition the decision transformer on maneuver-specific instruction for determining the next actions. We also present pool diversity and centralizing losses to enhance the discriminative power of our maneuver pool. Our formulation allows to learn maneuver-specific feature prototypes, and to recall the most relevant features during air combat maneuvering.

Our main contributions can be summarized as follows:

- We collect a large and diverse offline dataset generated via hand-designed controllers, human demonstrators, and expert policies for air combat, which can be used to learn a policy without further interaction.
- Inspired by the flight manual [2], we propose a decision transformer architecture with learnable and lightweight maneuver pool, called maneuver-conditioned decision transformer (M-DT). The maneuver pool conditions the decision transformer on maneuver-specific instruction for making in-flight decisions.
- We demonstrate the effectiveness and flexibility of the M-DT with extensive experiments and ablation studies on the high-fidelity flight simulation environment [20].

II. RELATED WORKS

A. Online RL for Air Combat

Online RL has become a standard technique for training agents that solve the air combat task. Considerable efforts have been dedicated to devising effective exploration techniques for high-dimensional simulation environments. The predominant approach, commonly referred to as the soft-actor critic (SAC) [21] based method, employs stochastic actors to ensure sample efficiency and sufficient exploration [9]–[13]. In addition, hierarchical structures simplify the air combat complexities, dividing decision-making into inner and outer loops to model agents at various abstraction levels [9], [10]. These methods effectively diminish the state-action spaces to a comparatively smaller sub-space, thereby mitigating the necessity for extensive exploration. Meanwhile, various strategies have been devised to prevent bias towards a specific scenario, reflecting the diversity and variation in the air combat environment. For instance, Li *et al*. [11] introduce a self-play algorithm to collect a diverse set of sample data generated by agents operating in different environments. Several lines of research introduce a curriculum learning strategy in online RL, which gradually increases the difficulty of the task contributing to a more comprehensive understanding of the range of scenarios encountered in air combat [13], [14]. However, all the methods based on online RL require active interactions with the environment to iteratively collect a new set of experiences. This process is time-consuming and presents challenges in effectively utilizing pre-collected static data. In this paper, we investigate the offline RL framework for air combat, which learns a policy from static datasets without further interaction. This is valuable in our setting since policies can be fine-tuned or updated using previously collected data for better generalization.

B. Transformers in offline RL

Motivated by the remarkable success of transformer models in natural language processing (NLP) and computer vision (CV), researchers have explored their applicability in solving offline RL problems. The seminal work of decision transformer (DT) [17] transforms the RL problem into a sequence modeling problem by autoregressively generating current actions conditioned on sequences of past states, actions, and desired returns. Monastirsky *et al*. [22] demonstrate the sim-to-real capabilities of DT in multi-goal object-throwing scenarios with a robotic arm. Building on the success of DT, subsequent research has emphasized its generalization capabilities to enhance multi-task and task adaptation performance. The multi-game decision transformer [18] extended DT to a multi-game setup, demonstrating its ability to generalize to various Atari games with human-level performance. Prompt-DT [23] designs the trajectory prompt, which contains the few-shot demonstrations, and encodes task-specific information to guide policy generation. Hyper-DT [24] incorporates a hyper-network to generate task-specific adaptation modules for DT based on task embeddings. These methods need clear task boundaries to provide task-specific demonstrations during training and testing. Thus, directly applying Prompt-DT and Hyper-DT to air combat is non-trivial since it is impossible to manually provide proper demonstrations required for every time-step.

III. PREREQUISITES

A. Air Combat Environment

We are concerned with learning in a Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$. The tuple

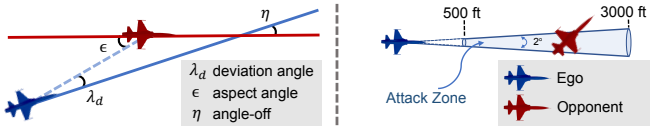


Fig. 2: Depictions of combat parameters and attack range.

is comprised of actions $a \in \mathcal{A}$, states $s \in \mathcal{S}$, transition function $P(s'|s,a)$, and reward function $r = \mathcal{R}(s,a)$. The MDP environment for air combat is simulated by integrating the JSBSim flight dynamics model (FDM), under the OpenAI Gym interface [20]. The JSBSim is a high-fidelity open-source FDM that is highly accurate for modeling the physics of flight and aerodynamics, and is used to model realistic state space $\mathcal{S}$ and transition function P. We consider the F-16 aircraft as ego and opponent agents. In this setting, the state at current time step t is represented by $s_t = [s_t^e, s_t^o]$. s_t^e contains aircraft aerodynamic states of the ego agent as:

$$s_t^e = [h^e, l^e, \phi^e, \theta^e, \psi^e, \omega^e, v^e, \dot{v}^e], \tag{1}$$

where h^e is the altitude and l^e denotes the remaining fuel. $(\phi^e, \theta^e, \psi^e)$ are the roll, pitch, and yaw angles in the body frame, respectively. $\omega^e = [\omega_x, \omega_y, \omega_z]$, $v^e = [v_x, v_y, v_z]$ and $\dot{v}^e = [\dot{v}_x, \dot{v}_y, \dot{v}_z]$ denote the angular rate, the velocity, and the acceleration vectors in the body frame, respectively. We encode information of the opponent agent in s_t^o as:

