Spacecraft Autonomous Decision-Planning for Collision Avoidance : a Reinforcement Learning Approach

Nicolas Bourriez, Adrien Loizeau, Adam Abdin

Spacecraft Autonomous Decision-Planning for Collision Avoidance : a Reinforcement Learning Approach

Nicolas Bourriez[1], **Adrien Loizeau**[1] **and Adam F. Abdin**[2,*]

[1]***CentraleSupélec, Université Paris-Saclay, Gif-sur-Yvette, France.***

[2]***Laboratoire Génie Industriel, CentraleSupélec, Université Paris-Saclay, Gif-sur-Yvette, France.***

[*]***Corresponding author. e-mail: adam.abdin@centralesupelec.fr***

[+]***Speaker***

Abstract

The space environment around the Earth is becoming increasingly populated by both active spacecraft and space debris. To avoid potential collision events, significant improvements in Space Situational Awareness (SSA) activities and Collision Avoidance (CA) technologies are allowing the tracking and maneuvering of spacecraft with increasing accuracy and reliability. However, these procedures still largely involve a high level of human intervention to make the necessary decisions. For an increasingly complex space environment, this decision-making strategy is not likely to be sustainable. Therefore, it is important to successfully introduce higher levels of automation for key Space Traffic Management (STM) processes to ensure the level of reliability needed for navigating a large number of spacecraft. These processes range from collision risk detection to the identification of the appropriate action to take and the execution of avoidance maneuvers. This work proposes an implementation of autonomous CA decision-making capabilities on spacecraft based on Reinforcement Learning (RL) techniques. A novel methodology based on a Partially Observable Markov Decision Process (POMDP) framework is developed to train the Artificial Intelligence (AI) system on board the spacecraft, considering epistemic and aleatory uncertainties. The proposed framework considers imperfect monitoring information about the status of the debris in orbit and allows the AI system to effectively learn stochastic policies to perform accurate Collision Avoidance Maneuvers (CAMs). The objective is to successfully delegate the decision-making process for autonomously implementing a CAM to the spacecraft without human intervention. This approach would allow for a faster response in the decision-making process and for highly decentralized operations.

Index Terms

Space Traffic Management; Space Situational Awareness; Collision Avoidance; Artificial Intelligence; Reinforcement Learning

I. Introduction

THE current space environment around the Earth is becoming increasingly populated by both active spacecraft (satellites and launch vehicles) and space debris. Thousands of pieces of debris, measuring at least 10cm in diameter, and millions of pieces larger than 1cm, traveling at extremely high speeds, can significantly damage a spacecraft upon collision (1). Collisions with space debris can generate more debris, which can then lead to further collisions, creating a chain reaction known

as the Kessler syndrome (2). This can lead to a severe threat to the long-term sustainability of Earth's orbits (3).

To minimize the risk of collisions with active or inactive objects in Earth's orbit, owners and operators (O/Os) of satellites must be aware of the collision risk to their assets (4) and need to implement proper actions to manage these risks. With mega-constellations rising and the significant increase in spacecraft orbiting the Earth, managing collision risks and developing proper strategies to improve space situational awareness (SSA) and space traffic management (STM) have become crucially important. Several challenges remain related to the accurate monitoring and tracking of space objects and the development of proper decision-support frameworks, leveraging advancements in Artificial Intelligence (AI) methods to use these data to make informed risk mitigation and operational management decisions.

As part of improving SSA capabilities, space agencies around the world are developing technologies for systems that can detect and track space objects and issue an alert when evasive action may be necessary. The global Space Surveillance Network (SSN) is an example of one such system used to track objects in Earth's orbit and monitor their trajectories (5; 6). The SSN is a network of ground-based radar and optical sensors used to track objects in Earth's orbit. A physics simulator uses SSN observations to predict the evolution of the state of objects over time. Each satellite, also known as a target/protected object, is compared to all other objects in the catalog to detect a conjunction or a close approach. When a conjunction between the target and another object, usually referred to as the chaser/debris, is detected, the SSN propagated states become available, and a Conjunction Data Message (CDM) is produced, containing information about the event. This information includes the time of closest approach (TCA) and the probability of collision. At the issue of these warnings, the satellite's owners and operators have to decide whether to take action to avoid a collision with the available information one to two days before the TCA. To make this decision, they must assess the collision risk, including the probability of collision and the potential consequences of the collision, to plan and implement a collision avoidance maneuver (CAM).

As the space environment becomes increasingly congested, the tasks of detecting, assessing, and planning maneuvers to avoid collisions become more challenging for manual human responses and existing decision processes. Traditionally, experts have been responsible for planning and executing spacecraft CAMs, a process that typically takes days to hours of preparation. However, due to the growing demands of STM, there is a need for intelligent onboard autonomous systems to handle spacecraft maneuvering tasks. These autonomous systems can manage tasks such as collision avoidance and station keeping on a larger scale with faster response times. Automating these processes can significantly enhance the safety and efficiency of space operations, ultimately contributing to a more sustainable and secure future in space operations (7). However, developing onboard autonomous collision avoidance systems is a challenging task. The optimal maneuver must balance multiple factors such as collision probability, propellant consumption, and mission objectives (8). Moreover, the optimal decisions may need to take into account the inherent uncertainties in objects' velocities, location, and monitoring information, increasing the problem's complexity (9). In addition, the adoption of appropriate computational frameworks is crucial for efficiently handling large volumes of data and improving prediction and decision-making accuracy under uncertainty. Addressing these issues requires adequate AI-based decision-support models and algorithms capable of evaluating the trade-off between these different objectives and operational constraints and efficiently finding the optimal solutions.

