2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

# Towards Fully Autonomous Orbit Management for Low-Earth Orbit Satellites based on Neuro-Evolutionary Algorithms and Deep Reinforcement Learning

Alexander Kyuroson, Avijit Banerjee, Nektarios Aristeidis Tafanidis, Sumeet Satpute and George Nikolakopoulos

*Abstract*— **The recent advances in space technology are focusing on fully autonomous, real-time, long-term orbit management and mission planning for large-scale satellite constellations in Low-Earth Orbit (LEO). Thus, a pioneering approach for autonomous orbital station-keeping has been introduced using a model-free Deep Policy Gradient-based Reinforcement Learning (DPGRL) strategy explicitly tailored for LEO. Addressing the critical need for more efficient and self-regulating orbit management in LEO satellite constellations, this work explores the potential synergy between Deep Reinforcement Learning (DRL) and Neuro-Evolution of Augmenting Topology (NEAT) to optimize station-keeping strategies with the primary goal to empower satellite to autonomously maintain their orbit in the presence of external perturbations within an allowable tolerance margin, thereby significantly reducing operational costs while maintaining precise and consistent station-keeping throughout their life cycle. The study specifically tailors DPGRL algorithms for LEO satellites, considering low-thrust constraints for maneuvers and integrating dense reward schemes and domain-based reward shaping techniques. By showcasing the adaptability and scalability of the combined NEAT and DRL framework in diverse operational scenarios, this approach holds immense promise for revolutionizing autonomous orbit management, paving the way for more efficient and adaptable satellite operations while incorporating the physical constraints of satellite, such as thruster limitations.**

## I. Introduction

Recent advancements in satellite miniaturization and their utilization in various fields such as communication and Earth observation have created the need for long-term orbit management and mission planning for LEO satellite constellations. To address these demands, Machine Learning (ML) [1] and Deep Learning (DL) [2] have gained exponential attention within the aerospace community for their applications in autonomous and real-time Guidance, Navigation and Control (GNC) systems [3] for future deployment over the past decades. Such pioneering direction represents a departure from conventional methods, offering real-time onboard capabilities that pave the way for more autonomous and resilient decision-making with long-duration missions [4]. Notably, the application of ML and DL in GNC systems is expected to allow for the execution of diverse periodic operations such as maneuver planning for station-keeping, ensuring the satellite formations stability in the presence of orbital perturbations, thereby enabling precise control to actively maintain the desired relative position of satellite within the constellation. Such advancements enable more fuel-efficient maneuvers with more effective and independent control, marking a significant leap forward in space exploration.

After the final orbital insertion to relocate the satellite from its parking orbit to its desired orbit, continuous perturbations necessitate periodic corrections to ensure the satellite remains within an acceptable tolerance range of its intended trajectory. To address this, defining a trajectory tracking problem involves using the nominal satellite trajectory, unaffected by perturbations, as the reference path. The goal is to determine the necessary maneuver plan or the control input to keep the satellite aligned with this reference trajectory. However, such station-keeping maneuvers become extremely complex given that the satellite is part of a constellation and its relative state, i.e., position and velocity, must be maintained with respect to other satellites within the formation [5]. Furthermore, other factors, such as fuel consumption and the duration of orbital correction, must be considered as they will impact operational capabilities.

DRL has been shown as a promising framework to not only autonomously control the movements of a satellite while maintaining it within its designated orbit but also reduce operational costs and maximize mission returns [6]. It must be noted that DRL algorithms can continuously learn and optimize control strategies by receiving feedback from the environment such as the position of satellite and its orbital dynamics, thereby making optimal state-dependent decisions accordingly [7]. DRL algorithms can be designed to consider various parameters such as gravitational forces, orbital perturbations, and other dynamic factors affecting the position of satellite. Such state-dependent learning processes allow the satellite to adapt to dynamic and unstable environmental conditions and execute maneuvers that optimize station-keeping, reducing the need for constant manual interventions.

Furthermore, due to the size of trained policies, which are based on Multi-Layered Perceptron (MLP) network architectures and therefore are computationally efficient, DRL agents can be deployed on platforms with limited computing power, to achieve autonomous capabilities such as collision avoidance [8], path-planning, trajectory optimization [9] as well as station-keeping [10]. Expanding on this concept, a complex model-free data-driven policy based on Proximal Policy Optimization (PPO) was proposed for propulsion-

Robotics and AI Group, Department of Computer, Electrical and Space Engineering, Luleå University of Technology Luleå Sweden akyuroson@gmail.com, {nektaf, aviban,sumsat,geonik}@ltu.se

less maneuvers that leverage differential drag states to solve multi-satellite constellation problems [5]. Moreover, similar DRL algorithm was used to perform station-keeping maneuvers for a satellite operating near a Sun-Earth $L_2$ Southern quasi-halo trajectory in an ephemeris model while utilizing Bayesian optimization for guided parameter selection to achieve a policy with high accuracy in a deterministic fashion [11].

Given the advantages mentioned above of DRL for use in orbit management, this work investigates the feasibility of utilization of DRL in combination with NEAT [12], [13] for station-keeping, which can lead to more adaptive and efficient control strategies, while enhancing the ability of satellite to autonomously manage its position within a tolerance range of its orbital parameters, thereby reducing operational costs and ensuring precise and consistent station-keeping throughout the mission. Therefore, the main contributions of this study are as follows: (a) A novel satellite environment for real-time simulation of orbital station-keeping that includes non-linear orbital dynamics and perturbations. (b) Employing NEAT for topology and parametric optimization as well as the creation of an expert agent for comparison with optimal policy achieved by DRL in the proposed environment under mission-specific constraints. (c) Utilization of model-free DPGRL algorithms in conjunction with LEO satellites with low-thrust constraints for maneuvers to the desired orbit, while incorporating dense reward scheme and reward shaping based on domain knowledge. (d) A preliminary comparative analysis of Deep Deterministic Policy Gradient (DDPG) [14], Twin Delayed Deep Deterministic Policy Gradient (TD3) [15] and NEAT to further improve learning while fine-tuning the system constraints and address catastrophic inference.