$$s_t^o = [\lambda_d, \epsilon, \eta, d, v^o, \dot{v}^o], \tag{2}$$

where $(\lambda_d, \epsilon, \eta)$ are the deviation angle, aspect angle, and angle-off as depicted in Fig. 2. d denotes the relative distance vector (radial distance, elevation angle, azimuth angle) expressed using the spherical coordinates between the ego and opponent agents. $(v^o, \dot{v}^o)$ are the opponent's velocity and acceleration vectors in the ego's body frame. We normalize the state observation (s_t^e, s_t^o) to $[-1, 1]$. The action space contains four control commands of the F-16 aircraft as follows:

$$a_t = [u_a, u_e, u_r, u_{th}], \tag{3}$$

where (u_a, u_e, u_r, u_{th}) are the continuous-valued aileron, elevator, rudder, and throttle commands, respectively. The reward function r is a weighted summation of the several factors as in [13]. It takes into account the damages, deviation angle, aspect angle, and minimum altitude (see [13] for more details). We set the maximum episode length and the minimum aircraft height to 3,000 (5 minutes) and 1,000 feet, respectively. The attack range is defined in Fig. 2, and the damage estimation method is the same as that of the AlphaDogfight Trial [9].

B. Decision Transformer

Decision Transformer (DT) treats offline RL as a sequential modeling problem which predicts the next actions autoregressively using a causal self-attention mask [17]. It models trajectories with tuples of state s_t, action a_t, and return-to-go $\hat{r}_t$ gathered at different time steps. At current timestep t, DT takes a sequence of trajectories $\tau_{t,H}$, *i.e.*, the most recent H-step history as input, and outputs action a_t as follows:

$$\begin{aligned} \tau_{t,H} &= \{\hat{r}_{t-H+1}, s_{t-H+1}, a_{t-H+1}, ..., \hat{r}_t, s_t\}, \\ a_t &= \mathrm{DT}(\tau_{t,H}). \end{aligned} \tag{4}$$

The return-to-go is obtained as $\hat{r}_t = \Sigma_{t'=t}^{T} r_i$ during training, where T is termination timesteps. At test time, we use $\hat{r}_t = G - \Sigma_{t'=0}^{t} r_i$ where G is the target return for an entire episode. DT concatenates the timestep embedding into a trajectory token to encode sequential sequence information, and is trained to predict a_t by minimizing mean squared error (MSE) loss when dealing with continuous action spaces.

IV. METHODS

In this section, we first describe our motivation and the proposed method for learning policies that can apply to air combat automatically. We then provide the training procedure and our loss functions in detail. We follow two key design principles: (1) learning maneuver-specific feature prototypes as a privileged context using diverse offline datasets and (2) imposing such knowledge into self-attention blocks in DT, analogous to the flight training process of beginning pilots using the flight manual [2] as depicted in Fig. 1.

A. Maneuver-Conditioned Decision Transformer

1) Overview: We show in Fig. 3 an overview of our offline RL framework, called M-DT. It mainly consists of four components: a query module (QM), maneuver pool, maneuver injection module (MIM), and DT. We first define a sequence of L-step state history $s_{t-L+1:t}$ at timestep t:

$$s_{t-L+1:t} = \{s_{t-L+1}, s_{t-L}, ..., s_t\}. \tag{5}$$

The state encoder f consisting of embedding and linear layers takes $s_{t-L+1:t}$, and extracts query features $\boldsymbol{q}_t = f(s_{t-L+1:t})$. The query features are then used to retrieve and update maneuver-specific information in the maneuver pool with the QM. We feed the retrieved information and $\tau_{t,H}$ into the DT module through the MIM for predicting the action a_t. The goal of MIM is to inject information from the maneuver pool into intermediate features in DT for guided decision-making. We train M-DT end-to-end without pretraining. At test time, we decrement the target return by the achieved reward repetitively until episode termination following [17].

2) Maneuver pool and conditioning: The maneuver pool contains M items recording diverse prototypical patterns of maneuvers. M is the parameter to model the variation of maneuvers, which can vary according to the environment and dynamics. In the flight manual [2], the maneuver is manually categorized in the form of BFM (e.g., defensive spiral and flat scissors), and recommended according to the current state based on past human experiences. Contrarily, we try to learn such knowledge from large offline datasets. Each item consists of a pair of keys and values. The individual key $\boldsymbol{\kappa}_i$ $(i = 1, ..., M)$ is used to address the items, and also implicitly categorizes maneuver patterns. It is a C-dimensional vector where C is the embedding dimension. Inspired by [25], we define the paired value as

