Postprint: TOC Reward Function for Multi-Target Trajectory Recovery via Deep Reinforcement Learning

Authors: He Liang, Xu Zhengguo, Jia Yu, Shen Chao, Li Yun

Date: 2019-05-10T00:00:00+00:00

Abstract

This study addresses the problem in trajectory detection where target geographical positions obtained by detectors typically constitute indistinguishable multi-target scenarios within the same frame, necessitating the reconstruction of individual trajectories and target discrimination from positional information. A method employing deep reinforcement learning for target trajectory restoration is proposed. Based on the physical characteristics of target trajectories, a mathematical model is extracted, and a Trajectory Osculating Circle (TOC) reward function is introduced by incorporating trajectory direction, curvature, and other attributes, enabling deep reinforcement learning to effectively restore multi-target trajectories and distinguish individual targets. First, the multi-target trajectory restoration problem is described and modeled into a framework processable by deep reinforcement learning; the TOC reward function is then applied to conduct experiments on this problem; finally, the mathematical derivation and physical interpretation of the reward function are presented. Experimental results demonstrate that the deep reinforcement network driven by the TOC reward function can effectively restore target trajectories, aligning with actual target trajectories in both heading and speed.

Full Text

Preamble

Vol. 37 No. 6 Application Research of Computers ChinaXiv Cooperative Journal Accepted Paper

TOC Reward Function for Multi-Target Trajectory Recovery via Deep Reinforcement Learning

He Liang[1], Xu Zhengguo[1], Jia Yu[1], Shen Chao[2], Li Yun[1] (1. National Key Laboratory of Science & Technology on Blind Signal Processing, Chengdu 610041, China; 2. MOE Key Laboratory for Intelligent Networks & Network Security, Xi' an Jiaotong University, Xi' an 710049, China)

Abstract: In trajectory detection, detectors typically receive multiple target geographical locations within a single frame that cannot be distinguished from one another, necessitating the reconstruction of individual trajectories and target identification. This paper proposes a deep reinforcement learning approach for trajectory recovery. Based on the physical characteristics of target trajectories, we extract a mathematical model and introduce a Trajectory Osculating Circle (TOC) reward function that incorporates trajectory direction and curvature, enabling deep reinforcement learning to effectively recover multi-target trajectories and distinguish between targets. We first formulate the multi-target trajectory recovery problem and model it for deep reinforcement learning processing. We then conduct experiments on multi-target trajectory recovery using the TOC reward function. Finally, we provide mathematical derivation and physical interpretation of this reward function. Experimental results demonstrate that the deep reinforcement learning network driven by the TOC reward function can effectively reconstruct target trajectories that align well with actual trajectories in terms of heading and speed.

Keywords: deep reinforcement learning; sequential decision; Q-function; trajectory osculating circle

0 Introduction

Reinforcement learning has demonstrated outstanding performance in decision-making problems and attracted considerable attention. The theoretical foundation of reinforcement learning was proposed in [**?**], which simulates how organisms automatically adjust their actions based on environmental influences to optimally adapt. Mnih et al. [**?**] successfully learned control policies from high-dimensional input using deep learning models via reinforcement learning, achieving success in Atari games. To simulate environmental adaptability, two key problems must be addressed: (a) extracting effective environmental information from high-dimensional inputs, such as identifying attack targets from visual images in games; and (b) making effective decisions based on environmental information to alter the environment toward favorable states and final outcomes [**?**, **?**]. Using Atari' s pixel information and game scores as input, reinforcement learning-trained decision-making can match the skills of professional gamers. This demonstrates that for dynamic decision-making problems, optimal final results do not necessarily require choosing the locally optimal action at each step; instead, reinforcement learning can gradually learn decision-making methods that optimize final outcomes through the learning process.