The remainder of this article is structured as follows. Section II outlines the related theories to implement the proposed framework. Thereafter, Section III presents a detailed description of the implemented DRL and NEAT framework and discusses the agent and environmental design for solving LEO satellite station-keeping. Furthermore, a detailed evaluation of achieved simulation results is presented in Section IV. Finally, we conclude this article by discussing the achieved results and future work in Section V.

## II. Background

In this Section, the orbital dynamics, encompassing both the equations of motion and the perturbations affecting the satellite, are presented. Moreover, a detailed description of the DRL algorithms that are employed for orbital station-keeping within the satellite orbital environment is provided, followed by an overview of evolutionary algorithms, specifically NEAT, for its use for Reinforcement Learning (RL).

### A. Orbital Dynamics

The orbital environment model for the satellite, used in both training as well as testing of the RL agent, is described in the Earth Centered Inertial (ECI) (J200) frame of reference

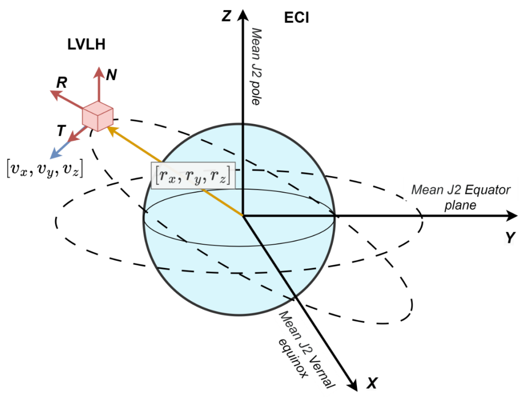

Fig. 1: The ECI and LVLH reference frames.

and is as follows [16], [17]:

$$\ddot{\mathbf{r}}(t) = -\frac{\mu}{||\mathbf{r}||^3}\mathbf{r} + \mathbf{a}_t + \mathbf{a}_{3b} + \mathbf{a}_{drag} + \mathbf{a}_g + \mathbf{a}_{srp}, \qquad (1)$$

where $\mathbf{a_t} = R^{\mathbb{I}}{}_{\mathbb{H}}\frac{\mathbf{U}}{m}$, and the satellite position $\mathbf{r}(t) = [r_x, r_y, r_z]^T$ and velocity $\mathbf{v}(t) = [v_x, v_y, v_z]^T$ are represented in ECI reference frame defined as $\mathbb{I} = \{X, Y, Z\}$ as depicted in Figure 1. Additionally, $\mu$ denotes the Earth's gravitational parameter, and $||r||$ indicates the distance of the satellite from the center of the Earth. The satellite is orbiting around the Earth, primarily under the influence of the gravitational field. However, its motion is affected under the influence of various perturbation factors such as non-spherical gravity $\mathbf{a}_g$ due to Earth's oblateness, atmospheric drag $\mathbf{a}_{drag}$, influences of the third body $\mathbf{a}_{3b}$ such as the Sun and Moon as well as the solar radiation pressure $\mathbf{a}_{srp}$. In the context of station-keeping for the LEO region, the non-spherical Earth's gravitational field and the air drag are the primary influential factors. To account for the effects of disturbances while maintaining the station-keeping objectives, controlled actuation, $\mathbf{a}_t$, is considered provided by the on-board low-thrust electric propulsion system. The thrust force is expressed as $\mathbf{U} = [F_R,\ F_T,\ F_N]^T$ in the Local-Vertical Local-Horizontal (LVLH) reference frame $\mathbb{H} = \{R, T, N\}$, where $R^{\mathbb{I}}{}_{\mathbb{H}}$ denotes the corresponding transformation matrix from $\mathbb{H}$ to $\mathbb{I}$.

External perturbations continuously deviate the satellite from its desired orbit. A brief description of the perturbation models used to construct the orbital environment is described in the following part.

**Earth's Gravitational Perturbation:** The gravitational field can be described as a potential, whose gradient provides acceleration due to gravity. The higher degree spherical harmonics gravity accelerations are given as [16]:

$$\mathbf{a}_g = \nabla\frac{\mu}{r}\sum_{n=2}^{\infty}\sum_{m=0}^{n} H_1 H_2, \qquad (2)$$

where $H_1 = P_{nm}(\sin\varphi)\,(C_{nm}\cos m\lambda + S_{nm}\sin m\lambda)$, $H_2 = \left(\frac{R_E}{r}\right)^n$, and $n$ and $m$ are the degree and order of spherical harmonics, $R_E$ is the Earth's radius, $r$ is the geocentric distance of the satellite, $P_{nm}$ is the Legendre polynomial with argument $\sin\varphi$, $\varphi$ and $\lambda$ are the respective latitude and longitude of the satellite and $C_{nm}$ and $S_{nm}$

are the harmonic coefficients of the potential. The Earth's gravity potential can be modeled accurately, using the higher degree and order Earth's Gravity Model (EGM). The EGM96 gravity model provides the data for spherical harmonic coefficients complete to the degree and order 360 [18].

**Atmospheric Drag:** Given that the satellite is operating in LEO, the acceleration drag experienced is calculated according to Equation 3, where $\mathbf{v}_{rel} = \mathbf{v} - \left(0,\ 0,\ \Omega_{Earth}\right)^T \times \mathbf{r}$ is the relative velocity of the satellite and earth, and $\Omega_{Earth}$, is the Earth's rotation, $C_r$, is the drag coefficient, and $A_d$, the flat plate area in nominal attitude.

$$\mathbf{a}_{drag} = -0.5\rho\left(\frac{A_d C_d}{m}\right)||\mathbf{v}_{rel}||^2\left(\frac{\mathbf{v}_{rel}}{||\mathbf{v}_{rel}||}\right). \tag{3}$$

**Third Body Perturbation:** The third-body gravity perturbations are exerted by the Sun and Moon. Thus, the gravity acceleration due to the third body effects can be expressed as [16]:

$$\mathbf{a}_i = \mu_i\left(\frac{\mathbf{r_{is}}}{||\mathbf{r_{is}}||^3} - \frac{\mathbf{r_i}}{||\mathbf{r_i}||^3}\right), \tag{4}$$

where $i$ represents Sun or Moon, and $\mu$ corresponds to their respective gravitational parameters. Furthermore, $\mathbf{r}_{is} = \mathbf{r}_i - \mathbf{r}$ are the respective position vectors from the satellite to the Sun and Moon, and $\mathbf{r}_i$ is the respective position vector from the Earth to the Sun and Moon. The position of the Sun and Moon is required to calculate the third body acceleration, where $\mathbf{a}_{3b} = \mathbf{a_{moon}} + \mathbf{a_{sun}}$. The position vectors of the celestial bodies were calculated via the SPICE Toolkit.