Recent studies have started to propose optimization and learning algorithms for improving orbital collision risk predictions and developing collision avoidance planning and execution algorithms. Some

researchers have used machine learning (ML) to improve the prediction of the probability of collision using historical conjunction data (10; 11), while others researchers have explored the use of deep learning to simulate the future states of objects and predict the probability of a collision (12; 13). Other works have proposed methodologies for automating the CAM execution using polynomial regression models (14) or robust Bayesian framework (15). In addition, studies developed analytical (16; 17) and semi-analytical (18) models to calculate expressions for the orbit modification to implement autonomous CAM, as well as heuristic approaches such as Particle Swarm Optimisation (PSO) for on-board trajectory generation for STM (19).

While these methods provide a wide variety of tools to develop autonomous CAM frameworks, they often rely on physics-based models to derive adequate action. These models often require a significant number of simplifications compared to the actual environment dynamics, making the decision-making output dependent on the model quality and the assumptions made. More recently, other tools have been developed based on novel maneuver optimization algorithm that combines domain knowledge with Reinforcement Learning (RL) algorithms (20). The RL-based approach allows the model to explore the environment, learning to map states (e.g., collision risks) to optimal actions (e.g., CAMs) without requiring explicit models of the state transitions (physics-based models) and has recently shown promising results in spacecraft CAMs planning (21; 22). However, existing RL applications to spacecraft CAM are based on modeling the decision problem as a Markov Decision Process (MDP), which assumes that the agent is capable of having perfect access to the state of the environment (e.g., accurate debris position and velocity). In reality, however, this is not the case since monitoring data on the state of the environment is imperfect and is characterized by a range of uncertainties. Monitoring data on the status of satellites' systems and orbital environments, such as the proximity to orbital debris and the position and velocity of the debris, are highly uncertain and do not necessarily represent the actual state of the system (i.e., the system state is not fully observable). Therefore, proper modeling of the CAM problem requires methods capable of dealing with partial observability of the status of the systems.

To this end, in this paper, we propose a novel RL-based approach for developing spacecraft autonomous CAM systems that consider aleatory and epistemic uncertainties and in which the decision-makers do not have complete knowledge of the status of the environment. We model the AI planning problem mathematically as a Partially Observable Markov Decision Process (POMDP) to take into account the problem's uncertainties and imperfect monitoring of the environment's status. Moreover, we propose a novel solution algorithm based on Deep Recurrent Q-Network (DRQN) capable of solving the proposed POMDP with continuous and infinite state space and discretized action space. To the best of our knowledge, this is the first implementation of the POMDP formalism to develop AI algorithms for spacecraft planning tasks.

The rest of the paper is organized as follows. Section (I), presented the problem context, the state-of-the-art and the paper contributions. In Section (II) we describe the proposed RL planning model for autonomous CAM planning and execution. The model is mathematically described within a POMDP formalism capable of realistically representing the uncertainties in the orbital environment. Section (III) details the DRQN solution algorithm used for training the AI agent. Section (IV) presents the results and validation of the training model. Finally, Section (V) discusses the conclusions and further research perspectives.

II. Reinforcement learning with partial observability for autonomous CAM planning and execution

We model the AI system with the function of planning and executing CAM decisions onboard the spacecraft (the agent) as a system that does not have explicit knowledge about the possible states of the objects in its surroundings (i.e., debris and non-cooperative spacecraft). Instead, The agent has only imperfect monitoring information from sensory monitoring data and external information sources. A novel RL approach is used to train the agent to autonomously make optimal CAM decisions that simultaneously minimize collision risk and fuel expenditure to perform the maneuvers. For this, we mathematically model the learning process as a Partially Observable Markov Decision Process (POMDP).

A POMDP is a generalization of Markov Decision Processes (MDPs) to decision-making situations in which the real system states are not fully observable (i.e., the monitoring data are insufficient to describe the system's real state). This realistic extension significantly increases the complexity of the model and requires advanced RL algorithms capable of finding the optimal solutions and training the agent in continuous and uncertain observation spaces. To the best of our knowledge, this is the first implementation of the POMDP formalism to develop AI algorithms for spacecraft planning tasks. In this section, we describe in detail the methodology developed in our approach for spacecraft autonomous CAM decision-planning.

A. Partially Observable Markov Decision Process (POMDP) collision avoidance maneuver model

The POMDP is a flexible mathematical framework for representing sequential decision problems (23; 24). Unlike in MDPs, in POMDP, the autonomous AI agent cannot directly observe the state of the environment. Instead, the agent only has access to observations that are generated probabilistically based on previous actions and imperfect data acquired from sensory monitoring. The POMDP framework, thus, models the inherent uncertainty in the space collision avoidance problem to obtain optimal decisions under aleatory and epistemic uncertainties.

To model the spacecraft decision-learning process under a POMDP framework, in our work, the model is defined using the tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$, $\mathcal{O}$, $\mathcal{Z}$, γ), where:

- $\mathcal{S}$: represents the state-space of the environment. It is a continuous space that describes the spacecraft's and space debris' position and velocity vectors in three-dimensional space (x, y, z, dx, dy, dz).
- $\mathcal{A}$: represents the action space of the spacecraft. In our model, the agent can only modify its position by applying impulsive thrust to change its velocity. The action space is discrete, with five possible thrust values, including 0.0, 0.01, 0.05, 0.1, and -0.05, in each of the three dimensions and at a specific time. As a result, the action space has a size of $5^4 = 625$.
- $\mathcal{T}$: represents the environment's state transition probability. We consider that transitions within the model are unknown (i.e., no analytical functions exist to model the state transitions) and that the agent can only learn through interaction with the environment.
- $\mathcal{R}$: represents the reward (or penalty) received by the agent after taking an action. In our framework, the reward received by the agent is based on three components: collision probability, fuel consumption, and trajectory deviation. Detailed explanations of these components are provided below.

