An Autonomous Sensor Tasking Approach for Large Scale Space Object Cataloging

Richard Linares*
University of Minnesota, Minneapolis, NM, 55455
Roberto Furfaro†
University of Arizona, Tucson, AZ, 85721

Abstract

The field of Space Situational Awareness (SSA) has progressed over the last few decades with new sensors coming online, the development of new approaches for making observations, and new algorithms for processing them. Although there has been success in the development of new approaches, a missing piece is the translation of SSA goals to sensors and resource allocation; otherwise known as the Sensor Management Problem (SMP). This work solves the SMP using an artificial intelligence approach called Deep Reinforcement Learning (DRL). Stable methods for training DRL approaches based on neural networks exist, but most of these approaches are not suitable for high dimensional systems. The Asynchronous Advantage Actor-Critic (A3C) method is a recently developed and effective approach for high dimensional systems, and this work leverages these results and applies this approach to decision making in SSA. The decision space for the SSA problems can be high dimensional, even for tasking of a single telescope. Since the number of SOs in space is relatively high, each sensor will have a large number of possible actions at a given time. Therefore, efficient DRL approaches are required when solving the SMP for SSA. This work develops a A3C based method for DRL applied to SSA sensor tasking. One of the key benefits of DRL approaches is the ability to handle high dimensional data. For example DRL methods have been applied to image processing for the autonomous car application. For example, a 256x256 RGB image has 196608 parameters (256*256*3=196608) which is very high dimensional, and deep learning approaches routinely take images like this as inputs. Therefore, when applied to the whole catalog the DRL approach offers the ability to solve this high dimensional problem. This work has the potential to, for the first time, solve the non-myopic sensor tasking problem for the whole SO catalog (over 22,000 objects) providing a truly revolutionary result.

1 Introduction

Space Situational Awareness (SSA) is an area of research motivated by the U.S. Air Force's mission to protect and ensure peaceful access to space. This task involves collecting tracking data on over 22,000 space objects (SOs), 1,100 of which are active, using optical and radar sensors. There has been a lot of work done on improving the information that can be derived from observational data [1–7], however, planning observations is still a challenge. The large number of SOs, currently being tracked and the forecasted growth of tracked objects, combined with the limited number of sensors available to track them, makes it difficult to maintain persistent surveillance. Therefore,

*Assistant Professor, Department of Aerospace Engineering and Mechanics. Email: rlinares@umn.edu.
†Associate Professor, Department of Systems & Industrial Engineering, robertof@email.arizona.edu.

sophisticated sensing planning strategies are required in order to decide which SO to observe and to make the correct trade-offs for protecting the space assets. The sensors in the SSN can be thought of as a collection of information-gathering agents. The system must make decisions on the types of measurements and the ordering of measurement sequences in order to optimize the performance of a sensing task, such as detection [8], classification [9], and tracking [10].

Development of approaches for optimal and adaptive allocation of sensing resources will be a significant contribution to the field of SSA and the warfighter. This work develops a new Deep Reinforcement Learning (DRL) approach for sensor tasking in SSA based on the Asynchronous Advantage Actor-Critic (A3C) method [11]. This method allows for an optimal solution to be learned over time from the SSN's actions, a solution that can handle the large number of SOs (>10,000[1]) and can be optimal for complex tasking scenarios.

The challenge for the SSA sensor tasking problem is to determine the tasking performance prior to obtaining the sensor measurement [12–14]. The nonlinearity of the SSA tasking problem introduces additional difficulty when it comes to the problem statement. It is well known that if the dynamics and measurement models are both linear, the evolution of the covariance matrix is deterministic. For the nonlinear problem, the covariance matrix evolution is no longer deterministic, but rather dependent on the observations and stochastic. Usually, information metrics for sensor tasking are only dependent on the covariance matrix, and therefore, the metric evolves deterministically given a sequence of control, thus making the optimal sensing problem deterministic for linear systems. For the nonlinear SSA problem, the optimal sensing problem will be stochastic in general since the information metrics are now measurement dependent.

The nonlinearity of the orbital dynamics problem requires the solution of the evolution of the non-Gaussian joint probability density function (pdf) given initial condition and uncertainty in the force models. This evolution is captured by solving the Kolmogorov Equation for the joint density function [15] or statistical moments for capturing sufficient statistics. Additionally, the long-term scheduling solution suffers from combinatorial increase in problem size when solving tasking problems of even moderate size. The complexity of a sensor tasking problem is discussed in reference [16], and although the work considers a different formulation than that of the SSA SM problem, the authors showed that the tasking problem is NP-hard. The sensor tasking problem suffers from the *curse of dimensionality*, where the complexity of the problem grows exponentially with dimension.

For example, consider a brute force solution of the tasking problem where the performance for all choices of the sensor tasking is evaluated. For 100 potential sensors, from which we are to choose 25, there are on the order of 10^{23} possible choices. Therefore, direct brute force solution is not possible with currently available computing resources and an approach that can overcome the *curse of dimensionality* of the SSA SM would provide a groundbreaking result. This work attempts to overcome this issue through the use of DNN that have been shown to be capable of tempering the *curse of dimensionality*. For example, DNN have been applied to the game GO [17], a notoriously complex game with on the order of 10^{170} board positions. The way DNNs overcome the *curse of dimensionality* is through learning high level, low dimensional abstractions for a problem and then using this low dimensional representation for the solution. The SSA PM can be thought of as having a low dimensional feature space[2] since all SOs share the same dynamics and there are a fixed number of regimes (LEO, MEO, GEO, etc).

The SM problem is a general challenge across many engineering applications and has been extensively studied, for example in tracking multiple objects using surveillance cameras [18], mobile robotic applications [19], among others. The sensor tasking problem has been formulated using both

[1] this is an estimate based on the DRL literature

[2] in the absence of chaos

an information theoretic framework [14] and in a general Bayesian framework [20]. Additionally, this work is investigates the sensor tasking problem in the context of dynamical systems which has been studied extensively [21, 22]. In recent past, the optimal sensor tasking problem has been investigated by the control systems and robotics communities and it is known as information-theoretic control [23], active sensing [24], and dual control [25] in those communities. Although, the sensor tasking problem has been studied extensively, there have been very few works that address the long-term tasking problem. In addition, most tasking formations are based on very specific user defined metrics and there hasn't been any work on developing general tasking approaches that can address a diverse set of objects[3]. Finally, most of the sensor tasking related work focuses on linear systems and the nonlinear problem has not been addressed adequately. These challenges, among others, are addressed by this work.

Most methods for sensor scheduling are myopic and only offer optimality for short-term benefit. Short-term or single time step optimal solutions for the SM problem have been proposed that use information entropy, tracking covariance, Fisher information, Cramer-Rao lower bound, and information divergence [26]. The non-myopic case, where an optimal solution is desired over large time horizons, is more challenging. When the performance metric is measured over an extended period of time, myopic approaches do not provide adequate solutions. Reference 26 formulates the sensor scheduling problem as a Partially Observable Markov Decision Process (POMDP) and solved it using the completely observable rollout method. This method allowed for the inclusion of long-term performance considerations. An alternative approach is that of Ref. 27, which used a Reinforcement Learning (RL)-based sensor scan optimization scheme for multi-target tracking. Ref. 27 used temporal difference learning utilizing an ϵ-greedy Gibbs method for exploration. The RL approaches offer a computationally efficient solution to dynamic programming problems and a way around the *curse of dimensionality* (due to action space size explosion). This work explores RL for SM using the policy gradient method.

