# Safe Capturing and Stabilizing of Diverse Debris Objects

Master thesis by Wiebke Retagne (2843582)
Date of submission: October 19, 2023

1. Review: Prof. Dr. rer. nat. Robert Roth
2. Review: Prof. Dr. rer. nat. Günther Waxenegger-Wilfing
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Physics Department
DLR, Institut für Kernphysik
AG Robert Roth

## Erklärung zur Abschlussarbeit
## gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Wiebke Retagne (2843582), die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 19. Oktober 2023

W. Retagne

W. Retagne

# Contents

# Abstract

With increased access to space, the challenge of space debris is becoming a more and more relevant topic. To keep space usable for future generations, space debris has to be mitigated. The most effective mitigation measure includes Active Debris Removal (ADR) of around 5 objects per year. This thesis investigates the challenge of the Attitude Control System (ACS) in ADR. The problem here is that often the exact size, shape and mass of the targeted debris objects are unknown. Furthermore, through the capturing process the mass and inertia of the satellite system is changed, making it difficult for the ACS to maintain a stable position. In this thesis an approach of Deep Reinforcement Learning (DRL) algorithm is investigated to solve this satellite attitude problem. A DRL approach will help in coping with the unknown inertia and will enable the ADR to handle diverse debris objects. Several different attitude control scenarios are investigated, with the complexity increasing continuously.

# 1 Introduction

In the last 10 years the landscape of space flight has changed drastically. Historically, only national and supranational agencies like NASA [1] or ESA [2] were able to put a satellite into space. With companies like SpaceX [3] offering rocket starts commercially, space has become more and more accessible. In 2021, the start-up sector in space saw a significant increase in funding. Relative to 2020, capital investment grew by a whopping 82% [4]. This offers several opportunities, but it also means that the number of players and the number of objects in the popular regions of space like the low-earth orbit and the geostationary orbit is increasing rapidly. Mega-constellations for commercial use by companies such as SpaceX, Amazon [5] and OneWeb [6] further contribute to the rising number of satellites. As a consequence, space debris has become a more and more relevant topic. To keep space usable for future generations, space debris has to be mitigated. One mitigation measure is the so-called "passivation", i.e. making sure that post mission break ups are minimized. This can be done by implementing measures for depleting the stored energy (tanks, batteries) and assessing the risk for components that cannot fully be depleted of their energy. Another important mitigation measure is the post mission disposal, where satellites de-orbit themselves after they have successfully completed their mission. While these measures can help in mitigating space debris, they are only feasible for satellites that are still command-able at the end of their life. If a mission is lost due to any reason then the only measure left is the Active Debris Removal (ADR). ADR describes the process of removing a space debris fragment through capturing this fragment and de-orbiting it. Fig. 1.1 shows that including ADR in mitigation measures ensures for the most effective mitigation of space debris. The company ClearSpace [7] has been tasked to do a first demonstration for an ADR. The ClearSpace-1 mission will target the Vespa (Vega Secondary Payload Adapter). It is set to launch in 2025.

The challenge here is that often the exact size, shape and mass of the targeted debris objects are unknown. Furthermore, through the capturing process the mass and inertia of the satellite system is changed. Classical attitude regulation systems are modeled to work in a known environment, with a known inertia. The ClearSpace demonstration also involves just one object, where the size and mass are well known. If the ADR is to be used widely, for around 5 objects per year, it is not feasible to build one dedicated satellite for each debris object that is to be removed. In this thesis an approach of Deep Reinforcement Learning (DRL) algorithm is investigated to solve this satellite attitude problem. A DRL approach would help in coping with the unknown inertia and would enable the ADR to handle diverse debris objects.

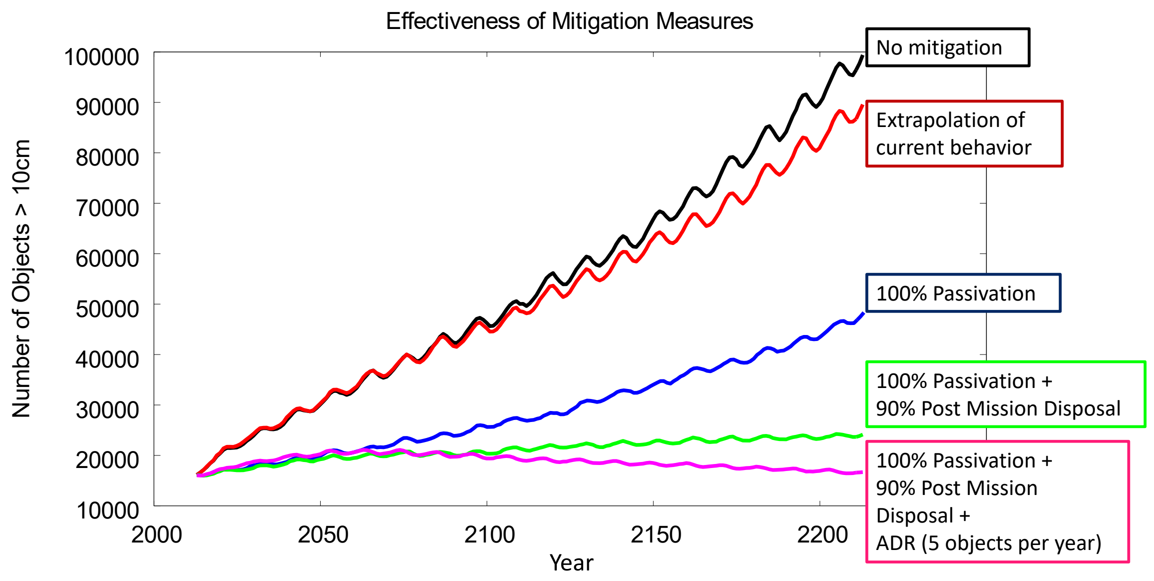


Figure 1.1: Comparison of different mitigation measures and their effect. The extrapolation of the current behavior regarding space debris shows a significant increase in space debris (red). The most effective mitigation measure includes ADR of around 5 objects per year (pink) [8].

DRL - and especially DRL for the spacecraft control applications - has been investigated in previous works. An overview can be found in [9]. The partially known space environment, the time-varying dynamics and the benefits of autonomy make it the perfect candidate for reinforcement learning. Allison et al. [10] investigated the use of a Proximal Policy Optimization (PPO) algorithm to find an optimal control strategy for the spacecraft attitude problem. The spacecraft was modeled as a rigid body and the state space is defined by the error quaternion vectors. The control policy obtained was able to achieve a good performance for a range between $0.1\,\text{kg}$ and $100,000\,\text{kg}$. As a simulation environment the Mujoco physics engine was chosen. In this work the Basilisk software [11] is used. This is an astrodynamic simulation software especially designed for spacecraft applications. In reinforcement learning (RL) the learned policy is very dependent on the simulation used in training. Choosing a simulator which closely reproduces the real dynamics helps to ensure that RL policies operate well in real environments. Additionally, in this thesis the Soft Actor Critic (SAC) algorithm (see section 2.2) is chosen to train the agent. This has been shown to achieve better results in continuous action spaces and encourages exploration. In this thesis the use of safety constraints will also be investigated. In RL, constraints are usually modeled through the reward function [12]. This can introduce a large amount of trial and error in the formulation of the reward function. In this work the proposal is to use constraints independent of the reward function that the agent has to consider like in [13]. This can help in guaranteeing stability.

# 2 Theoretical Background

This chapter lays the basis for the following discussions. In the subsequent sections, the theory behind Reinforcement Learning and the Soft Actor Critic Algorithm (SAC) are discussed.

## 2.1 Reinforcement Learning

The idea of Reinforcement Learning (RL) is based on an interaction between an agent and an environment. This can be compared to to how humans learn by trial and error. How this basic concept of learning can be translated into a computational approach shall be discussed in this section. The section is based on [12].

