Dynamic Sensor Tasking for Space Situational Awareness via Reinforcement Learning

Richard Linares*
University at Minnesota, Minneapolis, MN, USA
Roberto Furfaro†
University of Arizona, Tucson, AZ, USA

Conference Paper

This paper studies the Sensor Management (SM) problem for optical Space Object (SO) tracking. The tasking problem is formulated as a Markov Decision Process (MDP) and solved using Reinforcement Learning (RL). The RL problem is solved using the actor-critic policy gradient approach. The actor provides a policy which is random over actions and given by a parametric probability density function (pdf). The critic evaluates the policy by calculating the estimated total reward or the value function for the problem. The parameters of the policy action pdf are optimized using gradients with respect to the reward function. Both the critic and the actor are modeled using deep neural networks (multi-layer neural networks).

The policy neural network takes the current state as input and outputs probabilities for each possible action. This policy is random, and can be evaluated by sampling random actions using the probabilities determined by the policy neural network's outputs. The critic approximates the total reward using a neural network. The estimated total reward is used to approximate the gradient of the policy network with respect to the network parameters. This approach is used to find the non-myopic optimal policy for tasking optical sensors to estimate SO orbits. The reward function is based on reducing the uncertainty for the overall catalog to below a user specified uncertainty threshold. This work uses a 30 km total position error for the uncertainty threshold.

This work provides the RL method with a negative reward as long as any SO has a total position error above the uncertainty threshold. This penalizes policies that take longer to achieve the desired accuracy. A positive reward is provided when all SOs are below the catalog uncertainty threshold. An optimal policy is sought that takes actions to achieve the desired catalog uncertainty in minimum time. This work trains the policy in simulation by letting it task a single sensor to "learn" from its performance. The proposed approach for the SM problem is tested in simulation and good performance is found using the actor-critic policy gradient method.

1 Introduction

The U.S. Air Force has maintained a catalog of Space Objects (SOs) since the dawn of the space age and the network of sensors that provides the data for this catalog is called the Space Surveillance Network (SSN) [1]. Over the past few decades, space technologies, and the satellites that support them, have become indispensable for modern economies. Economic and military drivers have led to large increases in the number of SOs and countries that have a presence in space. This growth in the number of SOs is stressing the current capability of the SSN and creating a need for new approaches for improving Space Situational Awareness (SSA). SSA has become a key mission area for the U.S. Air Force, which is tasked with collecting tracking data on over 22,000 SOs, 1,100 of which are active, currently being tracked. The solution to this task involves solving a large scale resource allocation and management problem. This paper studies the Sensor

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

Management (SM) problem for SSA, where the goal is to maintain knowledge over a given set of SOs using a limited number of sensing platforms.

The SM problem is a general challenge across many engineering applications and has been extensively studied. 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, information divergence, and information entropy [2]. 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 2 formulates the sensor scheduling problem as a Markov decision process (MDP) 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. 3, which used a Reinforcement Learning (RL)-based sensor scan optimization scheme for multi-target tracking. Ref. 3 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 policy gradient method.

Sensor tasking for SSA is usually based on priorities, such as estimating atmospheric reentry, 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 [4]. Using these Gabbard class based observation 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 closer to optimal tasking solutions [5–8]. These sensor scheduling methods are myopic, only optimal for a single time step into the future. Reference 5 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 9 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 5 and 9 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 [7] 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. [10], which used a Gaussian sum filter to represent the pdf of each object. Reference 11 was the first paper to study the non-myopic SM problem for SSA and was able to show a clear improvement over myopic approaches. Reference 11 solved this problem using information space receding horizon control and using stochastic optimization. This work was later extended to non-Gaussian SO pdfs [8]. Reference 8 applied the AEGIS-FISST approach to this problem by developing a tasking strategy that can include the data association process using Random Finite Sets.