Sensor tasking for SSA is usually based on priorities, such as estimating atmospheric re-entry, limiting time elapsed since last observation, and particular military interest among other things. A simple method for solving the tasking problem relies on binning objects. SOs are binned into classes based on perigee and apogee heights, and these SM methods provide a general suggested amount of observations per day for objects in these bins, which are referred to as Gabbard classes [28]. Using these Gabbard class based observations per day guidelines, resources can be distributed to maintain a catalog. In practice, methods based on object binning and heuristic rules can control uncertainty growth in the catalog, but these types of methods are sub-optimal and not based on fundamental principles. Advanced, statistically rigorous methods for SSA sensor tasking have been developed that provide close to optimal myopic tasking solutions [18,29–31]. These sensor scheduling methods are myopic and only optimal for a single time step into the future. Reference 29 studied a method based on the covariance of SOs to solve the tasking problem. This work used a metric based on the reduction in covariance in a given measurement to determine the "best" observations to make at the current time step. Reference 32 also used covariance information to determine both a Fisher-Information Matrix (FIM) based and a hybrid tasking approach that used covariance information to develop a myopic sensor tasking strategy.

References 29 and 32 did not use multi-time step optimization and also used linearized uncertainty models, which only account for the mean and covariance of the SO pdf. Reference 30 overcame these two limitations by using Lyapunov exponents to account for future uncertainty growth and the Unscented Kalman Filter (UKF) to account for higher order uncertainty. Furthermore, this work was extended to non-Gaussian probability density functions (pdfs) by Ref. 30, which used a Gaussian sum filter to represent the pdf of each object. Reference 33 was the first

[3]to the knowledge of the authors

paper to study the non-myopic SM problem for SSA and was able to show a clear improvement over myopic approaches. Reference 33 solved this problem using information space receding horizon control and stochastic optimization. This work was later extended to non-Gaussian SO pdfs [31]. Reference 31 applied the AEGIS-FISST approach to this problem by developing a tasking strategy that can include the data association process using Random Finite Sets.

The benefit of the proposed DRL approach is that it can improve over time and that it can learn autonomy tasks that have not been explicitly programed into it. It can also learn from examples and learn optimal policies from sub-optimal behavior. Another advantage is that only a subset of sensors are required to follow the RL policy while the other sensors can continue to use the traditional approaches. The RL approach can learn from both its own actions and actions of other sensors using traditional approaches.

RL based methods have recently seen successful applications to a range of tasks from robotic control tasks, locomotion [34,35] to manipulation [36,37] and autonomous vehicle control [38]. The ability of RL approaches to generalize well to different domains and learn directly from experience makes it promising for the SSA SM problem. DRL algorithms have now been used for applications outside of simulated settings or relatively simple hardware task to complex robotic tasks. Reference 39 demonstrated a DRL algorithm based on off-policy training of deep Q-functions for complex robotic 3D manipulation tasks. This work also showed that these approaches are efficient enough to train on real physical robotic systems and not just in simulated environments. Therefore, this work proposes the a DRL-based SM approach which can be trained on real sensors or in a simulated environment.

Answering questions such as "How should I plan my observations to discover new and pressing threats to my space assets?" have been left largely unaddressed due to a number of technical challenges. For one, translating this question into a mathematical objective can be difficult. Even with a mathematical objective, the problem is difficult to solve because of the large number of parameters required for the solution. Therefore, the research community has focused largely on a much simpler task of designing SM solutions for maintaining a catalog. Under the RL framework, high level goals such as "classify rocket body, debris, and payloads", "find new objects", and/or "keep overall catalog error below a threshold" can be specified, and the RL approach can then be used to learn how to translate its actions into strategies that meet these high level goals optimally.

This paper develops a DRL approach for the SM problem applied to SSA using the A3C method [11]. The A3C method is a policy gradient method similar that of Reference 40, however A3C has many benefits over traditional policy gradient methods. The key advantage that this work leverages is ability of the A3C method to train asynchronously which allows it to be parallelized and to be apply to a network of robotic agents. This shows that the A3C method can solve larger problems than that of traditional actor-critic methods [40].

The organization of this paper is as follows. First, the problem statement is given for the dynamic sensor tasking problem. Following this the state representation and dynamics for the uncertainty state of SOs is discussed. Next, the A3C method and neural network models are discussed. Then the simulation orbital and measurement models and parameters are provided. Additionally, results are shown for simulated examples. Finally, discussions and conclusions are provided. This paper discusses the theory involved behind the proposed algorithms and results from simulation trials are shown.

2 Problem Statement

As discussed earlier, the SSA SM problem can be formulated as a MDP using the sufficient statistics (the minimum set of parameters needed to completely describe a random process). The SM prob-

lem can be expressed using a discrete-time system model, with the state of the system at time step k denoted by $\mathbf{x}_k$. The system dynamics provides the transition from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$ given $\mathbf{u}_k$, where $\mathbf{u}_k \in \mathcal{R}^\ell$ denotes the current control action, and this transition may be stochastic. Therefore, it is meaningful to represent this transition with a probability distribution $\mathbf{x}_{k+1} \sim p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k)$, where $\mathbf{x}_k$, $\mathbf{x}_{k+1} \in \mathcal{R}^n$ denotes the current and next state, respectively. Since the SSA SM problem is nonlinear and therefore stochastic in nature, the action is modeled probabilistically. The probabilistic model for the actions of the agent is given by a policy $\mathbf{u}_k \sim \pi(\mathbf{u}_k|\mathbf{x}_k)$ where the randomness in the policy can also enable exploration of the policy space while also providing optimality for stochastic control problems.

The MDP requires that $\mathbf{x}_k$ be a Markov state that describes the state of the system. This assumption implies that the future states depend only on the current state and not on the events that occurred before it (otherwise known as the Markov property). For the orbit estimation problem, the state information needed is captured by the pdf for each satellite. If we assume that the pdfs are Gaussian, this implies that we only need the mean and covariance for each SO. Therefore, this work uses the Gaussian assumption to define the pdf for each SO. Relaxing this assumption will be considered for future work. Given the means, $\boldsymbol{\mu}_k^i$, and the covariances, P_k^i, for each SO the state of the MDP can be described. Moreover, since the SSA SM problem is nonlinear, this work utilizes the UKF to account for nonlinearities in propagating the mean and covariance. The dimensionality of this state is $n = N^2 + N$, where the covariance and mean contributes N^2 and N parameters, respectively. To further reduce the dimension of the state vector, this work will also consider the assumption that the covariance matrix can be captured by the diagonal elements or the variance for each dimension of the SO state vector. Under this assumption the number of state parameters is given by $n = 2N$ and this state at time k can be written as

$$\mathbf{x_k} = \left[\boldsymbol{\mu}_k^1, \cdots \boldsymbol{\mu}_k^N, \quad \text{diag}\left\{P_k^1\right\}, \cdots, \text{diag}\left\{P_k^N\right\}\right]^T \tag{1}$$