Before deep learning, reinforcement learning suffered from poor stability and

was limited to low-dimensional input problems due to constraints in memory complexity, computational complexity, machine learning algorithms, and sampling complexity [**?**, **?**]. Deep neural networks have more hidden layers and hyperparameters than traditional neural networks [**?**], with activation functions evolving from Sigmoid to ReLU [**?**]. The strong function approximation and learning capabilities of deep neural networks provide powerful tools for reinforcement learning to solve high-dimensional complex problems. Reinforcement learning combined with deep learning is known as deep reinforcement learning (DRL) [**?**]. However, simple combination does not guarantee learning stability, prompting a series of studies focused on stabilization. Current main approaches include: (a) introducing replay mechanisms where agents periodically revisit previously experienced games to strengthen learning of strategies for similar environmental problems [**?**, **?**], though this consumes significant memory for storing historical game information; and (b) utilizing multiple parallel agents with decoupled data to ensure environmental information stability for each agent [**?**], while also leveraging GPU acceleration [**?**, **?**] or adopting distributed deep reinforcement learning architectures [**?**].

Deep reinforcement learning can handle more complex problems beyond the reach of traditional reinforcement learning, with stability substantially improved through increased memory space and parallel technology development. Using deep neural networks to process high-dimensional visual information from Atari games enables reinforcement learning to more stably address decision-making problems in these games. AlphaGo' s success in defeating human players also utilized deep reinforcement learning combined with classical search algorithms [**?**]. Many other complex and difficult-to-operate games can be completed via deep reinforcement learning, such as FlappyBird [**?**, **?**] and TORCS [**?**]. These experiments were conducted on the OpenAI Gym platform [**?**].

Target trajectory clustering problems [**?**] primarily aim to mine navigation trajectories of various targets from massive trajectory data and cluster them based on target attribute information, generally using target speed and distance information to comprehensively determine the clustering results of points belonging to each target. This problem can typically be addressed using clustering methods like K-means on spatial geographic location and identity information [**?**]. However, traditional clustering methods like K-means focus primarily on distance information during clustering [**?**], requiring more complex distance metric functions to introduce other attributes or mapping input data to corresponding high-dimensional spaces via kernel functions, though kernel function design lacks intuitiveness. This paper uses reward function design in deep reinforcement learning to intuitively introduce mathematical characteristics of geometric properties of trajectories, resulting in a more intuitive and effective reward function that avoids the complex and non-intuitive design work of kernel functions in clustering algorithms.

Deep reinforcement learning enables computers to effectively handle complex decision-making games. Beyond gaming, trajectory recovery problems can also

be solved via deep reinforcement learning. The trajectory recovery problem addressed in this paper is as follows: detectors obtain geographical location information of multiple targets at a certain frequency but cannot distinguish between them. The goal is to draw trajectories for each target. Due to detector limitations, missed detections and false alarms exist. This problem can be characterized as a dynamic programming problem requiring decisions on how to connect points at each moment. This paper employs deep reinforcement learning to solve it.

---

1 Sequential Decision Problems and Reinforcement Learning

Reinforcement learning is a learning process that accumulates experience through repeated environmental experiments to achieve intelligent decision-making. Reinforcement learning involves an agent, environment, and reward, with interactions between agent and environment completed through reward mechanisms. At each time step, the environment is in some state. The agent observes the environmental state and takes an action based on a certain policy. This action affects the environment, causing it to transition to the next state. The agent then obtains the new environmental state and a reward feedback for its action from the changed environment. The selection of different policies for environmental states at different times constitutes a typical sequential decision problem. Mathematical descriptions between policies and environments require state value functions and state-action value functions, while exploration and exploitation are necessary during decision-making to obtain global optimal solutions.

1.1 Sequential Decision-Making

Sequential decision-making refers to decisions made at each time point in chronological order based on environmental state conditions. Policies are formed from decisions at each stage. During reinforcement learning, actions beneficial to achieving goals are retained while detrimental actions are discarded based on interactive data of rewarded environmental actions at each stage.

1.2 State Value Function

A policy is the basis for action selection. After adopting a certain policy, the expected reward obtained is the state value function. The reinforcement learning process requires constructing an indicator function for each policy based on previous experimental results, thereby gradually converging to the optimal policy according to this indicator. For example, the greedy policy selects the policy that yields the maximum current state value at each decision step, though this approach may not necessarily result in a globally optimal policy.

1.3 State-Action Value Function

After taking a certain action, the corresponding reward value is the state-action value function. This action is derived from the policy and determined by the reward feedback from the environment under different actions in the current state. Different policies lead to different actions in the same state. During optimization, both state value functions and state-action value functions are needed to guide agent learning toward better and ultimately optimal performance.

