ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 19, April 2017

# Reinforcement Learning Applied in Third Person Shooter Games

T.Sarada Kiranmayee[1], Akhil Mantha[2], Manideep Kumara Reddy[3], Karan Khosla[4]
[1] *Asst. Professor in CSE Department, SRM University*
[2]*UG Scholar, CSE Department, SRM University*
[3]*UG Scholar, CSE Department, SRM University*
[4]*UG Scholar, CSE Department, SRM University*
[1]tskiranmayee@gmail.com
[2]akhilmantha@live.com
[3]manideepreddy2007@gmail.com
[4]karankhosla99@gmail.com

***Abstract*—Reinforcement learning (RL) is a popular machine learning technique that has many successes in learning how to play classic style games. Applying RL to third person shooter (TPS) games is an interesting area of research as it has the potential to create diverse behaviors without the need to implicitly code them. This paper investigates the tabular Sarsa RL algorithm applied to a purpose built TPS game. The first part of the research investigates using RL to learn bot controllers for the tasks of navigation, item collection, combat individually and vehicle maneuvering. Results showed that the RL algorithm was able to learn a satisfactory strategy for navigation control, but not to the quality of the industry standard path finding algorithm. The combat controller performed well against a rule-based bot, indicating promising preliminary results for using RL in TPS games. The second part of the research used pertained RL controllers and then combined them by a number of different methods to create a more generalized bot artificial intelligence (AI). The experimental results indicated that RL could be used in a generalized way to control a combination of tasks in TPS bots such as navigation, item collection, and combat.**

## I. INTRODUCTION

Over the past few decades reinforcement learning has evolved a lot and has been applied to lot of multiagent systems and robotics. Many researchers have applied RL to games and have created game bots. RL has been applied to a lot of classic games like Backgammon, PAC-MAN. However there has been little research done in applying RL to modern video games where the complexity is more.

Modern video games are becoming more complex and particularly TPS. TPS games are becoming more complex and also have a complex environment. This paper will show that the bot can know an environment and the objects present in it.

TPS bot AI generally consists of path finding, driving, picking up and using objects in the environment and using different weapons like sniper, handgun and perform combat. Bot AI in most of the games uses rule-based systems, state machines and goal-based machines. In TPS the success of the bot will be depended on their ability to learn new skills and adapt to their dynamic and complex environments. In TPS to bridge this gap we are using policy search method instead of value function approximation.

One of the main disadvantages of RL is that with increased complexity of the problem, it results in the scalability issues.

To overcome this, the task is split into two. The first task is that the bot studies its environment and figures the path and the second task is to perform combat or vehicular maneuvering. The sarsa (λ) Algorithm is used as the algorithm for the following problem. Hierarchical RL is used to train the bot for combat and navigation. The single task controller will be used to form a bot with multiple behavior set. The bot can relearn the same or performed tasks by using the RL Algorithm.

The sarsa (λ) algorithm uses a tabular approach in which it uses a look-up table to store the sate action pairs. As the complexity of the TPS environment increases we can use numerous ways to address the problem, such as approximating the value functions and abstracting the sensor inputs. The Sarsa algorithm with eligibility traces, termed Sarsa , is used for updating the policy values. Eligibility

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 19, April 2017

traces are a method to speed up learning by allowing past actions to benefit from the current reward, and also allowing for sequences of actions to be learned.

## II. SARSA (Λ) ALGORITHM

The Sarsa Algorithm is an On-Policy Algorithm forTD-learning. The major difference between it and Q-Learning, is that the maximum reward for the next state is not necessarily used for updating the Q-values. Instead, a new action, and therefore reward, is selected using the same policy that determined the original action. The name Sarsa actually comes from the fact that the updates are done using the quintuple.

Where s,a are the original state and action, r is the reward observed in the following state and s', a' are the new state action pairs. From the following algorithm,it uses two steps for determining the next state action pairs. A state-action pair is the mapping of how well an action performs in a state and is stored by the policy. The policy evolves over time and provides the path an agent should take to reach the maximum reward for the task. The two most common type of policy representaion are tabular and genralization approaches.

The tabular approach uses a look-up table to store the numerical values of the state-action pairs, while the genrealization approach uses a function approximator to model and generalize the state-to-action mapping. In this paper we will implement the tabular approach in the complex environment of TPS games. The previously calculated TD is used as the learning rate (α) and eligibility trace (λ) to update each state-action pair in the policy.

