Optimizing Flight Paths Through Anti-Aircraft Gun Fire with Machine Learning

Esben Lund, Bernt Almklov, and Runhild A. Klausen
Norwegian Defence Research Establishment FFI
NORWAY

Esben.Lund@ffi.no Bernt.Almklov@ffi.no

Runhild-Aae.Klausen@ffi.no

ABSTRACT

We present a methodology intended to optimize the flight path through a flight corridor occupied by enemy anti-aircraft guns. This is relevant for all kinds of aircraft, missiles, and drones moving through air space that is fully or partially controlled by such guns. To this end we use Q-learning - a type of reinforcement (machine) learning - which tries to find the optimal strategy to avoiding the anti-aircraft guns through repeated semi-random flight path trials. Q-learning can produce an optimal flight path through the enemy fire without modeling the anti-aircraft guns directly. An adversary response is still needed, but this can come from a black box simulation, user input, real data, or any other source. Here, we use an in-house tool for generating the anti-aircraft fire. This tool simulates a close-in weapons system (CIWS) guided by a fire control radar and Kalman flight path prediction filters. Q-learning can also be supplemented with neural networks - so-called deep Q-learning (DQN) - to handle even more complex problems. In this work, we present results for a subsonic flight corridor pass of one anti-aircraft gun position using classic Q-learning (no neural networks).

1.0 INTRODUCTION

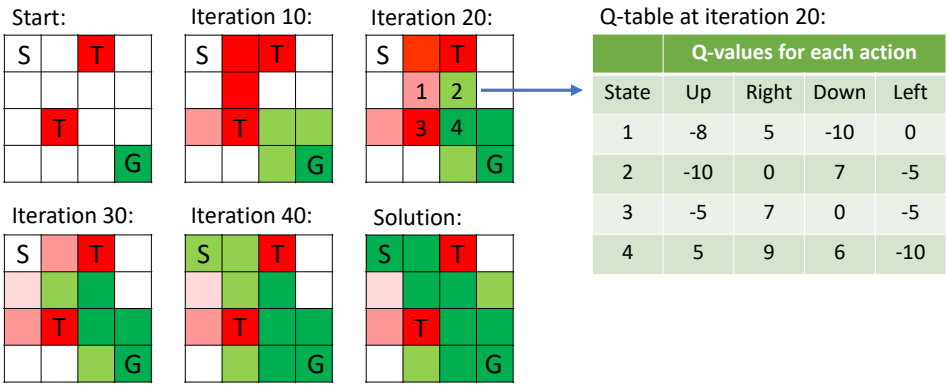
Machine learning has been around for many years, but has seen a surge of interest in recent years due to major improvements in some applications; mainly image and speech recognition, and reinforcement learning. Speech recognition is used in cell phones and appliances, such as Apple's Siri or Amazon's Alexa, while image recognition is found in many places; such as surveillance, autonomous vehicles, medicine, and social media apps. Both speech and image recognition use deep neural networks trained specifically for each application [1]. Reinforcement learning is a more general form of machine learning, not restricted to images or sound [2]. The main principle of reinforcement learning is to set up a challenge, such as a game, with one or more agents (players) in an environment. The agents play a round of the game, and are subsequently rewarded by a judge (the environment) according to their performance. The agents then update their notebooks (aggregated reward tables) to remember (and later repeat) smart moves, or avoid bad ones. After a sufficient amount of games played, the agents will settle on any available optimal solutions. This works fine for simple problems, such as finding your way through a maze, but for more complex challenges, e.g. Go or Chess, the amount of possible states during a game often prohibits building any kind of table containing all states. This has limited meaningful applications of reinforcement learning for many years. However, recently researchers have started using neural networks to replace these state tables. The rewards gained during the game are used to train a neural network to recognize smart moves or patterns, instead of building a state table. These networks can provide good approximations to the state tables, at fractions of the size. A company called Deepmind (acquired by Google) has had great success with this approach in training agents to play Atari games, Go, and Chess [3–7]. Their success has spawned a new interest in reinforcement learning. It has, however, turned out to be difficult to use reinforcement learning in more complex, real life situations, so practical applications of reinforcement learning are still rare.

In this paper we use a form of reinforcement learning called Q-learning to simulate an object (missile, drone, etc.) flying towards a target located next to an anti-aircraft gun. The adversaries in the game are the defending anti-aircraft gun and the incoming flying object, which is rewarded for avoiding the anti-aircraft gun's bullets. The flying object is the only learning agent in the game. The anti-aircraft gun is part of the environment, hence will not improve its performance over time. Furthermore, the flying object is initially flying in a straight line with a fixed speed towards the target, before learning to deviate from the straight path to avoid the anti-aircraft gun's bullets. The anti-aircraft gun is simulated by in-house software taking into account fire control radar properties, ballistics, and flight path predictions. The purpose of the game is to find the flight path with the optimal survivability for the incoming flying object.