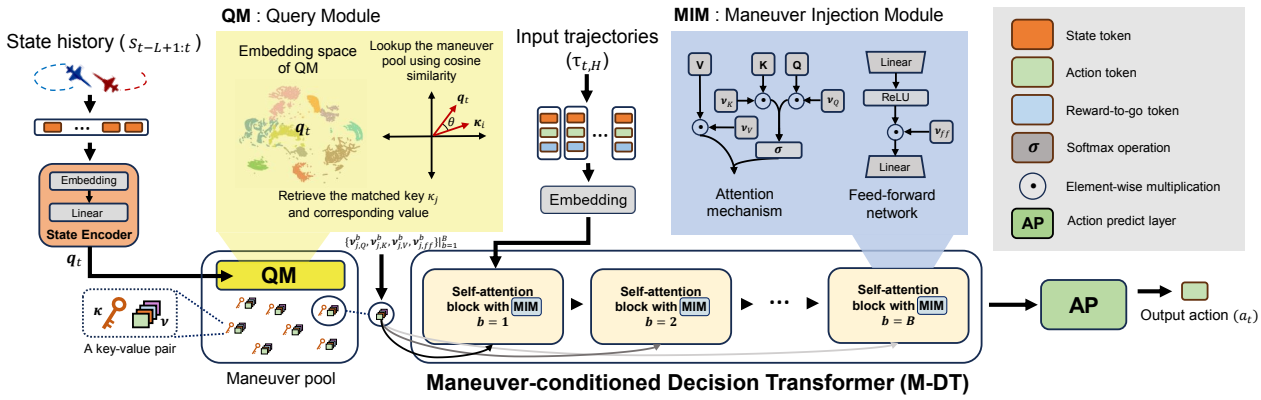


Fig. 3: Overview of M-DT for air combat. M-DT is built upon DT architecture. We augment DT by introducing a lightweight maneuver pool and a query mechanism for guiding the decision process in DT. To this end, M-DT takes both $\tau_{t,H}$ and $s_{t-L+1:t}$, where $s_{t-L+1:t}$ is used to lookup value vectors in the maneuver pool (Eq. (6)). The retrieved value vectors are then injected into self-attention blocks in DT through MIM (Eq. (7)).

a set of trainable vectors $\{\boldsymbol{\nu}^b_{i,Q}, \boldsymbol{\nu}^b_{i,K}, \boldsymbol{\nu}^b_{i,V}, \boldsymbol{\nu}^b_{i,ff}\}^B_{b=1}$. Each set is associated with a query, key, value, and feed-forward activation of each self-attention block b in DT. These vectors parameterize maneuver-specific instructions, and act as a privileged context for guiding the decision process in DT, as illustrated in Fig. 3.

Next, we describe how to retrieve the appropriate value according to $\boldsymbol{q}_t$ computed from the state history $s_{t-L+1:L}$ (QM). Given a query $\boldsymbol{q}_t$, we measure the similarity between $\boldsymbol{q}_t$ and all keys to lookup the closest key $\boldsymbol{\kappa}_j$:

$$\boldsymbol{\kappa}_j = \underset{\boldsymbol{\kappa}_i \in \mathbf{K}}{\operatorname{argmax}} \cos(\boldsymbol{q}_t, \boldsymbol{\kappa}_i), \tag{6}$$

where $\cos(\cdot,\cdot)$ returns the cosine similarity. We denote the set of all keys by $\mathbf{K} = \{\boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, ..., \boldsymbol{\kappa}_M\}$. The corresponding value vectors $\{\boldsymbol{\nu}^b_{j,Q}, \boldsymbol{\nu}^b_{j,K}, \boldsymbol{\nu}^b_{j,V}, \boldsymbol{\nu}^b_{j,ff}\}^B_{b=1}$ are then retrieved. We feed these vectors into DT through the MIM that exploits maneuver-specific information encoded in the retrieved value, and explicitly injects to the features of each self-attention block. We adopt $(IA)^3$ method [25] introduced in the NLP literature to implement this injection. Note that the attention block in DT is composed of two sequentially connected self-attention and feed-forward layers. Using the notation from [25], the overall process of MIM is:

$$\begin{aligned} \text{softmax}\left(\frac{(\boldsymbol{\nu}_Q \odot \boldsymbol{Q})(\boldsymbol{\nu}_K \odot \boldsymbol{K}^\top)}{\sqrt{n}}\right)(\boldsymbol{\nu}_V \odot \boldsymbol{V}), \\ (\boldsymbol{\nu}_{ff} \odot ReLU(\boldsymbol{W}_1 x))\boldsymbol{W}_2, \end{aligned} \tag{7}$$

where $(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ denotes query, key, and value in the self-attention layer and $\odot$ represents to element-wise multiplication. n is the normalization factor. $\boldsymbol{W}$ is the weight matrix and x is the input for the feed-forward layers. In Eq. 7, the indexes (j, b) of value are omitted for simplicity. We train the maneuver pool with a large number of offline trajectories, enabling the most representative features to be stored and transferred to DT through the MIM. All values are initialized to ones, and remain unchanged during the early stage of training (warm-up). We initialize all keys to $[-1, 1]$.

B. Training

We train our M-DT using MSE, pool centralizing, and pool diversity losses end-to-end. The warm-up strategy is applied to the values in the maneuver pool, fully leveraging the sequential modeling ability and high capacity of DT. Given the ground-truth actions a^*_t in offline trajectories, the MSE loss makes the predicted actions a_t from M-DT similar to a^*_t by penalizing the L2 distance:

$$\mathcal{L}_{MSE} = ||a^*_t - a_t||^2, \tag{8}$$

where $a_t = \text{M-DT}(\tau_{t,H}, s_{t-L+1:t})$. It is important to store representative and discriminative maneuver-specific features in the pool. To this end, we add additional constraints including the centralizing and diversity losses. The centralizing loss encourages the queries to be close to the nearest key in the maneuver pool as:

$$\mathcal{L}_{cent} = -\cos(\boldsymbol{q}_t, \boldsymbol{\kappa}_j), \tag{9}$$

where j is an index of the nearest key for $\boldsymbol{q}_t$, and is obtained by solving Eq. 6 during training stage. It helps similar state histories $s_{t-L+1:t}$ to be mapped closely in the embedding space of QM. The diversity loss aims to regularize key-value pairs to become different from others, and prevent them from being identical as follows:

$$\begin{aligned} \mathcal{L}_{div} = \max(\| \mathbf{K}\mathbf{K}^\top - I \|_F - \chi_{\mathbf{K}}, 0) \\ + \max(\| \mathbf{V}\mathbf{V}^\top - I \|_F - \chi_{\mathbf{V}}, 0), \end{aligned} \tag{10}$$

where we denote the set of all values by $\mathbf{V} = \{\boldsymbol{\nu}_i\}^M_{i=1}$. Note that here we apply the diversity loss for $\mathbf{V}$ to each self-attention block b and feature (Q, K, V, ff) separately. $\|\cdot\|_F$ is the Frobenius norm and χ denotes a learnable vector that controls the extent of overlap between pool items. Intuitively, each key (or value) is trained to be orthogonal when $\chi_{\mathbf{K}}$ (or $\chi_{\mathbf{V}}$) is close to 0. The overall objective of M-DT is a weighted summation of all loss functions defined as:

$$\mathcal{L}_{All} = \mathcal{L}_{MSE} + \alpha\mathcal{L}_{cent} + \beta\mathcal{L}_{div}, \tag{11}$$

where α and β are the balancing parameters.

V. EXPERIMENTS

In this section, we commence by outlining the procedures of constructing our offline dataset for air combat, along with presenting implementation details. Next, we provide ablation

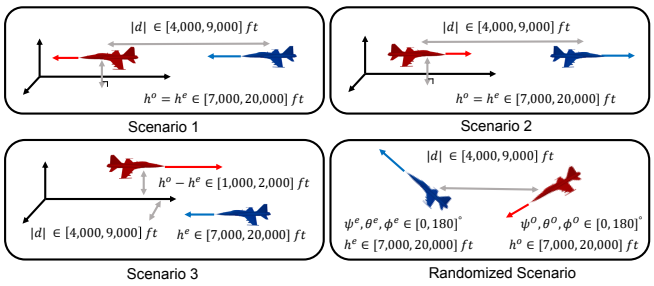


Fig. 4: Combat scenarios are designed based on the difficulty associated with initial geometric positions. Scenarios 1, 2, and 3 are employed to collect an offline dataset and evaluate the proposed method. Randomized Scenario is used for evaluation only. The superscripts e and o denote the ego and the opponent agents, respectively.

studies to analyze the importance of components of M-DT. Quantitative comparisons between ours and state-of-the-art offline RL methods are then presented. Finally, we provide a detailed analysis of the maneuvers generated by M-DT.

A. Experimental Setup

1) Combat Scenario: In the following, we design combat scenarios to reflect the diversity and variation in the air combat environment. Based on the flight manual, air combat task has varying degrees of difficulty depending on the initial geometric position between the ego and the opponent. We detail combat scenarios in Fig. 4, each varying in difficulty based on initial angle and energy level. For collecting offline datasets, we design three scenarios: (1) Scenario 1, the easiest situation, features equalized energy levels with the ego agent having an angular advantage. (2) Scenario 2 equalizes energy levels again but places the ego agent at an angular disadvantage. It indicates how effectively counter-attacks are executed, and it may become less challenging for experienced egos. (3) Scenario 3 places the ego at a disadvantageous energy level with a neutral angle. In each scenario, the ego and the opponent maintain uniform postures, with distances $|d|$ varying randomly from 4,000 to 9,000 ft between them. Initial altitudes of the ego and the opponent agents (h^e, h^o) vary within the range of 7,000 to 20,000 ft, satisfying scenario constraints. Additionally, we introduce Randomized Scenario featuring randomized initial attitude angles ($\phi^e, \theta^e, \psi^e, \phi^o, \theta^o, \psi^o$) to evaluate agents with initial geometric positions which are not encountered during training. In this scenario, horizontal distances are set randomly between 4,000 and 9,000 ft, and altitudes vary independently within the range of 7,000 to 20,000 ft.

TABLE I: Statistics of our offline dataset used for training. R_{norm} is the normalized average return as defined in Eq. (12).

Data source	R_{norm}	# of episodes	# of samples
Human	51.96	112	0.14M
HDP	50.22	1,500	2.95M
Online RL	100	1,500	1.78M

(a) (b)

Fig. 5: (a) Interactive simulator (b) A snapshot of the data collection procedure, *i.e.,* professional human pilot (top) vs HDP opponent (bottom).

2) Data Collection: To reflect the heterogeneous nature of data collected in practice, we consider a set of behavior policies for the ego agent: Human, Hand-designed policy (HDP) [8], and Online RL [13]. During data collection, we use Scenario 1 to 3 and fix the opponent agent to HDP.

Human. Human dataset is collected using an interactive simulator equipped with stick, pedal, and throttle lever. It allows human pilots to directly manipulate the ego aircraft. Fig. 5 illustrates the simulator and provides an example of the data collection procedure. Human dataset consists of a total of 112 episodes (eight hour-long trajectories).

Hand-designed policy (HDP). We use the model of [8] derived from a flight manual. It is based on a behavior tree and pure pursuit algorithm, achieving superhuman performance. The dataset is collected by generating 500 episodes for each Scenario 1 to 3, ensuring that all maneuvers outline in behavior tree are executed.

Online RL. We train an online agent by using the method proposed in [13]. It is based on soft actor-critic (SAC) [21] and curriculum learning. The agent is trained to engage against the HDP opponent through online interactions with the environment. The resulting agent is then used to collect trajectory samples in Scenario 1 to 3. The collected dataset used for offline training is summarized in Table I. Note that Randomized Scenario is not included in the training set.