1.4 Exploration and Exploitation

Exploitation involves searching for optimal values within a specified region, but this optimum may not be globally optimal. Therefore, exploration must be introduced to develop unknown regions in hopes of finding the global optimum. [Figure 1: see original paper] shows a function where we expect to find its global minimum point within an interval. Assuming current exploitation is limited to finding the minimum value at a certain point, the exploration process can expand the search interval with a certain probability, thereby providing opportunities to discover the global optimum.

Exploitation is the process of finding local optima within intervals, while exploration provides opportunities to escape local optima and find global optima. Practical applications demonstrate that combining exploration and exploitation can effectively locate global optima.

---

2 Multi-Target Trajectory Recovery Problem Modeling

[Figure 2: see original paper] illustrates the general reinforcement learning framework. The agent needs to conduct a series of interactions with the environment. At each moment, the environment is in some state. The agent selects a certain action based on the current environmental state combined with its historical behavioral criteria and policy. This action correspondingly affects the environmental state, and the environment responds to the agent' s action by providing the next state and a reward for the current action. This process repeats until the environment no longer responds or changes, i.e., the task is completed.

Upon task completion, the agent further obtains a total score from the final outcome. This total score differs from the reward value obtained after each action, representing the overall evaluation based on task completion. This score corresponds to completing one task and is therefore called the episodic score. The agent' s task is to repeatedly complete multiple episodes to identify operations that yield high episodic scores, hoping to achieve the highest possible score.

Reinforcement learning obtains strategies for responding to various environmental states and collects corresponding rewards through continuous trial and error, ultimately achieving optimal performance.

2.1 Multi-Target Trajectory Recovery Problem and Modeling Condition Analysis

Detectors obtain geographical locations of several target points at a certain frequency. However, due to detector limitations, they cannot determine which target each detected point corresponds to, while missed detections and false alarms also exist. For example, detectors might capture information as shown in [Figure 3: see original paper].

At each time moment, the detector locates target positions but cannot distinguish between targets from the detection results. For instance, the five detection moments in [Figure 3: see original paper] show results where only four targets were detected at one moment and six targets at another, corresponding to false alarms and missed detections respectively. The multi-target trajectory detection problem involves integrating these detection results to obtain each target' s trajectory, as shown in [Figure 4: see original paper].

In [Figure 4: see original paper], circled numbers represent various targets and their corresponding trajectories. Arrows indicate reconstructed trajectory directions, solid circles at arrow tails denote target starting positions, hollow circles indicate target termination positions, dashed hollow circles along arrow lines represent positions where targets were detected during their path, gray circles indicate detector false alarms, and gray dashed circles indicate missed detections. The multi-target trajectory recovery problem aims to reasonably judge the overall trend of target trajectories, eliminate unreasonable false alarm points, add reasonable missed detection points, and thereby reconstruct the true target trajectories.

In practical scenarios, detector limitations mean only geographical location information of multiple targets at each time point can be obtained, i.e., the detector cannot provide which specific target each point in each frame corresponds to. However, the targets of interest exhibit certain movement patterns rather than purely random spatial motion. Target speeds remain relatively stable without sudden fluctuations, and targets generally move along pre-planned trajectories with minor adjustments rather than sharp turns. For the multi-target trajectory recovery problem, the following assumptions are considered during modeling:

a) Regarding detector limitations, two main assumptions exist: (a) At each detection time point (i.e., each frame), only geographical location information of multiple targets is detected without inter-target distinguishability; (b) Detected targets in each frame may contain false alarms or missed detections, i.e., noise points exist within frames.

b) Regarding target movement regularity and rationality, three assumptions are considered: (a) Target speed remains relatively stable without sudden acceleration or deceleration in short periods; (b) Target trajectories are relatively smooth without sharp turns; (c) Target trajectories exhibit periodic regular motion within a certain range, i.e., targets perform circuit

tasks within specific regions.