### 2.1.1 The Reinforcement Learning Problem

The Reinforcement Learning Problem can be characterized through the interaction of the so-called agent with an environment. The agent is the learner and decision maker, while the environment provides the background and gives feedback to those decisions. In contrast to supervised learning the correct input/output pairs are not given, nor are sub-optimal actions explicitly corrected. To describe this agent-environment interaction more formally the "action", the "state" and the "reward" are defined:
The **agent** interacts with the environment in a discrete **time step** $\mathbf{t}$ through an **action** $\mathbf{a_t}$. Afterwards the agent receives feedback in the form of the **state** $\mathbf{s_{t+1}}$, which contains information about the environment. To evaluate the action the agent also receives a numerical **reward** $\mathbf{r_{t+1}}$ signal. The goal is to maximize the sum of rewards in the long run. The sum of these rewards $r$ is also called the **return** $\mathbf{R}$.

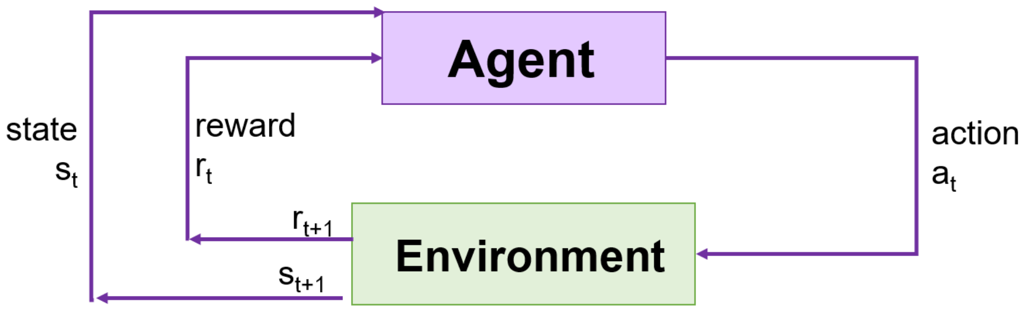


Figure 2.1: The agent environment interaction. The agent observes the state $s_t$, takes an action $a_t$ and receives a reward $r_{t+1}$ and the next state $s_{t+1}$ from the environment. Modified from [12].

### 2.1.2 Markov Decision Process

Mathematically, the RL problem can be represented as a Markov Decision Process (MDP). If an environment satisfies the **Markov property** that generally means the present state is only dependent on the immediate previous state. If the states are deterministic the next state can be predicted based on the current state and the action taken in this state. If the states are non-deterministic a state probability distribution can be predicted. In both cases the associated reward with the state or state distribution can be predicted. The MDP can then be defined as

$$MDP := (S, A, T, P, r), \tag{2.1}$$

with

$$\begin{align} S &:= \text{State space}, \tag{2.2}\\ A &:= \text{Action space}, \tag{2.3}\\ T \subseteq \mathbb{N} &:= \text{ set of time steps}, \tag{2.4}\\ P : S \times A \times T \times S \to [0,1] &:= \text{State-transition probability function}, \tag{2.5}\\ r : S \times A \times T \to \mathbb{R} &:= \text{Reward function}. \tag{2.6} \end{align}$$

The action space includes all actions. $A(s_t)$ is the set of available actions in state $s_t$ at the time step $t \in T$. The state-transition probability function $P$ gives the probability of the transition from state $s_t$ to the subsequent state $s_{t+1}$, when taking action $a_t$ . The reward function is the associated reward with this transition. If the state and action space are finite then the MDP is also finite [12].

### 2.1.3 Discounted Rewards

As mentioned before, the goal of the agent should be to maximize the return and not only the immediate reward. In the simplest case the return $R_t$ is the sum of the future rewards until the terminal state $T$.

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T. \tag{2.7}$$

Here, a distinction between **episodic tasks** and **continuous tasks** needs to be made. An episodic task is characterized through repeated interactions, which have a fixed terminal state $T$. In continuous tasks the interaction between agent and environment cannot naturally be broken into identifiable episodes. The final time step here is $T = \infty$. As a result, the return itself could become infinite. In general, a balance needs to be found between selecting an action that has a big immediate reward and selecting one which might provide a bigger reward in the future. To achieve this it can be useful to introduce **discounted rewards**. A discount factor $\gamma$ is used to rate the future rewards and therefore, determine how these are considered:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \text{ with } 0 \le \gamma \le 1. \tag{2.8}$$

The value of the discount factor $\gamma$ determines how far-sighted the agent is. If $\gamma = 0$ the agent only takes the immediate rewards into account. As $\gamma$ approaches 1 the agent gives more and more weight to the future rewards.

### 2.1.4 Policies and Value Functions

To determine what action the agent should take next, an estimation is needed of how good it is to be in a certain state in regards to the overall goal of maximizing the return. For this the **policy** $\boldsymbol{\pi(a_t, s_t)}$ can be defined, as the probability of selecting action $a_t$ in state $s_t$. To evaluate this the **state-value function** $\mathbf{V^\pi(s)}$ is defined. Informally, this is the expected ($\mathbb{E}$) discounted return when starting in state $s_t$ and following a policy $\pi$.

$$V^\pi(s_t) = \mathbb{E}_\pi[R_t|S = s_t] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|S = s_t\right]. \tag{2.9}$$

Furthermore, the **action-value function** $\mathbf{Q^\pi(s_t, a_t)}$ is defined. This is the expected return when starting in state $s_t$ and taking action $a_t$ following a policy $\pi$.

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi[R_t|S = s_t, A = a_t] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|S = s_t, A = a_t\right]. \tag{2.10}$$

The state value function and the action value function can be defined recursively. These recursive definitions are called the **Bellmann equations**

$$V^\pi(s_t) = \mathbb{E}_{a_t\sim\pi(\cdot|S), s_{t+1}\sim P(\cdot|s_t,a_t)}\left(r_{t+1} + \gamma V^\pi(s_{t+1})\right), \tag{2.11}$$

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}\sim P(\cdot|s_t,a_t)}\Big(r_{t+1} + \gamma\mathbb{E}_{a_{t+1}\sim\pi(\cdot|s_{t+1})}\left(Q^\pi(s_{t+1}, a_{t+1})\right)\Big), \tag{2.12}$$

where the notation $\pi(\cdot|S)$ refers to a function call the policy $\pi$ takes from the states $S$. The "$\sim$" symbolizes that the action $a_t$ is selected according to the policy $\pi$. The goal of the RL problem is to find an optimal policy. To find this the optimal value functions are defined as

$$V^*(s_t) = \max_\pi V^\pi(s_t) \quad \forall s_t \in S, \tag{2.13}$$

$$Q^*(s_t, a_t) = \max_\pi Q^\pi(s_t, a_t) \quad \forall s_t \in S, \quad \forall a_t \in A. \tag{2.14}$$

A policy is considered better or equal to another policy if the expected return is greater than or equal to that of the other policy for all states. The **expected return** $\mathbf{J}(\boldsymbol{\pi})$ associated to a policy is given by

$$J(\pi) = \mathbb{E}_{t\sim P(\cdot|\pi)}\left\{R_t\right\}. \tag{2.15}$$

The reinforcement learning problem then can be written as

$$\pi^* = \arg\max_\pi J(\pi), \tag{2.16}$$

where $\pi^*$ describes the optimal policy. While there can be many optimal policies $\pi^*$, they all share the same optimal state-value function $V^*(s_t)$ and action-value function $Q^*(s_t, a_t)$. An optimal solution can be found by employing the Bellmann equations 2.12 and using Eq. 2.10 and Eq. 2.9 to find the **Bellmann optimality equations**

$$V^*(s_t) = \max_a \mathbb{E}\left\{r_{t+1} + \gamma V^*(s_{t+1})\,|\,S = s_t\right\}, \tag{2.17}$$

$$Q^*(s_t, a_t) = \mathbb{E}\left\{r_{t+1} + \gamma\max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})\,|\,S = s_t, A = a_t\right\}. \tag{2.18}$$

Solving these equations leads to the optimal state-value or action-value function. Using these, an optimal policy can be determined by maximizing the associated value function.