- O: represents the observation space of the environment. Similar to the state space, observations are continuous and are defined by the position and velocity vectors of the spacecraft and space debris (x, y, z, dx, dy, dz).
- $\mathcal{Z}$: represents the observation model detailed below. The observation model is characterized by two types of uncertainties: epistemic uncertainties stemming from the SGP4 model and aleatory uncertainties resulting from the monitoring sensors. In our approach, both uncertainties are modeled as a Gaussian distribution.
- γ: represents the learning factor of the RL model ($\in [0, 1]$) and is set to 0.99 in our problem.

An essential component of the presented model is the observation model denoted $\mathcal{Z}$. $\mathcal{Z}(o \text{ — } s, a, s_0)$ is the probability or probability density of receiving observation o about the debris characteristics in state s_0, given the previous state and action were s and a, respectively. Information about the state may be inferred from the entire history of previous actions and observations and the initial information, b_0. Thus, in a POMDP, the agent's policy is a function mapping each possible history, $h_t = (b_0, a_0, o_1, a_1, o_2, \ldots, a_{t-1}, o_t)$, to an action. In some cases, each state's probability can be calculated based on the history of observations. This distribution is known as a belief (b), with $b_{t(s)}$ denoting the probability of state s. The belief is a sufficient statistic for optimal decision-making.

There exists a policy, π, such that when $a_t = \pi(\text{bt})$, the expected cumulative reward is maximized for the POMDP. Given the POMDP model, each subsequent belief can be calculated using Bayes' rule. However, the exact update is computationally intensive, so approximate approaches such as particle filtering are usually used. For our model, the approach implemented to calculate the belief state is discussed in Section (III).

A critical precursor to achieving the training objective is the formulation of an appropriate reward structure. The rewards serve as a critical feedback mechanism for guiding the learning process and shaping the agent's behavior. The cumulative sum of rewards obtained by the agent is a fundamental component of the loss function in our RL training model. This cumulative reward encapsulates the agent's performance to achieve optimal CAMs across each episode and serves as a critical signal for RL, facilitating the optimization of the agent's policy over time. In our formulation, we utilize a reward system characterized by predefined thresholds for various components, each of which contributes to the agent's cumulative sum reward. The key reward components and their associated thresholds are summarized in Table I:

TABLE I: Reward Components and Threshold Values

Reward Component	Threshold Value
Collision Probability	10^{-4}
Fuel Level	10 units
Trajectory Deviation (a)	100
Trajectory Deviation (e)	0.01
Trajectory Deviation (i)	0.01
Trajectory Deviation (W)	0.01
Trajectory Deviation (w)	0.01

- *Collision probability reward:* The agent receives negative rewards when the collision probability exceeds the threshold of 10^{-4}, aligning with NASA's standards for collision risk mitigation in satellite operations.

- *Fuel level reward*: The fuel level is subject to a threshold of 10 units, imposing a negative reward for each action that consumes fuel; this ensures efficient fuel management.
- *Trajectory deviation reward*: Trajectory deviation is evaluated with respect to the differences in the first five osculating Keplerian elements (a, e, i, W, and w). Thresholds of 100, 0.01, 0.01, 0.01, and 0.01 are applied to each respective element, and the agent receives negative rewards proportional to its deviation from these desired values.

B. Collision avoidance data generation and simulation model

To ensure proper training of the spacecraft's autonomous decision-planning model, the quality of the data used is of significant importance. Particularly data related to conjunction events or near-conjunction scenarios to which the agent could be trained. Existing CDMs and Two Line Elements (TLEs) data sets can typically be used as training data. However, historical data on collision events are limited. Furthermore, collisions are typically catastrophic events that result in the destruction of the objects involved. This results in a lack of available data on the dynamics of a collision event. Therefore, available CDMs and TLEs data are not sufficient for training the RL agent to perform CAM optimally.

Instead, we rely on a custom simulator to generate simulated collision events with a wide range of customizable parameters and scenarios. This provides a much larger and more diverse dataset to train the agent, allowing it to generalize better and adapt to new situations. Additionally, the simulator allows for the collection of detailed data during the simulated collision events, providing a better understanding of the dynamics involved.

C. Conjunction simulation model

We simulate conjunction events using an adaptation of the simulator developed in (21). This simulator uses Keplerian elements to instantiate debris objects and generate their positions within a specific range of the satellite.

*1) **Instantiating debris positions using Keplerian Elements**:* To generate collision scenarios between the protected spacecraft and space debris, the Pykep library (25) was used to instantiate debris positions relative to the spacecraft. Pykep provides space flight mechanics computations based on perturbed Keplerian dynamics. The instantiation of the debris position can be defined using Keplerian elements such as semi-major axis, eccentricity, inclination, right ascension of the ascending node, argument of periapsis, and true anomaly. The model can, thus, generate debris positions within a specific range of the satellite, which is essential for creating different instances of trajectory scenarios.