Actual targets may not fully comply with these assumptions. On one hand, some target speeds fluctuate. By adjusting the regularization coefficient of the corresponding trajectory distance stability term in the subsequent reward function design, the method can adapt to certain speed fluctuations. However, if speed fluctuations are too large, errors in judging target distances based solely on geographical location information will increase accordingly. On the other hand, if too many false alarm noise points exist in each frame of detection results, they may be incorrectly considered as trajectories of other non-existent targets. This performance can be controlled by reducing the regularization coefficient of the target count term.

Further analysis of the speed stability assumption reveals that target speeds do change in practice. In the scenarios considered in this paper, targets move according to pre-planned trajectories without sharp acceleration or deceleration. Therefore, this assumption corresponds to relatively stable target speeds, allowing for minor changes but not drastic variations. The reward function term designed for this assumption encourages reconstructed trajectories to maintain speed stability, preventing target points with large distances between adjacent frames from being assigned to the same target. Under this assumption, the designed reward function allows target speeds to vary but minimizes drastic changes, which aligns with actual target movement patterns. The goal is to reconstruct target trajectories with speed variations within reasonable ranges rather than dramatic fluctuations.

2.2 Environmental State

The currently detected target positions are regarded as the current system state, denoted as s_t at time t, with all states constituting the state space S.

As shown in [Figure 5: see original paper], each point s_i represents the spatial state of the system at different times. However, due to timing errors and limitations in actual received data, the temporal sequence is unknown and requires reasonable inference to select an appropriate trajectory point order.

2.3 Action

When the system is in environmental state s_t, the action space at time t is determined by the previous $t-1$ states, i.e., previously selected states no longer participate in action selection. Therefore, the action a_t at time t serves as the trajectory point at time t in [Figure 6: see original paper], i.e., $a_t \in \{p_i | i = 1, 2, ...\}$, meaning subsequent candidate points can only be selected from remaining undesignated detection points.

At time t, assuming trajectory points have been selected for times t_1 and t_2, i.e., $s_{t_1} = p_1$ and $s_{t_2} = p_2$, the action set becomes $\{p_3, p_4, p_5\}$.

2.4 TOC Reward Function

Actual targets rarely change speed abruptly, while detection points for target localization are obtained under fixed sampling rates. Therefore, under reasonable trajectory conditions, distances between adjacent points should be similar. Additionally, considering that target motion may not follow straight lines, direct connections yield shorter distances than actual curved paths (the straight line segment is the shortest connection between two points in a plane). Accordingly, we define the reward obtained by taking action a_t when the system is in state s_t at time t as:

$$\text{reward}_t(s_t, a_t) = -\text{Var}(\{d(s_i, s_{i+1}) | i = 1, 2, \dots, t-1\} \cup \{d(s_t, a_t)\})$$

where Var denotes sample variance and $d(\cdot, \cdot)$ represents the distance between two points, also simply denoted as $d_{i,i+1}$. This term is called the trajectory distance stability term.

Furthermore, if targets exhibit phased regular patterns—for example, moving along certain curved trajectories for some time before switching to another regular curve—the reward may not need to consider variance across all previous states but only within certain time windows. Thus, we define the reward function considering the previous n states as:

$$\text{reward}_t(s_t, a_t) = -\text{Var}(\{d(s_i, s_{i+1}) | i = t-n, \dots, t-1\} \cup \{d(s_t, a_t)\})$$

When $n = t$, this corresponds to considering all states.

Additionally, since targets rarely exhibit sharp turns, selecting actions that result in appropriate distances but sharp turns should receive negative rewards. We characterize this using curvature, preferring trajectories with minimal curvature. The corresponding reward function adds a curvature regularization term:

$$\text{curv}_t(s_{t-2}, s_{t-1}, s_t, a_t) = \frac{4}{d(s_{t-2}, s_{t-1}) + d(s_{t-1}, s_t) + d(s_t, a_t)}$$

This expression estimates target trajectory curvature using three adjacent sampling points. It can be proven that when sampling intervals are sufficiently small and three sampling points are sufficiently close, this estimate approaches the true curvature of the target' s motion trajectory, consistent with the definition of the osculating circle in particle kinematics. The solution and explanation of this expression are provided in the appendix.