### 2.1.5 Deep Reinforcement Learning

Classic RL solution methods are based on tabular methods. Tabular methods use tables, or arrays to represent the value functions or policies. They explicitly store and update the value of each state or state-action space in a table. This makes them well-suited for small state spaces but causes them to have a problem with scalability. For large MDPs there are too many states and actions to visit all of them in training. Hence, the state-value function becomes increasingly large for complex problems. Saving them in a tabular manner gives rise to memory problems. A solution is to replace the value tables through using approximation methods. With approximation methods the value function can be estimated via function approximation. In Deep Reinforcement Learning (DRL) neural networks serve as non-linear function approximations. One of these DRL methods is the Soft Actor Critic Algorithm, which will be discussed in the following section.

## 2.2 Soft Actor Critic Algorithm

The Soft Actor Critic (SAC) algorithm is an off-policy maximum entropy deep reinforcement learning algorithm with a stochastic actor. The RL problem in Eq. 2.15 is extended by an entropy objective. The goal is to maximize the expected reward and the entropy. SAC makes use of three major concepts.

- Actor critic with separate policy and value function networks,
- Off-policy algorithm and
- Entropy maximization.

These concepts and the implementation of SAC shall be discussed in the following. This section is based on [14].

### 2.2.1 Actor Critic Algorithms

Actor critic algorithms are usually based on policy iteration. Policy iteration describes the alternation between policy evaluation and policy improvement. In the policy evaluation step, the Q-value function is updated on the basis of the actor. This is referred to as the **critic**. In the policy improvement step, the **actor** improves the policy $\pi_\phi$ by updating its parameters $\phi$ in the direction provided by the critic.

### 2.2.2 Off-policy Algorithm

The off-policy algorithm helps in increasing the sample efficiency through the reuse of previously collected data. The SAC algorithm makes use of an **experience replay buffer**, which is a set $\mathcal{D}$ of previous experience. These are used to update the policy and the critic network.

### 2.2.3 Entropy Maximization

The reward term is extended by an entropy objective $\mathcal{H}(\pi(\cdot|s_t))$.

$$J(\pi) = \sum_{k=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right], \tag{2.19}$$

where $\alpha$ describes the temperature parameter. This determines the relative importance of the entropy term against the reward term. Therefore, it controls the stochasticity of the optimal policy. The $\rho_\pi(s_t)$ and $\rho_\pi(s_t, a_t)$ term describe the state and state-action marginals of the trajectory distribution induced by a policy $\pi(s_t, a_t)$. The extension by the entropy objective has the advantage of incentivizing the policy to explore the state-action space widely.

### 2.2.4 Implementation

In practice, SAC makes use of neural networks as function approximators for the parameterized soft Q-function $Q_\theta(s_t, a_t)$ and the tractable policy $\pi_\phi(a_t, s_t)$, where $\theta, \phi$ describe the parameters of the neural networks. All of these are optimized with **stochastic gradient descent**. For this derivation the temperature parameter $\alpha$ is treated as constant, but later an extension of SAC is described which adjusts the temperature automatically to match an entropy target. The soft Q-function is trained by minimizing the soft Bellmann residual error:

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(s_t, a_t) - (r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\bar{\theta}}(s_{t+1})])\right)^2\right]. \tag{2.20}$$

The value function is implicitly parameterized through the soft Q-function through

$$V(s_t) = \mathbb{E}_{a_t\sim\pi}\big[Q(s_t, a_t) - \alpha \log \pi(a_t|s_t)\big]. \tag{2.21}$$

The $\log$ term represents the entropy term $\mathcal{H}$ as the Shannon entropy. The Q-function is updated through the gradient estimate:

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(a_t, s_t)\Big[Q_\theta(s_t, a_t) - \big(r(s_t, a_t) + \gamma\left[Q_{\bar{\theta}}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\phi(a_{t+1}|s_{t+1})\right]\big)\Big]. \tag{2.22}$$

The **target network function**, denoted by $Q_{\bar{\theta}}$, is introduced to stabilize training in the algorithm. In order to achieve this, a second network is used, which is updated through the use of **polyak averaging**, a technique that involves computing a moving average of the Q-network. This is updated once per update of the main network, and lags behind it. To prevent overestimation of the Q-values, two Q-functions are trained independently and the minimum of their respective target networks are used to update the actor.

The policy parameters can be learned directly by minimizing the **Kullback-Leibler Divergence (DKL)**. The DKL measures how one probability distribution is different from a second. In SAC the distribution of the policy function is compared to the distribution of the Q-function.

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}} \left[\mathbb{E}_{a_t\sim\pi_\phi} \left[\alpha \log(\pi_\phi(a_t|s_t)) - Q_\theta(a_t, s_t)\right]\right]. \tag{2.23}$$

To minimize this the reparametrization trick is used. The policy is reparameterized using an NN transformation:

$$a_t = f_\phi(\epsilon_t; s_t), \tag{2.24}$$

where $\epsilon$ represents an input noise vector sampled form a Gaussian distribution. The policy term then modifies to:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} \left[ \alpha \log \pi_\phi(f_\phi(\epsilon_t; s_t)|s_t) - Q_\theta(s_t, f_\phi(\epsilon_t; s_t)) \right]. \tag{2.25}$$

This again can be estimated using gradient descent

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(a_t|s_t) + (\nabla_{a_t} \alpha \log \pi_\phi(a_t|s_t) - \nabla_{a_t} Q(s_t, a_t)) \nabla_\phi f_\phi(\epsilon_t; s_t). \tag{2.26}$$

## Automating Entropy Adjustment for Maximum Entropy RL

In the previous calculations the temperature parameter $\alpha$ was assumed to be constant. In practice, finding the optimal temperature is not trivial and introduces a parameter that needs to be tuned. To counteract this, the process is automated by formulating a different maximum entropy reinforcement learning objective. The problem is formulated as a **constrained optimization problem** where the average entropy of the policy is constrained.

$$\max_{\pi_0:T} \mathbb{E}_{\rho_\pi} \left[ \sum_{t=0}^{T} r(s_t, a_t) \right] \quad \text{s.t.} \quad \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[ -\log(\pi_t(a_t|s_t)) \right] \quad \geq \quad \mathcal{H} \; \forall t, \tag{2.27}$$

where $\mathcal{H}$ is the desired minimum entropy. This can be rewritten through an (approximate) dynamic programming approach solving for the policy backward through time

$$\max_{\pi_0} \left( \mathbb{E}\left[ r(s_0, a_0) \right] + \max_{\pi_1} \left( \mathbb{E}[...] + \max_{\pi_T} \mathbb{E}[r(s_T, a_T)] \right) \right). \tag{2.28}$$

Starting from the last time step, the constrained maximization is changed to the dual problem.

$$\max_{\pi_T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[ r(s_T, a_T) \right] = \min_{\alpha_T \geq 0} \max_{\pi_T} \mathbb{E} \left[ r(s_T, a_T) - \alpha_T \log \pi(a_T|s_T) \right] - \alpha_T \mathcal{H}, \tag{2.29}$$

where $\alpha_T$ is the dual variable. The optimal dual variable $\alpha_T^*$ can be solved for by

$$\arg\min_{\alpha_T} \mathbb{E}_{(s_t,a_t)\sim\pi_t^*} \left[ -\alpha_T \log \pi_T^*(a_T|s_T; \alpha_T) - \alpha_T \mathcal{H} \right] \tag{2.30}$$

Through using the recursive definition of the soft Q-function, subject to the entropy constraints and using the dual problem the following equation can be determined

$$\max_{\pi_{T-1}} \left( \mathbb{E}[r(s_{T-1}, a_{T-1})] + \max_{\pi_T} \mathbb{E}[r(s_T, a_T)] \right) = \tag{2.31}$$

$$\min_{\alpha \geq 0} \max_{\pi_{T-1}} \left( \mathbb{E}[Q_{T-1}^*(s_{T-1}, a_{T-1})] - \mathbb{E}[\alpha_{T-1} \log \pi(a_{T-1}|s_{T-1})] - \alpha_{T-1}\mathcal{H} \right) + \alpha_T^* \mathcal{H}. \tag{2.32}$$

In this way, the algorithm can proceed backwards in time and can recursively optimize Eq. 2.27. The optimal dual variable $\alpha_t^*$ can be found similarly after solving for $Q_t^*$ and $\pi_t^*$ as

$$\alpha_t^* = \arg\min_{\alpha_t} \mathbb{E}_{a_t \sim \pi_t^*} \left[ -\alpha \log \pi_t^*(a_t|s_t; \alpha_t) - \alpha \bar{\mathcal{H}} \right]. \tag{2.33}$$

In summary, the solution to Eq. 2.33 and the policy and soft Q-function updates described previously constitute the core of the SAC algorithm. In addition to learning the Q-functions and the policy the temperature term $\alpha$ is also learned by minimizing the dual objective in Eq. 2.27. This is done by dual gradient descent. The gradients for $\alpha$ are updated with the following objective

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t} \left[ -\alpha \log \pi_t(a_t|s_t) - \alpha \bar{\mathcal{H}} \right] . \tag{2.34}$$

The SAC algorithm alternates between collecting experience from interacting with the environment and updating the function approximators using stochastic gradients. These batches are sampled from the replay buffer $\mathcal{D}$. Through extending the reward by an entropy objective, SAC encourages exploration and stabilizes training in respect to hyperparameters. Fig. 2.2 shows the pseudocode for the SAC algorithm.