We can now parametrize the policy using a deep neural network. The parameters of the neural network which include weights and basis are defined by the vector $\boldsymbol{\theta}$. Therefore, with this parameterization of policy, we can now optimize over policies by optimizing over the parameters $\boldsymbol{\theta}$. An agent (telescope system) has a current state $\mathbf{x}_k \in S$ (the information state of the catalog [33]) at each discrete time step k and chooses an action $\mathbf{u}_k \in U$ according to a policy parameterized policy $\pi_{\boldsymbol{\theta}}$. For the policy $\pi_{\boldsymbol{\theta}}$, a reward signal $r_k(\mathbf{x}_k, \mathbf{u}_k)$ is given for a transition to a new state $\mathbf{x}_{k+1}$. A critical element of the RL problem definition involves the specification of the reward function. The reward is defined to provide the agent positive feedback for achieving the objective and negative feedback for actions that do not achieve the desired objective. A simple, yet useful reward model is to give the agent -1 reward for time steps where the goal is not achieved and $+1$ reward once the desired goal is achieved. The general objective of RL is to maximize an expectation over the discounted return, $J(\boldsymbol{\theta})$, given as:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[r_k + \gamma r_{k+1} + \gamma^2 r_{k+2} + \cdots\right] \tag{2}$$

where $\gamma \in [0, 1)$ is a discount factor and the notation $r_k = r_k(\mathbf{x}_k, \mathbf{u}_k)$ is used. Q-learning is a popular RL method which defines a Q-function that represents the total reward or the total "cost" to go for a policy π [41]. Once the Q-function is determined, the action with the highest value or estimated total reward is taken at each time step. Therefore, the policy can be solved using the Q-function. Here the agent is the telescope network and the possible actions are making observations of a particular SO. The Q-function of a policy π is:

$$Q^\pi(\mathbf{x}_k, \mathbf{u}_k) = \mathbb{E}_\pi\left[\sum_{i=k}^{\infty} \gamma^{i-k} r_i\right] \tag{3}$$

where the function estimates the total discounted reward for policy π from state $\mathbf{x}_k$ assuming that action $\mathbf{u}_k$ is taken and then all following actions are sampled from policy π. Q-network method uses neural networks parameterized by $\boldsymbol{\theta}$ to represent $Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k; \boldsymbol{\theta})$, but we drop the dependency notation for simplicity [41]. Q-networks are optimized by minimizing the following loss function at each iteration i:

$$\mathcal{L}(\boldsymbol{\theta}) = \left(r_k + \gamma \max_{\mathbf{u}_{k+1}} Q^{\pi^*}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) - Q^{\pi^*}(\mathbf{x}_k, \mathbf{u}_k) \right)^2 \tag{4}$$

This equation uses the Bellman optimality condition [41] to relate $Q^{\pi^*}(\mathbf{x}_k, \mathbf{u}_k)$ to $Q^{\pi^*}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})$, and this equation can be optimized using the stochastic gradient descent approach [42]. Note that this relationship is for the optimal policy π^* and therefore the value function learned from the Bellman equation can be used to determine the optimal policy by taking the action that maximizing $Q^{\pi^*}(\mathbf{x}_k, \mathbf{u}_k)$.

Although, the SSA SM problem is high-dimensional, all satellites share the sample dynamical environment and there are a low number of distinct orbital regimes (LEO, GEO, HEO, MEO, GTO, etc.). Therefore, a low dimensional representation of the problem should be possible. The challenge lies in finding such a representation. This work uses a DNN based solution to learn a low dimensional representation [43] of the SM problem from interactions with the simulated environment.

3 State Dynamics

The dynamics of the mean and covariance can be solved for using the Extended Kalman Filter (EKF) equations for propagation and update. The forecast step of in the EKF is given by

$$\dot{\boldsymbol{\mu}}_k^i = \mathbf{f}(\boldsymbol{\mu}_k^i, t) \tag{5}$$

$$\dot{P}_k^i = F(\boldsymbol{\mu}_k^i, t) P_k^i + P_k^i F^T(\boldsymbol{\mu}_k^i, t) + G(t) Q(t) G^T(t) \tag{6}$$

where the function $\mathbf{f}(\boldsymbol{\mu}_k^i, t)$ is a nonlinear dynamics function. The term $G(t)Q(t)G^T(t)$ represents the process noise which is neglected in this work. The propagation of the covariance matrix is done using the linearized dynamics where the matrix $F(\boldsymbol{\mu}_k^i, t)$ can be written as

$$F(\boldsymbol{\mu}_k^i, t) = \left. \frac{\partial \mathbf{f}}{\partial \boldsymbol{\mu}} \right|_{\boldsymbol{\mu}_k^i} \tag{7}$$

where the covariance propagation is done linearly and the pdfs for each SO is assumed to be Gaussian. Under the Gaussian assumption the action $\mathbf{u}_k$, which represents taking a measurement of a particular SO (denoted by i) results in a change in the mean and covariance for the i^{th} SO given by

$$\boldsymbol{\mu}_k^i = \boldsymbol{\mu}_k^i + K_k \left[\tilde{\mathbf{y}}^i - \mathbf{h}(\boldsymbol{\mu}_k^i, t) \right] \tag{8}$$

$$K_k = P_k H_k^T(\boldsymbol{\mu}_k^i, t) \left[H_k(\boldsymbol{\mu}_k^i, t) P_k H_k^T(\boldsymbol{\mu}_k^i, t) + R_k \right]^{-1} \tag{9}$$

$$H_k(\boldsymbol{\mu}_k^i, t) = \left. \frac{\partial \mathbf{h}}{\partial \boldsymbol{\mu}} \right|_{\boldsymbol{\mu}_k^i} \tag{10}$$

where $\tilde{\mathbf{y}}^i \in \mathcal{R}^m$ is the observation vector provided by the sensor when making an observation of the i^{th} SO. The matrix R_k represents the measurement error covariance matrix. The function $\mathbf{h}(\boldsymbol{\mu}_k^i, t)$ is a nonlinear observation function and the measurement update uses a linearized version of this

function to update the covariance matrix under the Gaussian assumption. The reward function used for this work is given by

$$r_k(\mathbf{x}_k, \mathbf{u}_k) = \begin{cases} -1 & \text{if} \quad \sigma > \bar{\sigma} \\ 1 & \text{if} \quad \sigma < \bar{\sigma} \end{cases} \tag{11}$$

where $\sigma = \max_i \left[\frac{1}{2} \text{Trace} \left\{ \sqrt{P_k^i [1:2, 1:2]} \right\} \right]$ ($\frac{1}{2}$ term is used for the simple planar case studied in this work) and $\bar{\sigma}$ is defined as the catalog accuracy threshold. There are many options for the reward function, some are shown in Figure 1. Future work will consider these alternative options.