Finally, considering that targets often move along relatively regular curves, the curvature change rate should not be excessive. The incentive function requires an additional regularization term using variance of historical curvature. Similar to the trajectory distance stability term, considering variance of curvature over the previous n states yields the curvature stability term:

$$\text{curvstable}_t = \text{Var}(\{\text{curv}_i | i = t-n, \ldots, t\})$$

Combining these factors based on actual target trajectory characteristics, the reward function corresponding to taking action a_t when the system is in state s_t at time t is:

$$r_t(s_t, a_t) = \text{reward}_t(s_t, a_t) - \lambda_1 \cdot \text{curv}_t - \lambda_2 \cdot \text{curvstable}_t$$

where n represents the state order and λ_1, λ_2 are regularization coefficients. When considering multiple targets, we further introduce the number of targets as an additional regularization term to correct the reward function. Adding a target count term to the above expression yields the Trajectory Osculating Circle (TOC) reward function proposed in this paper.

2.5 Q-Function

The reward function only provides the reward feedback from the environment after the agent takes the current action. What the agent needs to learn is not maximizing the immediate environmental reward but achieving optimal final results through sequential decision-making. Therefore, the agent' s strategy is to select actions that maximize future rewards given a specific state s_t. The future reward under specific state s_t and action a_t is called the Q-function, expressed as $Q(s_t, a_t)$. The agent' s task at time t is to find the action a_t that maximizes $Q(s_t, a_t)$.

However, in practical problems, the Q-function is difficult to express explicitly, and identical Q-functions may not exist for every state-action pair. Therefore, iterative methods are needed to gradually update the Q-function:

$$Q_{k+1}(s_t, a_t) = Q_k(s_t, a_t) + \alpha[r_t + \gamma \max_{a'} Q_k(s_{t+1}, a') - Q_k(s_t, a_t)]$$

where α represents the learning rate and γ represents the discount factor. In deep reinforcement learning, this Q-function is represented by a deep learning network called a Deep Q-Network (DQN) [**?**], with system state s_t as input and Q-values for each action as output. Additionally, deep reinforcement learning defines a memory buffer that stores information about the current state, reward, action, next transitioned state, and whether the state is terminal at specific moments. Fixed-size memory segments are periodically and randomly selected from this buffer for batch training of the Q-function' s deep neural network.

3 Algorithm Flow

The overall algorithm flow is shown in [Figure 7: see original paper], primarily comprising three processes: environment recognition, DQN, and action execution. First, a Convolutional Neural Network (CNN) performs state recognition and corresponding reward function calculation for the environment. The recognized state is fed into a DQN trained on Q-values calculated via the reward function to obtain Q-values for different actions in the current state, and the action maximizing the Q-value is executed in the environment.

3.1 Environment Recognition

For deep reinforcement learning inputs, this problem primarily involves target conditions after obtaining geographical location information. Considering that an image should possess translation and rotation invariance—meaning the selection of coordinate origin and orthogonal directions in the target plane should not affect trajectory discrimination—we adopt CNN for recognition. Environment recognition uses CNN, placing target positions at each sampling moment into a grid network of specified dimensions, using integer coordinates to represent each sampling point's location. The presence of target points in grids along with each target' s speed and bearing information serves as CNN input, while the target' s state is used as sample labels for CNN training. The resulting network can serve as the agent' s visual module, providing the agent with current environmental state information s_t and corresponding reward r_t after taking certain actions. On the other hand, since the functional form of the environment' s reward was designed in the previous section, we can directly construct the corresponding neural network for this function without training a reward-generating network from the environment.

The CNN identifies target positions in the current frame' s image data. Unlike complex object detection problems where targets have intricate geometric structures, each frame' s data consists of detected data points. Therefore, we only need to grid the input frame and output whether target points exist in each grid. The CNN input is the gridded frame image with outputs equal to the number of grids, each indicating target presence in the corresponding grid.

3.2 DQN

The environmental state includes position information connected to the current target trajectory, as well as speed and bearing information, forming a multi-dimensional vector $s_t = [s_{t1}, s_{t2}, \dots, s_{tk}]^T$. Thus, the system state is k-dimensional, serving as DQN input with corresponding k-dimensional input layer. The DQN' s purpose is to provide Q-function values for different actions based on the environmental state. Therefore, we need to recursively calculate Q-function values for each action under each state through rewards to train the DQN, and access these data via a sample replay buffer for DQN training when needed. The DQN network outputs Q-function values for each action in the

current state, i.e., $Q(s_t) = \{Q(s_t, a_1), Q(s_t, a_2), \ldots, Q(s_t, a_l)\}$. Here, the action set corresponds to the number of obtained target trajectory points l, making the DQN output dimension l.