**Algorithm 1** Soft Actor-Critic

```
Input: $\theta_1, \theta_2, \phi$                                        ▷ Initial parameters
  $\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$                         ▷ Initialize target network weights
  $\mathcal{D} \leftarrow \emptyset$                                       ▷ Initialize an empty replay pool
  for each iteration do
      for each environment step do
          $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$                        ▷ Sample action from the policy
          $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$                 ▷ Sample transition from the environment
          $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$   ▷ Store the transition in the replay pool
      end for
      for each gradient step do
          $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$     ▷ Update the Q-function parameters
          $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$                   ▷ Update policy weights
          $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$                    ▷ Adjust temperature
          $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$      ▷ Update target network weights
      end for
  end for
Output: $\theta_1, \theta_2, \phi$                                       ▷ Optimized parameters
```

Figure 2.2: The SAC algorithm. It makes use of target networks with the parameters $\bar{\theta}_i$ and alternates between collecting experience from the environment, storing these in the replay buffer $\mathcal{D}$ and updating the Q-function, the policy weights and the temperature through gradient descent from batches sampled from the replay buffer $\mathcal{D}$ [14].

## 2.3 Astrodynamic Framework

This chapter provides the physical background to simulate a spacecraft. It describes the assumptions made in the simulation and gives an overview about the software used. For the astrodynamic framework the Basilisk software [11] is used. This software provides Python modules written in C/C++. It is being developed by the University of Colorado AVS Lab [15] and the Laboratory for Atmospheric and Space Physics (LASP) [16]. For visualization the accompanying software Vizard is used [17].

### 2.3.1 Coordinate Frames

Before specifying the spacecraft dynamics a common reference frame needs to be defined. In the simulation two different coordinate frames are used. The first one is the **earth-centered inertial frame (ECI)** $\mathcal{N}$, displayed in Fig. 2.3. The center of its origin is at the center of the earth. For inertial coordinate frames a fixed reference direction has to be defined. Here, the vernal equinox ♈ is used. This describes the line of intersection between the equator and the ecliptic plane. The sun passes this point twice per year. In the ECI, the $I$ axis points towards the vernal equinox, the $J$ axis is $90°$C to the east in the equatorial plane and the $K$ axis extends through the North pole [18].

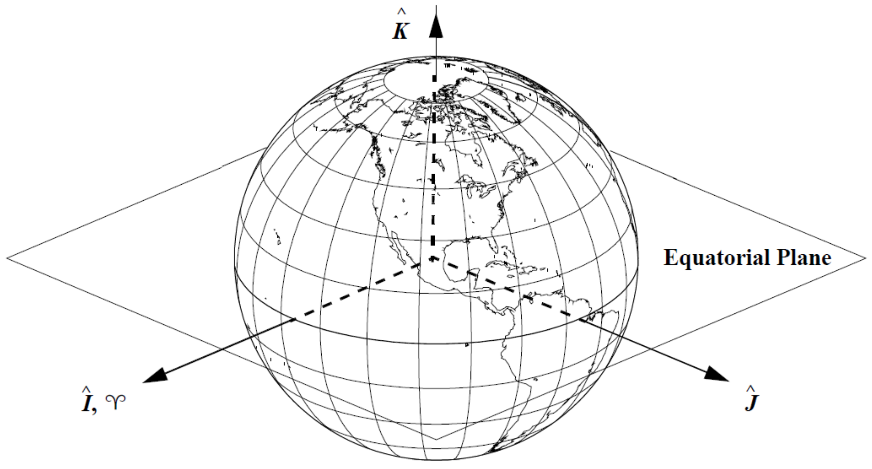


Figure 2.3: The earth-centered inertial frame $\mathcal{N}$ [18]. The center of its origin is at the center of the earth. The vernal equinox ♈ defines the fixed reference direction for the ECI. It describes the line of intersection between the equatorial plane and the ecliptic plane.

Another coordinate frame used is the **spacecraft body frame** $\mathcal{B}$, displayed in Fig. 2.4. This describes a reference frame that is fixed to the rigid body of the spacecraft [19]. The origin of the frame is denoted as $B$. $B_c$ describes the center of mass of the hub. $r_{\mathcal{B}/\mathcal{N}}$ describes the position of the spacecraft in reference to the inertial ECI frame $\mathcal{N}$ .

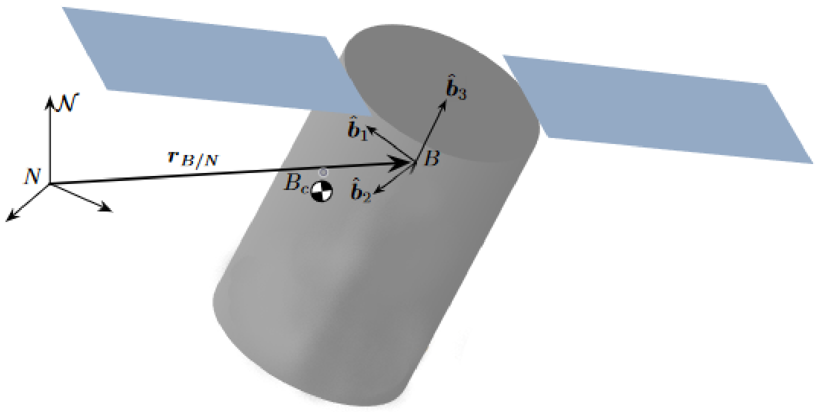


Figure 2.4: The spacecraft body-centered frame $\mathcal{B}$. $B_c$ describes the center of mass of the hub. Modified from [19].

The angular velocity $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$ between the ECI $\mathcal{N}$ and the spacecraft body frame $\mathcal{B}$ is given by

$$\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} = [\omega_1, \omega_2, \omega_3]^T. \tag{2.35}$$

### 2.3.2 Spacecraft Dynamics

In Basilisk the spacecraft is modeled as a rigid body. In the following the spacecraft is referred to as the "hub". The rotational dynamics with respect to the body-centered frame $B$ and the center of mass location $B_C$ follow **Euler's law for a rigid body**:

$$\mathbf{H}_{hub,B} = [I_{hub,B_C}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + m_{hub}\mathbf{r}_{B_C/B} \times \dot{\mathbf{r}}_{B_C/B}, \tag{2.36}$$

where $\mathbf{H}_{hub,B}$ describes the angular momentum of the hub, $I_{hub,B_C}$ is the inertia matrix of the spacecraft hub, $m_{hub}$ is the mass of the spacecraft hub, $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$ is the angular velocity between the inertial and body frame as defined in Eq. 2.35 and $r_{B_C/B}$ and its derivative is the location and velocity of the point B in the body frame.

## Reaction Wheels

Satellites are often equipped with momentum exchange devices to perform rotational maneuvers. A common choice for these are **reaction wheels (RWs)**. A reaction wheel is a body fixed wheel, which is spun up or down to exert a torque on the spacecraft. Due to conservation of angular momentum spinning up the reaction wheel causes the spacecraft to counter rotate, thus changing the attitude. In the simulation, the reaction wheels are connected to the rigid body hub according to Fig. 2.5.