**Solar Radiation Pressure:** For the Solar Radiation Pressure (SRP) induced acceleration, the cannonball disturbance model and dual conical shadow model [19] are considered. The SRP induced acceleration is given by:

$$a_{srp} = \left(\frac{SC_r A_s}{m}\right)\frac{-\mathbf{r_{ss}}}{||\mathbf{r_{ss}}||^3}, \tag{5}$$

where $S$ is the solar flux calculated from the conical shadow model, $C_r$ is the surface reflectivity coefficient, $A_s$ is the exposed area, $c$ is the speed of light, and $m$ is the mass of the spacecraft.

## *B. Deep Reinforcement Learning*

DRL can be described as a learning paradigm, where an agent attempts to learn an optimal policy, $\pi$, by interacting with its environment, $\mathcal{E}$; see Figure 2. Based on received observations, $x_t$, at each time-step, $t$, the agent takes an action, $a_t \in \mathbb{R}^n$, which will result in a scalar reward feedback, $r_t$. The resulting policy maps each state to a probability distribution for a given action, $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$. By utilizing the Markov Decision Process (MDP), the DRL can be modeled based on state space, $\mathcal{S}$, action space $\mathcal{A}$, initial state distribution $\mathcal{P}(s_1)$, transition dynamics $\mathcal{P}(s_{t+1}|s_t, a_t)$, a reward function $\mathcal{R}(s_t, a_t)$ and a discount factor $\gamma \in [0, 1]$. To obtain the optimal policy by maximizing the expected return from the initial distribution, $\mathcal{J} = \mathbb{E}_{r_i, s_i \backsim E, a_i \backsim \pi}[\mathcal{R}_1]$ is the main goal in DRL framework.

*1) DDPG:* DDPG can be described as an off-policy RL algorithm that has adopted an actor-critic architecture as shown in Figure 3. By combining elements of value-based

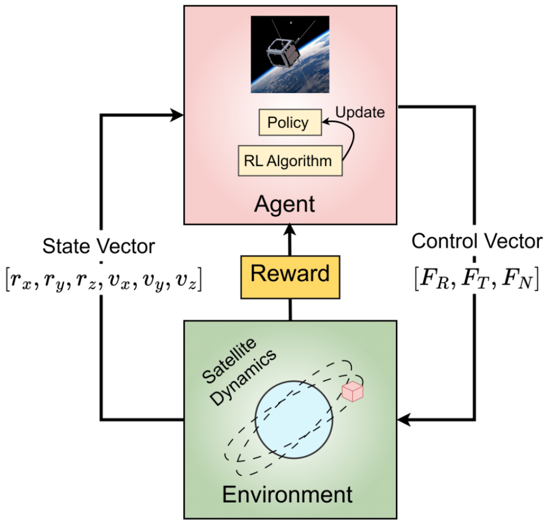

Fig. 2: The layout of the proposed DRL framework.

and policy-based methods, it can concurrently learn a $Q$-function and a policy based on the sampled batches from its experience pool [14]. By directly taking specific actions based on the current state, the actor-network learns a deterministic policy. Moreover, experience replay is employed to stabilize training, and target networks are used to compute more stable $Q$-value estimates. Critic Network updates its parameters, $\theta^Q$, based on the Temporal Difference (TD) method, where the loss function for the critic network is given by:

$$L(\theta^Q) = \frac{1}{N}\sum_i \left(Q(s_i, a_i|\theta^Q) - y_i\right)^2, \tag{6}$$

where $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$ is the target value for the critic network and $\gamma$ the discount factor.

*2) TD3:* To address the overestimation of $Q$-value, which can be observed in actor-critic methods such as DDPG, TD3 adopts an improved clipping variant of double $Q$-learning by utilizing two critic networks that have access to the same pool of sampled experiences [15]. To update the critic networks, the minimum $Q$-value is selected based on the following expression:

$$L = \left(r + \gamma \min_{i=1,2} Q\prime(s_{t+1}, a_{t+1}|\theta_i^{Q\prime}) - Q(s_{t+1}, a_{t+1}|\theta_i^{Q\prime})\right)^2, \tag{7}$$

where $\theta_1^Q$ and $\theta_2^Q$ represent the corresponding parameters of critic networks and $\theta_1^{Q\prime}$ and $\theta_2^{Q\prime}$ are the parameters of the target networks. Moreover, $r(s_t, a_t)$ is the reward feedback based on the deterministic action $a_t = \mu(s_t|\theta^\mu)$. To minimize the loss function, Equation 7 is used to update both $\theta_1^Q$ and $\theta_2^Q$. Furthermore, by adding exploration noise, $\epsilon$, to the action, the value function can be updated in a smoother manner while reducing the $Q$-function error exploitation. It must be noted that based on the domain-specific problem, $\epsilon$ can be either represented by uncorrelated Gaussian distribution [15], $\mathcal{N}(0, \sigma_a)$, where $\sigma_a$ represents the standard deviation observed in $n$ steps, or temporally correlated noise such as Ornstein-Uhlenbeck (OU) process [14]. The regularization of $a_{t+1}$ by adding clipped noise, $\epsilon$, can facilitate the smoothing of the target policy and increase the exploration ability of the algorithm. The TD3 employs the delayed target network