3.3 Action Execution

After calculating Q-function values for each action based on the current environmental state, we must decide which action to execute in the environment. The Q-function value represents an estimate of the current action considering multi-step reward feedback, so we simply select the action corresponding to the maximum Q-value to apply to the environment. The executed action is $a_t = \arg\max_a Q(s_t, a)$. After the environment changes accordingly, the environment recognition CNN further learns the changed environmental state and corresponding reward, repeating this cycle until all target point trajectory information is identified.

---

4 Experimental Analysis

We conducted experiments on the OpenAI Gym platform, which conveniently enables deep reinforcement learning environment construction and DQN implementation. To demonstrate the effectiveness of the proposed TOC reward function, we experimentally evaluated the impact of each term in the TOC reward function on learning results using simulated trajectory data, and presented trajectory recovery results on real datasets.

4.1 Experimental Setup

When constructing the environment with OpenAI Gym, the environmental state consists of target geographical locations detected by the detector, with actions also being various positions. During reward function calculation, if the selected action is already a trajectory point, the policy reward is 0; otherwise, the reward value is calculated using the TOC reward function combined with points already assigned to the trajectory.

The TOC reward function contains three regularization terms corresponding to curvature magnitude, curvature stability, and target count. To demonstrate the TOC function' s impact on deep reinforcement learning for multi-target trajectory mining, we also treat the trajectory distance stability term as a regularization term in experiments. When adjusting one regularization coefficient, all others are set to 1. The true target trajectories used in experiments are shown in [Figure 8: see original paper], with latitude and longitude as horizontal and vertical coordinates respectively.

[Figure 8: see original paper] shows four targets distinguished by four colors and shapes, where targets 1 and 2 perform linear motion while targets 3 and

4 perform circular motion. Each point on the trajectories represents the accurate geographical location at different times. In reality, detector measurements contain loss, false alarms, and missed detections. Here, random noise is added as Gaussian noise with mean 0 and variance 0.01, while false alarm and missed detection rates are both set at 5%, as shown in [Figure 9: see original paper].

The trajectories in [Figure 9: see original paper] represent simulated data after noise addition, with missed detection points marked by crosses and false alarm points marked by X symbols. Since missed detection points do not exist in experimental data, they can be ignored. For completeness in trajectory recovery experiments, we first consider both false alarm and missed detection points, with results presented below.

4.2 Trajectory Distance Stability Term

[Figure 10: see original paper] shows recovery effects with regularization coefficients of 0.1 and 10 for the trajectory distance stability term. As the proportion of the trajectory distance stability term in the reward function increases, the reconstructed target trajectories increasingly tend toward equal distances between adjacent detection points for each target, potentially causing errors in target count estimation when this regularization coefficient becomes too large.

4.3 Curvature Term

[Figure 11: see original paper] shows recovery effects with and without the curvature term, and with increasing curvature regularization coefficients. The curvature term describes trajectory curvature, as normal targets rarely exhibit sharp turns. Larger regularization coefficients indicate stronger control toward straight-line trajectories. Results show that as the curvature term increases, target trajectories become progressively straighter, often sacrificing accurate target count estimation. In the right figure, most target trajectories are reconstructed as straight lines.

Testing the proposed TOC reward function with comprehensive consideration of trajectory distance stability, curvature, curvature stability, and target count terms yields deep reinforcement learning results shown in [Figure 14: see original paper].

4.4 Curvature Stability Term

[Figure 12: see original paper] shows recovery effects with and without the curvature stability term, and with increasing curvature stability regularization coefficients. The curvature stability term describes phenomena when targets perform circular motion—though curvature exists everywhere, it remains constant for circular motion. This provides good descriptive capability for regularly moving targets. Experimental results show that as this coefficient increases, target trajectories are increasingly identified as circular motion (since circular curvature is stable), often sacrificing accurate target count estimation.

