Double Deep Q Networks for Sensor Management in Space Situational Awareness

Benedict Oakes
CDT in Distributed Algorithms
University of Liverpool
Liverpool, United Kingdom
sgboakes@liverpool.ac.uk

Dominic Richards
Artificial Intelligence Group
STFC Hartree Centre
Daresbury, United Kingdom
dominic.richards@stfc.ac.uk

Jordi Barr
Cyber and Information Systems Division
Defence Science and Technology Laboratory
United Kingdom
jmbarr@dstl.gov.uk

Jason Ralph
Electrical Engineering and Electronics
University of Liverpool
Liverpool, United Kingdom
jfralph@liverpool.ac.uk

***Abstract*—We present a novel Double Deep Q Network (DDQN) application to a sensor management problem in space situational awareness (SSA). Frequent launches of satellites into Earth orbit pose a significant sensor management challenge, whereby a limited number of sensors are required to detect and track an increasing number of objects. In this paper, we demonstrate the use of reinforcement learning to develop a sensor management policy for SSA. We simulate a controllable Earth-based telescope, which is trained to maximise the number of satellites tracked using an extended Kalman filter. The estimated state covariance matrices for satellites observed under the DDQN policy are greatly reduced compared to those generated by an alternate (random) policy. This work provides the basis for further advancements and motivates the use of reinforcement learning for SSA.**

***Index Terms*—Reinforcement Learning, Sensor Management, Space Situational Awareness**

I. Introduction

In an era of regular and frequent launches of satellites to low Earth orbit (LEO), the possibility of collisions between resident space objects continues to increase, and poses a significant threat to space based infrastructure. The Kessler Syndrome – a cascade of collisions that could render any satellite use in LEO extremely difficult and costly – becomes an increasing risk [1]. Between one third and one half of the capacity of LEO space has already been occupied [2].

Space Situational Awareness (SSA) is the understanding of the complex orbital domain, involving man made objects, and natural phenomena [3]. Ground-based surveillance and tracking of man made objects in orbit can be achieved with a variety of instruments, including radars and optical telescopes. Measurements are required to be able to predict the trajectories of the objects, to assess the risk of potential collisions. However, measurements must be made repeatedly, as the orbit of any satellite is subject to change. These changes may be small perturbations, but the accumulation of small changes over time can be significant. In the LEO environment, there are several factors that could affect the orbit of satellites - most notably intentional manoeuvres, atmospheric drag, or solar radiation pressure could alter the orbit from a predicted trajectory. With limited sensor availability, efficient sensor management (SM) algorithms are necessary for long-term SSA. Given the large number of objects in LEO, the problem suffers from a combinatorial explosion as the number of possible actions increases [4]. The European Space Agency is investing in improving the long-term sustainability of the space domain [5], and employing novel methods to improve SSA and help accomplish this goal. Objects orbiting the earth in LEO have short orbital periods, meaning they cannot be observed reliably from a single site; and these sites are often constrained to making measurements in clear weather and of restricted patches of sky. Therefore, using multiple sensors located around the globe is highly beneficial, but comes with a cost and is a considerable SM challenge.

Deep reinforcement learning (DRL) is one possible solution to this problem. DRL is the combination of standard reinforcement learning algorithms with neural networks to solve Markov decision processes (MDPs). DRL has been applied to various fields with large action spaces, and has produced impressive results [6]- [8].

II. Filtering and State Estimation

In this paper, we aim to estimate $\{X\}$, the set of state vectors describing satellites' positions and velocities, using measurements $\{Y\}$. The optimal state estimation algorithm for linear, Gaussian systems is the well known Kalman filter (KF). The Kalman filter is the best linear estimator for reducing the mean square error [9]. For slightly nonlinear systems, adaptations of the KF exist to attempt state estimation while overcoming some of these non-linearities. The extended Kalman filter (EKF) employs state transition and measurement functions, as opposed to simple matrices, to propagate the state estimates. However, the covariance is propagated linearly through the step, so the EKF is only suitable for systems with modest non-linearities. The unscented Kalman filter (UKF)

develops this further, by generating sigma points around the target position, and propagating these through the non-linearity, and reforming the covariance [10]. In this paper we will only use an EKF for simplicity but the approach is readily extendable to UKF or other state estimation methods.