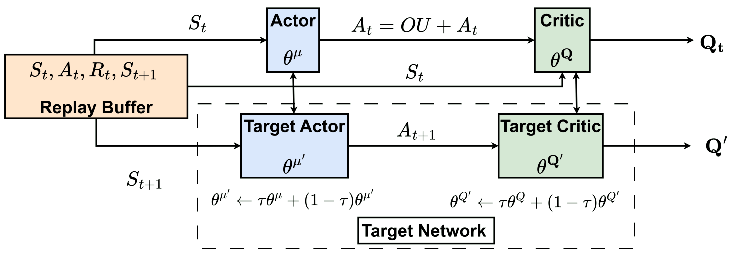

Fig. 3: DDPG algorithm architecture flowchart.

update method by using State–Action–Reward–State–Action (SARSA)-style regularization technique to prevent accumulations of error and to ensure small TD error during the training. The soft update rule for the target network is given by $\theta_i^{Q\prime} \leftarrow \tau \cdot \theta_i^{Q} + (1-\tau)\theta_i^{Q\prime}$, where $\theta_i^{Q}$, $\theta_i^{Q\prime}$ and $\tau$ represent the parameters of the main and the target network and the updating rate, respectively.

### C. Neuro-Evolutionary Topology Optimization

NEAT can be described as an evolutionary algorithm, where both the weights and topologies of Neural Networks (NNs) are evolved such that for any given control problem [20], it can discover an optimal policy without any reliance on indirect inference based on the value function. Furthermore, it has been shown that NEAT can efficiently survey policy space for the environments where the gradients are complex and difficult to calculate [12]. Combined with its capability to mitigate the impact of initialization as well as initial topology configuration of NNs on learning performance, NEAT has been considered to be vital for the optimization of domain-specific MLPs in DRL framework [21]. However, due to its initialization approach, where extremely rudimentary NNs without any hidden layers are utilized to gradually evolve into highly sophisticated networks, it has been deemed sample inefficient [12]. Therefore, other variations of NEAT such as Hypercube-based Neuro-Evolution of Augmenting Topology (HyperNEAT) [22] and adaptive HyperNEAT [23] were proposed to address some of the shortcomings of NEAT.

The topology optimization via NEAT is achieved by incrementally augmenting the initial population of generated NNs via mutation operators in addition to adding nodes and links after each generation assessment [13]. Thus, discovering the most optimal topology with the minimum required number of nodes and their related weights for a given environment. Furthermore, NEAT utilizes speciation by measuring the number of excess, $E$, and disjoint, $D$, genes between a given pair of genomes to calculate their compatibility distance, $\delta$, thus avoiding any topological override between different populations for any given species [12]. The compatibility distance can be expressed as $\delta = c_1\frac{E}{N} + c_2\frac{D}{N} + c_3\overline{W}$, where $c_1$, $c_2$, and $c_3$ are the associated weights to adjust the importance of each factor while $\overline{W}$ and $N$ represent the average weight differences of matching genes and the number of genes, respectively. During the reproduction phase, NEAT leverages explicit fitness sharing, thereby preventing the takeover of the entire population by any single species to allow possible future evolution and crossovers [20]. The adjusted fitness, $f_i'$, for given organism, $i$, is given by:

$$f_i' = \frac{f_i}{\sum_{j=1}^{n} sh\left(\delta(i,j)\right)}, \tag{8}$$

where $sh(\delta(i,j)) \in \{0,1\}$ and represents the sharing function that depending on the compatibility threshold, $\delta_t$, reduce $\sum_{j=1}^{n} sh\left(\delta(i,j)\right)$ to the number of organisms in a given species. It must be noted that based on $f_i'$, the number of offsprings are adjusted such that the lowest performing species are eliminated before generating the next generation [22].

## III. Methodology

In this Section, the proposed orbital environment and its related constraints are presented. Moreover, a brief description of DRL algorithms as well as NEAT and their related hyperparameters are provided. The reward function design for addressing the station-keeping problem is given in detail, and its subsequent components are further analyzed.

### A. Station-keeping in DRL Framework

Figure 2 illustrates the proposed architecture of the satellite environment and its utilization in conjunction with DRL framework for autonomous orbital management, specifically optimal station-keeping. To determine the optimal maneuvers required for driving the satellite back to its nominal reference trajectory based on its current state, $\mathbf{x}(t)$, the desired nominal trajectory without any perturbations is calculated and utilized as the target state in DRL algorithm to determine the optimal control input, $\mathbf{U}$, for maintaining the satellite on its reference trajectory. Furthermore, the reward feedback, $\mathcal{R}$, is proposed based on the convergence towards the nominal state and allows the optimization of the policy, $\pi$, during the training phase by interacting with the simulated environment that encompasses the satellite dynamics.

*1) Experiment Setup:* The station-keeping problem can be described as the optimal trajectory tracking problem and is given by:

$$\min_{U} \quad V(s(t)) = \int_{\tau=t}^{t_f} \|(s(\tau) - s_d(\tau)\|_2^Q + \|U(t)\|_2^R \mathrm{d}\tau, \tag{9}$$

where $t_f$ denotes a specific final time, while $s_d(\tau)$ and $s(\tau)$ represent the desired and actual states of the spacecraft at any given time instance $\tau$, respectively. Furthermore, Equation 9 is subjected to $\dot{s}(t) = As(t) + \frac{1}{m}BU + D$, where $U(t) \in [\mathbf{F}_{min},\ \mathbf{F}_{max}]$ represents the thrust magnitude with its minimum and maximum thresholds. In the proposed environment, the low-thrust constraints are applied such that

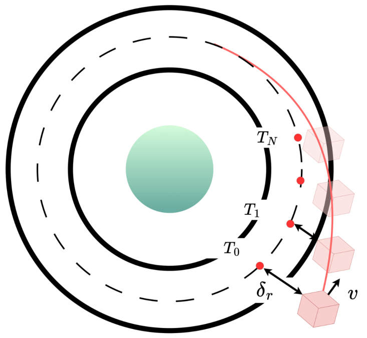

Fig. 4: Satellite trajectory tracking, where $\delta_r$ is the error between the reference and actual position at time-step $T_n$.

the maximum thrust magnitude is 18 mN. The time-step $T_n$ is set to 10 s with the thrust duration also fixed at 10 s. Moreover, the state initialization is randomized such that the initial velocity and position of the satellite have been affected by the perturbations without any prior interference with position error, $\delta_r \leq 45km$, as illustrated in Figure 4.