4.5 Target Count Term

[Figure 13: see original paper] shows recovery effects with and without the target count term, and with increasing target count regularization coefficients. Target count can be provided when known or automatically learned via deep reinforcement learning, though this term' s regularization coefficient requires careful tuning. If the target count is known, this term should be avoided. Results show that as this coefficient increases, target trajectories tend to be connected as a single target' s trajectory, i.e., the method expects fewer targets and mixes all targets into one trajectory.

4.6 Comparative Analysis with Existing Trajectory Clustering Algorithms

This section uses real target geographical location datasets to compare the proposed TOC reward function method with trajectory clustering algorithms. Trajectory clustering methods using spatial information rely primarily on spatial distance for clustering, exhibiting obvious spatial limitations—mainly considering trajectory points within certain Euclidean distances. In contrast, the deep reinforcement learning network based on the TOC reward function comprehensively considers multiple indicators including trajectory distance, curvature, and target count.

Furthermore, we compare target count determination performance with existing trajectory clustering algorithms. Automatic target count selection in trajectory clustering algorithms can be achieved by calculating clustering results under different target counts, as shown in [Figure 16: see original paper]. The figure shows that as target count increases, each target point is clustered into more categories, reducing clustering error within each category. Generally, the inflection point is selected as the ideal category number, yielding a target count of 2 from this algorithm. In contrast, the TOC reward function with deep reinforcement learning accurately reconstructs 3 targets matching the actual count.

---

5 Conclusion

In practical applications, detectors often can only obtain target geographical locations, requiring target distinction and trajectory reconstruction. This paper constructs a mathematical model based on the physical meaning of target trajectories and proposes the TOC reward function with mathematical proof. Through comparative experiments on simulated data evaluating each TOC reward function component and real-data experiments, we demonstrate the TOC reward function' s effectiveness in measuring trajectory distance stability, curvature, curvature stability, and target count. Experiments show that adjusting coefficients for trajectory distance stability, curvature, curvature stability, and target count terms can effectively control deep reinforcement learning performance to reconstruct trajectories matching actual target behavior.

Remaining issues include: (a) The TOC function exhibits training instability when guiding deep reinforcement learning for trajectory recovery, sometimes failing to converge to ideal results; (b) Additional physical information about actual target trajectories should be considered to add more regularization terms to the TOC function for identifying the most realistic trajectory among multiple reasonable reconstructions. Due to detector limitations, this paper focuses on scenarios where only geographical location information is known, introducing other prior information such as speed and trajectory features for recovery. The proposed TOC reward function primarily targets spatial geographical information. Incorporating more physical information would yield more accurate and reasonable trajectory reconstruction.

Currently, additional physical information that could be considered includes: (a) Heading information, which depends on detection methods and is obtainable through many detection approaches, allowing comprehensive consideration and design of heading parameter regularization terms to be added to the TOC reward function; (b) Target type information, which requires other detection methods. If target types are known, they can further distinguish targets within the same frame, enabling reasonable association of known-type target trajectories and improving reconstruction accuracy. In practical problems where physical information such as target type can accurately distinguish targets, this information can be used to pre-identify distinguishable targets, after which the proposed method can process remaining indistinguishable targets to achieve better trajectory reconstruction results.

References

[1] Sutton R S, Barto A G. Reinforcement learning: an introduction [J]. IEEE Trans on Neural Networks, 1998, 9(5): 1054.

[2] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning [EB/OL]. (2013-01-01). https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf.

[3] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518 (7540): 529-533.

[4] Duan Yan, Chen Xi, Houthooft R, et al. Benchmarking deep reinforcement learning continuous control [C]//Proc of International Conference on Machine Learning. 2016: 1329-1338.

[5] Kai A, Deisenroth M P, Brundage M, et al. A Brief Survey of Deep Reinforcement Learning [J]. IEEE Signal Processing Magazine, 2017, 8 (6).

[6] Strehl A L, Li L, Wiewiora E, et al. PAC model-free reinforcement learning [C]//Proc of International Conference on Machine Learning. New York: ACM Press, 2006: 881-888.

[7] Lecun Y, Bengio Y, Hinton G. Deep learning [J]. Nature, 2015, 521 (7553): 436.