Earlier studies done with reinforcement learning on flying objects include homing-phase missile guidance law design [8], and obstacle avoidance for UAV's [9]. Our group has also used deep reinforcement learning for battlefield strategy planning [10].

2.0 Q-LEARNING

The core of Q-learning is to define a problem, set up a table containing all possible states of the problem (the so-called Q-table), and calculate values of the quality of all available actions at these states through simulations [2]. A well-formed problem will converge on a local or global optimal solution, illustrated in Figure 1. Furthermore, the Q-table can be replaced by a neural network to create a deep Q-learning network (DQN). We have chosen to simplify matters initially by not using a neural network. This traditional Q-learning approach has been satisfactory for the problem presented here, but it might be insufficient for more complex cases.



Q-table at iteration 20:

	Q-values for each action			
State	Up	Right	Down	Left
1	-8	5	-10	0
2	-10	0	7	-5
3	-5	7	0	-5
4	5	9	6	-10

Figure 1: Illustration of a simple grid problem solved by Q-learning. The top left grid shows the starting layout with the starting point (S), the green goal (G), and two red traps (T) to be avoided, while the bottom right grid shows the optimal path from S to G reached by Q-learning. The Q-learning process tries many random paths from the starting point, and registers the success of potential future moves (Q-values) in each square (state). Green indicates positive Q-values, while red shows negative Q-values. The optimal solution is built over many iterations, four of which are shown here. The table of all Q-values in all states is called the Q-table. Part of the Q-table at iteration 20 is shown on the right.

Every iteration of the actual Q-learning process starts by updating the current flight path, which is done by either picking the best Q-value from the Q-table, or choosing a random direction, at every step along the flight path. This is called exploitation vs. exploration, and the frequency of random exploratory steps is called the exploration rate (set to 0.3 here) [11]. After updating the flight path, rewards are calculated along the current track by feeding it into the anti-aircraft gun simulation, which simulates the full encounter and returns a list of minimum separations between the fired bullets and the incoming flying object, along the

track. These separations are converted into rewards, larger separations giving higher rewards, which are then used to update the Q-table along the current track;

$$Q^{new}(s_t, a_t) = (1 - \alpha) \times Q^{old}(s_t, a_t) + \alpha \times (r_t + \gamma \times \max Q(s_{t+1}, a))$$

where α is the learning rate, r_t is the reward received when going from state s_t to state s_{t+1}, a_t is the action taken, and γ is the discount factor. The learning rate (set to 0.1 here) determines to what extent newly acquired information overrides old information. A factor of 0 makes the agent learn nothing, while a factor of 1 makes the agent discard all prior knowledge. The discount factor (set to 0.99 here) determines the importance of future rewards. A factor of 0 will only consider the current reward (r_t), while a higher factor also will include long-term rewards. MaxQ(s_{t+1}, a) is the highest Q-value of all possible actions at state s_{t+1}. Only the states and actions visited by the current track are updated. For a complex problem with a large Q-table, it often takes millions of runs to build a complete Q-table.

3.0 LAYOUT

Reinforcement learning is a process with one or more agents trying to maneuver through an environment in an optimal way. The environment reacts to the agent(s), giving negative or positive feedback (rewards) along the way. In this paper the environment consists of an anti-aircraft gun with a target next to it. The gun is simulated by a program written at FFI which takes many factors into account, such as; the fire control radar, bullet ballistics, and flight path prediction. The mechanical properties of the gun, e.g. maximum horizontal and vertical turning speed, are also considered. The flight path prediction is handled by a Kalman filter, and the gun is set up as a typical 30 mm Close-In Weapons System (CIWS) with an effective range of 3 km. It fires constantly at the incoming object, starting before the object comes within range to take bullet travel time into account. The gun, and target, are located 10 m above the ground, with a 30 m separation, shown in Figure 2.

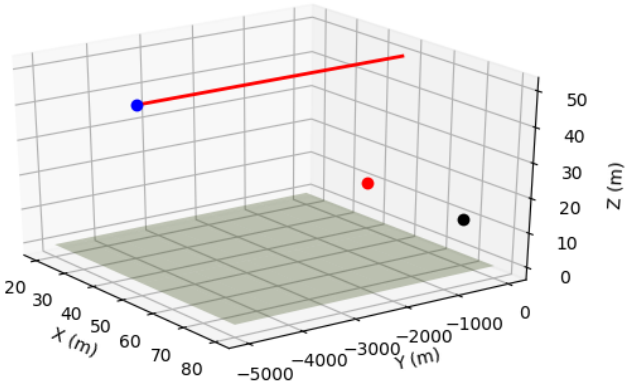


Figure 2: The layout of the simulation, with the blue dot showing the starting point of the flying object 50 m above the ground, 5 km from the target (red dot) and anti-aircraft gun (black dot). The starting point is at X = 50 m, the target is at X = 40 m, and the gun is at X = 70 m. Both the target and gun are located 10 m above the ground. The red line shows the initial flight path (right and into the paper) before alterations by the reinforcement learning process. Note the different scale on the Y-axis.