1: Initialize Q(s, a) = 0, set e(s, a)=0 for all s, a
2: Repeat for each training game
3: Repeat for each update step t in the game
4: Set s' to the current state
5:Select an action a'
6:Take an action a', observe reward r
7: $\delta \leftarrow r + \gamma Q(s', a')$
8:
9: For all s, a:
10: $Q(s, a) \leftarrow Q(s, a) +$
11: $e(s,$
12:
13: Until end game condition is true

TABLE I

Pseudo code for Sarsa Algorithm

At each time step, the RL algorithm chooses an action depending on the current action selection method, a process known as the dilemma of exploration versus exploitation. This dilemma is the tradeoff between the RL algorithm exploiting the knowledge currently in the policy and exploring the state space for better actions to perform. A number of methods have been proposed to deal with this issue such as greedy selection and softmax [1]. The literature shows that the -greedy method is popular and successful in many different scenarios [2], [3]. The parameter εis used to control the ratio between exploration and exploitation by a random chance system. Once an action is chosen, the environment calculates a reward indicating how well the agent performed based on a reward function. The agent's internal policy is then updated according to an update (learning) function. Several RL algorithms have been developed over the years including temporal difference (TD), Q-learning, and Sarsa. The Sarsa algorithm, similar to Q-learning, has successfully been applied to MAS using computer game environments [13], [14].

## III. RELATED WORK

The application of RL toward modern video games remains poorly explored. Despite preliminary findings displaying promising results. An RL algorithm to a racing car game and dealt with the complexity of the environment by evaluating the state at discrete points in time [2]. Actions are then considered as a continuous function of the state–action pair. In a fighting simulation game, Graepel applied the Sarsa algorithm to teach the non-player characters (NPCs) to play against the hard-coded AI [12]. The results indicated that the game agents displayed various behaviors.

In Merick's research a role-playing game is controlled using Q-Learning and greedy exploration function. Generally in TPS games the object types and how to interact with them are defined before the game starts.

Minh in his research has developed deep reinforcement learning method and applied it to Atari 2600, which has an arcade-learning environment. RL has been applied to classical games like PAC-MAN and Backgammon. It has been extensively used in robotic domain. It is very little applied in complex video game environment like TPS.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 19, April 2017

## IV. EXPERIMENTAL SETUP AND ARCHITECTURE

A TPS game is used as the test bed for all the experiments described. A commercial game is used instead of a purpose built game. But there is a drawback with the commercial game. According to Laird's FPS research [24], commercial game engines may have insidious problems that cannot be changed such as the bots having an advantage over the human player. For example, in *Quake 2* the bots had faster turning circles than the human-controlled bots [6].

The game testbed has the basic features of commercial TPS games such as walls, power up items, bots and vehicle maneuvering. Bots are equipped with the ability to turn left, turn right, move forwards, move backwards, strafe left, strafe right, pick up items, and shoot. All bots in the game have exactly the same capabilities and parameters, e.g., turn speed (0.1), speed (0.2 m/update cycle), ammo clip (unlimited), weapon type (handgun), and hit points (50). The guns have a cool down timer of 1 s, to avoid a constant stream of fire.

When a state–action pair occurred, the eligibility trace was set to 1.0, instead of incrementing the current trace by 1.0, as the former case encourages faster learning times [10]. A small learning rate was used in all experiments, and was linearly de- creased during the training phase according to where d is the discount rate applied at each iteration, is the initial learning rate (0.2), is the target end-learning rate (0.05), and is the total number of iterations for the training phase (5000). These values were chosen from success in preliminary runs. [9] discussed about Enhancement of TCP Throughput using enhanced TCP Reno Scheme. Mobile Ad-Hoc Networks (MANETs) have been an area for active research over the past few years due to their potentially widespread application in military and civilian communications.

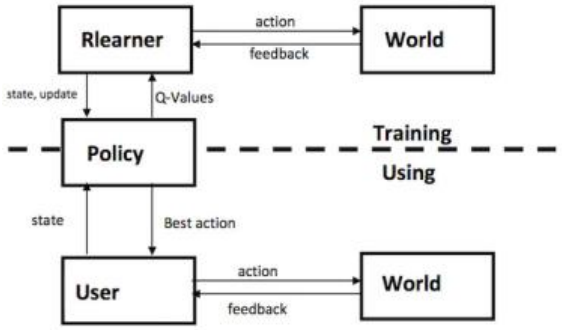


Fig1. Reinforcement Learning Architecture

In this research we placed emphasis more on modularization and reuse of code. The upper part corresponds to the training of our agent, showing its interactions with the world and the policy. The lower part corresponds to the user, a program or an applet that is using the "knowledge" of the trained policy to perform its task. Both the RLearner and the user, interact with their own world, which holds the current state and defines the rules. It provides feedback about the next state, the validity of a certain action and the reward for a certain action. The policy links it all together. It is updated by the RLearner based on the experience it is having with the world and later on consulted by the user for choosing an optimal action. Both the training and the consulting of the policy can also be done simultaneously depending on the application. Usually training is done to some extent before using the policy.