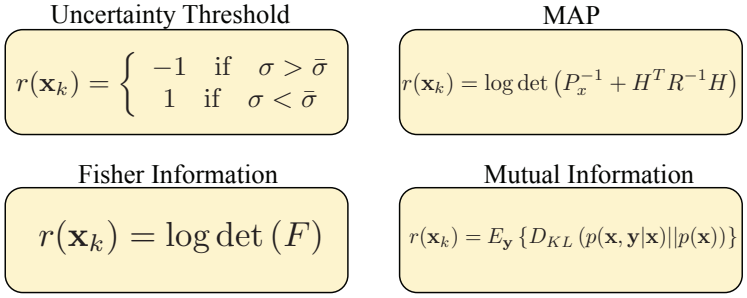


Figure 1: **Possible Tasking Metrics for SSA SM Problem.**

4 Asynchronous Actor-Critic Method

In general, RL approaches can be divided into two main classes, value-based and policy-based methods. Value-based methods estimate the value function and extract the policy from the value function, such as in Q-learning. These methods have poor convergence properties since small changes in the value function can have drastic changes in the policy. Alternatively, policy-based methods optimize the policy directly and therefore have smoother convergence and are easier to analyze. However, policy-based methods can converge to local maxima, are sample inefficient, and may suffer from having high variance estimates of the gradient. Actor-critic methods combine the benefits of both value-based and policy-based methods by using information from an estimated value function while optimizing the policy directly [44].

This work considers a parameterization of the policy by $\boldsymbol{\theta} \in \mathcal{R}^{d_{\boldsymbol{\theta}}}$. These parameters are learned from trajectories sampled from a nominal policy and from a reward function $r_k = r(\mathbf{x}_k, \mathbf{u}_k)$. A sequence of state and actions form a trajectory $\boldsymbol{\tau} = [\mathbf{x}_{0:T}, \mathbf{u}_{0:T}]$, where T is a time horizon which can be finite or infinite. The probability distribution for a trajectory is given by

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = \left[\prod_{i=0}^{T} p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k) \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) \right] p(\mathbf{x}_0) \tag{12}$$

Then the goal of policy learning process is to maximize the reward by finding a policy denoted by the parameters $\boldsymbol{\theta}$ that maximizes:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} \left[\sum_{k=0}^{T} \gamma^k r_k \right] \tag{13}$$

The reward for a trajectory can be expressed as $r(\boldsymbol{\tau}) = \sum_{k=0}^{T} \gamma^k r_k$. Then the expectation in Eq (13) can be written as

$$J(\boldsymbol{\theta}) = \int_{\mathbb{T}} r(\boldsymbol{\tau}) p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) d\boldsymbol{\tau} \tag{14}$$

Note from the equation above that $r(\boldsymbol{\tau})$ weights trajectories but is not a function of the policy parameters $\boldsymbol{\theta}$, i.e. the reward structure for the environment is independent of the policy used. The policy gradient approach updates the policy parameters based on $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ which requires gradients of the probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$. The gradients can be calculated using the REINFORCE trick [45] which uses the following relationship

$$\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \tag{15}$$

$$= p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \sum_{k=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k | \mathbf{x}_k) \tag{16}$$

Importantly, this derivative can be computed without knowledge of the probability distribution for a trajectory Eq. (12). Using the probability distribution for a trajectory Eq. (12) the gradient is given by $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = \sum_{k=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k | \mathbf{x}_k)$. Note the derivatives of $p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_k)$ do not have to be computed and no model needs to be maintained since the policy is nondeterministic. If a deterministic policy is used, computing $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$ would require the derivative $\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_k) = \nabla_{\mathbf{u}_k} \log p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_k) \nabla_{\boldsymbol{\theta}} \pi(\mathbf{u}_k)$ to compute $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$ and, hence, it would require a system model.

Then with the REINFORCE trick the gradient of the expected reward with respect to the policy is given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \sum_{k=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k | \mathbf{x}_k) d\boldsymbol{\tau} \tag{17}$$

$$= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} \left[\sum_{k=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k | \mathbf{x}_k) r(\boldsymbol{\tau}) \right] \tag{18}$$

As the expectation $\mathbb{E}\{\cdot\}$ can be replaced by sample averages only the derivative $\sum_{k=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k | \mathbf{x}_k)$ is needed for determining the gradient. Note that this derivative can be calculated for each term in the sum independently.

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} \left[\sum_{k=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k | \mathbf{x}_k) r(\boldsymbol{\tau}) \right] \tag{19}$$

The policy gradient theorem [44] generalizes the likelihood ratio approach to multi-step MDPs and allows the total reward $r(\boldsymbol{\tau})$ to be replaced with long-term value $Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k)$ given the policy π. Then the policy gradient from Eq. (17) can be rewritten as:

$$\nabla_{\theta} J(\boldsymbol{\theta}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k | \mathbf{x}_k) Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k) \right] \tag{20}$$

The actor-critic method uses a critic to estimate the action-value function, $Q(\mathbf{x}_k, \mathbf{u}_k) \approx Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k)$, where the critic's neural network introduces a new set of parameters $\mathbf{w} \in \mathcal{R}^{d_{\mathbf{w}}}$. Therefore, in the actor-critic method there are two sets of parameters that are determined, the policy parameters,

$\boldsymbol{\theta}$, and the parameters for critic value function, $\mathbf{w}$. Actor-critic algorithms follow an approximate policy gradient

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k) \right] \tag{21}$$

$$\Delta\boldsymbol{\theta} \approx \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k) \tag{22}$$

The critic is updated using the Bellman equation [41] converted into a loss function over $\mathbf{w}$ which can be optimized using stochastic gradient descent. The loss function is given by

$$L(\mathbf{w}) = \left(Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k) - (r(\mathbf{x}_k) + \gamma Q^{\pi}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})) \right)^2 \tag{23}$$

The equation above is used to calculate a gradient for the parameters $\mathbf{w}$ which is used to update the parameters given training examples. Then the parameters $\boldsymbol{\theta}$ and $\mathbf{w}$ are updated using Eq. (22) and Eq. (37), respectively. The update rule for these parameters for the actor-critic method is given by

$$\mathbf{w}^{+} = \mathbf{w}^{-} - \beta_{\mathbf{w}} \nabla_{\mathbf{w}} \, L(\mathbf{w})|_{\mathbf{w}=\mathbf{w}^{-}} \tag{24}$$

$$\boldsymbol{\theta}^{+} = \boldsymbol{\theta}^{-} + \beta_{\boldsymbol{\theta}} Q^{\mathbf{w}}(\mathbf{x}_k, \mathbf{u}_k) \, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k)|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{-}} \tag{25}$$

where $\beta_{\boldsymbol{\theta}}$ and $\beta_{\mathbf{w}}$ are the learning rates for policy and critic parameters, respectively. Both $Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k)$ and $\pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k)$ are modeled as neural networks where their respective parameters are the neural networks parameters.