3) Metrics: We report the averaged win rate (Win), lose rate (Lose), remaining health of ego (Health), and damage incurred by opponent (Damage). We also report R_{norm} by normalizing average returns R roughly to the range between 0 and 100 following the protocol of [19]:

$$R_{norm} = 100 \times (R - R_{random})/(R_{expert} - R_{random}). \quad (12)$$

A normalized average return of 0 corresponds to the average returns (R_{random}) of an agent taking actions at random across the action space. A score of 100 corresponds to the average returns (R_{expert}) of a domain-specific expert. In our case, this is the performance of the online agent used to collect Online RL dataset. During the evaluation, the aforementioned metrics are measured over 100 episodes for each scenario separately.

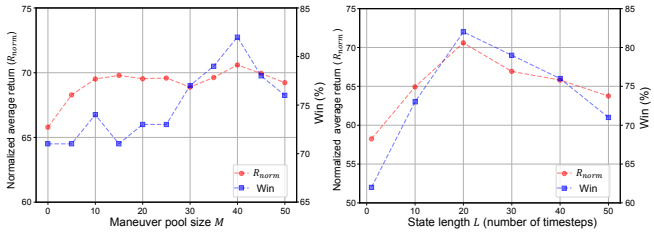


Fig. 6: Ablation: The effect of maneuver pool size and state length. (a) "R_{norm}" and "Win" with regards to maneuver pool size M, given $L = 20$. (b) "R_{norm}" and "Win" with regards to state length L, given $M = 40$.

4) Training Configuration: The proposed method is implemented using the PyTorch framework. We set B, H, and C to 8, 20, and 512, respectively for DT architecture. The same configuration is used for all DT-based methods in the comparisons. Our models are trained for 1,000,000 iterations using the AdamW optimizer with weight decay of 10^{-4}, and the learning rate of 10^{-4}. The overall loss function in Eq. 11 is minimized with $\alpha = 0.1$ and $\beta = 10^{-5}$. The hyperparameters (M and L) of maneuver pool are set to 40 and 20, respectively based on our ablation study. For evaluation, we select the checkpoint based on validation performance in Randomized Scenario across 100 validation seeds.

B. Ablation Study

1) Effect of Hyperparameters for M-DT: Recall that there are two key hyperparameters: the size of the maneuver pool M, and the state length L. Intuitively, M decides the total number of maneuvers present in air combat, and L decides the number of timesteps that are used to extract maneuver-specific information. Increasing the M shows a positive effect on performance as shown in Fig. 6(a), suggesting the necessity of a sufficiently large M to encode every maneuver in diverse engagement scenarios. Conversely, Fig. 6(b) indicates that M-DT performs optimally with an L of 20 timesteps, implying that the best maneuvers are extracted with 2 seconds of information. Taking fewer timesteps as input led to worse performance due to a lack of perception of the surrounding situation while taking more timesteps led to a marginal performance drop.

2) Effect of Dataset Diversity: We demonstrate that our method can result in better performance by integrating diverse data sources into the offline dataset. Table II (rows 1, 2, 6) compares the performance based on the specific datasets utilized. Results demonstrate higher performance using the full dataset compared to the expert dataset alone, providing evidence that the model learns more effectively with varying levels of data distribution. For example, the expert dataset never incurs a penalty for falling while the non-expert dataset does. Consequently, the final trained model accounts for this distinction, exhibiting more anti-fall maneuvers. The analysis of Table II (row 2) is described in Section V.D.

3) Effect of Loss Functions: Subsequently, we conduct experiments to observe the effect of our proposed loss function. To this end, we train M-DT with different combinations of loss functions and present the results in Table II (rows 3-6).

TABLE II: Ablation studies on the utilized dataset and components of objective functions. It presents the average performance in Randomized Scenario across 100 different test seeds.

Dataset			Objective		R_{norm}	Win	Lose	Health	Damage
Online RL	HDP	Human	$\mathcal{L}_{cent}$	$\mathcal{L}_{div}$					
✓	✗	✗	-	-	64.63	69	21	74.84	68.04
✗	✓	✓	-	-	53.53	61	27	**78.76**	38.17
-	-	-	✗	✗	67.49	73	25	64.82	76.73
-	-	-	✗	✓	69.06	76	16	66.53	74.11
-	-	-	✓	✗	70.06	80	15	64.44	77.68
✓	✓	✓	✓	✓	**70.59**	**82**	**15**	66.03	**82.88**

Applying each of the centralizing loss $\mathcal{L}_{cent}$ and the diversity loss $\mathcal{L}_{div}$ encourages M-DT to discriminate learned features from the offline dataset, resulting in improved performance. The performance is further improved when both $\mathcal{L}_{cent}$ and $\mathcal{L}_{div}$ are applied simultaneously.

C. Comparison with Baselines

We compare the decision-making ability of M-DT with various baselines, including both conventional and state-of-the-art learning-based methods. Since there are no offline RL methods dedicatedly designed for air combat, we re-implement the existing methods on our environment using source codes provided by the authors. For fair comparisons, all the offline RL methods are implemented using the same DT architecture as described in Section V-A.4, and are trained until convergence.