Recent advancements in Deep Reinforcement Learning (DRL) [12,13] have demonstrated ground breaking results across a number of domains. In particular, DRL has been used to successfully develop an artificial intelligence approach that can defeat expert human GO players [13], an astonishing accomplishment. The generality of DRL and the ground breaking results, motivates the exploration of these approaches for SSA applications. Here the term "deep" refers to any neural network learning approach with more than one hidden layer. Deep learning approaches mimic the function of the brain by learning nonlinear hierarchical features from data that build in abstraction [14]. In an end-to-end fashion, DRL approaches can be used to process data directly to "learn" a control policy from training data.

This paper develops a DRL approach [15] for the SM problem applied to SSA. RL approaches define a value function, which represents the total reward for possible actions at the current state. For the SSA problem, the agent is the telescope sensor and the possible actions are the decisions to make observations of a particular satellite. This paper develops a simulation environment where a neural network can be trained to perform a policy that will reduce the overall satellite catalog error. The reward model development is

critical to the performance of the RL policy. This paper focuses on a reward function that rewards the policy's ability to maintain a given catalog accuracy. The primary model that is investigated is based on the trace of the SO covariance matrix, where the goal is to reduce the positional uncertainty of all SOs in the catalog to below a threshold within minimum time (or observations). This approach rewards policies that take measurements that maximally reduce the overall catalog uncertainty over time.

Stable methods for training RL approaches based on neural networks exist, but most of these approaches are not suitable for high dimensional systems. Reference 16 developed an effective approach for high dimensional systems and this paper leverages these results and applies this approach to decision making in SSA. The action 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 that are possible at a given time. Therefore, efficient RL approaches are required when solving the SM problem for SSA. This paper implements the policy gradient method [16] for RL applied to SSA sensor tasking.

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 policy gradient method is 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

Given a discrete-time system model, we can denote the state of the system at time step k by $\mathbf{x}_k$. The system dynamics provide 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 $\mathbf{x}_{k+1} \sim p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k)$ and $\mathbf{x}_k$, $\mathbf{x}_{k+1} \in \mathcal{R}^n$ denotes the current and next state, respectively. The actions are modeled probabilistically and are generated by a policy $\mathbf{u}_k \sim \pi(\mathbf{u}_k|\mathbf{x}_k)$ where the randomness in the policy can enable exploration of the policy space while also providing optimality for certain classes of control problems.

An agent (telescope system) has a current state $\mathbf{x}_k \in S$ (the information state of the catalog [11]) at each discrete time step k and chooses an action $\mathbf{u}_k \in U$ according to a policy π. For the policy π, a reward signal r_k is given for a transition to a new state $\mathbf{x}_{k+1}$. The general objective of RL is to maximize an expectation over the discounted return, $J(\boldsymbol{\theta})$, given as:

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

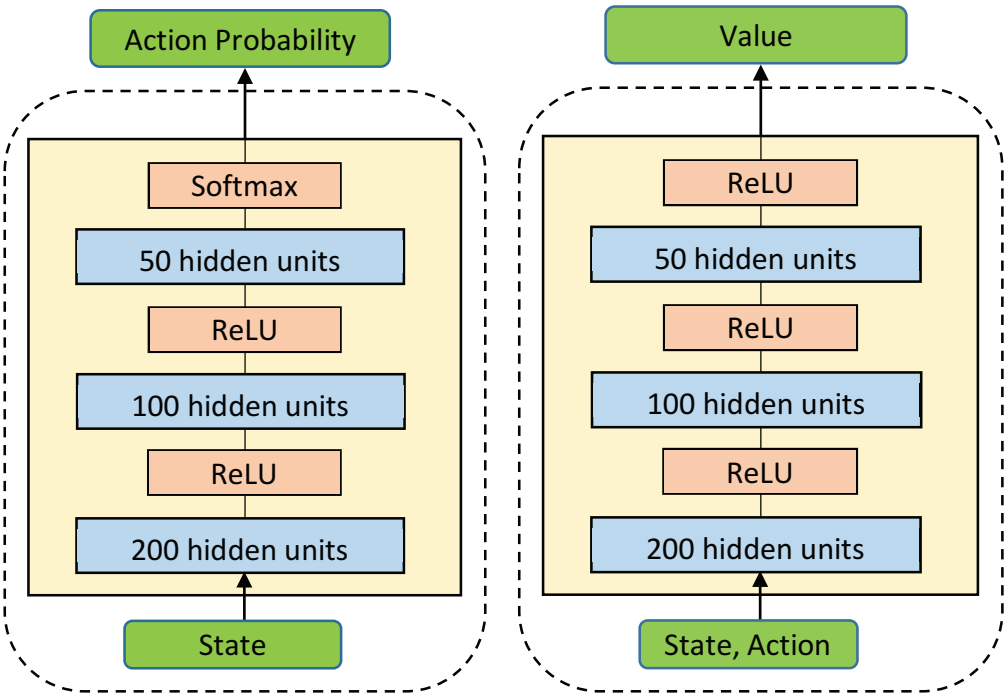
where $\gamma \in [0, 1)$ is a discount factor. Q-learning is a popular RL method which defines a Q-function that represents the total reward or the total "cost" to go for a policy π [15]. Once the Q-function is determined, the action with the highest value or estimated total reward is taken at each time step. Therefore, the policy can be solved for 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:

$$Q^\pi(\mathbf{x}_k, \mathbf{u}_k) = \mathbb{E}_\pi\left[\sum_{i=k}^{\infty} \gamma^{i-k} r_i\right] \tag{2}$$

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 [15]. Q-networks are optimized by minimizing the following loss function at each iteration i:

$$\mathcal{L}(\boldsymbol{\theta}) = \left(r_k + \gamma \max_{\mathbf{u}_{k+1}} Q^\pi(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) - Q^\pi(\mathbf{x}_k, \mathbf{u}_k)\right)^2 \tag{3}$$

This equation uses the Bellman optimality condition [15] 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 stochastic gradient descent. For the SSA problem this work develops a simulation environment where a RL approach is trained to perform a policy that will reduce the overall catalog uncertainty.



(a) Actor (b) Critic

Figure 1: Neural Network Architecture for the Actor and Critic Networks.

3 State Representation and Dynamics

The Markov decision process (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 we need is the pdf for each satellite and 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. Given the means, $\boldsymbol{\mu}_k^i$, and the covariances, P_k^i, for each SO the state of the MDP can be described. The dimensionality of this state is $n = N^2+N$, where the covariance and mean contributes N^2 and N parameters, respectively. To reduce the dimension of the state vector this work assumes 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

$$\mathbf{x_k} = \left[\boldsymbol{\mu}_k^1, \cdots \boldsymbol{\mu}_k^N, \quad \text{diag}\left\{P_k^1\right\}, \cdots, \text{diag}\left\{P_k^N\right\}\right]^T \tag{4}$$

This state is used in solving the MDP problem using policy gradient method. 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

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

where $\sigma = \max_i \left[\frac{1}{2}\text{Trace}\left\{\sqrt{P_k^i\left[1:2, 1:2\right]}\right\}\right]$ ($\frac{1}{2}$ term is used for the simple planar case studied in this work) and $\bar{\sigma}$ is defined as the catalog accuracy threshold.

4 Policy Gradient

This work considers a parameterization of the policy by $\boldsymbol{\theta} \in \mathcal{R}^d$. These parameters are learned from trajectories sampled from a nominal policy and from a reward function $r_k = r(\mathbf{x}_k, \mathbf{u}_k)$. Then the goal of policy learning process is to maximize the reward by finding a policy denoted by the parameters $\boldsymbol{\theta}$ that maximizes:

$$J(\boldsymbol{\theta}) = E\left[\sum_{i=0}^{\infty} \gamma^k r_k\right] \tag{12}$$

The probability distribution for a trajectory given by a sequence of states $\mathbf{x}_{0:k+1}$ is denoted by