4.0 Q-TABLE SETUP

To allow the use of a classic Q-learning approach (no neural networks), it is adamant to limit the size of the Q-table. If it does not fit within the computer memory, updating and accessing it becomes prohibitively slow. To achieve this, the flight path is intersected by grids at 25 m intervals, giving 200 intersections for the

whole 5 km track. Moreover, these grids are divided into 51 vertical and 51 horizontal bins (25 on each side of the central bin), i.e. 2601 bins in each grid. Each bin is 2 m x 2 m, giving a total grid coverage of 102 m x 102 m. Multiplying 2601 bins by 200 grids produces 520 200 bins (states) in total for the Q-table. Furthermore, the flying object can choose between nine actions at every state (bin) when moving to the next intersection; straight forward, up, up to the right, right, down to the right, down, down to the left, left, and finally, up to the left, shown in Figure 3. This gives every state nine Q-values, i.e. 4 681 800 Q-values for the whole Q-table, well within the capacity of most computers.

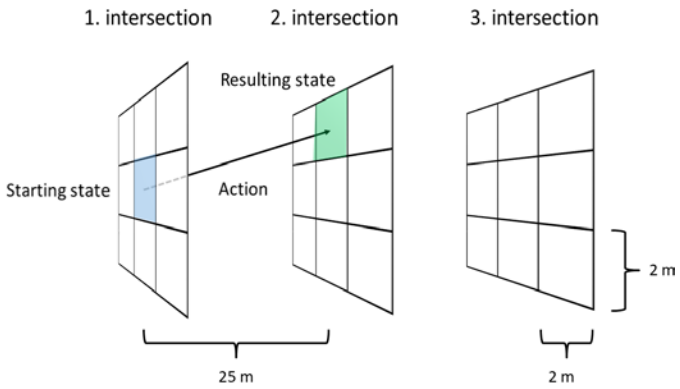


Figure 3: The first three intersections out of 200. Each intersection contains 51 x 51 states (3 x 3 in the figure), with each state having nine possible actions and resulting states in the following intersection. Intersections are 25 m apart, and each bin (state) is 2 x 2 m.

3.1 Rewards

The rewards are calculated by taking the logarithm of the minimum separation between the anti-aircraft bullets and the flying object divided by 10 (limited to the range 1–100 m), which produces rewards ranging from -1 (1 m separation) to 1 (100 m separation). The zero point is at 10 m separation. This keeps the average reward close to zero (avoiding divergence in the Q-values), and reduces the gradient of the rewards at large separations where the benefit of increasing the separation is negligible. The reward for reaching the target is fixed at 500.

5.0 RESULTS AND CONCLUSION

Figure 4 shows the Q-values after 500 000 simulations. Q-values below three are excluded from the plot to make it less cluttered. The black smoothed line shows the optimal track (maximum Q-values) from start to finish. It spirals towards the target (with a 10 m radius) after coming within the 3 km range of the anti-aircraft gun. Running further simulations does not change the flight pattern significantly, indicating that this is a local or global optimal flight path for avoiding this particular anti-aircraft gun and flight path predictor.

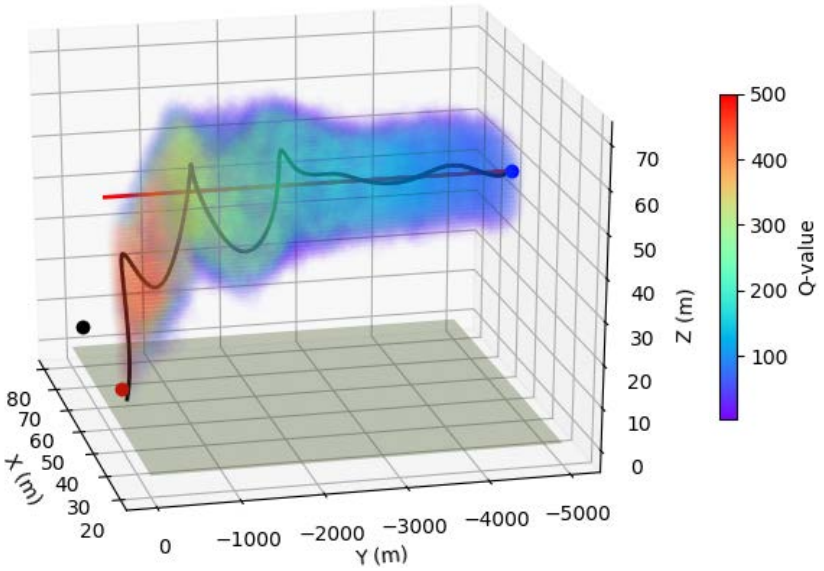


Figure 4: Q-values and optimal track (black line) after 500 000 simulations. The flight starts at Y=-5000 m (blue dot), and ends at the target (Y=0, red dot). The black dot shows the anti-aircraft gun position (30 m from the target), and the red line shows the initial track.