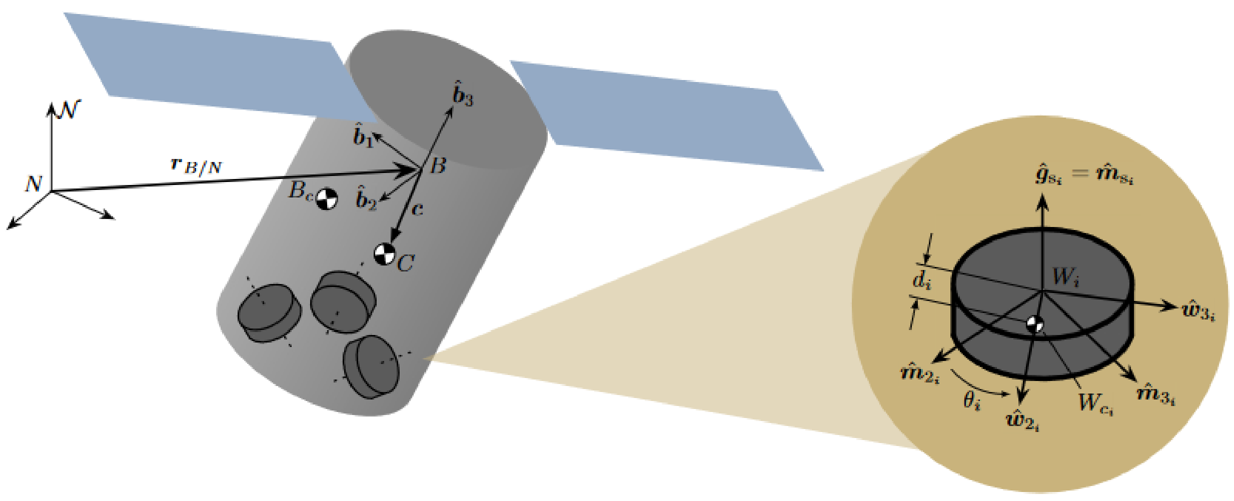


Figure 2.5: The RWs and their respective frames. They are attached to the spacecraft rigid body hub. The center of masses for the rigid body hub, for the reaction wheels and for the spacecraft including the reaction wheels are denoted as $B_C$, $W_{C_i}$ and $C$, respectively [19].

The frame and variable definitions used for the problem are

$$B_c := \text{center of mass of spacecraft}, \tag{2.37}$$

$$B := \text{ origin of the B-frame} \tag{2.38}$$

$$N_{rw} := \text{Total number of RWs with center of mass labeled as } W_{C_i}, \tag{2.39}$$

$$\mathcal{B} := \{\hat{b}_1, \hat{b}_2, \hat{b}_3\}, \text{ body-fixed frame}, \tag{2.40}$$

$$\mathcal{W}_i := \{\hat{g}_{s_i}, \hat{w}_{2_i}, \hat{w}_{3_i}\}, \text{ wheel-fixed frame of } i\text{-th RW}, \tag{2.41}$$

$$\mathcal{M}_i := \{\hat{m}_{s_i}, \hat{m}_{2_i}, \hat{m}_{3_i}\}, \text{ body-fixed motor frame of } i\text{-th RW}. \tag{2.42}$$

The variable definition can also be seen in Fig. 2.5. $\hat{g}_{s_i}$ is aligned with the spin axis of the RW. This is denoted by the $s_i$ subscript. $\hat{w}_{2_i}$ is perpendicular to $\hat{g}_{s_i}$ and points in the direction of the center of mass of the reaction wheel $W_{C_i}$, $\hat{w}_{3_i}$ completes the coordinate system according to the right hand rule. The $\mathcal{M}_i$ frame is defined as being equal to the $\mathcal{W}_i$ frame at the beginning of the simulation and is offset by an angle of $\theta_i$ by the $\hat{m}_{s_i} = \hat{g}_{s_i}$ axes. Variable $d_i$ is the center of mass offset of the RW. The time derivative of a vector $\mathbf{v}$ as seen by the inertial frame $\mathcal{N}$ is denoted as ${}^{\mathcal{N}}d\mathbf{v}/dt = \dot{\mathbf{v}}$ and with respect to the body-fixed frame $\mathcal{B}$ as ${}^{\mathcal{B}}d\mathbf{v}/dt = \mathbf{v}'$. The point $C$ in the body hub defines the center of mass of the system $sc$ including reaction wheels.

The vector $\mathbf{c}$ is defined as pointing from the origin of the body frame $B$ to the systems center of mass as

$$\mathbf{c} = \frac{1}{m_{sc}} \left( m_B \mathbf{r}_{B_C/B} + \sum_{i=1}^{N_{rw}} m_{W_i} \mathbf{r}_{W_{Ci/B}} \right). \tag{2.43}$$

Considering reaction wheels, the angular momentum of the spacecraft $sc$ in Eq. 2.36 is extended through the angular momentum of the $i$-th reaction wheels:

$$\mathbf{H}_{sc,B} = \mathbf{H}_{hub,B} + \sum_{i=1}^{N_{rw}} \mathbf{H}_{rw_i,B}, \text{ with} \tag{2.44}$$

$$\mathbf{H}_{rw_i,B} = [I_{rw_i,W_{C_i}}](\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + \Omega_i \hat{\mathbf{g}}_{s_i}) + m_{rw_i} \mathbf{r}_{W_{C_i}/B} \times \dot{\mathbf{r}}_{W_{C_i}/B}, \tag{2.45}$$

where $\Omega_i$ is the wheel speed of the $i$-th RW. To develop the equation of motions the time derivative of the angular momentum needs to be calculated

$$\dot{\mathbf{H}}_{sc,B} = \dot{\mathbf{H}}_{hub,B} + \sum_{i=1}^{N_{rw}} \dot{\mathbf{H}}_{rw_i,B}. \tag{2.46}$$

The hub and the reaction wheel angular momentum derivative can be found after some calculation according to [19] as

$$\dot{\mathbf{H}}_{hub,B} = [I_{hub,B}]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + [\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][I_{hub,B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \tag{2.47}$$

$$\dot{\mathbf{H}}_{rw_i,B} = [I_{W_i,B}]'\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + [I_{W_i,B}]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + [\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][I_{W_i,B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \tag{2.48}$$

$$+ [I_{W_i,W_{C_i}}]'\Omega_i \hat{\mathbf{g}}_{s_i} + [I_{W_i,W_{C_i}}]\dot{\Omega}_i \hat{\mathbf{g}}_{s_i} \tag{2.49}$$

$$+ m_{W_i} \mathbf{r}_{W_{C_i}/B} \times (d_i \dot{\Omega}_i \hat{\boldsymbol{\omega}}_{3_i} - d_i \dot{\Omega}_i^2 \hat{\boldsymbol{\omega}}_{2_i}) \tag{2.50}$$

$$+ m_{W_i} \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \times (\mathbf{r}_{W_{C_i}/B} \times \mathbf{r}'_{W_{C_i}/B}). \tag{2.51}$$

The tilde operator designates the skew symmetric matrix, i.e. $[\tilde{\omega}]\boldsymbol{v} = \boldsymbol{\omega} \times \boldsymbol{v}$. This notation has the benefit of being frame independent and represents the cross product in a compact manner. The full rotational equation of motion is

$$m_{sc}[\tilde{\mathbf{c}}]\ddot{\mathbf{r}}_{\mathcal{B}/\mathcal{N}} + [I_{sc,B}]\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} + \sum_{i=1}^{N_{rw}} J_{s_i} \hat{\mathbf{g}} s_i \dot{\Omega}_i = -[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][I_{sc,B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} - \sum_{i=1}^{N_{rw}} (\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \times J_{s_i} \Omega_i \hat{\mathbf{g}}_{s_i}) + \mathbf{L}_B, \tag{2.52}$$

where $\mathbf{L}_B$ is the torque about point $B$. The motor torque equation describes how the motor torque $u_{s_i}$ relates to the wheel speed derivative $\dot{\Omega}_i$ and can be written as

$$\dot{\Omega}_i = \frac{u_{s_i}}{J_{s_i}} - \hat{\mathbf{g}}_{s_i}^T \dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}. \tag{2.53}$$