III. Reinforcement Learning

Reinforcement learning (RL) is a machine learning method in which an intelligent agent must make decisions to maximise its received reward, which is determined by the results of the actions it takes [11], [12]. RL differs from other machine learning learning areas in that the model can be unknown, the agent need only know the actions and the reward, as well as some observation about the environment's transition into new time steps, based on the environment's evolution over time. Observations are usually related to some value in the environment that determines the amount of reward returned. This can be ideal for SM applications, particularly in SSA, where we do not need to model a potentially complex environment for the agent to interpret. This means RL can work in much higher dimensions than other dynamic programming approaches.

Markov decision processes (MDPs) are the underlying formulations that RL algorithms are built upon. MDPs operate discretely, where at each time step, an action is made. The state will react to this action via a transition, and a reward is given. The transition is defined as $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s'|S_t = s]$, where $\mathcal{P}_{ss'}$ is the state transition probability, s is the Markov state at time t and s' is the successor state. The goal of an MDP is to find a policy, matching states to actions, to receive the maximum reward.

Previous work into RL for SSA includes applications of DRL in [13] and [14]. They show proof-of-concept results for applying RL to the SSA problem, using Actor-Critic methods. The actor refers to the policy, that asks the estimated value function, or critic, about the next possible state values, which the critic improves during learning. More recently, extensions to the previous work were completed that added more complexity and required less intensive compute resources to run [15], [16]. We present the first implementation of a DDQN to the sensor management problem for SSA, as opposed to the Actor-Critic methods cited above.

A. Q-learning

Q-learning is a simple value iteration update on a Markov decision process. Q-values, or quality-values, are state-action values, and refer to the expected reward gained by taking a certain action in a given state. Q-learning attempts to first find Q-values for a range of states and actions, and to then exploit the Q-values by selecting the action that returns the highest reward at any state, in a greedy policy. Q-learning is different from a Q-value iteration algorithm, as the transition probabilities and rewards are initially unknown.

Q-learning is defined by:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot \max Q(s_{t+1}, a) - Q(s_t, a_t)) \tag{1}$$

where Q is the expected reward and is a function of action a and state s at time t. α is the learning rate, and $0 < \gamma < 1$ is the discount factor. α is a tuning parameter that determines how quickly the algorithm learns new information. γ is a parameter required for convergence of the algorithm, and determines how much weight is given to information in the future. If γ is close to 1, the future is valued almost as much as the present. If γ is close to 0, the immediate information is much more highly valued [17].

B. Deep Q Network

A DQN is an implementation of Q-learning. The main issue with Q-learning is that it does not scale to larger problems with larger action and state spaces. Deep Q-learning was developed to overcome this. Deep Q-learning uses neural networks (NNs) and experience replay to use a random sample of prior actions instead of just the most recent action. Some well-known DQNs use convolutional NNs: hierarchical layers of tiled convolutional filters to mimic the effects of receptive fields [18]. Receptive fields are defined as the association of input fields to output fields. Experience replay is the use of batches of sampled transitions for better data efficiency and stability. DQNs are the term given to implementations of Q-learning algorithms applied to such NNs.

C. Double Deep Q Network

It has been shown that DQNs commonly overestimate action values in certain situations, and produce over-confident Q-values [19]. To solve this problem, Double Deep Q Networks were developed. In DQNs, the max operator is used to select and evaluate actions, which leads to overly confident value estimates. By using two sets of weights θ and θ', and using one to determine the policy and the other to evaluate it, this problem is effectively overcome.

IV. Problem Simulation

In this scenario, we create a satellite simulation using a Python package Pysatellite: a Github repository being developed by the author [20]. This package implements orbit generation, reference frame transformations, and target tracking – in this scenario through the use of an EKF. We generate 25 LEO satellites using a Keplerian model, visualised in Fig. 1. Higher order terms such as solar radiation pressure and atmospheric drag will be included in further iterations. In this paper, it is assumed that all satellites are following circular orbits at a radius from the centre of the Earth of $R = 7 \times 10^6$ metres. For this implementation, we find that using an EKF is adequate to handle the non-linearities of the system, but in future work, a UKF or particle filter may be more suitable.