- **Hand-designed policy (HDP)** [8]. This is the policy used to collect our offline dataset (see Section V-A.2).
- **Behavior cloning (BC)**. BC is trained only using Online RL data in Table I, and is not conditioned on the return-to-go. The result of BC helps show the effect of return-to-go tokens in transformers.
- **Decision transformer (DT)** [17]. We train DT using our offline dataset to show the effectiveness of M-DT.
- **Prompt-DT** [23]. Prompt-DT is a variant of DT, and integrates task-specific information sampled from expert policies into DT through prepending. It allows us to compare the conditioning method of M-DT with that of Prompt-DT for air combat.
- **Hyper-DT** [24]. We also compare another conditioning method proposed in Hyper-DT. It leverages adapter layers trained from expert demonstrations to insert task-specific information to DT.

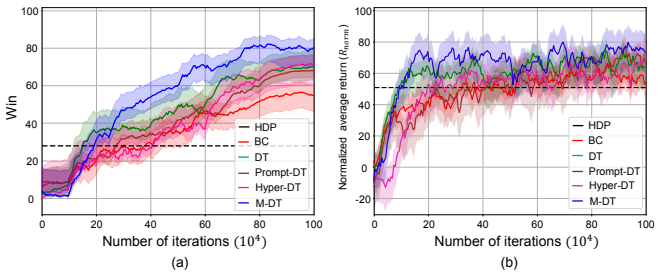


Fig. 7: Validation curves of M-DT and other baselines. We report (a) Win and (b) R_{norm} in Randomized Scenario according to the number of training iterations. All runs are shown with mean (curve) and standard deviation (shaded area) computed over 100 different validation seeds.

TABLE III: Quantitative evaluation results of M-DT and other baselines. It presents the average performance in each scenario across 100 different test seeds.

Method	Randomized Scenario					Scenario 1					Scenario 2					Scenario 3				
	R_{norm}	Win	Lose	Health	Damage	R_{norm}	Win	Lose	Health	Damage	R_{norm}	Win	Lose	Health	Damage	R_{norm}	Win	Lose	Health	Damage
HDP [8]	50.85	28	33	70.21	27.43	71.20	48	0	64.96	45.79	26.85	3	53	54.39	33.10	52.61	28	37	66.86	36.92
BC	54.78	55	42	56.85	63.51	95.89	100	0	97.03	99.74	73.63	83	16	81.89	65.26	49.62	40	58	54.19	42.92
DT [17]	65.79	71	24	64.91	73.16	96.31	100	0	**100.00**	99.42	76.89	85	14	78.78	77.99	61.26	61	39	62.83	48.01
Prompt-DT [23]	65.55	65	25	61.53	74.19	99.49	100	0	98.95	99.76	76.53	87	8	**87.95**	71.57	61.92	65	27	66.51	48.82
Hyper-DT [24]	67.55	72	18	**74.33**	70.75	95.84	100	0	97.89	98.56	78.22	91	6	86.64	70.72	61.59	66	30	68.32	45.47
M-DT	**70.59**	**82**	**15**	66.03	**82.88**	**101.83**	100	0	99.00	**100.00**	**78.96**	**91**	**5**	86.99	**95.95**	**63.25**	**68**	**22**	**68.44**	**51.56**

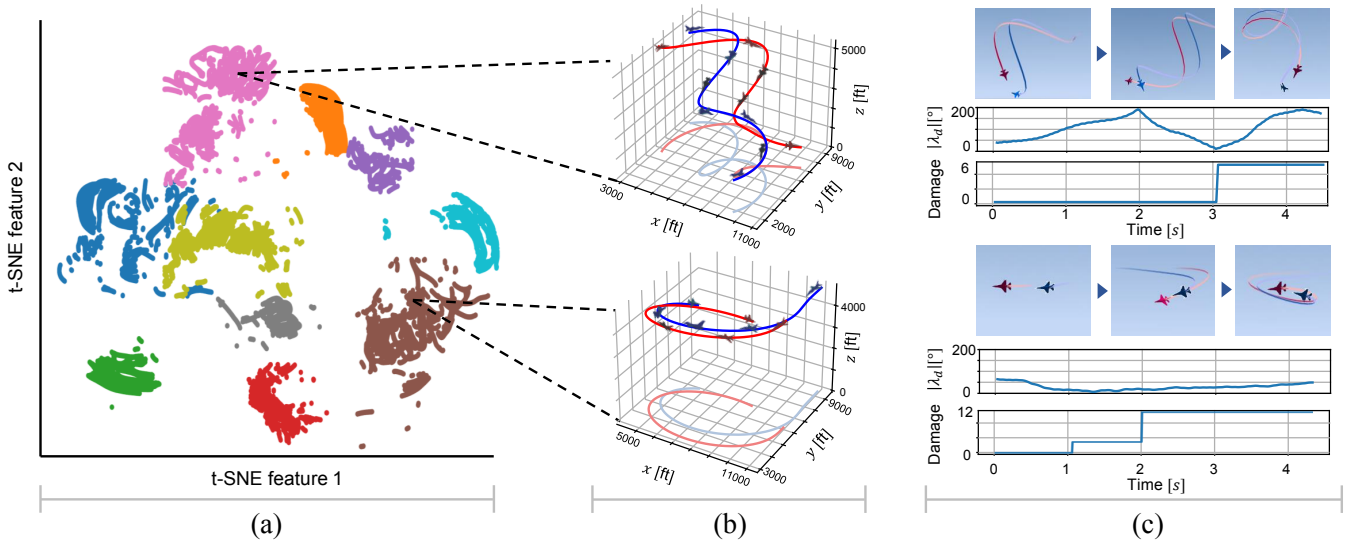


Fig. 8: Learned flight maneuver. (a) t-SNE clustering of state embeddings for the ten most used maneuver keys. Similar maneuvers are clustered together while dissimilar maneuvers are apart. (b) Trajectory graphs of Ego (M-DT, blue) and opponent (HDP, red) while performing the generated maneuvers. (c) Snapshots of generated maneuvers along with graphs of deviation angle and incurred damage during maneuvers.