Combining Eq. 2.52 and Eq. 2.53 yields

$$m_{sc}[\tilde{\mathbf{c}}]\ddot{\mathbf{r}}_{\mathcal{B}/\mathcal{N}} + ([I_{sc,B}] - \sum_{i=1}^{N_{rw}} J_{s_i} \hat{\mathbf{g}}_{s_i} \hat{\mathbf{g}}_{s_i}^T)\dot{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}} \tag{2.54}$$

$$= -[\tilde{\boldsymbol{\omega}}_{\mathcal{B}/\mathcal{N}}][I_{sc,B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} - \sum_{i=1}^{N_{rw}} (\hat{\mathbf{g}}_{s_i} u_{s_i} + \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} \times J_{s_i} \Omega_i \hat{\mathbf{g}}_{s_i}) - [I'_{sc,B}]\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}} + \mathbf{L}_B. \tag{2.55}$$

In this thesis the satellite is equipped with three balanced reaction wheels, each aligned with one spin axis $\hat{\mathbf{g}}_{s_i}$. This enables each reaction wheel to control one spin axis, without influencing the others. The equations of motions described above are used to model the rotational behavior in Basilisk. The problem that arises with using reaction wheels is **saturation**. This phenomenon describes that the reaction wheels have stored more momentum than their maximum wheel speed allows. This can happen either through external forces or if the attitude control system exceeds the maximum wheel speed. As a consequence attitude control with reaction wheels is no longer possible. The excess momentum needs to be dumped using a secondary attitude control system [19, 20].

### 2.3.3 Modified Rodrigues Parameters

As an attitude representation Modified Rodrigues Parameters (MRP) are selected. The MRP vector $\sigma$ is defined in terms of the Euler parameters. The Euler parameters are defined with respect to the principal rotation $\Phi$ as

$$\beta_0 = \cos\left(\frac{\Phi}{2}\right), \tag{2.56}$$

$$\beta_i = e_i \sin\left(\frac{\Phi}{2}\right), \quad i = 1, 2, 3. \tag{2.57}$$

The MRP vector $\sigma$ is then defined as

$$\sigma_i = \frac{\beta_i}{1+\beta_0}, \quad i = 1, 2, 3, \tag{2.58}$$

with the inverse equations

$$\beta_0 = \frac{1-\sigma^2}{1+\sigma^2}, \tag{2.59}$$

$$\beta_i = \frac{2\sigma_i}{1+\sigma^2}. \tag{2.60}$$

The MRP can be expressed in terms of the principal rotation elements as

$$\sigma = \tan\frac{\Phi}{4}\hat{e}. \tag{2.61}$$

For small angles, the small angle approximation holds for the tangents. In comparison to the Euler parameters, this improves the domain of linearity. The projection of the alternate Euler parameter vector $-\beta$ results in a distinct set of shadow images $\sigma_i^S$. Each MRP is an equally valid attitude description. Therefore, one can switch between these two vectors by using

$$\sigma_i^S = \frac{-\beta_i}{1-\beta_0} = \frac{-\sigma_i}{\sigma^2}. \tag{2.62}$$

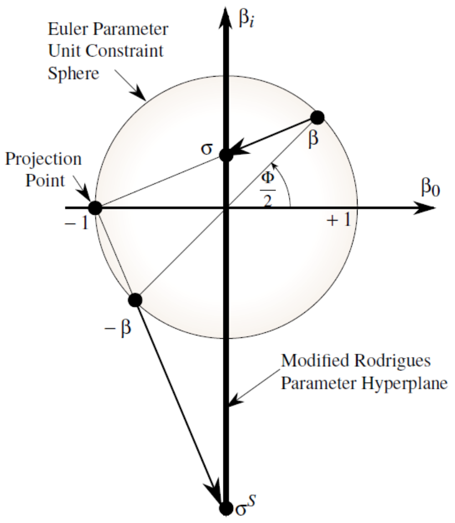

Euler Parameter Unit Constraint Sphere
β_i
Projection Point
σ
β
Φ/2
β_0
−1
+1
−β
Modified Rodrigues Parameter Hyperplane
σ^S

Figure 2.6: The stereographic projection of the MRPs and its shadow images into the Euler Parameter unit sphere. $\sigma$ describes the attitude that is projected unto the point $\beta = (-1, 0, 0, 0)$. As a $360°$ degree principal rotation is approached ($\beta_0 \to -1$), the projection of the corresponding point on the constraint sphere goes to infinity. But the projection of the alternate euler vector $-\beta = (1, 0, 0, 0)$ results in a distinct set of shadow images, avoiding the singularity. Each vector can be used as an attitude description [20].

While the original MRP and the shadow MRP are both individually singular, combining these can avoid singularity. The MRP and its shadow set have completely opposite singular behaviors. The original MRP has a singularity at $\Phi = 360°$ the shadow set has its singularity at $\Phi = 0°$. One set defines the short rotation, while the other one defines the long rotation. Implementing both of these, a complete $360°$ rotation can be described without singularity. The only problem that needs to be addressed is the discontinuity when switching between these. A typical switching surface is $\sigma^2 = 1$. In general this behavior can be summarized by

$$|\sigma| \leq 1 \text{ if } \Phi \leq 180°, \tag{2.63}$$
$$|\sigma| \geq 1 \text{ if } \Phi \geq 180°, \tag{2.64}$$
$$|\sigma| = 1 \text{ if } \Phi = 180°. \tag{2.65}$$

Using this representation has several benefits: The MRP linearizes well for small angles. In the simulation, the switching between these two sets happens seamlessly and the resulting attitude is always given as the shortest rotational path. Therefore, the MRP has a bounded maximum norm of 1, which is very useful to interpret the feedback gained from the attitude error [20].

# 3 Scenarios

Before investigating the space debris scenario, where the mass changes, a few simpler scenarios are tested. This is to study the feasibility of using a DRL and especially a SAC algorithm for the Attitude Control System (ACS) of a satellite. After starting with the simplest scenario the complexity is increased continuously. In the following chapter these different scenarios are presented and discussed.