Detections are generated by simulating a telescope on the surface of the Earth which measures azimuth, elevation, and range with additive Gaussian noise. Fig. 2 shows the satellite paths from the telescope's point of view. The measurements are transformed into the Earth-centred inertial (ECI) reference frame, a Cartesian frame with the origin at the centre of the Earth, through which the Earth rotates. The

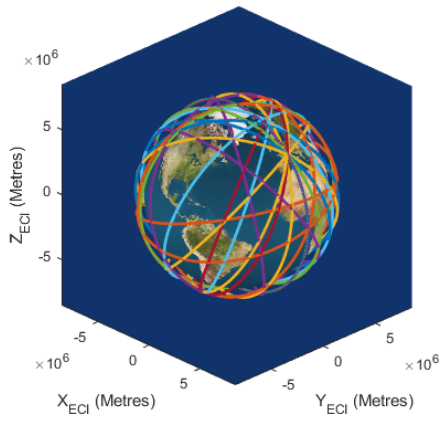

Fig. 1. LEO satellite paths plotted around the earth in the ECI reference frame

EKF operates in an ECI reference frame, with a state vector $X = (x, y, z, x_v, y_v, z_v)$, which encodes the Cartesian position and velocity of the satellite. Measurements are transformed from azimuth, elevation, and range to ECI via the following method:

$$Y_{AER} = \begin{bmatrix} \phi \\ \theta \\ R \end{bmatrix} \tag{2}$$

$$Y_{NED} = \begin{bmatrix} R \cdot \cos(\theta) \cdot \cos(\phi) \\ R \cdot \cos(\theta) \cdot \sin(\phi) \\ -R \cdot \sin(\theta) \end{bmatrix} \tag{3}$$

$$Y_{ECEF} = \begin{bmatrix} -\sin(\phi_0)\cos(\lambda_0) & -\sin(\lambda_0) & -\cos(\phi_0)\cos(\lambda_0) \\ -\sin(\phi_0)\sin(\lambda_0) & \cos(\lambda_0) & -\cos(\phi_0)\sin(\lambda_0) \\ \cos(\phi_0) & 0 & -\sin(\phi_0) \end{bmatrix} \cdot Y_{NED} \tag{4}$$

$$Y_{ECI} = \begin{bmatrix} \cos(\omega t) & -\sin(\omega t) & 0 \\ \sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot Y_{ECEF} \tag{5}$$

where ϕ, θ, and R are the azimuth, elevation and range coordinates of the satellite respectively, Y_{NED} refers to the commonly used North, East, Down reference frame, ϕ_0 and λ_0 are the latitude and longitude positions of the sensor, respectively, ω is the Earth rotation rate, and t is the sidereal time. In our simulated problem, we define our own ECI and ECEF reference frames related to the time elapsed between frames. For dealing with real data, ECI and ECEF reference frames that relate to common time bases must be used. The method explained here does not account for higher-order specificities associated with real-world reference frames, but is suitable for a simple geoid simulation.

A measurement noise matrix is generated in the AER frame for each measurement. We simulate ideal diffraction limited measurements.

$$P_{AER} = \begin{bmatrix} \sigma_\theta^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_r^2 \end{bmatrix} \tag{6}$$

where σ_θ and σ_r are the standard deviations expected for ideal angle and range measurements. These are converted to the ECI frame by calculating the Jacobian matrix of the measurement through the above transformation from AER to ECI and applying it to the measurement noise matrix:

$$P_{ECI} = J \cdot P_{AER} \cdot J^T \tag{7}$$

where J is the calculated Jacobian matrix.

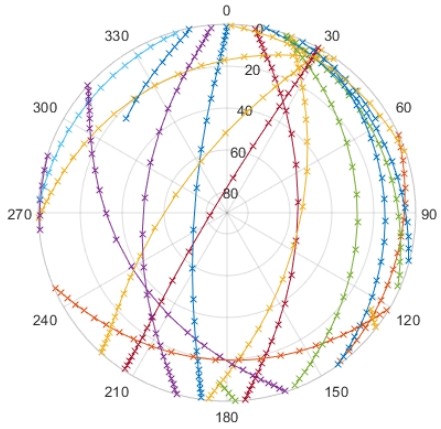

Fig. 2. Satellite paths crossing an Earth based telescope's Field of View

For the learning environment, we use Tensorflow Agents, an accessible and approachable RL framework in Python. Agents allows users to create their own environments, and apply them to a range of different RL algorithms, including DQN, Deep Deterministic Policy Gradient (DDPG), and others; for our simulation we used a DDQN. The DDQN uses a replay buffer and stochastic gradient descent to calculate the loss.