This work uses a modification of the traditional actor-critic method to provide reduced variance estimates of the policy gradient. It can be shown that subtracting a state-dependent bias from the critic's value estimate can reduce the variance in estimating the gradient while still producing an unbiased estimate [44]. Therefore the following gradient calculation should have lower variance then that of Eq. (17):

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} \left[\sum_{k=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) (r(\boldsymbol{\tau}) - b(\mathbf{x}_k)) \right] \tag{26}$$

Typically the state-dependent function, $b(\mathbf{x}_k)$, used for reducing the variance is chosen to be the state-value function, $V(\mathbf{x}_k)$, which can be related to the state-action value function via the following relationship:

$$Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k) = r_k + \gamma V^{\pi}(\mathbf{x}_{k+1}) \tag{27}$$

Then the function $A^{\pi}(\mathbf{x}_k, \mathbf{u}_k) = Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k) - V^{\pi}(\mathbf{x}_k)$, where $A^{\pi}(\mathbf{x}_k, \mathbf{u}_k)$ is called the advantage function, can be used in place of $r(\boldsymbol{\tau}) - b(\mathbf{x}_k)$ providing a reduced variance estimate of the gradient. The advantage function measures how much better a particular action is then the action specified by π. Additionally, using Eq. (28) the advantage function can be expressed in terms of $V^{\pi}(\mathbf{x}_k)$ only using the following relationship:

$$A^{\pi}(\mathbf{x}_k, \mathbf{u}_k) = r_k + \gamma V^{\pi}(\mathbf{x}_{k+1}) - V^{\pi}(\mathbf{x}_k) \tag{28}$$

Note that the terms r_k, $\mathbf{x}_k$, and $\mathbf{x}_{k+1}$ are observed during one transition step of the model. Then the Advantage Actor-Critic (A2C) method provides a reduced variance estimate of the policy gradient with the following gradient expression

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} \left[\sum_{k=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) A^{\pi}(\mathbf{x}_k, \mathbf{u}_k) \right] \tag{29}$$

where now the critic's neural network is the state-value function and this function introduces a new set of parameters $\mathbf{v} \in \mathcal{R}^{d_{\mathbf{v}}}$. Note that to calculate the advantage function Eq. (28) can be used which only requires an approximation of $V^{\pi}(\mathbf{x}_k)$ and observing the reward during transitions. The Bellman equation for the state-value function can be converted into a loss function over $\mathbf{v}$ which can be optimized using stochastic gradient descent. The loss function is given by

$$L(\mathbf{v}) = \left(V^{\pi}(\mathbf{x}_k) - (r_k + \gamma V^{\pi}(\mathbf{x}_{k+1}))\right)^2 \tag{30}$$

Additionally, to promote exploration for the policy and for avoiding local minima, an entropy term is added to the policy loss function. The gradient described in Eq. (31) can be converted into a loss function and an entropy term can be added, the following is the A2C policy loss with an entropy term:

$$L(\boldsymbol{\theta}) = \sum_{k=0}^{T} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) A^{\pi}(\mathbf{x}_k, \mathbf{u}_k) - \beta_H \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) \tag{31}$$

where β_H is the entropy cost tuning parameter, higher values of β_H promote policy with more randomness and therefore more exploration. Then the update rule of the A2C method is given by

$$\mathbf{v}^{+} = \mathbf{v}^{-} - \beta_{\mathbf{v}} \, \nabla_{\mathbf{v}} L(\mathbf{v})|_{\mathbf{v}=\mathbf{v}^{-}} \tag{32}$$

$$\boldsymbol{\theta}^{+} = \boldsymbol{\theta}^{-} + \beta_{\boldsymbol{\theta}} \, \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{-}} \tag{33}$$

Both $V^{\pi}(\mathbf{x}_k)$ and $\pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k)$ are then approximated using a neural network. The current policy $\boldsymbol{\theta}^{-}$ is evaluated and the parameters for $V^{\pi}(\mathbf{x}_k)$ and $\pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k)$ are updated using transition data $\mathbf{x}_k$, $\mathbf{x}_{k+1}$, and r_k. Both parameter sets, $\boldsymbol{\theta}$ and $\mathbf{v}$ are updated after each roll-out of the policy.

Finally, the A2C approach is made asynchronous by freezing the policy during training. This process is highlighted in Figure 2 which was taken from Ref. 46. From the figure it can be seen that a master model is kept and distributed to each agent or each CPU. Then the master model is held fix while each agent samples the environment. Each agent accumulates gradient updates but the master model is not updated until all agents are finished with sampling. Then once all agents are finished, the master model updates are computed by adding up all of the gradients computed by each agent. Once the master model is updated the agents all sync to the new model and start the gradient accumulation process again. This is the process behind the A3C method and used for parallelizing the training for this work.

5 Policy and Value Function Neural Networks

As is routinely done with large reinforcement learning problems, the policy and value functions for this work are both represented using deep neural networks. The neural network for the policy used here can be expressed in the following functional form:

$$\pi(\mathbf{u}_k|\mathbf{x}_k) = \text{soft}\left(W_4^a \mathbf{f}\left(W_3^a \mathbf{f}\left(W_2^a \mathbf{f}\left(W_1^a \mathbf{x} + \mathbf{b}_1^a\right) + \mathbf{b}_2^a\right) + \mathbf{b}_3^a\right) + \mathbf{b}_4^a\right) \tag{34}$$

where $W_1^a \in \mathcal{R}^{h_1,n}$, $b_1^a \in \mathcal{R}^{h_1,1}$, $W_2^a \in \mathcal{R}^{h_2,h_1}$, $b_2^a \in \mathcal{R}^{h_2,1}$, $W_3^a \in \mathcal{R}^{h_3,h_2}$, $b_3^a \in \mathcal{R}^{h_3,1}$, $W_4^a \in \mathcal{R}^{\ell,h_3}$, and $b_4^a \in \mathcal{R}^{\ell,1}$, $\mathbf{f}$ is the activation function, and the function soft $(\cdot)$ is the softmax function. Note that the parameters h_1, h_2, and h_3 are the number of hidden units for the first, second, and third layers, respectively. The number of hidden units is used to increase the dimensionality of the policy model. However, increasing the number of hidden units also makes the learning model computationally

and date expensive, and therefore this is a tuning parameter. The softmax function is used to output values that are normalize to be positive to sum to 1 and the softmax can be written by

$$\text{soft}\,(\mathbf{a}) = \frac{\exp(\mathbf{a})}{\sum_i \exp(\mathbf{a}(i))} \tag{35}$$