*2) Action & Observation Space:* The state observation for a given satellite is determined based on satellite dynamics and the perturbations present in its environment, as depicted in Figure 2. The observation space, $\mathbf{x}(t)$, contains both the velocity vector, $\mathbf{v}(t)$, as well as the position vector, $\mathbf{r}(t)$, of the satellite in a given time-step and is obtained from the orbital propagator. Both velocity and position vectors are included in observation space for the DRL algorithm due to the station-keeping problem formulation, where both the position and velocity of the satellite must be tightly tracked to maintain a nominal orbit.

Furthermore, the action vector, $a_t$, is the control input, which is expressed as $a_t = [\mathbf{F_R},\ \mathbf{F_T},\ \mathbf{F_N}]$ in LVLH reference frame and corresponds to the exerted forces from the satellite thrusters. The normalized thrust vector elements with their corresponding range $\in [-1, 1]$ are denoted by $\mathbf{F}_R$, $\mathbf{F}_T$ and $\mathbf{F}_N$. By considering six onboard thrusters, which are aligned with all three axes of LVLH frame of reference, the actuation command for the combined thrusters located on the same axis can be described as acceleration and deceleration along a given axis for a given state.

*3) Reward Function Design:* The reward function and its design is a crucial element of DRL framework as it directly impacts the learning outcomes of a given agent [17]. Therefore, a well-formulated reward function is required to enable the agent to efficiently explore the environment for optimal policy convergence. To comply with the standard reward formulation based on policy gradient frameworks

TABLE I: Hyperparameters for the DRL algorithms.

| Parameters | DDPG Values | TD3 Values |
|---|---|---|
| *Learning rate critic* | 0.00025 | 0.0001 |
| *Learning rate actor* | 0.000025 | 0.00001 |
| *Discount factor* | 0.99 | 0.99 |
| *Update interval* | 2 | 2 |
| *Target network update rate* | 0.001 | 0.001 |
| *Batch size* | 64 | 64 |
| *OU noise* ($\sigma$) | 0.15 | 0.15 |
| *OU noise* ($\zeta$) | 0.0 | 0.0 |
| *OU noise* ($\phi$) | 0.2 | 0.2 |

of RL and MDP, a policy given as $\pi_\theta$ can be found such that the agent must optimize the parameter $\theta$ by maximizing the expected accumulated reward $\mathcal{J}(\theta) = \hat{\mathbb{E}}_{s\backsim\rho^\pi,a\backsim\pi_\theta}[r(s,a)]$. According to the policy gradient theorem, the gradient of $\mathcal{J}(\theta)$ with respect to $\theta$ can be expressed as following:

$$\nabla_\theta \mathcal{J}(\pi_\theta) = \hat{\mathbb{E}}_{s\backsim\rho^\pi,a\backsim\pi_\theta}[\nabla_\theta log\pi_\theta(s,a)Q^\pi(s,a)], \quad (10)$$

where $Q^\pi$ represents the state-action value function. To realize fully autonomous station-keeping, state-specific thresholds for both position and velocity are defined such that extreme divergence from optimal orbit will result in the termination of the simulation with an extremely high penalty. Furthermore, the reward function, $\mathcal{R}_{tot}$, has been decomposed into multiple segments to achieve fuel-optimal maneuvers to reach the nominal trajectory within the least feasible steps. Therefore, a dense reward scheme is proposed to facilitate the agent in its exploration endeavor while minimizing the samples required to reach an optimal policy. The total reward function is expressed as a combination of linear and exponential functions and is formulated as follows:

$$\mathcal{R}_{tot}(s,a) = \mathcal{R}_{state} + \mathcal{R}_{position} + \mathcal{R}_{bound} + \mathcal{R}_{action} - \mathcal{R}_{step}, \quad (11)$$

where $\mathcal{R}_{state}$ represents the unified state reward based on $\{\mathbf{s}_t, \mathbf{s}_{t+1}, \mathbf{s}_{target}\}$ and is given by:

$$\mathcal{R}_{state} = -ln\left(\|\mathbf{s}_{t+1} - \mathbf{s}_{target}\|\right) + ln\left(\|\mathbf{s}_t - \mathbf{s}_{target}\|\right). \quad (12)$$

Moreover, $\mathcal{R}_{position}$ is a compounded reward to assess the approximation to the nominal position by exponentially rewarding the agent trajectory for given actions. This will result in closer proximity to the desired orbit and can be expressed as:

$$\mathcal{R}_{position} = e^{\left(20-\frac{\|\delta_r\|}{\psi}\right)}, \quad (13)$$

where $\psi = 10^3$ is a scaling factor and $\delta_r = \bar{\mathbf{r}}_{t+1} - \bar{\mathbf{r}}_{target}$ represents the error between the reference and actual position as shown in Figure 4. Equation 13 provides an auxiliary reward when the agent crosses the maximum tolerable trajectory threshold, $\delta_r \leq 20km$. Furthermore, to minimize the consumption of propellant, the agent is rewarded based on the magnitude of control input, which is given by:

$$\mathcal{R}_{action} = \beta \cdot (1 - \|\bar{\mathbf{a}}\|), \quad (14)$$

where $\beta = 10^2$ is an empirical weight chosen such that the fuel consumption is scaled with respect to $\mathcal{R}_{position}$, thereby resulting in less fuel utilization when the satellite is within the desired boundary threshold, $\delta_r \leq 1km$. To ensure that the exploration is performed in an optimal and sample-efficient fashion, the $\mathcal{R}_{bound}$ is utilized to incentivize any exploration where the satellite exceeds its initial divergence outside of selected boundaries. Therefore, $\mathcal{R}_{bound}$ is solely activated when the achieved error between the nominal and actual position in a new state, $\bar{\mathbf{r}}_{t+1} \geq 35km$, and is calculated as follows:

$$\mathcal{R}_{bound} = -\left(10 \cdot \psi + e^{\left(\frac{\|\delta_r\|}{\psi}-40\right)}\right). \quad (15)$$

To minimize the number of steps while achieving the desired orbit, $\mathcal{R}_{step}$ is used to encourage the agent to converge to the solution by minimizing its effort and can be expressed by $\mathcal{R}_{step} = \gamma \cdot \kappa$, where $\gamma = 5 \cdot 10^{-4}$ is an empirically selected weight and $\kappa$ represents the current step.