We state the following definitions for clarity: iterations refer to the number of training episodes that have occurred. Episodes refer to one full run of an environment, made up of t time-steps. At each iteration, the step transitions are added to a circular buffer that stores the last n number of iterations. During learning, a small sample of the buffer is used to calculate the loss, instead of just the last transition. This provides two benefits: as each transition is sampled many times, a higher data efficiency is achieved, and using uncorrelated transitions leads to a better stability of data.

We model a telescope control scenario, with a sensor state $T = (\phi, \theta)$, where ϕ is the telescope's azimuth pointing, and θ is the telescope's elevation pointing. At each time step of the environment, the agent can choose from 5 possible

actions: move up, down, left, right, or do nothing. As the environment is discretised, we assume that when an action is taken, the telescope state in the next time-step will be equal to the maximum distance travelled in that direction, based on a telescope slew rate of $2^o/s$. Up and down refer to the telescope's elevation pointing, and left and right refer to the telescope's azimuth pointing. If actions are taken that would be unfeasible, such as the telescope pointing below the horizon, no action is taken. As elevation can cross the zenith at $\pi/2^c$, and azimuth is bounded $0 < \theta < 2\pi^c$, actions that would take the telescope direction out of this range are wrapped. For each satellite, we find the difference between the centre of the telescope Field of View (FoV) and the satellite's azimuth and elevation position.

$$d_\phi = |s_\phi - T_\phi| \tag{8}$$

$$d_\theta = |s_\theta - T_\theta| \tag{9}$$

where d_ϕ and d_θ are the differences in azimuth and elevation, respectively, s_ϕ and s_θ are the satellite's azimuth and elevation positions, respectively, and T_ϕ and T_θ are the telescope's centre azimuth and elevation pointing, respectively. If both azimuth and elevation are within the FoV of the telescope, then a reward is given. The reward is cumulative in each time-step, so if multiple satellites are detected, the reward will increase accordingly. Algorithm 1 shows the basic operation of the RL environment.

Algorithm 2 shows the training loop to improve the agent's policy, and select the best action at each time-step. The agent will train for a certain number of iterations, and the performance is periodically evaluated using a sample of episodes which are executed with the current policy. We then run another episode to generate measurements using the current policy, which are used in the EKF.

Algorithm 1 RL environment

```
t = time-step
a = action
r = reward
s = satellite, s_p = satellite position
T(φ, θ) = sensor state with field of view FoV
o = observation
while episode not ended do
  t += 1
  Get a for current t
  if a < limit then
    apply a
  else
    wrap and apply a
  end if
  for each s do
    o = T(φ, θ) − s_p
    if o within FoV then
      measurement = True
      r += 1
    else
      measurement = False
      r += 0
    end if
  end for
  return r, o
  if last t then
    end episode
  end if
end while
```

Algorithm 2 DDQN Training

```
t = time-steps
r = reward
Y = measurements
for each iteration do
  collect multiple t, save to replay buffer
  sample buffer, update network
  if iteration = evaluation interval then
    calculate r̄ of 10 episodes with current network
    generate set of Y from current network
    use Y in EKF
  end if
end for
```

V. Results

After training the above environment on a DDQN for 20,000 iterations, and sampling the average reward at every 1,000 iterations, Fig. 3 shows that the DDQN agent clearly outperforms the same environment run with a random policy, where at a given time-step, an action is picked at random, instead of choosing the action that will maximise the cumulative reward. We choose to compare against a random policy to show a baseline for learning, and to show clearly the improvement of the DDQN over increasing iterations. Future work will compare the DDQN to other RL implementations. Each iteration trains the agent with 10 episodes of the environment, with each episode consisting of 20 time-steps. We use a seeded simulation to create the same satellite orbits, and run the environment for 5 sets of iterations to generate average returns.