*2) **Projection with SGP4 at each time step**:* To accurately simulate the motion of objects in space, we use the Simplified General Perturbations 4 (SGP4) model, which is widely used for orbit propagation. SGP4 takes into account the gravitational effects of the Earth, the Moon, and the Sun, as well as atmospheric drag and solar radiation pressure. Thus, the simulation model projects the debris trajectory using SGP4 at each time step. This allows us to accurately simulate the objects' positions and velocity and predict their future positions, including potential collisions between the objects and the satellite.

3) ***Retrograde collision reconstruction and debris velocity adjustment****:* In this section, we present a method for the retrograde reconstruction of collisions, starting from their conjunction points and backtracking to an initial time step. The methodology involves the instantiation of the environment, including the protected spacecraft's parameters and its osculating Keplerian elements. For each instantiated debris, the collision time is stochastically determined, followed by the projection of the protected object to the collision time using the SGP4 model. Subsequently, the debris's position is generated with a controlled proximity based on the projected object's position. Finally, the debris's velocity is adjusted in order to simulate the correct direction of motion leading to a collision based on a probability distribution.

1) *Initialization of the environment*: At the start time, we instantiate the protected spacecraft, setting its characteristics such as radius, gravity (μ), fuel level, and Earth's gravitational parameter ($\mu_{\text{central\_body}}$). Additionally, six osculating Keplerian elements are defined: semi-major axis (a) in meters; eccentricity (e) $\in [0, 1)$; inclination (i) in radians; longitude of the ascending node (W) in radians; argument of periapses (w) in radians; and mean anomaly (M) in radians.
2) *Debris instantiation*: for each debris that we choose to instantiate (n_{debris}),
   a) The collision time is randomly determined between *start_time* and *end_time* following a uniform distribution.
   b) Once the collision time is established, we project our protected object up to that collision time using the SGP4 model. This generates a position vector and a velocity vector projected at that precise collision moment.
   c) The debris's position is instantiated based on the projected object's position, with a certain determined proximity. The method to ascertain this collision proximity is as follows: the generation of the debris's position follows a normal distribution centered around the projected object's position (expressed in meters), with a standard deviation σ (a hyperparameter).
   d) Once the debris's position at collision instant c is established, it is important to calculate the debris's velocity as well. However, to update the velocity vector of debris in space, we simulate the correct direction of motion for our debris, which will collide with the protected object according to a specific probability distribution.

We generate the debris's velocity following Eq. (1):

$$\begin{aligned} &rotate\_velocity(\text{vel}, \text{pos}, \theta) = \\ &\|\text{vel}\| \cdot \left(\frac{\cos(\theta)}{\|\text{vel}\|} \cdot \text{vel} + \frac{\sin(\cos(\theta)}{\|w\|} \cdot w \right) \end{aligned} \tag{1}$$

where:

$$\begin{aligned} vel &= \text{velocity vector} \\ pos &= \text{position vector} \\ \theta &= \text{random angle in radians} \\ w &= pos \times vel \end{aligned}$$

It is important to point out that θ is randomly chosen between two possible ranges of angles, each chosen with equal probability $\frac{1}{2}$. These ranges are hyperparameters, acting as the minimum angle between debris and protected object at collision time (radians) ($\leq \pi/4$ by default). Finally, some noise is added to the xdebris' velocity in order to simulate variations or uncertainties in the magnitude of

the velocity, which can be due to factors such as measurement errors, external influences, or modeling inaccuracies. This is expressed in the following equation:

$$\mathbf{v}_{\text{new}} = \mathbf{v} \times \mathcal{N}(1, \sigma_{\text{vr}})$$

where:

- $\mathbf{v}_{\text{new}}$ is the new velocity vector after applying the scaling.
- $\mathbf{v}$ is the original velocity vector.
- $\mathcal{N}(1, \sigma_{\text{vr}})$ is a normal distribution with mean 1 and standard deviation σ_{vr}.
- σ_{vr} is the standard deviation controlling the randomness of the scaling.

Finally, in order to comprehensively assess the performance and robustness of the algorithms developed for collision prediction and avoidance in dynamic environments, a large number of scenarios are generated through a systematic approach, where the parameters governing the behaviors of debris and protective systems are drawn from probability distributions. The developed approach involves a controlled variation of environmental factors that can influence the complexity of collision scenarios. To effectively simulate a range of plausible conditions, key parameters such as the number of debris particles, the temporal span of the environment, the debris positional deviations, the velocity ratio fluctuations, and the minimum angle between debris and protective systems at the point of potential collision, are stochastically determined.