*4) Terminal Constraints:* The terminal constraints play a significant role in the DRL training process, as it prevents sample-inefficient training. Therefore, the terminal position error is defined as follows:

$$T_{constraints} = \begin{cases} True & \|\bar{r}_{t+1} - \bar{r}_{target}\| \geq 50km \\ False & Otherwise. \end{cases} \tag{16}$$

Given that the terminal criterion is reached, the training episode is halted, and the agent is penalized and receives a $-10^4$ penalty. Furthermore, in any successful episode, where $\|\delta_r\| \leq 1km$, the agent is rewarded with an additional $10^4$ points.

*5) Exploration Noise:* The exploration of the environment by the agent is facilitated by incorporating noise into the control inputs generated by the actor-network as depicted in Figure 3. This strategy increases the efficacy of DRL based control by effectively expanding the agent exploration into previously unseen state-action space during the training process [14]. Furthermore, the addition of noise creates inherent robustness in the generated model for real-world scenarios, given the existence of noise in the acquired data during live missions.

The OU process, which models the velocity of Brownian particles in the presence of friction, is utilized during the training phase for both DDPG and TD3 as temporally correlated noise. The OU process is formulated as follows:

$$\delta \mathbf{a} = \phi(\zeta - a) + \sigma \mathcal{N}(0, 1), \tag{17}$$

where $\phi$, $\zeta$, and $\sigma$ represent the decay rate, long-term mean, and variation of the generated noise, respectively [14]. The associated values for each parameter of OU process for both DRL algorithms are provided in Table I.

*6) Hyperparameters:* Both DDPG and TD3 have been considered as DRL algorithm candidates to evaluate the proposed satellite environment and obtain an optimal policy for the station-keeping problem. The main driving factor for selecting these algorithms is their wide utilization in application with both observation and action space in the continuous domain [14]. The topology for actor and critic networks for both algorithms was selected after experimenting with networks of various sizes. A similar network topology is used throughout all training and evaluation phases to maintain neutrality and accurately assess the capabilities of both algorithms. Specifically, the actor and critic networks are constructed with two deep, fully connected layers comprising 400 and 300 neurons, respectively, activated by the ReLU function. Furthermore, the output layer utilizes the hyperbolic tangent function, tanh, as its activation mechanism, enabling the clipping of the thrust vector, $\mathbf{a}_t$, within its predetermined bounds. Details regarding the remaining hyperparameters for both algorithms are provided in Table I.

### *B. Station-keeping with NEAT*

Similar to the proposed DRL framework depicted in Figure 2, NEAT algorithm is used to leverage the proposed satellite environment for autonomous orbit management, specifically addressing the station-keeping problem with defined constraints as stated briefly in the previous section.

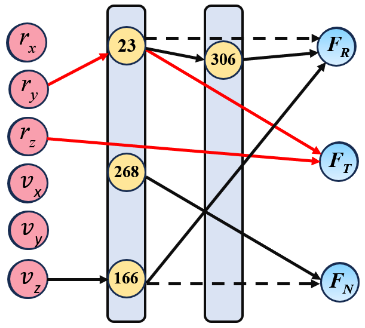

Fig. 5: NNs topology from NEAT for the best-achieved fitness with two hidden layers, where yellow nodes indicate clusters of neurons while the color of edges, red and black, indicate a negative or positive correlation, respectively.

*1) Experiment Setup:* A similar environment and experimental setting as DRL framework is deployed, where the main goal is evolution and optimization of the feature network, thereby determining the optimal topology for the defined problem to study the correlation between state and action space.

*2) Hyperparameters:* NEAT has been employed to obtain an optimal policy that satisfies similar constraints and criteria as DRL algorithms. Furthermore, the initial number of hidden layers was set to 2 so that similar MLP as DDPG and TD3 actor-network is created to further investigate the selected NNs architecture and possibly discover other optimal topology for the station-keeping problem for low-thrust satellites. It must be noted the termination condition was selected such that given the maximum population, the average fitness of the species must be $\geq 10^4$. The remaining hyperparameters to configure NEAT are provided in Table II.

## IV. Results

In this section, the results from both DRL algorithms as well as NEAT for station-keeping are presented. Furthermore, the component-wise residual for the state vector is provided to analyze the behavior of policy generated by both methods. It must be noted that both the environmental framework and the agents have been implemented in the Python 3.8 environment, while Pytorch is used for the implementation of DRL algorithms.

TABLE II: Hyperparameters for NEAT algorithm.

| **Parameters** | **Values** |
|---|---|
| $Population\ size\ (p)$ | 50 |
| $Max\ generations\ n_2$ | 100 |
| $Add\ node\ rate\ (m_n)$ | 0.02 |
| $Add\ connection\ rate\ (m_l)$ | 0.5 |
| $Weight\ init\ mean$ | 0.0 |
| $Weight\ init\ std$ | 1.0 |
| $Weight\ mutate\ rate\ (m_w)$ | 0.3 |
| $Weight\ mutate\ power$ | 0.81 |
| $Activation\ functions$ | ReLU & tanh |
| $Aggregation\ functions$ | mean, max & sum |
| $Crossover\ rate\ (c)$ | 0.1 |

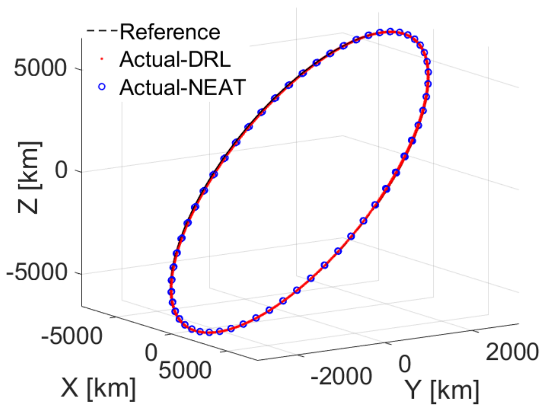

Fig. 6: Correction of satellite trajectory utilizing DRL and NEAT.