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1}) = p(\mathbf{x}_0) \prod_{i=1}^{k+1} p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k) \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) \tag{13}$$

Then the expectation in Eq (12) can be written as

$$J(\boldsymbol{\theta}) = \int r(\mathbf{x}_{0:k+1}) p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1}) d\mathbf{x}_{0:k+1} \tag{14}$$

The policy gradient approach updates the policy parameters based on $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ which requires gradients of the probability distribution $p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1})$. The gradients can be calculated using the REINFORCE trick [16] which uses the following relationship

$$\nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1}) = p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1}) \tag{15}$$

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

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1}) r(\mathbf{x}_{0:k+1}) d\mathbf{x}_{0:k+1} \tag{16}$$

$$= \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1}) r(\mathbf{x}_{0:k+1})\right] \tag{17}$$

As the expectation $E\{\cdot\}$ can be replaced by sample averages only the derivative $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1})$ is needed for the determining the gradient. Importantly, this derivative can be computed without knowledge of the probability distribution for a trajectory Eq. (13). Using probability distribution for a trajectory Eq. (13) the gradient is given by

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1}) = \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) \tag{18}$$

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 $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1})$ 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 $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}_{0:k+1})$ and, hence, it would require a system model. The policy gradient theorem generalizes the likelihood ratio approach to multi-step MDPs and allows the total reward $r(\mathbf{x}_{0:k+1})$ 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. (16) can be rewritten as:

$$\nabla_{\theta} J(\boldsymbol{\theta}) = \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) Q^{\pi}(\mathbf{x}_k, \mathbf{u}_k)\right] \tag{19}$$

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

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

$$\Delta\boldsymbol{\theta} \approx \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) Q^{\mathbf{w}}(\mathbf{x}_k, \mathbf{u}_k) \tag{21}$$

The critic is updated using the standard Temporal Difference (TD) update which is used in standard Q-Learning approaches. The TD is calculated from $\mathbf{x}_k, \mathbf{x}_{k+1}, \mathbf{u}_k, \mathbf{u}_{k+1}$ and $r(\mathbf{x}_k)$ and given by

$$Q^{\mathbf{w}}(\mathbf{x}_k, \mathbf{u}_k) = Q^{\mathbf{w}}(\mathbf{x}_k, \mathbf{u}_k) + \alpha \left(r(\mathbf{x}_k) + \gamma \max_{\mathbf{u}_{k+1}} Q^{\mathbf{w}}(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})\right) \tag{22}$$

The above equation can be converted into a loss function over $\mathbf{w}$ which can be optimized using stochastic gradient descent. The loss function is given by

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

The equation above is used to calculate a gradient for the parameters $\mathbf{w}$ which is used to update the

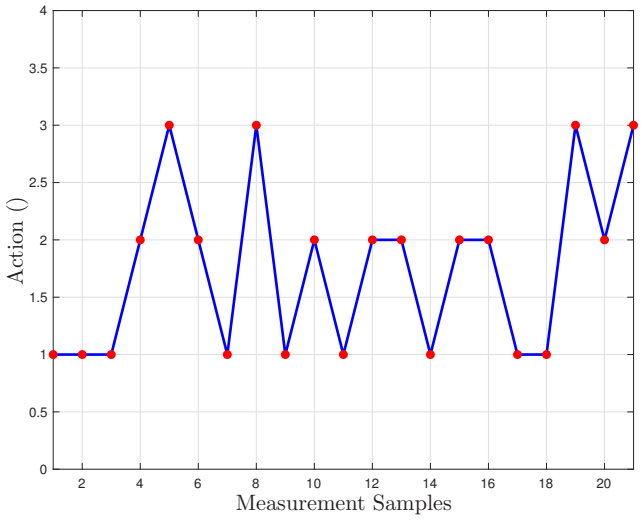


Figure 2: SO Observation Actions for 3 SO Example.