III. Solution algorithm

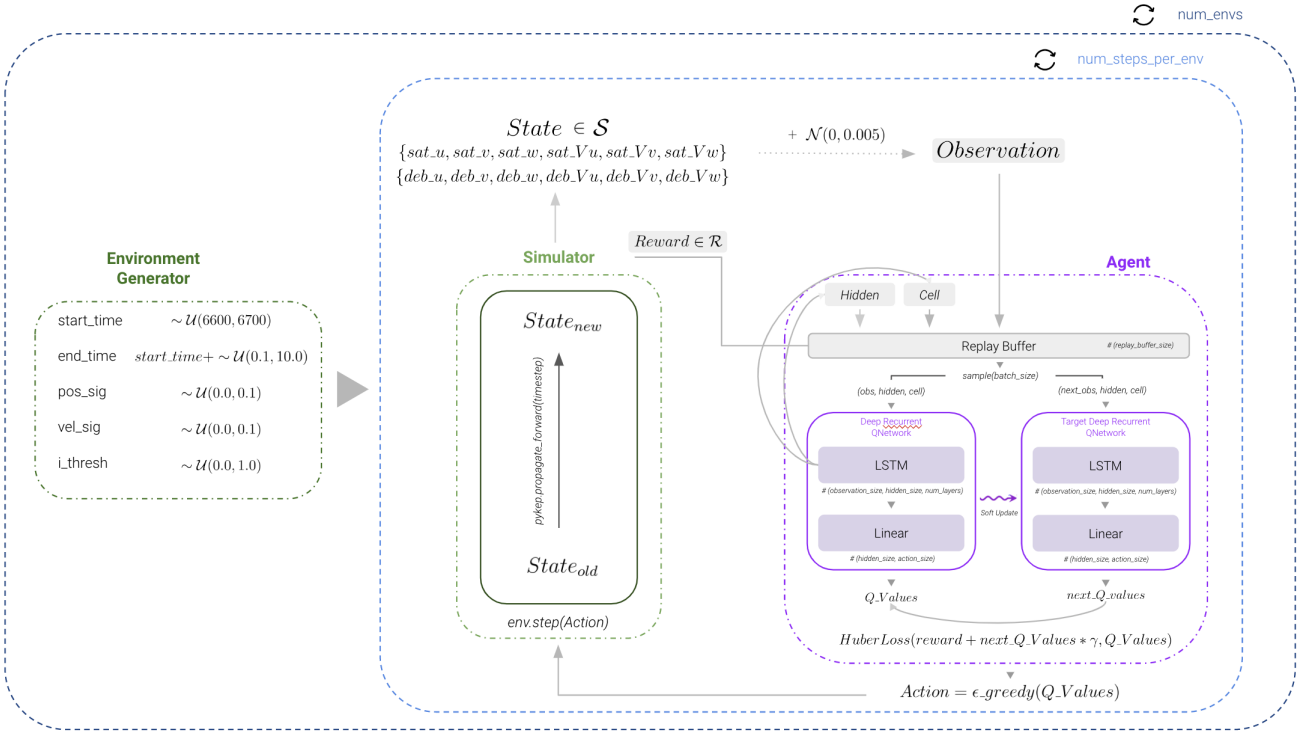


Fig. 1: Architecture of the proposed Deep Recurrent Q-Network (DRQN) algorithm

To solve our proposed POMDP-CAM planning and execution model, an appropriate solver should be implemented. Different RL algorithms can be developed to train the AI system proposed. Given the problem-specific characteristics, Deep Recurrent Q-Network (DRQN) has been found to be the most

capable algorithm for training the autonomous CAM system. This section details the DRQN algorithm and discusses its advantages and limitations for training the agent in the context of our problem.

A. *Deep Recurrent Q-Network (DRQN)*

The proposed CAM learning model is a continuous model-free POMDP with infinite state space and discretized action space. Only a few state-of-the-art algorithms exist to solve these types of models with infinite $mathcalS$ calls and a non-linear function approximator. Deep Q-Network (DQN) has shown promising results for solving the more simple MDPs with similar modeling characteristics. Recent research works have proposed extensions for DQN for solving POMDP problems by adding a Recurrent Neural Network (RNN) layer to the Q-Network (26). The DRQN can serve as a non-linear function approximator for our proposed POMDP-CAM planning model since the state space of space debris is extremely large, and traditional linear approximators cannot capture the complex relationships between different state variables. DRQNs, on the other hand, can learn and represent these non-linear relationships effectively.

Furthermore, the RNN layer in the DRQN architecture plays the role of belief updater by keeping track of the history of past observations and actions. Thus, the latent, hidden state can be considered equivalent to a classical "belief" in the POMDP model. For instance, if the agent observes a sudden change in the position of a piece of debris, the RNN layer can consider past observations and update the agent's belief on whether this change will likely result in a collision.

1) ***Solver Architecture****:* For each conditional environment, a set of stochastic parameters is generated by the environment generator in order to simulate the debris and the protected object paths from start time to end time, with a certain probability of collision. For each step in this environment, the simulator updates its state s. Gaussian noise is added to simulate the model's imperfect monitoring (partially observability), translating it to the observation o. This observation is passed to the agent, i.e., the DRQN algorithm, which will take action accordingly.

The implementation of the DRQN consists of two main components (26): an RNN layer (in that case, a Long Short-Term Networks (LSTM)) and a fully connected layer that approximates the Q-values for each action, given the current state as seen in Figure (1). The LSTM layer helps the network maintain a belief of past observations and actions. Furthermore, a ReplayBuffer and a Target Q-Network are added to enhance the learning process and address some of the DRQN's limitations. On one side, the ReplayBuffer is a memory buffer that stores past experiences (i.e., observation, action, reward, following observation, terminal) and randomly samples a batch of experiences to train the DRQN. This technique helps de-correlate the data from identical sequences and avoid overfitting when the same data is used multiple times. By storing past experiences, the ReplayBuffer also allows the DRQN to learn from experiences that occurred earlier in training, thus increasing the efficiency of the learning process. On the other side, Target Q-Network is a separate network that is softly updated with the weights of the DRQN (27). This network aims to provide a more stable and consistent target for the DRQN to learn from. Without the Target Q Network, the DRQN would be learning from a constantly changing estimate. By soft updating the Target Q Network, the DRQN can learn from a more stable and consistent target, which helps accelerate the learning process and improve the final performance.