## 3.1 Simulation Architecture

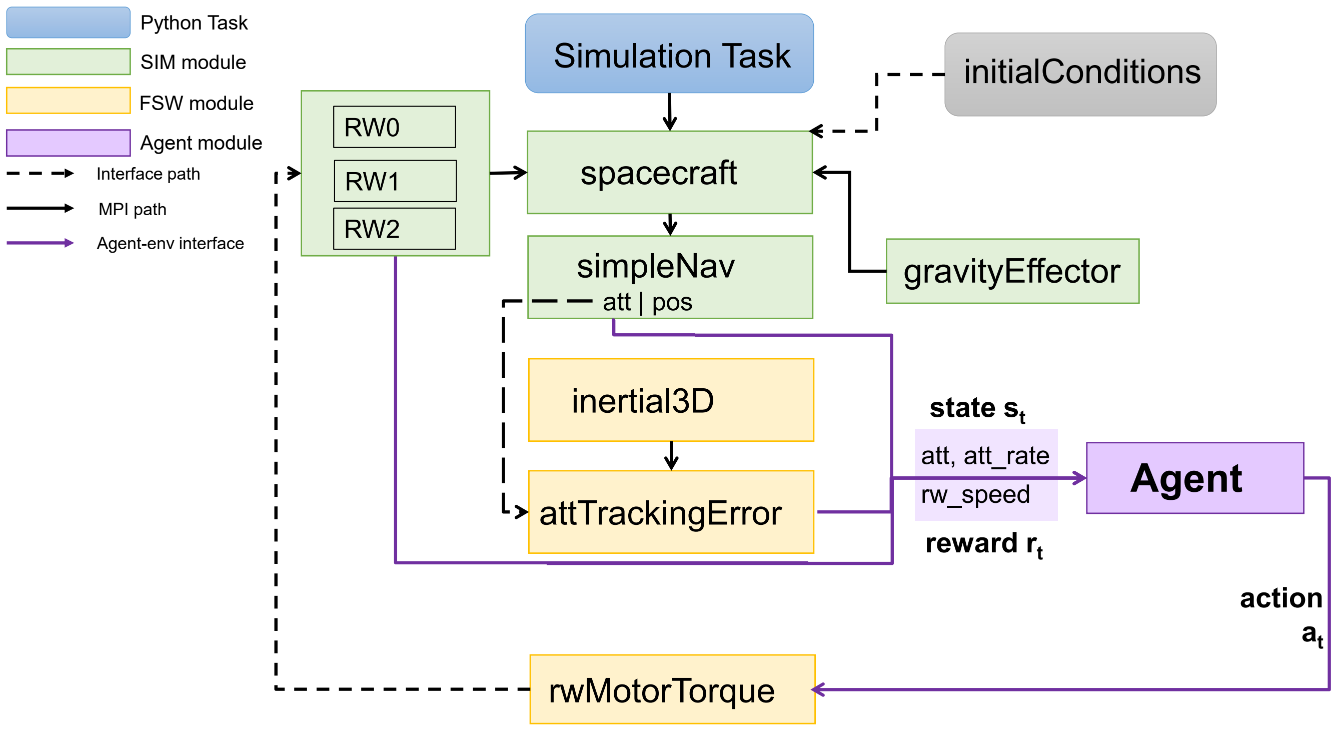


Figure 3.1: The simulation architecture and agent interface. The simulation (SIM) and flight software (FSW) modules communicate through the message passing interface (MPI). The agent receives the state $s_t$ from the environment. The state contains the attitude $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$, the attitude rate $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$ and the wheel speed $\boldsymbol{\Omega}$. The agent then gives an action $a_t$ in the form of a torque vector into the environment. The communication between agent and environment is based on an OpenAI Gym interface. Modified from [21].

In Fig. 3.1 the simulation architecture is shown. The *spacecraft* module describes the spacecraft as a rigid body. The *gravityEffector* module provides the necessary orbital dynamics for a low earth orbit. The *simpleNav* module provides the current attitude, attitude rate and position of the spacecraft. The *inertial3D* module provides the attitude goal. In the *attTrackingError* module this is used to compute the difference between the desired attitude and the current attitude. This is then given as a feedback to the agent. The agent provides a torque input to the *rwMotorTorque* module, which maps these onto the three reaction wheels. The reaction wheel speed $\Omega$ is then given as a feedback to the agent. Through the *initialConditions* module the different scenarios described in the following can be implemented. The agent and the simulation communicate through an OpenAI Gym interface. The agent is trained using the SAC algorithm from rllib [22]. The simulation environment has no aerodynamic, gravitational, or solar radiation pressure effects.

## 3.2 Stabilizing the Satellite with Reaction Wheels

A satellite equipped with reaction wheels is simulated. These are the used for the attitude control. Three reaction wheels are set up to control the three axes of the satellite. For describing the attitude tracking error $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$ and the attitude rate $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$ the MRPs described in section 2.3.3 are used. As action $\mathbf{a}$ the torque vector $u$ on the reaction wheels is used.

$$\mathbf{a} = [u_1, u_2, u_3]. \tag{3.1}$$

The reaction wheels are modeled after a standard set of reaction wheels like the Honeywell HR16 [23]. In compliance with this the action space boundaries are set to $u_{\min} = -0.2\,\text{Nm}$ and $u_{\max} = 0.2\,\text{Nm}$.
The state $\mathbf{s}$ is defined, with the reaction wheel speed $\boldsymbol{\Omega}$ as the following

$$\mathbf{s} = [\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}, \boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}, \boldsymbol{\Omega}]. \tag{3.2}$$

**Constant Initial Attitude**
This scenario is the simplest scenario. The satellite is given a constant, fixed initial attitude of $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}\text{Init}} = [1, 0, 0]$ and no initial attitude rate $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}\text{Init}} = [0, 0, 0]$. In *inertial3D* the desired attitude is set to $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}\text{Desired}} = [0, 0, 0]$. Therefore, the agent has the goal to rotate the satellite by 180°.

**Vary Initial Attitude**
In this scenario, instead of using a fixed initial attitude, different initial attitudes $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}\text{Init}}$ and attitude rates $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}\text{Init}}$ are sampled through the *initialConditions* module. The sampling happens uniformly: All values within the given range have the same probability of being chosen. The goal state is $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}\text{Desired}} = [0, 0, 0]$. The agent has to handle several different initial attitudes and attitude rates, bringing the satellite to the desired orientation each time.

**Vary Initial Mass**
In this scenario the same assumptions as in the scenario "From a Fixed Starting Position" are employed. Instead of a fixed mass the mass is now sampled uniformly. This simulates the change of mass that occurs after the capture of the space debris object. The agent has the goal to rotate the satellite by 180°.
An overview of the different scenarios can be found in table 3.1.

Table 3.1: Comparison of the different scenarios. The trained agents are named for a more convenient reference in the rest of the work. Shown are initial and desired values and the mass of the simulated satellite.

| Scenario | Agent | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{Init}}$ | $\sigma_{\mathcal{B}/\mathcal{N}\text{Desired}}$ | $\omega^i_{\mathcal{B}/\mathcal{N}\text{Init}}[\frac{\text{rad}}{\text{s}}]$ | Mass[kg] |
|---|---|---|---|---|---|
| Constant Init Att | Constable | $[1,0,0]$ | $[0,0,0]$ | $[0,0,0]$ | 330 |
| Vary Init Att | Spinny | $0-1$ | $[0,0,0]$ | $0-0.1$ | 330 |
| Location Pointing | Spinny | $[1,0,0]$ | Time Dependent | $[0,0,0]$ | 330 |
| Vary Init Mass | Massive | $[1,0,0]$ | $[0,0,0]$ | $[0,0,0]$ | $10-1000$ |

### 3.2.1 Reward Modeling & RL Training Parameters

The main objective of the agent is to reach the desired orientation. For this purpose an attitude error is defined, on which the reward is based on. The attitude error is defined as the difference between the desired orientation and the current orientation of the satellite for each coordinate.

$$\sigma_{\mathcal{B}/\mathcal{N}} = \left[\sigma^1_{\mathcal{B}/\mathcal{N}}, \sigma^2_{\mathcal{B}/\mathcal{N}}, \sigma^3_{\mathcal{B}/\mathcal{N}}\right], \text{ with} \tag{3.3}$$

$$\sigma^i_{\mathcal{B}/\mathcal{N}} = \left(\sigma^i_{\mathcal{B}/\mathcal{N}\text{Desired}} - \sigma^i_{\mathcal{B}/\mathcal{N}\text{Current}}\right). \tag{3.4}$$

Using this definition the reward is

$$r = \begin{cases} -1, \text{ if } |\Omega_i| \geq \Omega_{max} = 6000\,\text{rpm} \\ -|\sigma_{\mathcal{B}/\mathcal{N}}|^2 + 0.1, \text{ if } \sigma^i_{\mathcal{B}/\mathcal{N}} < 0.05, \quad \forall i \\ -|\sigma_{\mathcal{B}/\mathcal{N}}|^2 + 0.05, \text{ if } \sigma^i_{\mathcal{B}/\mathcal{N}} \text{ and } \sigma^j_{\mathcal{B}/\mathcal{N}} < 0.05, \quad \forall i \neq j \\ -|\sigma_{\mathcal{B}/\mathcal{N}}|^2, \text{ else.} \end{cases} \tag{3.5}$$

The first term of the reward function is to avoid saturation of the reaction wheels, as described in section 2.3.2. The next objective is to minimize the norm of the attitude error $-|\sigma_{\mathcal{B}/\mathcal{N}}|^2$. Empirically it could be observed, that these two conditions alone did not achieve the desired orientation in each coordinate, since the norm of the attitude error is in a similar small range, even if not all coordinates are near the desired location. Therefore, conditions were added, which consider the coordinates separately and give additional incentive to the agent to bring all coordinates below a certain threshold. The condition in the second row gives an additional reward, if all coordinates are below 0.05. The condition in the third row gives a smaller additional reward, if at least two coordinates are below the threshold.
With $0 \leq |\sigma_{\mathcal{B}/\mathcal{N}}| \leq 1$ the minimum and maximum reward $r$ can be calculated

$$-1 \leq r \leq 0.1. \tag{3.6}$$

Using this definition and the number of steps in one episode the minimum and maximum return $R$ for one episode is

$$R_{\text{max}} = r_{\text{max}} \cdot \#\text{steps} = 0.1 \cdot 60 = 6. \tag{3.7}$$

$$R_{\text{min}} = r_{\text{min}} \cdot \#\text{steps} = -1 \cdot 60 = -60. \tag{3.8}$$

The reward function is the same throughout all scenarios. The RL hyperparameters used for the training are shown in table 3.2.