## V. RESULTS

Navigation Task:

The aim of the navigation task is to investigate a bot could learn to maneuver in the environment. The reward function consisted of the objectives 1) Minimize collisions 2) Maximize the distance travelled 3) Maximize the number of bots killed. Small values were chosen for the first two objectives, as the occurrence of them over the training phase iteration was very high. If the reward values were higher, then the item collection reward would be negligible when it occurred.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 19, April 2017

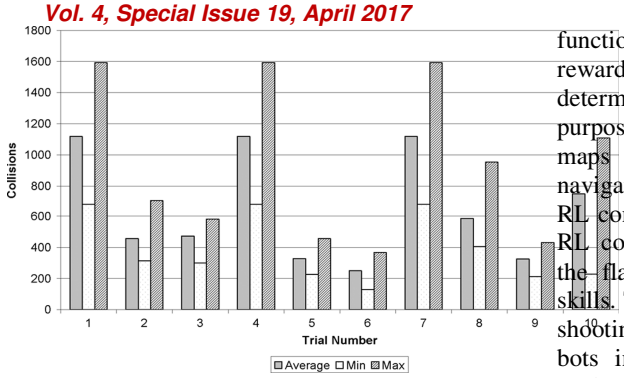


Fig2. Collision Graph

Core Combat Cycle:

The first thing to recognize is that the figure contains all kinds of hidden complexities. For example, for each of the arrow we have a question of "when it is appropriate to follow this transition". Some of the transitions are voluntary (giving up search) other transitions are forced by perception, from combat we are forced transition.

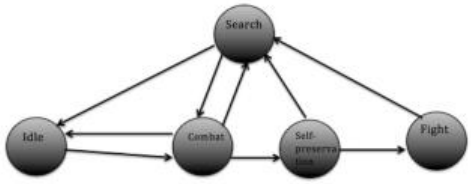


Fig3. Core Combat Cycle

Combat Maps:

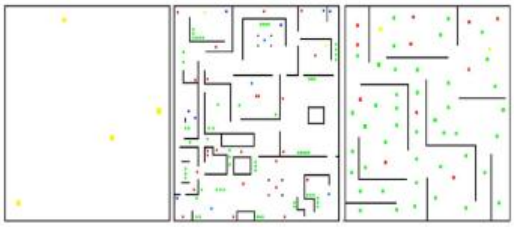

Fig4. A) The arena map B) The maze map C) The core combat map

## VI.CONCLUSION

The proposed approach is implemented on a complex environment unlike the limited state space. This paper presented an approach to apply Reinforcement Learning in Third Person Shooter games where the complexity of the environment is more. The Sara Algorithm is used to implement reinforcement learning and to maximize the reward function. In this Algorithm instead of new action, a reward is selected using the same policy that determined the original action. In the general-purpose task experiments, both the arena and maze maps showed similar trends in combat and navigation statistics, indicating the robustness of the RL controllers. The hierarchical RL and rule-based RL controllers performed significantly better than the flat RL controller in combat and navigation skills. The hierarchical RL bot performed best in the shooting accuracy objective, outperforming all other bots in the experiment. The rule-based RL bot performed slightly better in the other objectives than the hierarchical RL bot, however the observational results showed that the hierarchical RL bot had a wider range of behaviors in combat scenarios. For example, in some situations with multiple enemies, the hierarchical RL bot would flee the scene whereas the rule-based RL bot tended to stay with the fight.

## REFERENCES

[1] Sutton, Richard.S and Barto Reinforecment Learning: An Introduction, MIT Press, 1998

[2] J. Manslow, "Using reinforcement learning to solve AI control problems," in *AI Game Programming Wisdom 2*, S. Rabin, Ed. Hingham, MA: Charles River Media, 2004.

[3] A. H. Tan and D. Xiao, "Self-organizing cognitive agents and reinforcement learning in multi-agent environment," in *Proc. Int. Conf. Intell. Agent Technol.*, Compiegne, France, 2005, pp. 351–357.

[4] J. Bradley and G. Hayes, "Group utility functions: Learning equilibria between groups of agents in computer games by modifying the reinforcement signal," in *Proc. Congr. Evol. Comput.*, 2005, vol. 2, pp. 1914–1921.

[5] S. Nason and J. E. Laird, "Soar-RL: Integrating reinforcement learning with soar," *Cogn. Syst. Res.*, vol. 6, no. 1, pp. 51–59, 2005.

[6] J. Laird, "It knows what you're going to do: Adding anticipation to a Quakebot," in *Proc. 5th Int. Conf. Autonom. Agent*, 2001, pp. 385–392.

[7] S. Cohen, O.Maimon, and E.Khmlenitsky, "Reinforcement learning with hierarchical decision making," in *Proc. 6th Int. Conf. Intell. Syst. Design Appl.*, 2006, vol. 3, pp. 177–182.

[8] M. Huber, "A hybrid architecture for hierarchical reinforcement learning," in *Proc. IEEE*

*Int. Conf. Robot. Autom.*, San Francisco, CA, 2000, vol. 4, pp. 3290–3295.

[9] Christo Ananth, Shivamurugan. C.,Ramasubbu. S, "Enhancement of TCP Throughput using enhanced TCP Reno Scheme", International Journal Of Advanced Research Trends In Engineering And Technology (IJARTET), Volume II, Special Issue XXV, April 2015

[10] Andrew Moore and Chris Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993. □

[11] Nahum Shimkin: Reinforcement Learning Algorithms

[12] T.Graepel, R.Herbrich, and J. Gold, "Learning to fight," in *Proc. Int. Conf. Comput. Games, Artif. Intell. Design Educ.*, 2004, pp. 193–200.