parameters given training examples. Then the parameters $\boldsymbol{\theta}$ and $\mathbf{w}$ are updated using Eq. (21) and Eq. (26), respectively. The update rule for these parameters is given by

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

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

where $\beta_{\boldsymbol{\theta}}$ and $\beta_{\mathbf{w}}$ are the learning rates for policy and critic parameters, respectively. Both $Q^{\mathbf{w}}(\mathbf{x}_k, \mathbf{u}_k)$ and $\pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k)$ are modeled as neural networks where their respective parameters are the neural networks parameters. The network architecture used for this work to represent both $Q^{\mathbf{w}}(\mathbf{x}_k, \mathbf{u}_k)$ and $\pi_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k)$ are shown in Figure 1. From Figure 1 the number of parameter required for both networks is given by

$$d_{\mathbf{w}} = d = 200 \cdot (1 + n) + 201 \cdot 100 + 50 \cdot 101 + 51 \cdot \ell \tag{26}$$

where this equation calculates the number of parameters for the weight matrices and biases needed for the network architecture shown in Figure 1. Note that the state parameters are given by $n = 2N$ and the number of action equals the number of SOs, $\ell = N_{\text{SO}}$.

5 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 [17]

$$\ddot{x} + \frac{\mu x}{r^3} = 0 \tag{27}$$

$$\ddot{y} + \frac{\mu y}{r^3} = 0 \tag{28}$$

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, $\sigma_a = 90$ Deg, $\mu_M = 0$ Deg, and $\sigma_M = 90$ Deg. Here μ and σ variables denote mean and standard deviation, 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

$$\phi = \text{atan2}\,(u_y, u_x) \tag{29a}$$

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 $\sigma_\phi^2 = 0.0085$ deg^2.

6 Simulation Results

This section discusses the initial proof of concept results for using policy gradient RL applied to SO tracking. Python and Tensorflow [18] are used as the simulation environment for this work. The policy gradient method is applied to the SO tracking problem by simulating the satellite orbits and observation dynamics. The observation dynamics is captured in the evolution of the mean and covariance for each SO. 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 5. The actor-critic method is then used to estimate the state and action value function online and this function is used to calculate the policy gradient. The policy gradient is then used to update the policy parameters.

Two simulation cases are considered in this work, one with just 3 SOs and one with 30 SOs. The first case with 3 SOs is a lower dimensional case with just 24 states. Figures 2 and 3 shows the results for the 3 SO case. Both cases run policy gradient method on 1000 experiments (Figure 3) where the SO are given an

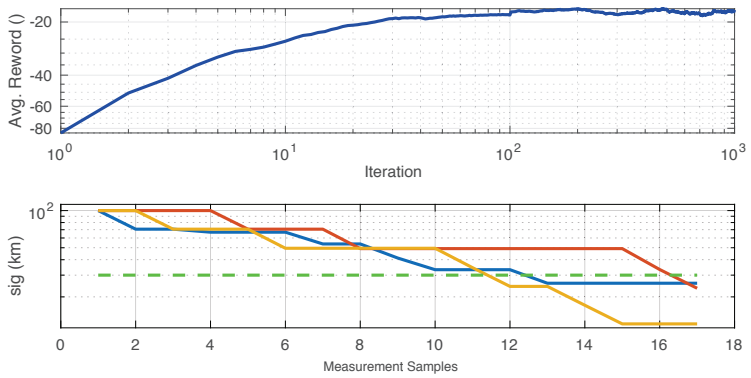


Figure 3: Average Position Error Variance vs Measurement Samples for 3 SO Example (Bottom) and Average Reward vs Training Iterations (Top).

initial uncertainty of 100 km in position coordinates and 0.1 km/s in velocity. Both experimental cases use 2000 time steps for data collection, where data is taken from one satellite every 30 secs. The second case uses a large number of SO and large time gaps between measurements tracks. For the second case 10 mins of continuous observations are simulated and a one hour gap is assumed between each 10 min window. During the 10 min window it is assumed that the sensors can point at the next satellite within 30 secs, which is reasonable for GEO. The time gaps here make this case a bit more realistic since in practice these gaps are expected.