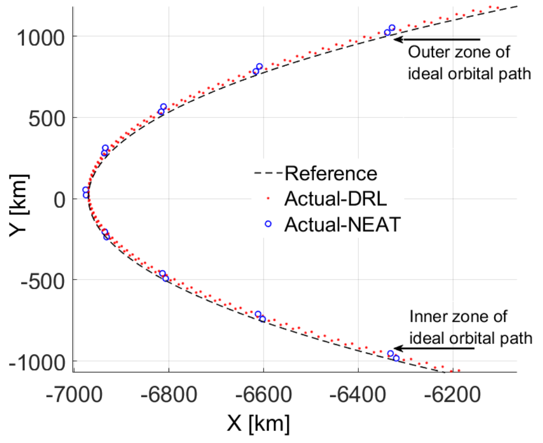

Fig. 7: Close-up view of a segment of satellite trajectory based on DRL and NEAT.

## A. Evaluation of Reward

The average episodic reward for both DDPG and TD3 during the training process is shown in Figure 8. As Figure 8 illustrates, although both DRL algorithms are capable of solving the station-keeping task, DDPG has the most unreliable performance due to catastrophic inference, which is one of the primary challenges of DRL algorithms given non-stationary data stream. To combat these issues, several strategies were devised for continual learning that align with the main reasons that TD3 was adopted due to its stable learning and faster convergence to the solution.

## B. Evaluation of DRL & NEAT

The optimal policy from TD3 is utilized to compare the trajectory and velocity error between the DRL framework and NEAT. It must be noted that optimized NNs topology devised by NEAT closely resembles the MLP architecture proposed for TD3 [15] as shown in Figure 5. Moreover, it has been shown that in the context of station-keeping, some state elements are not vital for achieving the nominal trajectory and velocity; see Figure 5.

Figure 6 and 7 illustrate the nominal trajectory of the satellite without perturbations as well as trajectory correction performed by TD3 and NEAT for station-keeping for a given

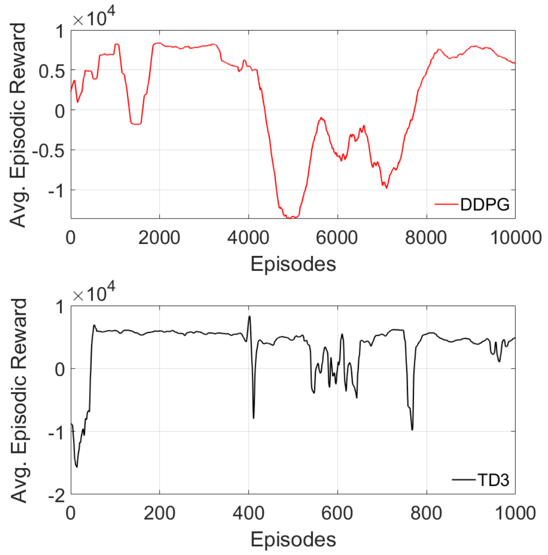

Fig. 8: Average episodic training reward for DDPG and TD3.

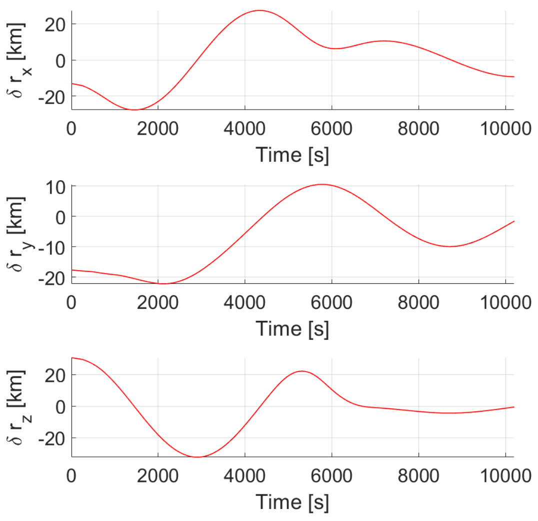

Fig. 9: Component-wise trajectory error for TD3.

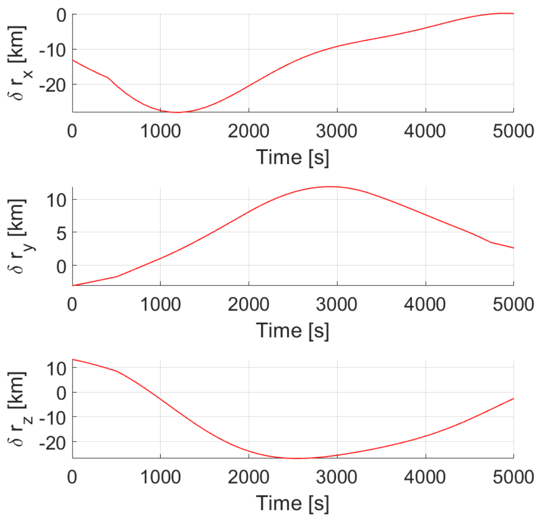

Fig. 10: Component-wise trajectory error for NEAT.