The activation function used for this work is the rectified linear unit (ReLU). The ReLU function is given by $\mathbf{f}(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x})$, where it is zero for negative input values and linear for positive input values. The state-value function neural network has the same form as the policy except for removal of the softmax output layer and is given by

$$V^{\pi}(\mathbf{x}_k) = W_4^v \mathbf{f}\left(W_3^v \mathbf{f}\left(W_2^v \mathbf{f}\left(W_1^v \mathbf{x} + \mathbf{b}_1^v\right) + \mathbf{b}_2^v\right) + \mathbf{b}_3^v\right) + \mathbf{b}_4^v \tag{36}$$

where $W_1^v \in \mathcal{R}^{h_1,n}$, $b_1^v \in \mathcal{R}^{h_1,1}$, $W_2^v \in \mathcal{R}^{h_2,h_1}$, $b_2^v \in \mathcal{R}^{h_2,1}$, $W_3^v \in \mathcal{R}^{h_3,h_2}$, $b_3^v \in \mathcal{R}^{h_3,1}$, $W_4^v \in \mathcal{R}^{1,h_3}$, and $b_4^v \in \mathcal{R}^{1,1}$. Note that the state-value function neural network also has the same size as the policy neural network except for the output layer which outputs to a scalar value. From Eqs. (34) and (36) the number of parameters required for both networks is given by

$$d_{\boldsymbol{\theta}} = \ell \cdot (h_3 + 1) + h_3 \cdot (h_2 + 1) + h_2 \cdot (h_1 + 1) + h1 \cdot (n + 1) \tag{37}$$

$$d_{\mathbf{v}} = (h_3 + 1) + h_3 \cdot (h_2 + 1) + h_2 \cdot (h_1 + 1) + h1 \cdot (n + 1) \tag{38}$$

where $d_{\boldsymbol{\theta}}$ and $d_{\mathbf{v}}$ are the number of parameters in the policy and value neural networks, respectively. The equation above calculates the number of parameters for the weight matrices and biases needed for the network architecture given in Eqs. (34) and (36). Note that the state parameters are given by $n = 2N$ and the number of action equals the number of SOs, $\ell = N_{\text{SO}}$.

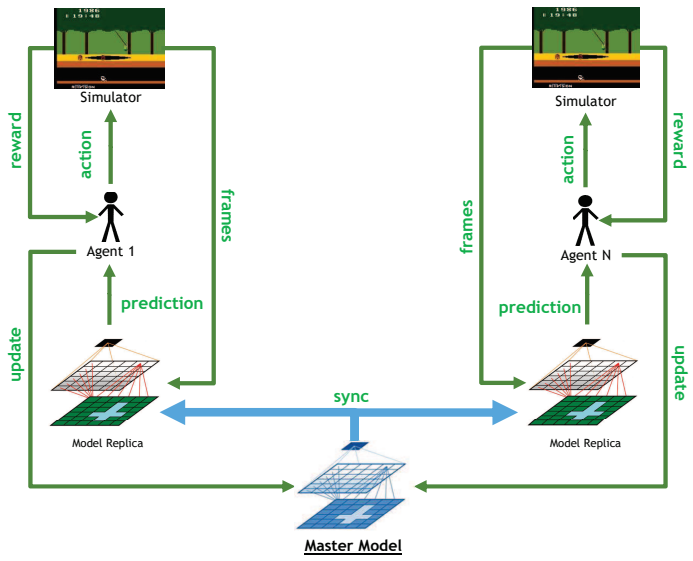


Figure 2: **Asynchronous Advantage Actor-Critic (A3C) Method (this figure was taken from reference 46).**

6 Simulation Models for Space Object Tracking

In this section, the numerical simulation for SO tracking is discussed. To show the effectiveness of the proposed ideas, we consider a population of near GEO SOs for our training set. The planar

equations of motion for a GEO objects assuming only 2 body forces are given by [47]

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} \tag{39}$$

where μ is the Earth's gravitational constant, $r = \|\mathbf{r}\|$, and the position vector is given by $\mathbf{r} = [x \quad y]^T$ and are assumed to be in inertial coordinates. The satellites used for the simulation are sampled randomly using orbital elements given by a, e, ω, and M, which are the semi-major axis, eccentricity, inclination, and mean anomaly, respectively. Note that only four orbital elements are needed to describe an orbit in the x-y plane. Then the initial orbital element state vector is given by $\mathbf{oe} = [a, \quad e, \quad i, \quad M]^T$. Random initial orbital element state vectors are sampled from a normal distribution for a, ω, and M and uniform distribution for e. The mean and standard deviation parameters for the normal distributions are given by $\mu_a = 42164$ km, $\sigma_a = 100$ km, $\mu_\omega = 0$ Deg, $\sigma_\omega = 90$ Deg, $\mu_M = 0$ Deg, and $\sigma_M = 90$ Deg. Here μ and σ variables denote mean and standard deviation for the orbital element samples, respectively. The eccentricity is sampled from a uniform distribution over the range $e \in [0.01, \quad 0.02]$. The initial orbital elements are then converted into an initial $\mathbf{r}$ and $\mathbf{v}$ and simulated for the policy training time window. This work uses an angle measurements and the angle observations are denoted by $\tilde{y} = \phi$ and $m = 1$. The observation model is given by

$$\phi = \text{atan2}\left(u_y, u_x\right) \tag{40}$$

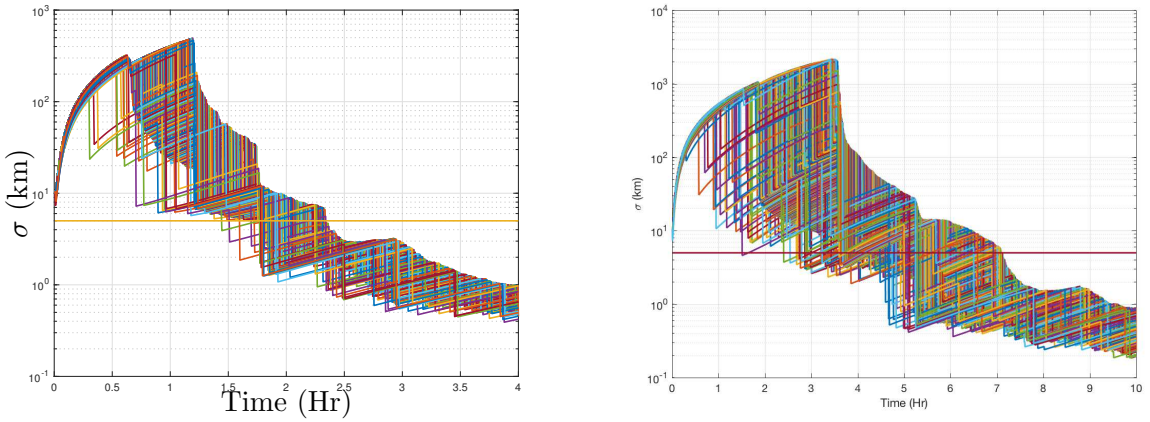
where $\mathbf{u} = [u_x, \quad u_y]^T$ denotes the position of the SO relative to the observer in inertial coordinates. The observer location is denoted by $\mathbf{r_{obs}}$ and the relative position is given by $\mathbf{u} = \mathbf{r} - \mathbf{r}_{obs}$. It is assumed the observations are corrupted with zero-mean white noise process with variance denoted by $\sigma_\phi^2 = 1$ arc-sec^2.

7 Simulation Results

This section discusses the initial proof-of-concept results for using the Asynchronous Advantage Actor-Critic (A3C) method applied to SO tracking. Python and Tensorflow [48] are used as the simulation environment for this work. All simulations were run on a Intel Xeon E5-2680 V4 2.4GHz (14 cores) processor with 64 GB of memory. Two simulation cases are shown, the first case uses 100 SOs and the second case uses 300 SOs, these results differ from Ref. 40 in the RL method used and in size of problem studied.