It is clear that after ∼8,000 iterations, the DDQN begins to converge on an optimal policy, that far exceeds the random policy shown, which has no obvious improvement over time, as expected. After this, the DDQN remains at or near the optimal policy, with no loss of return from catastrophic forgetting, a common problem in RL algorithms [21]. Catastrophic forgetting occurs as an agent explores its environment, it may learn things that break its previously learnt information, causing the agent to forget the past information and return poor rewards. The shaded region shows the standard deviation of 5 runs of the same environment in the DDQN. The standard deviation decreases once the algorithm reaches the plateau, showing its increased confidence in this region of training. The maximum possible return of the DDQN is less than the number of satellites simulated because only some of the satellites will cross the telescope field of view during the length of the environment simulation, as exemplified in Fig. 4. Increasing

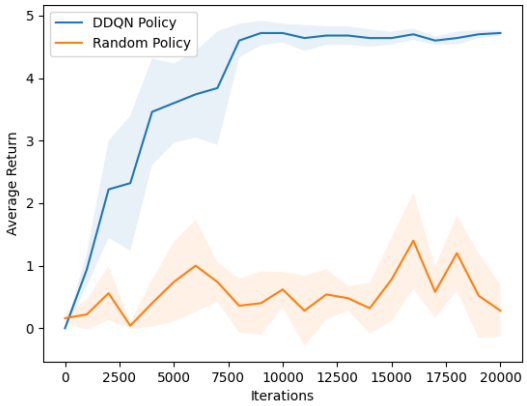

Fig. 3. Average reward from 10 episodes returned by an environment trained using a DDQN. The trained policy in blue shows vast improvements against the random policy. Shaded regions show the standard deviation of 5 training runs

the number of steps in each episode would lead to a higher maximum possible return, but training would take far longer due to the increased action space over the larger number of steps, and thus would require more iterations to converge.

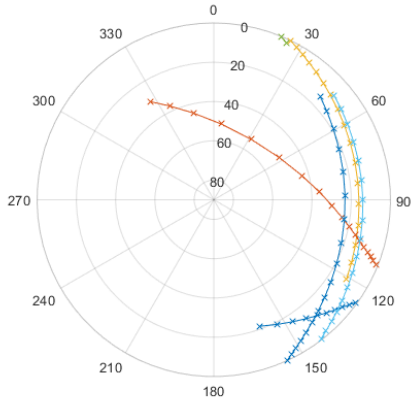

Fig. 4. Satellite paths crossing an Earth based telescope's FoV over a limited number of time steps - most satellites will not appear in telescope FoV

In Fig. 5, we show the log of the trace of the covariance matrices associated with each satellite after they have been tracked for the length of the episode. The value for each point in the graph is the trace of the covariance matrix after applying an EKF to the satellite for n steps, where n is the number of steps used in the RL environment, based on the measurements generated in the DDQN.

If a satellite is captured within the telescope FoV, a measurement is generated; conversely if the satellite is not observed by the telescope, no measurement can be made. As the policy improves and the number of satellites seen in the FoV of the telescope increases, more measurements are generated as iterations increase. By having more measurements for each satellite, the EKF is able to reduce the uncertainty of the

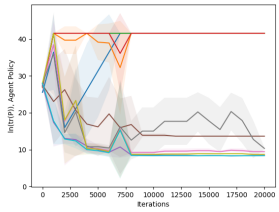

Fig. 5. Log-trace of final covariance matrices for tracked satellites at the end of an episode. Each line represents a different satellite visible in the telescope FoV. Shaded regions show the standard deviation of 5 training runs

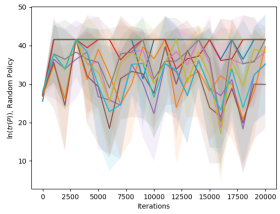

Fig. 6. Log-trace of final covariance matrices for tracked satellites at the end of an episode, with a random policy. Each line represents a different satellite visible in the telescope FoV. Shaded regions show the standard deviation of 5 training runs

target position and velocity, which can be seen in the lines at the bottom of the graph. Where limited or no measurements are made, the EKF can only predict the satellite position and velocity, giving increasing uncertainty – seen at the top of the graph. We see that over half of the visible satellites are measured more consistently as the DDQN trains, leading to reductions in the final uncertainty of the satellite. We compare this with Fig. 6, which is the result of tracking on measurements generated from a random policy. Here we see no overall improvement on the tracking performance.