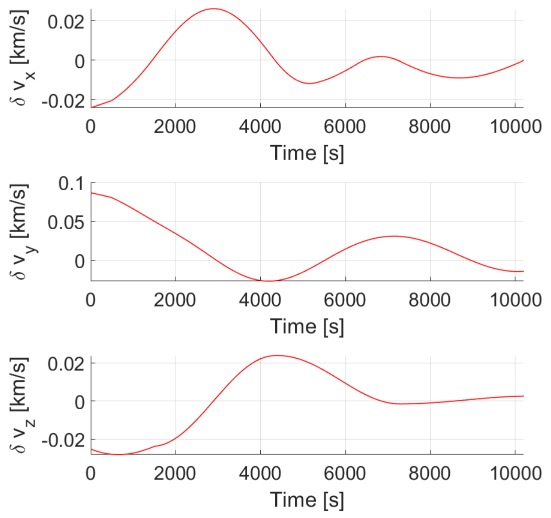

Fig. 11: Component-wise velocity error for TD3.

satellite in the presence of perturbations. It must be noted that the target orbit for this study is considered to be a circular orbit with a radius of 600 km, and an inclination of 70 deg. Furthermore, Figure 9 and 10 depict the residual between the target and the actual position of satellite for TD3 and NEAT as the correctional maneuver is performed. Similarly, the error between the desired and the actual velocity of the satellite for TD3 is shown in Figure 11. NEAT algorithm produces similar velocity residuals, which indicates that both algorithms can perform trajectory maneuvers such that the nominal trajectory is reached in the presence of perturbations.

## V. Conclusions

This article introduces a novel approach towards achieving a fully autonomous orbital station-keeping for LEO satellites through the amalgamation of NEAT and DRL. The quest for complete autonomy becomes essential in response to the pressing need for an autonomous decentralized orbit management strategy for vast satellite constellations. Towards this, the proposed methodology provides a framework for continuous, precise, and adaptable positioning of satellites without consistent human intervention. By synergizing DRL and NEAT while integrating dense reward schemes, the proposed approach strives to optimize the orbital deviation to empower satellites to independently maintain their orbits despite external perturbations, staying within acceptable tolerance margins. The proposed concept showcases the potential of AI-driven solutions in advancing autonomous control systems for complex real-world applications, departing from conventional methods and paving the way for more efficient and resilient satellite operations. Population-based optimization, prioritized replay buffer, and other variations of NEAT can be investigated to optimize the results further.

## References

[1] B. Li, J. Huang, Y. Feng, F. Wang, and J. Sang, "A machine learning-based approach for improved orbit predictions of LEO space debris with sparse tracking data from a single station," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 6, pp. 4253–4268, 2020.

[2] H. Li, S. Chen, D. Izzo, and H. Baoyin, "Deep networks as approximators of optimal low-thrust and multi-impulse cost in multitarget missions," *Acta Astronautica*, vol. 166, pp. 469–481, 2020.

[3] D. Izzo, M. Märtens, and B. Pan, "A survey on artificial intelligence trends in spacecraft guidance dynamics and control," *Astrodynamics*, vol. 3, pp. 287–299, 2019.

[4] H. Li, H. Baoyin, and F. Topputo, "Neural networks in time-optimal low-thrust interplanetary transfers," *Ieee Access*, vol. 7, pp. 156 413–156 419, 2019.

[5] B. Smith, R. Abay, J. Abbey, S. Balage, M. Brown, and R. Boyce, "Propulsionless planar phasing of multiple satellites using deep reinforcement learning," *Advances in Space Research*, vol. 67, no. 11, pp. 3667–3682, 2021.

[6] A. Harris, T. Teil, and H. Schaub, "Spacecraft decision-making autonomy using deep reinforcement learning," in *29th AAS/AIAA space flight mechanics meeting, hawaii*, 2019, pp. 1–19.

[7] Y. Cai, E. Zhang, Y. Qi, and L. Lu, "A review of research on the application of deep reinforcement learning in unmanned aerial vehicle resource allocation and trajectory planning," in *2022 4th International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, 2022, pp. 238–241.

[8] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 528–535.

[9] C. J. Sullivan and N. Bosanac, "Using reinforcement learning to design a low-thrust approach into a periodic orbit in a multi-body system," in *AIAA Scitech 2020 Forum*, 2020, p. 1914.

[10] D. Miller and R. Linares, "Low-thrust optimal control via reinforcement learning," in *29th AAS/AIAA Space Flight Mechanics Meeting*, vol. 168. American Astronautical Society Ka'anapali, Hawaii, 2019, pp. 1817–1834.

[11] S. Bonasera, N. Bosanac, C. J. Sullivan, I. Elliott, N. Ahmed, and J. W. McMahon, "Designing sun–earth l2 halo orbit stationkeeping maneuvers via reinforcement learning," *Journal of Guidance, Control, and Dynamics*, vol. 46, no. 2, pp. 301–311, 2023.

[12] Y. Peng, G. Chen, H. Singh, and M. Zhang, "NEAT for large-scale reinforcement learning through evolutionary feature learning and policy gradient search," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 490–497.

[13] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[15] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018.

[16] D. A. Vallado, *Fundamentals of astrodynamics and applications*. Springer Science & Business Media, 2001, vol. 12.

[17] J. Hu, H. Yang, S. Li, and Y. Zhao, "Densely rewarded reinforcement learning for robust low-thrust trajectory optimization," *Advances in Space Research*, 2023.

[18] NASA and NIMA, "EGM96 the NASA GSFC and NIMA joint geopotential model." [Online]. Available: https://cddis.nasa.gov/926/egm96/egm96.html

[19] E. Gill and O. Montenbruck, *Satellite orbits: Models, methods and applications*. Springer, 2013.

[20] K. O. Stanley and R. Miikkulainen, "Competitive coevolution through evolutionary complexification," *J. Artif. Int. Res.*, vol. 21, no. 1, p. 63–100, feb 2004.

[21] S. Whiteson, P. Stone, K. O. Stanley, R. Miikkulainen, and N. Kohl, "Automatic feature selection in neuroevolution," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 2005, pp. 1225–1232.

[22] S. Risi and K. O. Stanley, "Enhancing ES-hyperNEAT to evolve more complex regular neural networks," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 1539–1546.

[23] S. Risi and K. O. Stanley, "Indirectly encoding neural plasticity as a pattern of local rules," in *International conference on simulation of adaptive behavior*. Springer, 2010, pp. 533–543.