The state dynamics is captured in the evolution of the mean and covariance for each SO which is done using an Extended Kalman Filter framework shown in Section 3. As the policy takes actions and chooses to observe a particular satellite, Eq. (8) is used to update the covariance. When measurements are made on an object the covariance is reduced and when no observation is made the covariance grows due to the two body dynamics discussed in Section 6. Therefore the evolution of the covariance captures the essentials of the SSA SM problem. This work holds the policy parameters constant over the simulation window, which is the time period considered for tasking the sensor. This work collects 10 roll-outs before updating the policy and value parameters. This is done to improve the estimates of the gradients and reduce the "noise" in both gradient calculations. Then with 10 roll-outs collected the A3C method is then used to update the state-value function parameters and this function is used to calculate the policy gradient update. The learning rates used for the policy and value networks where $\beta_{\mathbf{v}} = 5 \cdot 10^{-3}$ and $\beta_{\boldsymbol{\theta}} = 1 \cdot 10^{-3}$, respectively. The value of the policy entropy loss term used for this work was $\beta_H = 1000$, this value gave the best results and promoted exploration for the policy. The $\bar{\sigma}$ used for training is 5 km in average position uncertainty. The measurement uncertainty used is $\sigma_\phi = 1$ arc-sec.

Two simulation cases are considered in this work, one with just 100 SOs and one with 300 SOs, where Ref. 40 considered only 3 SOs and 30 SOs. The first case with 100 SOs is the "lower" dimensional case but it still has 800 states. The 300 SO example has fairly high dimensionality with 2,400 states. The simulation results for both cases are shown in Figure 3. Figures 3(a) and 3(b) shows the results for the 100 and 300 SO cases, respectively. Both cases are trained for 10,000 update steps where the policy is rolled-out 10 times for each update. The SOs are given an initial uncertainty of 100 km in position coordinates and 0.1 km/s in velocity for both cases and the state vectors for the SOs are sampled from the distributions discussed in Section 6. For both cases the time between observations is consider to be 30 sec. The 100 SO case consider tasking a sensor for 2 Hrs and the 300 SO case consider tasking a sensor for 20 Hr. The 300 SO case is a bit unrealistic because optical sensors usually don't make day time observations and therefore are limited to only 8 Hrs of continuous tasking. Future work will consider multi-night scenarios. The 100 SO case



(a) 100 SO Example.

(b) 300 SO Example.

Figure 3: **Initial Simulation Results for the Asynchronous Advantage Actor-Critic (A3C) Method.**

was shown to converge well. However, this work showed very initial results and therefore, it is expected that if the learning rate are tuned better or even scheduled, that critic loss can be further reduced. From Figures 3(a) and 3(b) it can be seen that it is more difficult for the policy method to reduce the covariance for the 300 SO case and from Figure 3(b) it can be seen that some SO covariance matrices are not reduced right away. Although some SOs have covariance matrices that grow initially, overtime the tasking method is able to reduce their covariance values below the $\bar{\sigma}$ value. This highlights the fact that there are more SOs than the policy can maintain with one sensor.

Additionally, using the A3C method allowed for the problem to be scaled up and a larger number of satellites used in the simulation over the policy gradient method discussed in Ref. 40. However, this work did not push the upper limit of the number of satellites. From the simulation studies shown here it was determine that simulating the environment was the slowest part of the process. The code for the environment was not optimized for speed and therefore, significant gains can be made here. For example, this work used general perturbation theory but special perturbation theory [47] is much faster and may have enough accuracy to model the key features in the SSA SM problem. The A3C method was found to stabilize the learning process, in that the policy more reliably converged to a good policy. Additionally, the A3C method can handle larger problems since it allows for parallel computations. Although simulation results shown in this section are very

preliminary, a training time speed-up was noticed for the A3C. The simulation scenarios were run on a relatively low number of cores (14 cores) and therefore a significant speed-up are expected when this A3C is run on High Performance Computing (HPC) resources.

8 Conclusions

This paper provides initial proof-of-concept results for solving the sensor tasking and Sensor Management (SM) problem for Space Situational Awareness (SSA) using the Asynchronous Advantage Actor-Critic (A3C) method. The A3C method is a Reinforcement Learning (RL) approach that can learn, from interacting with the environment, how to optimally make decisions. There are two general classes of RL approaches, policy-based and value-based approaches, the A3C method combines the benefits of both. Furthermore, the A3C method can learn asynchronously from a group of agents that are interacting with the environment. This work leverages the A3C method's ability for asynchronous learning to parallelize the method and therefore speed up learning. It was shown that initial proof-of-concept results are achievable for relatively high dimensional problems, an improvement over previous work. The A3C method has the beneficial properties of producing better estimates of the policy gradient, with less noise, which can improve convergence. This work showed initial results for two simulation cases, the first case used 100 SOs and the second case used 300 SOs. These results show that the A3C method can handle these relatively high dimensional examples and good performance for both cases was shown. However, the results need to be explored further. For example, the size of the neural network was not optimized and therefore improvements on the results shown here may be possible. Additionally, the larger of the two examples considered here used 300 SOs and the upper limit on size was not explored. Finally, the neural networks were only train for 10,000 steps and longer training times were not explored. These analysis along with comparisons to traditional SM problem solutions will be considered for future work.

References

[1] Linares, R., Jah, M. K., Crassidis, J. L., and Nebelecky, C. K., "Space object shape characterization and tracking using light curve and angles data," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 1, 2013, pp. 13–25.

[2] Wetterer, C. J., Linares, R., Crassidis, J. L., Kelecy, T. M., Ziebart, M. K., Jah, M. K., and Cefola, P. J., "Refining space object radiation pressure modeling with bidirectional reflectance distribution functions," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 1, 2013, pp. 185–196.

[3] Linares, R. and Crassidis, J. L., "Resident Space Object Shape Inversion via Adaptive Hamiltonian Markov Chain Monte Carlo," *AAS/AIAA Space Flight Mechanics Meeting*, pp. 2016–514.

[4] Linares, R., Jah, M. K., Crassidis, J. L., Leve, F. A., and Kelecy, T., "Astrometric and photometric data fusion for inactive space object mass and area estimation," *Acta Astronautica*, Vol. 99, 2014, pp. 1–15.

[5] Linares, R., Shoemaker, M., Walker, A., Mehta, P. M., Palmer, D. M., Thompson, D. C., Koller, J., and Crassidis, J. L., "Photometric Data from Non-Resolved Objects for Space Object Characterization and Improved Atmospheric Modeling," *Advanced Maui Optical and Space Surveillance Technologies Conference*, Vol. 1, 2013, p. 32.

[6] Linares, R., Crassidis, J. L., and Jah, M. K., "Space object classification and characterization via multiple model adaptive estimation," *Information Fusion (FUSION), 2014 17th International Conference on*, IEEE, 2014, pp. 1–7.

[7] Linares, R. and Furfaro, R., "Space Object classification using deep Convolutional Neural Networks," *Information Fusion (FUSION), 2016 19th International Conference on*, IEEE, 2016, pp. 1140–1146.