In Fig. 7 and Fig. 8, we show the outcome of tracking in the final iteration, when the agent has attained the optimal reward. In the trained run, we see that a majority of the satellites are detected by the telescope, meaning the measurements made are able to reduce the uncertainty. Some satellites are never seen in the FoV, which is shown by the top line in the graph, where the EKF becomes increasingly uncertain about its state. In the random action run, we can see that no satellite has consistent measurements, meaning that the uncertainties are not lowered as well. This shows a SM algorithm that is better than random pointing, proved by a covariance-based metric.

VI. Future Work

In future work, we hope to expand on the work completed here, with the inclusion of target tracking performance metrics. Two such metrics that would likely prove fruitful in this scenario are the Posterior Cramér-Rao Bound (PCRB) [22] and the Generalised Optimal Sub-Pattern Assignment (GOSPA) [23]. The PCRB would be useful in situations where the geometry affects the resulting information of the targets, such as in cases where the covariance of a target is very thin but long. The GOSPA metric will be more suitable in scenarios where there is clutter, false detections, and missed targets [24], all of which are likely in the SSA domain. Further improvements will be made to increase the complexity of the satellite dynamics and how they are tracked, including more robust reference frame transformations, for example. Including effects like solar radiation pressure and atmospheric drag will increase the realism of the scenario, and will require more advanced tracking algorithms, such as a UKF. Other

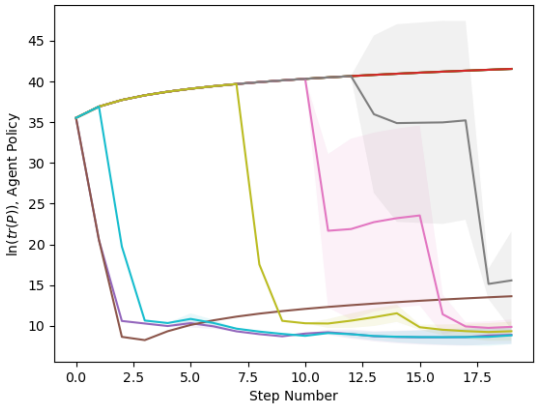

Fig. 7. Satellite covariances during the final trained episode. Each line represents 1 satellite. Shaded regions show the standard deviation of 5 training runs

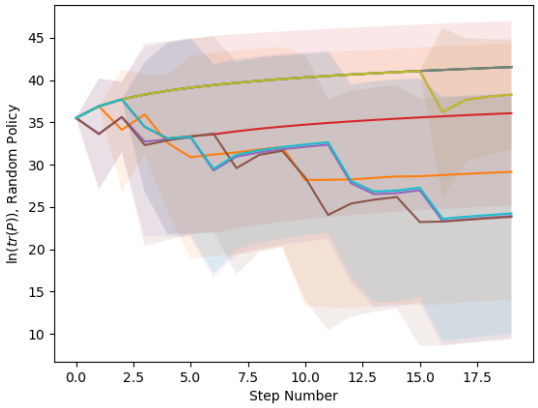

Fig. 8. Satellite covariances during the final random episode. Each line represents 1 satellite. Shaded regions show the standard deviation of 5 training runs

advancements include using angle-only measurement models, and continuous control agents to more accurately reflect the use of real telescopes.

VII. Conclusions

In this paper, we simulate a controllable Earth-based telescope viewing satellites in low Earth orbit in a reinforcement learning environment. We present a novel application of a Double Deep Q Network to space situational awareness. We maximise the number of satellites observed during a time period, and increase the number of successful measurements made. We use the generated measurements in an extended Kalman filter, in which we see a significant reduction in position and velocity uncertainty for observed satellites as a result of increasing observations, as opposed to observations made from a random policy. This forms the basis of a framework for future research into applying deep reinforcement learning to space situational awareness.