After the 1000 experiments the policy is used to generate a tasking sequence and these are given in Figure 2 for the 3 SO case. Figure 2 shows the action performed by this policy and it is seen that the policy learns to spread out the measurements over all the SOs initially but after some period focuses mostly on two of the three SOs. The position covariance is summarized in Figure 3, where it can be seen that the policy initially reduces the covariance for all of the SO drastically and then maintains a steady state value after that.

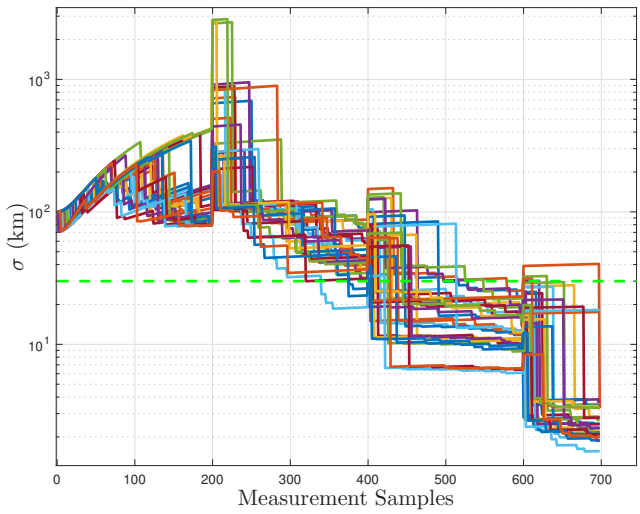


Figure 4: Average Position Error Variance vs Measurement Samples for 30 SO Example.

Figures 4 shows the results for the 30 SO case. This case is more difficult for the policy gradient method and from Figure 4 it can be seen that some SO covariance matrices are not reduced right away. Although some SOs have covariance matrices that growth 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 then the policy can

maintain with one sensor.

7 Conclusions

This paper provides initial results for solving the sensor tasking and management problem for Space Situational Awareness (SSA) using the policy gradient (PG) method. This work uses the PG method to solve a Markov decision process to determine the optimal tasking for maximizing some reward. The reward function used for this work is based on reducing the uncertainty for the overall catalog to below a user specified uncertainty threshold. The policy is "learned" from data using an actor-critic PG Reinforcement Learning (RL) method. A single sensor tasking system is simulated for multiple trials using a small catalog and the current policy. Then based on these simulations an update to the policy is calculated to maximize the reward. Two simulation cases are studied in this work, cases with 3 and 30 SOs, respectively. The first case was shown to have good performance using the PG method. The approach learned a policy that reduced the uncertainty for the three SO case rapidly, and then maintained it at a steady state value. The second case did not have as good of a performance since a larger number of SOs were considered, but overall it did well. For future work, we will consider the multi-sensor tasking problem and analyze the cases where not enough observations are available to observe all of the SOs. In these cases, the RL approach should make a decision on which SO should be focused on and which errors should be allowed to grow.

References

[1] Vallado, D. and Griesbach, J. D., "Simulating space surveillance networks," *Advances in the Astronautical Sciences*, Vol. 142, No. 11-580, 2011, pp. 2769–2787.

[2] 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.

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

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] Spencer, D. B. and Williams, P. S., "Managing Space Situational Awareness Using the Space Surveillance Network," Tech. rep., DTIC Document, 2013.

[11] 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.

[12] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., "Human-level control through deep reinforcement learning," *Nature*, Vol. 518, No. 7540, 2015, pp. 529–533.

[13] 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.

[14] Lee, H., Pham, P., Largman, Y., and Ng, A. Y., "Unsupervised feature learning for audio classification using convolutional deep belief networks," *Advances in neural information processing systems*, 2009, pp. 1096–1104.

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

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

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

[18] 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.