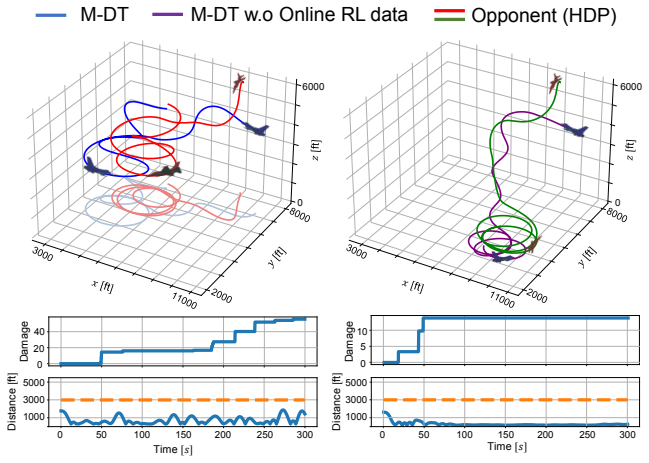
For Prompt-DT and Hyper-DT, we additionally construct the demonstration dataset with the expert policy of [13]. Given $s_{t-L+1:t}$, we sample the most similar trajectory segment from the demonstration dataset using the cosine similarity since it is impossible to manually provide proper demonstrations. The sampled trajectory segments are then used as expert demonstrations for training Prompt-DT and Hyper-DT. We also try using the stochastic sampling method as in [23], but find performance to be worse.

Table III and Fig. 7 present the quantitative results of M-DT and the compared methods. We find that Prompt-DT and Hyper-DT show better performance than DT in in-domain evaluation as in Table III(Scenario 1 to 3). Contrarily, we observe that their conditioning methods using expert demonstrations do not generalize well at Randomized Scenario, and the improvements are marginal as shown Fig. 7 and Table III(Randomized Scenario). Rather than directly using expert demonstrations, M-DT learns maneuver-specific feature prototypes, and deeply interacts with self-attention blocks in DT, achieving consistent improvements across various scenarios. These results validate the discriminative power and generalization ability of M-DT in generating appropriate maneuvers.

D. Maneuver Analysis

1) Contextualized Maneuvers Generated by M-DT: For a deeper understanding of what the M-DT learns, we visually examine how the maneuver pool selects the most appropriate maneuver based on engagement circumstances, as illustrated in Fig. 8. We sample state embeddings from 10 episodes in Randomized Scenario and cluster them into two embedding-space dimensions using t-SNE. Fig. 8(a) demonstrates that state embeddings are properly clustered for corresponding keys and each key is appropriately separated. Fig. 8(b) illustrates the trajectories of the mapped maneuvers. When comparing these maneuvers to the BFM categorized by fighter tactics [2], they resemble Rolling Scissors and Lag Pursuit. We investigated the tactical impact of each maneuver by analyzing the deviation angle and the inflicted damage from the perspective of the blue aircraft (M-DT), as shown in Fig. 8(c). The reduced deviation angle and increased damage prove that each maneuver effectively provides a tactical advantage.

2) M-DT without Online RL data: Since exploration is necessary for online RL training, constructing offline datasets using online RL agents may not be feasible due to poor sample efficiency and safety [26]. To alleviate real-world trial-and-error costs, we train M-DT using limited data sources from human pilots and the HDP model that do not require an online training process. Fig. 9(a) shows that M-DT trained from the entire data sources performs more aggressive and diversified maneuvers. Meanwhile, M-DT trained without online RL data (53.53/61, Table II) obtains better performance compared to HDP (50.85/28, Table III), in



(a) (b)

Fig. 9: Combating trajectories performed by M-DT with different data sources but initialized under the same conditions. (a) M-DT generates diverse maneuvers using offline data supported by Online RL agents, achieving effective damage. (b) M-DT trained with near real-world data sources performs cautious maneuvers while maintaining a low altitude and keeping opponents within short range.

terms of R_{norm} and Win. Fig. 9(b) additionally demonstrates its preference for safe maneuvers, aligning with the typical behavior of humans or HDP in real-world scenarios where riskier actions are generally avoided. These results indicate that M-DT performs well even when trained on limited data sources, suggesting the potential for real-world applications.

VI. CONCLUSION

In this paper, we presented M-DT, a novel offline RL framework for decision-making in air combat. We first constructed a high-fidelity air combat simulation environment and collected extensive datasets relevant to real-world air combat scenarios. M-DT leverages DT architecture to obtain its sequential modeling ability and maneuver pool which records prototypical representations of diverse maneuvers. By conditioning DT on maneuver-specific instruction, M-DT successfully learns complex tactical maneuvers. The experimental results demonstrate that M-DT is effective across various engagement scenarios and outperforms other offline RL baselines. Although we considered various dataset collection procedures, all the data were obtained in the simulation environment which may not capture the full characteristics of real-world environments. In future work, we will consider applying M-DT to real aircraft with more diverse combat scenarios.