2) ***Loss Function****:* The selection of an appropriate loss function is essential in training deep RL models, particularly in the presence of outliers. For the proposed DRQN implementation, we employ the Huber loss function on the cumulative sum of rewards. This choice is driven by the necessity to

robustly handle outliers, which can significantly impact our learning process. Huber loss, also known as smooth mean absolute error, offers the advantage of being less sensitive to extreme reward values, a common occurrence in complex environments featuring rare and substantial rewards or penalties. It is a combination of mean squared error (MSE) and mean absolute error (MAE), which allows it to be robust to outliers while maintaining smoothness and differentiability. Mathematically, the Huber loss function is defined as:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta) & \text{otherwise.} \end{cases}$$

Where:

- a is the difference between the predicted and actual values, $a = y_{\text{pred}} - y_{\text{true}}$.
- δ is a hyperparameter that determines the threshold at which the loss function changes from quadratic to linear. It should be chosen carefully as it affects the model's performance.

The Huber loss function behaves like MSE when the difference $|a|$ is smaller than a threshold δ, and MAE when the difference $|a|$ is greater than δ. This allows it to handle outliers more robustly than MSE, which squares the differences and thus gives more weight to larger differences. Additionally, the Huber loss function is differentiable at 0, which makes it suitable for optimization algorithms that require smooth and differentiable functions, such as gradient descent.

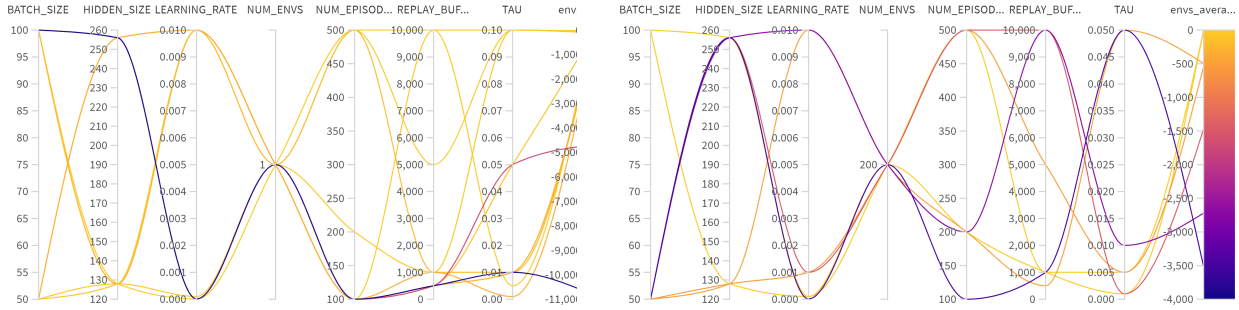
*3) **Constraints**:* One of the main challenges of using a DRQN is the algorithm's sensitivity to hyperparameters, particularly the `tau` and `replay_buffer_size` values. Small changes in the hyperparameter values could cause the network to diverge, making it difficult to train. Therefore, we perform a thorough investigation using hyperparameter sweeps to find the best values of these parameters. The results of hyperparameter tuning are discussed in the next section.

IV. Results and discussion

This section presents the initial results of the study. We first focus on presenting the results related to the efficacy of the DRQN model in learning optimal CAM policies. Given its inherent complexity, a central focus of this evaluation section is to elucidate the strategy for attaining solution convergence, both in a single training environment and across a large number of environments. The aim is to demonstrate the adaptability and robustness of our DRQN model to learn optimal CAM planning and execution policies under a variety of circumstances.

A. *Hyperparameters tuning*

We employ a grid search method for hyperparameter tuning of the model. Grid search operates by exhaustively evaluating a predefined set of hyperparameter values over a structured grid. Each unique combination of hyperparameters is systematically assessed, allowing for a comprehensive exploration of the hyperparameter space. In our problem, grid search enabled us to methodically examine various hyperparameter configurations, including neural network (NN) architecture specifications (e.g., hidden size layer), training parameters (e.g., learning rate, batch size), and exploration strategies (e.g., epsilon-greedy exploration). By systematically evaluating these hyperparameter choices on the average reward sum, we sought to identify the optimal configuration that would yield improved convergence and learning performance for our DRQN model. Figures (2a) and (2b) illustrate the grid search made for both 1 and 200 environments.



(a) Grid Search over 1 environment

(b) Grid Search over 200 environments

Fig. 2: Model hyperparameter tuning.

B. *Training of Spacecraft CA agent*

The process of hyperparameter optimization allowed the identification of the best parameter configurations. Subsequently, the next step is to select a configuration that would consistently converge across different stages of training. The hyperparameters selected for both training scenarios (under one environment and under 200 training environments) can be found in Tables (II) and (III), respectively.