[8] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks [C]//Proc of International Conference on Artificial Intelligence and Statistics. 2011: 315-323.

[9] Li Yuxi. Deep reinforcement learning: an overview [EB/OL]. 2017. https://arxiv.org/abs/1701.07274

[10] Lampe T, Riedmiller M. Approximate model-assisted Neural Fitted Q-Iteration [C]//Proc of International Joint Conference on Neural Networks. Piscataway, NJ: IEEE Press, 2014.

[11] Schulman J, Levine S, Moritz P, et al. Trust region policy optimization [J]. Computer Science, 2015: 1889-1897.

[12] Hasselt H V, Guez A, Silver D. Deep reinforcement learning with double Q-learning [EB/OL]. 2015. https://arxiv.org/pdf/1509.06461.pdf

[13] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning [C]// Proc of the 33rd International Conference on Machine Learning. 2016: 359-365.

[14] Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay [EB/OL]. 2015. https://arxiv.org/abs/1511.05952.

[15] Babaeizadeh M, Frosio I, Tyree S, et al. Reinforcement learning through asynchronous advantage actor-critic on a GPU [EB/OL]. https://openreview.net/pdf?id=r1VGvBcxl.

[16] Nair A, Srinivasan P, Blackwell S, et al. Massively PArallel methods for deep reinforcement learning [EB/OL]. 2015. https://arxiv.org/pdf/1507.04296.pdf.

[17] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search [J]. Nature, 2016, 529(7587): 484-489.

[18] Pilcer L S, Hoorelbeke A, Andigne A D. Playing flappy bird with deep reinforcement learning [C], IEEE Trans on Neural Networks, 2015, 16 (1): 285-286.

[19] Qi Hang, Gong Jiang, Xu Lunbo. 3D flappy bird with reinforcement learning, 2016.

[20] Sallab A, Abdou M, Perot E, et al. Deep reinforcement learning framework for autonomous driving [J]. Electronic Imaging, 2017, 2017 (19): 70-76.

[21] Brockman G, Cheung V, Pettersson L, et al. OpenAI Gym [EB/OL]. https://gym.openai.com/.

[22] Wang Zengfu, Pan Quan, Lang Lin, et al. Dynamic track cluster algorithm based on subtractive clustering [J]. Journal of System Simulation, 2009, 21(16): 5240-5243, 5246.

[23] Chen Yong. A data mining method for clustering target tracks [J]. Radio Engineering, 2015, 45(3): 22-24.

[24] Xing Yanni, Qian Yurong, Nan Fangzhe, et al. Survey of optimization on K-means algorithm in Spark [J]. Application Research of Computers, 2020, 37(3). http://www.arocmag.com/article/02-2020-03-001.html.

Appendix: Derivation of Curvature Stability Term in TOC Reward Function

For a smooth curve, curvature is defined as $K = \frac{d\phi}{ds}$, where ds is the arc length differential and $d\phi$ is the tangential angle differential. When the curve is expressed in Cartesian coordinates as $y = y(x)$, curvature can be expressed as $K = \frac{y''}{(1+y'^2)^{3/2}}$, yielding the curvature radius $R = \frac{1}{K}$.

In our problem, we can only obtain discrete points on the curve at relatively large intervals. We can approximate curvature using the circumradius of three points on the curve. We first present the method for calculating the circumradius given three points, then clarify its relationship with the curvature radius in the limit.

As shown in [Figure 17: see original paper], for three given points, assume point A is the previously taken strategy s_{t-2}, point B is the current system state s_{t-1}, and point C is the taken action a_t. The corresponding circumradius R satisfies the law of sines:

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} = 2R$$

where $a = d(s_{t-2}, s_{t-1})$, $b = d(s_{t-1}, a_t)$, and $c = d(s_{t-2}, a_t)$. The circumradius formula is:

$$R = \frac{abc}{\sqrt{(a+b+c)(-a+b+c)(a-b+c)(a+b-c)}}$$

In the limit state where $\Delta_1, \Delta_2 \to 0$, this approximation converges to the true curvature radius. This conclusion aligns with particle kinematics, where the osculating circle of a particle' s trajectory is defined as the limit of the circumcircle.

Note: Figure translations are in progress. See original paper for figures.