The original and optimal tracks from the simulations, shown in Figure 4, are presented in greater detail from four different angles in Figure 5, with the spiralling pattern clearly visible.

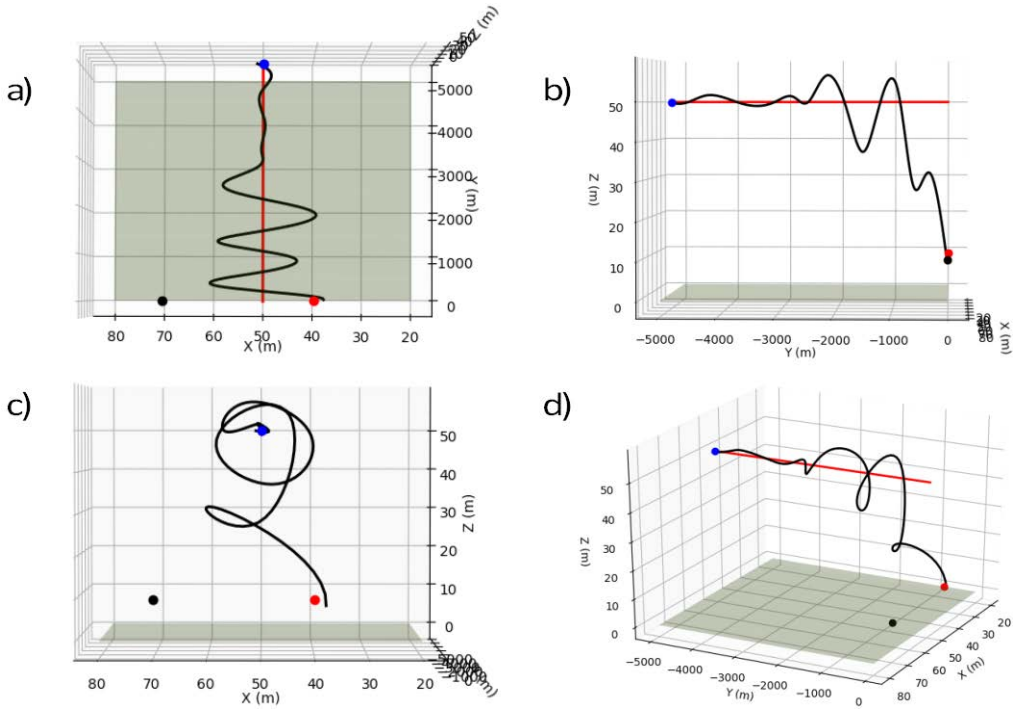


Figure 5: The optimal track (black) after 500 000 simulations from four different angles; a) top looking down, b) side view, c) front view, and d) side/front view. The blue dot shows the starting position, the red dot is the target, the black dot indicates the position of the anti-aircraft gun, and the red line shows the initial straight flight path.

To conclude this paper; by setting up a well-formed and simplified problem, nontrivial solutions to the optimal flight paths through anti-aircraft gun fire can be reached by classic Q-learning without using neural networks, due to the extensive computing power and memory storage of modern PC's.

6.0 REFERENCES

[1] Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press (www.deeplearningbook.org) (2016).

[2] Sutton R. and Barto A. Reinforcement Learning: An Introduction, MIT Press (2017).

[3] Mnih V. et al. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602v1 (2013).

[4] Mnih V. et al. Human-level control through deep reinforcement learning. Nature 518: 529-533 (2015).

[5] Silver D. et.al. Mastering the game of Go with deep neural networks and tree search. Nature 529: 484-489 (2016).

[6] Silver D. et.al. Mastering the game of Go without human knowledge. Nature 550: 354-359 (2017).

[7] Silver D. et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. arXiv:1712.01815v1 (2017).

[8] Gaudet B. et. al. Missile Homing-Phase Guidance Law Design Using Reinforcement Learning. AIAA Guidance, Navigation, and Control Conference 2012. 10.2514/6.2012-4470.

[9] Singla A. et. al. Memory-based Deep Reinforcement Learning for Obstacle Avoidance in UAV with Limited Environment Knowledge. CoRR abs-1811-03307 (2018)

[10] Klausen R. A. et. al. Battlefield Strategy from Deep Reinforcement Learning. NATO STO-MP-IST-160 W4-1 (2018).

[11] Silver D. UCL Course on RL, Advanced Topics (2015).