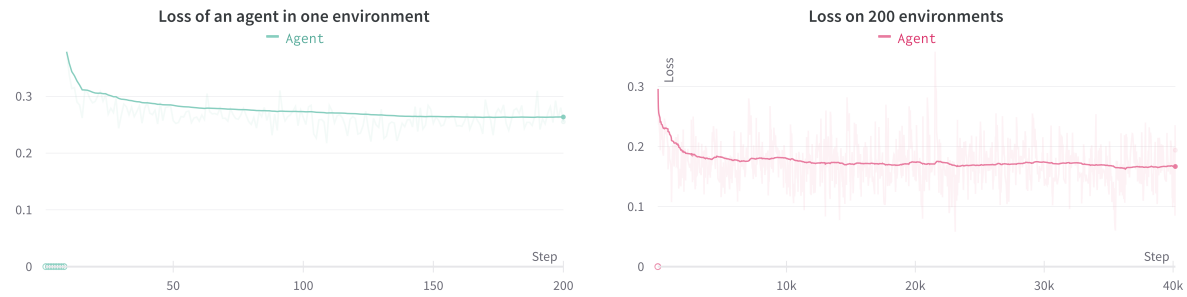
TABLE II: Hyperparameters used for evaluation on 1 environment

Hyperparameter	Value
Batch size	50
Hidden size	128
Learning rate	0.0001
Number of episodes	200
Replay buffer size	1,000
Tau	0.1
Number of environments	1

TABLE III: Hyperparameters used for evaluation on 200 environments

Hyperparameter	Value
Batch size	100
Hidden size	128
Learning rate	0.0001
Number of environments	1
Number of episodes	200
Replay buffer size	1,000
Tau	0.1

To evaluate the performance of the DRQN agent, we compare it to a baseline approach that uses a simple threshold to trigger a collision avoidance maneuver. Figure 3a illustrates the agent's loss trajectory over each training step within a single collision avoidance environment. These results show that the agent is successfully learning to autonomously conduct CAM maneuvers, as it illustrates the progressive improvement in maximizing its cumulative reward over the simulations. Furthermore, the training is extended to encompass a more complex scenario involving 200 distinct environments. The results show that the agent trained with this configuration attained better training results than those under a single training environment. Figure 3b shows the loss trajectory and the average cumulative reward profile of one of the successful agents in this more complex multi-environment setting. These results show that our proposed approach is capable of training an autonomous CAM system to plan and execute CAMs effectively, as evident by the significantly decreasing loss funtion. The trained spacecraft AI system performed well in terms of collision avoidance effectiveness, fuel consumption, and computational efficiency. In particular, the spacecraft was able to adapt to changing environments and make more efficient maneuvers in response to the available information. It is worth noting that the prospect of fine-tuning hyperparameters in future work remains an avenue for further enhancing the capabilities of our DRQN algorithm.



(a) Results of training on one environment

(b) Results of training on 200 environments

Fig. 3: Loss of an agent (spacecraft autonomous CAM system) trained on different environments: decreasing loss signifies the agent's improvement in learning to perform optimal CAM planning and executions.

V. Conclusions

While significant improvements in Space Situational Awareness (SSA) activities and Collision Avoidance (CA) technologies are allowing for tracking and maneuvering spacecraft away from potential debris collision risks with increasing accuracy and reliability, these procedures still primarily involve a high level of human intervention to make the necessary decisions. This decision-making strategy will not be sustainable for an increasingly complex space environment. It is, therefore, important to successfully introduce higher levels of automation for key Space Traffic Management (STM) processes to ensure the reliability needed for navigating large constellations of spacecraft. These processes include collision risk detection, the identification of the appropriate action to take, and the execution of avoidance maneuvers. In this work, we developed an implementation of autonomous CA capabilities for spacecraft based on Reinforcement Learning (RL) techniques. We propose to model the spacecraft CA training model using a novel Partially Observable Markov Decision Process (POMDP) solved with an efficient Deep Recurrent Q-Network (DRQN) algorithm. This allows the proper training of the AI system onboard the spacecraft as a system with imperfect monitoring information on all the possible states of the objects in its surroundings. This is particularly relevant to the practical setting of SSA for managing collision risk considering uncertainties in space debris positions and velocities.

The model proposed takes as input the current state of the environment, which includes the positions and velocities of the satellite and debris, as well as the CDM information, and outputs an action corresponding to a maneuver to avoid a potential collision. The agent is trained to maximize the expected cumulative reward over time. The reward function used includes the distance between the satellite and the debris, the probability of collision, and the fuel cost of the maneuver. Our results demonstrate the potential of using deep RL methods, such as DRQN, for developing autonomous CA systems in space. This approach could help mitigate the increasing risk of collisions with space debris and ensure the safety of space assets. To the best of our knowledge, this is the first implementation of the POMDP formalism and the DRQN solution algorithm to develop AI systems for spacecraft planning tasks.

Finally, several factors should be considered for the applicability and robustness of our approach in practical implementations. First, it should be noted that our model was trained on synthetic data generated to approximate realistic assumptions. However, the actual performance of our model in operational settings may deviate due to variations in the physical characteristics of space debris, satellite trajectories, and other environmental factors. Consequently, care should be taken to validate the proposed approach in controlled settings before practical implementations. Second, the precision

and accuracy of CA estimations hinge on exogenous variables, such as the reliability of satellite position and velocity predictions delivered by orbital propagators like the Simplified General Perturbations 4 (SGP4) model. Any errors in these models can influence the performance of the collision avoidance system. Thus, continuous efforts to enhance the precision of these systems remain pivotal to enhancing the efficacy of our approach.

Authors contributions

A.Abdin conceived the original idea of the study, the technical methodology proposed, and supervised the work. A.Loizeau and N.Bourriez contributed equally to developing and implementing the software, writing the first draft of the paper, and providing the initial analysis of the results. A.Abdin adapted and revised the manuscript.

Acknowledgements

The authors would like to thank Matthieu Roux (Ph.D. candidate at the Laboratory of Industrial Engineering, CentraleSupélec) for his valuable insights and suggestions on the implementation of the methodology proposed.

References

[1] J. Radtke, C. Kebschull, and E. Stoll, "Interactions of the space debris environment with mega constellations—using the example of the oneweb constellation," *Acta Astronautica*, vol. 131, pp. 55–68, 2017.