REFERENCES

[1] V. Minh, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] R. L. Shaw, "Fighter combat," *Tactics and Maneuvering: Naval Inst. Press, Annapolis, MD, USA*, 1985.

[3] G. H. Burgin and L. Sidor, "Rule-based air combat simulation," *Tech. Rep.*, 1988.

[4] J. Eklund, J. Sprinkle, and S. Sastry, "Implementing and testing a nonlinear model predictive tracking controller for aerial pursuit/evasion games on a fixed wing aircraft," in *Proc. Am. Control. Conf.*, vol. 3, 2005, pp. 1509–1514.

[5] J. S. McGrew, J. P. How, B. Williams, and N. Roy, "Air-combat strategy using approximate dynamic programming," *J. Guid. Control. Dyn.*, vol. 33, no. 5, pp. 1641–1654, 2010.

[6] N. Ernest, K. Cohen, E. Kivelevitch, C. Schumacher, and D. Casbeer, "Genetic fuzzy trees and their application towards autonomous training and control of a squadron of unmanned combat aerial vehicles," *Unmanned Syst.*, vol. 3, no. 3, pp. 185–204, 2015.

[7] H. Shin, J. Lee, H. Kim, and D. H. Shim, "An autonomous aerial combat framework for two-on-two engagements based on basic fighter maneuvers," *Aerosp. Sci. Technol.*, vol. 72, pp. 305–315, 2018.

[8] K. Yang, S. Kim, Y. Lee, C. Jang, and Y. -D. Kim, "Manual-based automated maneuvering decisions for air-to-air combat," *J. Aerosp. Inf. Syst.*, pp. 1–9, 2023.

[9] A. P. Pope, J. S. Ide, D. Mićović, H. Diaz, J. C. Twedt, K. Alcedo, T. T. Walker, D. Rosenbluth, L. Ritholtz, and D. Javorsek, "Hierarchical reinforcement learning for air combat at DARPA's AlphaDogfight Trials," *IEEE Trans. Artif. Intell.*, pp. 1–15, 2022.

[10] J. Chai, W. Chen, Y. Zhu, Z. -X. Yao, and D. Zhao, "A hierarchical deep reinforcement learning framework for 6-DOF UCAV air-to-air combat," *IEEE Trans. Syst. Man. Cybern. Syst.*, vol. 53, no. 9, pp. 5417–5429, 2023.

[11] B. Li, J. Huang, S. Bai, Z. Gan, S. Liang, N. Evgeny, and S. Yao, "Autonnomous air combat decision-making of UAV based on parallel self-play reinforcement learning," *CAAI Trans. Intell. Technol.*, vol. 8, no. 1, pp. 64–81, 2023.

[12] H. Seong and D. H. Shim, "TempFuser: Learning tactical and agile flight maneuvers in aerial dogfights using a long short-term temporal fusion transformer," *arXiv preprint arXiv:2308.03257*, 2023.

[13] J. Bae, H. Jung, S. Kim, S. Kim, and Y. -D. Kim, "Deep reinforcement learning-based air-to-air combat maneuver generation in a realistic environment," *IEEE ACCESS*, vol. 11, no. 1, pp. 26427–26440, 2023.

[14] Q. Yang, J. Zhang, G. Shi, J. Hu, and Y. Wu, "Maneuver decision of UAV in short-range air combat based on deep reinforcement learning," *IEEE ACCESS*, vol. 8, pp. 363–378, 2020.

[15] R. F. Prudencio, M. R. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *IEEE Trans. Neural. Netw. Learn. Syst.*, 2023.

[16] A. Nair, A. Gupta, M. Dalal, and S. Levine, "AWAC: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[17] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision Transformer: Reinforcement learning via sequence learning," *Adv. Neural. Inform. Process. Syst.*, vol. 34, pp. 15084–15097, 2021.

[18] K. -H. Lee, O. Nachum, M. Yang, L. Lee, D. Freeman, W. Xu, S. Guadarama, I. Fischer, E. Jang, H. Michalewski and I. Mordatch, "Multi-game decision transformers," *Adv. Neural. Inform. Process. Syst.*, vol. 35, pp. 27921–27936, 2022.

[19] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[20] J. Berndt, "JSBSim: An open source flight dynamics model in C++," in *Proc. AIAA Model. Simul. Technol. Conf. Exhib.*, 2004, p. 4923.

[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[22] M. Monastirsky, O. Azulay, and A. Sintov, "Learning to throw with a handful of samples using decision transformers," *IEEE Robot. Autom. Letters*, vol. 8, no. 2, pp. 576–583, 2022.

[23] M. Xu, Y. Shen, S. Zhang, Y. Lu, D. Zhao, J. Tenenbaum, and C. Gan, "Prompting decision transformer for few-shot policy generalization," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 24631–24645.

[24] M. Xu, Y. Lu, Y. Shen, S. Zhang, D. Zhao, and C. Gan, "Hyper-decision transformer for efficient online policy adaptation," in *Proc. Int. Conf. Learn. Represent.*, 2022.

[25] H. Liu, D. Tam, M. Muqeeth, J. Motha, T. Huang, M. Bansal, and C. Raffel, "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," *Adv. Neural. Inform. Process. Syst.*, vol. 35, pp. 1950–1965, 2022.

[26] G. Zhou, L. Ke, S. Srinivasa, A. Gupta, A. Rajeswaran, and V. Kumar, "Real world offline reinforcement learning with realistic data source," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023. pp. 7176–7183.