References

[1] D.J. Kessler, B.G. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt", *Journal of Geophysical Research*, Volume 83, Issue A6, pp. 2637-2646, 1978

[2] C. Pardini, L. Anselmo, "Evaluating the impact of space activities in low earth orbit", *Acta Astronautica*, Volume 184, pp. 11-22, 2021

[3] J. A. Kennewell, B. Vo, "An overview of space situational awareness" *Proceedings of the 16th International Conference on Information Fusion*, pp. 1029-1036., 2013

[4] N. Adurthi, P. Singla, M. Majji, "Mutual Information Based Sensor Tasking with Applications to Space Situational Awareness", *Journal of Guidance, Control, and Dynamics*, Volume 43, No. 4, pp. 767-789, 2020

[5] European Space Agency, "ESA's Space Environment Report 2021", https://www.esa.int/Safety_Security/Space_Debris/ESA_s_Space_Environment_Report_2021, accessed 2022.

[6] D. Silver, J. Schrittwieser, K. Simonyan, et al. "Mastering the game of Go without human knowledge", *Nature* 550, pp. 354–359, 2017

[7] O. Vinyals, I. Babuschkin, W.M. Czarnecki, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning", *Nature* 575, pp. 350–354, 2019

[8] M. Jaderberg and W.M. Czarnecki, I. Dunning et al. "Human-level performance in 3D multiplayer games with population-based reinforcement learning", *Science*, 364, 6443, pp. 859-865, 2019

[9] J. Humpherys, P. Redd, J. West, "A Fresh Look at the Kalman Filter", *SIAM Review*, Volume 54, No. 4, pp 801-823, 2012

[10] S.J. Julier, J.K. Uhlmann, "New extension of the Kalman filter to nonlinear systems", *Signal processing, sensor fusion, and target recognition VI*, Vol. 3068, pp. 182-193, International Society for Optics and Photonics, 1997

[11] L. Kaelbling, M. Littman, A. Moore, "Reinforcement learning: A survey", *Journal of artificial intelligence research*, Vol. 4, pp. 237–285, 1996

[12] R.S. Sutton, A.G. Barto, *"Reinforcement learning: An introduction"*, Malaysia: MIT press, 2018

[13] R. Linares, R. Furfaro, "An autonomous sensor tasking approach for large scale space object cataloging", *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, pp. 1-17, 2017

[14] R. Linares, R. Furfaro, "Dynamic sensor tasking for space situational awareness via reinforcement learning", *Advanced Maui Optical and Space Surveillance Technologies Conference*, p. 36, Maui, HI: Maui Economic Development Board, 2016

[15] A.E. Harvey, K.B. Laskey, "Online Learning Techniques for Space Situational Awareness (Poster)", *22th International Conference on Information Fusion (FUSION)*, pp. 1-7, 2019

[16] A.E. Harvey, K.B. Laskey, K. Chang, "Machine Learning Applications for Sensor Tasking with Non-Linear Filtering", 2021

[17] G. A. Rummery, M. Niranjan, "On-Line Q-Learning Using Connectionist Systems", *Cambridge University Engineering Department*, Cambridge, England, Technical Report, 1994

[18] V. Mnih, K. Kavukcuoglu, D. Silver, et al. "Human-level control through deep reinforcement learning", *Nature*, Volume 518 (7540), pp. 529–533, 2015

[19] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning", *Association for the Advancement of Artificial Intelligence*, Vol. 30, No. 1, 2016.

[20] B. Oakes, "Pysatellite", https://github.com/sgboakes/pysatellite, last accessed: 14/05/2022

[21] A. Géron, *"Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems"*, Beijing: O'Reilly, September 2019.

[22] M. L. Hernandez, A. Farina, "Posterior Cramér-Rao Bound for Target Tracking in the Presence of Multipath", *2018 21st International Conference on Information Fusion (FUSION)*, pp. 151-158, 2018

[23] A. S. Rahmathullah, Á. F. García-Fernández, L. Svensson, "Generalized optimal sub-pattern assignment metric", *2017 20th International Conference on Information Fusion (FUSION)*, pp. 1-8, 2017

[24] Á. F. García-Fernández, M. Hernandez, S. Maskell, "An analysis on metric-driven multi-target sensor management: GOSPA versus OSPA", *2021 IEEE 24th International Conference on Information Fusion (FUSION)*, pp. 1-8., 2021