[2] H. Krag, M. Serrano, V. Braun, P. Kuchynka, M. Catania, J. Siminski, M. Schimmerohn, X. Marc, D. Kuijper, I. Shurmer *et al.*, "A 1 cm space debris impact onto the sentinel-1a solar array," *Acta Astronautica*, vol. 137, pp. 434–443, 2017.

[3] T. J. Muelhaupt, M. E. Sorge, J. Morin, and R. S. Wilson, "Space traffic management in the new space era," *Journal of Space Safety Engineering*, vol. 6, no. 2, pp. 80–87, 2019.

[4] S. Le May, S. Gehly, B. Carter, and S. Flegel, "Space debris collision probability analysis for proposed global broadband constellations," *Acta Astronautica*, vol. 151, pp. 445–455, 2018.

[5] A. Horstmann and E. Stoll, "Investigation of propagation accuracy effects within the modeling of space debris," in *7th European Conference on Space Debris*, 2017.

[6] A. K. Mashiku and M. D. Hejduk, "Recommended methods for setting mission conjunction analysis hard body radii," in *2019 AAS/AIAA Astrodynamics Specialist Conference*, no. GSFC-E-DAA-TN71115-1, 2019.

[7] K. L. Hobbs and E. M. Feron, "A taxonomy for aerospace collision avoidance with implications for automation in space traffic management," in *AIAA Scitech 2020 Forum*, 2020, p. 0877.

[8] E.-H. Kim, H.-D. Kim, and H.-J. Kim, "Optimal solution of collision avoidance maneuver with multiple space debris," *Journal of Space Operations*, vol. 9, no. 3, pp. 20–31, 2012.

[9] E. Denenberg and P. Gurfil, "Debris avoidance maneuvers for spacecraft in a cluster," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 6, pp. 1428–1440, 2017.

[10] F. Pinto, G. Acciarini, S. Metz, S. Boufelja, S. Kaczmarek, K. Merz, J. A. Martinez-Heras, F. Letizia, C. Bridges, and A. G. Baydin, "Towards automated satellite conjunction management with bayesian deep learning," *arXiv preprint arXiv:2012.12450*, 2020.

[11] M. Vasile, V. Rodríguez-Fernández, R. Serra, D. Camacho, and A. Riccardi, "Artificial intelligence in support to space traffic management," in *68th International Astronautical Congress*, 2017.

[12] L. Sánchez, M. Vasile, and E. Minisci, "Ai to support decision making in collision risk assessment," in *70th International Astronautical Congress*, 2019.

[13] L. S. Fernandez-Mellado and M. Vasile, "On the use of machine learning and evidence theory to improve collision risk management," *Acta Astronautica*, vol. 181, pp. 694–706, 2021.

[14] K. A. Ramaneti, C. Krishna, A. Ahmed, and R. Rajesh, "Autonomous space debris collision avoidance system," in *Advances in Automation, Signal Processing, Instrumentation, and Control: Select Proceedings of i-CASIC 2020*. Springer, 2021, pp. 2489–2501.

[15] C. Greco, L. Sánchez, M. Manzi, and M. Vasile, "A robust bayesian agent for optimal collision avoidance manoeuvre planning," in *8th European Conference on Space Debris*, 2021.

[16] J. L. Gonzalo, C. Colombo, and P. Di Lizia, "Analytical framework for space debris collision avoidance maneuver design," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 3, pp. 469–487, 2021.

[17] J. Gonzalo Gòmez, C. Colombo *et al.*, "Collision avoidance algorithms for space traffic management applications," in *71st International Astronautical Congress*, 2020.

[18] J. Gonzalo Gomez, C. Colombo, P. Di Lizia *et al.*, "A semi-analytical approach to low-thrust collision avoidance manoeuvre design," in *70th International Astronautical Congress*, 2019.

[19] E. Lagona, S. Hilton, A. Afful, A. Gardi, and R. Sabatini, "Autonomous trajectory optimisation for intelligent satellite systems and space traffic management," *Acta Astronautica*, vol. 194, pp. 185–201, 2022.

[20] S. Willis, D. Izzo, and D. Hennes, "Reinforcement learning for spacecraft maneuvering near small bodies," in *AAS/AIAA Space Flight Mechanics Meeting*, vol. 158, 2016, pp. 1351–1368.

[21] L. Gremyachikh, D. Dubov, N. Kazeev, A. Kulibaba, A. Skuratov, A. Tereshkin, A. Ustyuzhanin, L. Shiryaeva, and S. Shishkin, "Space navigator: A tool for the optimization of collision avoidance maneuvers," *arXiv preprint arXiv:1902.02095*, 2019.

[22] Q. Qu, K. Liu, W. Wang, and J. Lü, "Spacecraft proximity maneuvering and rendezvous with collision avoidance based on reinforcement learning," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 6, pp. 5823–5834, 2022.

[23] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Partially observable markov decision processes for artificial intelligence," in *KI-95: Advances in Artificial Intelligence*, I. Wachsmuth, C.-R. Rollinger, and W. Brauer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 1–17.

[24] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, 01 2005.

[25] D. Izzo, "esa/pykep: Major update." November 2017, https://doi.org/10.5281/zenodo.1063506 [Accessed: January-2023. [Online]. Available: https://doi.org/10.5281/zenodo.1063506

[26] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 aaai fall symposium series*, 2015.

[27] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.