[8] Forsyth, D. A. and Ponce, J., "A modern approach," *Computer vision: a modern approach*, 2003, pp. 88–101.

[9] Hager, G. D., *Task-directed sensor fusion and planning: a computational approach*, Vol. 99, Springer Science & Business Media, 2012.

[10] Hager, G. and Mintz, M., "Computational methods for task-directed sensor data fusion and sensor planning," *The International Journal of Robotics Research*, Vol. 10, No. 4, 1991, pp. 285–313.

[11] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K., "Asynchronous methods for deep reinforcement learning," *International Conference on Machine Learning*, 2016, pp. 1928–1937.

[12] Cai, C. and Ferrari, S., "Information-driven sensor path planning by approximate cell decomposition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 39, No. 3, 2009, pp. 672–689.

[13] Chu, M., Haussecker, H., and Zhao, F., "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," *International Journal of High Performance Computing Applications*, Vol. 16, No. 3, 2002, pp. 293–313.

[14] Zhao, F., Shin, J., and Reich, J., "Information-driven dynamic sensor collaboration," *IEEE Signal processing magazine*, Vol. 19, No. 2, 2002, pp. 61–72.

[15] Vishwajeet, K., Singla, P., and Jah, M., "Nonlinear uncertainty propagation for perturbed two-body orbits," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 5, 2014, pp. 1415–1425.

[16] Grant, M., Boyd, S., and Ye, Y., "CVX: Matlab software for disciplined convex programming," 2008.

[17] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, Vol. 529, No. 7587, 2016, pp. 484–489.

[18] Kreucher, C., Kastella, K., and Hero III, A. O., "Information based sensor management for multitarget tracking," *Proc. of SPIE Vol*, Vol. 5204, 2003, p. 481.

[19] Kwok, C. and Fox, D., "Reinforcement learning for sensing strategies," *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, Vol. 4, IEEE, 2004, pp. 3158–3163.

[20] Crary, S. and Jeong, Y., "Bayesian optimal design of experiments for sensor calibration," *Solid-State Sensors and Actuators, 1995 and Eurosensors IX.. Transducers' 95. The 8th International Conference on*, Vol. 2, IEEE, 1995, pp. 48–51.

[21] Gupta, V., Chung, T. H., Hassibi, B., and Murray, R. M., "On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage," *Automatica*, Vol. 42, No. 2, 2006, pp. 251–260.

[22] Oshman, Y., "Optimal sensor selection strategy for discrete-time state estimators," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 30, No. 2, 1994, pp. 307–314.

[23] Grocholsky, B., *Information-Theoretic Control of Multiple Sensor Platforms*, Ph.D. thesis, The University of Sydney, 2002.

[24] Stachniss, C., *Exploration and mapping with mobile robots.*, Ph.D. thesis, University of Freiburg, 2006.

[25] Alpcan, T. and Shames, I., "An Information-based learning approach to dual control," *IEEE transactions on neural networks and learning systems*, Vol. 26, No. 11, 2015, pp. 2736–2748.

[26] Li, Y., Krakow, L., Chong, E., and Groom, K., "Approximate stochastic dynamic programming for sensor scheduling to track multiple targets," *Digital Signal Processing*, Vol. 19, No. 6, 2009, pp. 978 – 989.

[27] Lilith, N. and Doğançay, K., "Reinforcement learning-based dynamic scheduling for threat evaluation," *Signal Processing Conference, 2008 16th European*, IEEE, 2008, pp. 1–5.

[28] Miller, J. G., "A new sensor allocation algorithm for the Space Surveillance Network," *Military Operations Research*, Vol. 12, No. 1, 2007, pp. 57–70.

[29] Hill, K., Sydney, P., Hamada, K., Cortez, R., Luu, K., Jah, M., Schumacher, P., Coulman, M., Houchard, J., and Naho'olewa, D., "Covariance-based network tasking of optical sensors," *Paper AAS 10-150 presented at the AAS/AIAA Space Flight Mechanics Conference, February*, 2010, pp. 14–17.

[30] Williams, P. S., Spencer, D. B., and Erwin, R. S., "Coupling of estimation and sensor tasking applied to satellite tracking," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 4, 2013, pp. 993–1007.

[31] Hussein, I., Sunberg, Z., Chakravorty, S., Jah, M., and Erwin, R., "Stochastic Optimization for Sensor Allocation Using AEGIS-FISST," *AMOS conference*, 2014.

[32] Erwin, R. S., Albuquerque, P., Jayaweera, S. K., and Hussein, I., "Dynamic sensor tasking for space situational awareness," *Proceedings of the 2010 American Control Conference*, IEEE, 2010, pp. 1153–1158.

[33] Sunberg, Z., Chakravorty, S., and Erwin, R. S., "Information Space Receding Horizon Control for Multisensor Tasking Problems," *IEEE transactions on cybernetics*, Vol. 46, No. 6, 2016, pp. 1325–1336.

[34] Kohl, N. and Stone, P., "Policy gradient reinforcement learning for fast quadrupedal locomotion," *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, Vol. 3, IEEE, 2004, pp. 2619–2624.

[35] Endo, G., Morimoto, J., Matsubara, T., Nakanishi, J., and Cheng, G., "Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot," *The International Journal of Robotics Research*, Vol. 27, No. 2, 2008, pp. 213–228.

[36] Theodorou, E., Buchli, J., and Schaal, S., "Reinforcement learning of motor skills in high dimensions: A path integral approach," *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 2397–2403.

[37] Peters, J., Mulling, K., and Altun, Y., "Relative entropy policy search," *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[38] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al., "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[39] Gu, S., Holly, E., Lillicrap, T., and Levine, S., "Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates," *arXiv preprint arXiv:1610.00633*, 2016.

[40] Linares, R. and Furfaro, R., "Dynamic Sensor Tasking for Space Situational Awareness via Reinforcement Learning," *Advanced Maui Optical and Space Surveillance Technologies Conference*, 2016.

[41] Sutton, R. S. and Barto, A. G., *Reinforcement learning: An introduction*, Vol. 1, MIT press Cambridge, 1998.

[42] LeCun, Y., Bengio, Y., and Hinton, G., "Deep learning," *Nature*, Vol. 521, No. 7553, 2015, pp. 436–444.

[43] Hinton, G. E. and Salakhutdinov, R. R., "Reducing the dimensionality of data with neural networks," *science*, Vol. 313, No. 5786, 2006, pp. 504–507.

[44] Peters, J. and Schaal, S., "Reinforcement learning of motor skills with policy gradients," *Neural networks*, Vol. 21, No. 4, 2008, pp. 682–697.

[45] Williams, R. J., "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, Vol. 8, No. 3-4, 1992, pp. 229–256.

[46] Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J., and Kautz, J., "Reinforcement learning through asynchronous advantage actor-critic on a gpu," 2016.

[47] Vallado, D. A., *Fundamentals of Astrodynamics and Applications*, Microcosm, 2007.

[48] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv preprint arXiv:1603.04467*, 2016.