Table 3.2: The training parameters including the RL hyperparameters and the environment parameters. The learning rates refer to three different learning rates: The actor network, the Q-Value network and the $\alpha$ parameter. As activation function the rectified linear unit (ReLU) is chosen. This function returns the input if the input is positive, and zero otherwise.

| Parameter | Value |
|---|---|
| learning rates | $3 \cdot 10^{-4}$ |
| discount $\gamma$ | 0.99 |
| replay buffer size | 1000000 |
| number of hidden units | 256 |
| entropy target | $-\dim(A) = -3$ |
| nonlinearity | ReLU |
| target update interval | 1 |
| target smoothing coefficient $\tau$ | $5 \cdot 10^{-3}$ |
| time step | $10s$ |
| episode length | $600\,s = 10\,\text{min}$ |

## 3.3 Scenario: Constant Initial Attitude

In this scenario a fixed initial attitudes $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}\text{Init}} = [1, 0, 0]$ (rotation by 180°) and attitude rates $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}\text{Init}} = [0, 0, 0]\frac{\text{rad}}{\text{s}}$ are set. The goal state is $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}\text{Desired}} = [0, 0, 0]$. For future references the agent in this scenario will be referred to as *Agent Constable*.

### 3.3.1 Training of Agent Constable

The Basilisk software is used to simulate the space environment. The training for Agent Constable revealed problems with the underlying simulation software Basilisk. A slow but steady increasing memory consumption during the training could be observed. The memory leak was confirmed by the developers of Basilisk but the source could not yet be identified. As a consequence, it is not possible to train the agent continuously for an extended period of time. As mitigation, checkpoints of the trained agent models are saved and used as a starting point to continue the training. This has no influence on the final trained agent, but has an effect on the RL metrics used to evaluate the training. In this case, using already trained models cause an overfitting in the beginning. This means that the return starts rather high at the beginning of the training and then drops due to the agent overfitting to the limited existing data. As the agent collects more experience the return rises again. To better discuss the results, the return and mean Q-values are displayed for all checkpoints of the training in Fig. 3.2. The Q-values represent the expected discounted return the Agent receives, when in a state $s_t$ and taking an action $a_t$, following a policy $\pi$. The Q-values are updated using the Bellmann optimality equations (Eq. 2.18). When comparing them with the reward, they can provide valuable insights about the training. Agent Constable was trained for 39058 episodes.

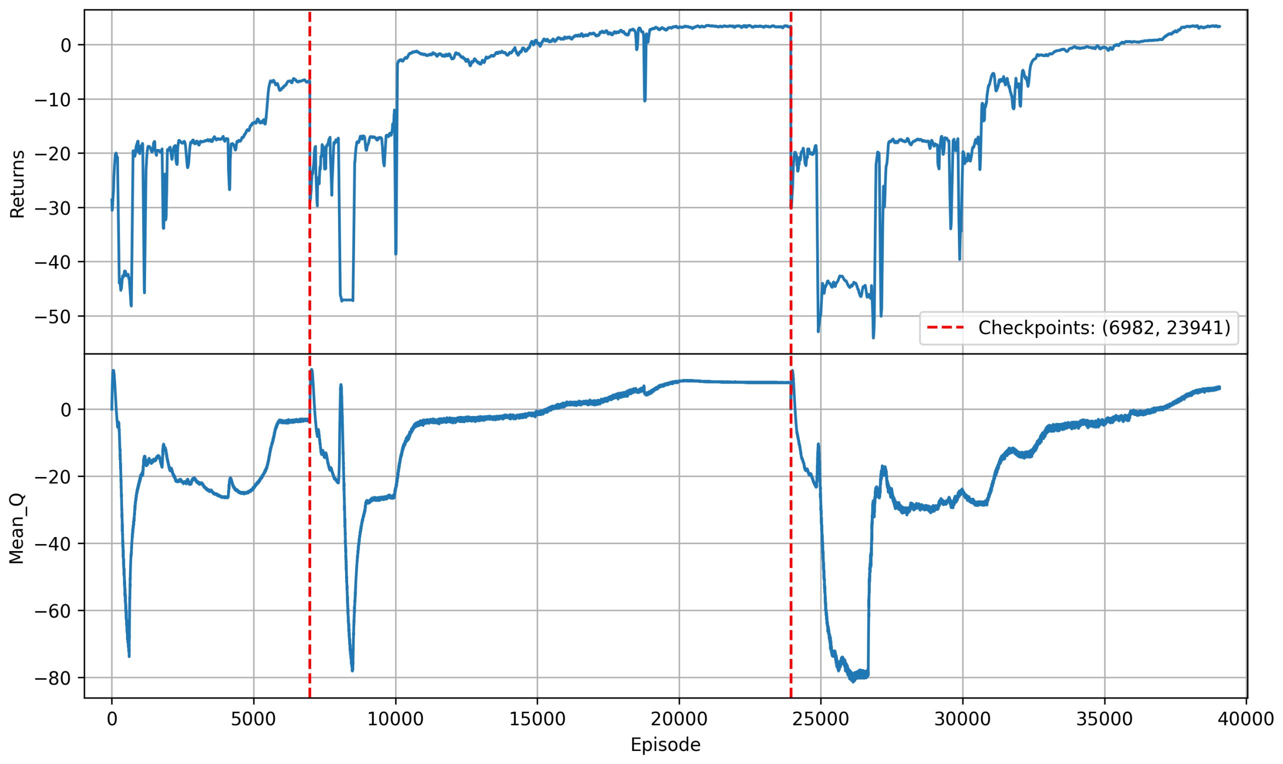


Figure 3.2: The return and Q-value over the course of the 39058 training episodes. Due to the memory leak the training was conducted in 3 sets: The first checkpoint is after 6982 episodes of training, the second checkpoint after 23941 episodes of training. The Q-value is able to approximate the return very accurately, even the dips in the return. The dips in the return are a sign that the agent was exploring and also considering actions that might not lead to a high return immediately. After the exploration phase the agent recovers and gradually brings the return up again.

In general, the Q-value is able to approximate the returns correctly. After the first checkpoint a peak in the Q-values appear, while there is a simultaneous dip in the return. This indicates that the agent is not able to approximate the expected return correctly yet. The agent is overestimating certain actions' values. This can indicate that the agent encountered something unexpected. The large negative return indicates that the actions of the agent have saturated the reaction wheels. The Q-value function is still learning which actions cause this saturation. After the first initial peak, the Q-value function also shows a large dip, indicating that the Q-value function has learned from this experience. In the end both the return and the mean Q-values converge.

### 3.3.2 Evaluation of Trained Agent Constable

To evaluate the agent, the trained agent is tested in the environment for 100 episodes. The 100 episodes are then evaluated on the basis of the following metrics:

**Return $R$ :** The mean return reached in each episode, including the standard deviation.

**$|\sigma_{\mathcal{B}/\mathcal{N}\text{final}}|$ :** The mean norm and standard deviation of the final 3D attitude error vector.

**$\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ :** All final attitude vectors are transformed into one list by concatenating their components. Then the mean and standard deviation of that list is calculated.

**Confidence Interval (CI) for $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ :** Gives an indication of where the final attitude error for all components will lie, with 95% confidence.

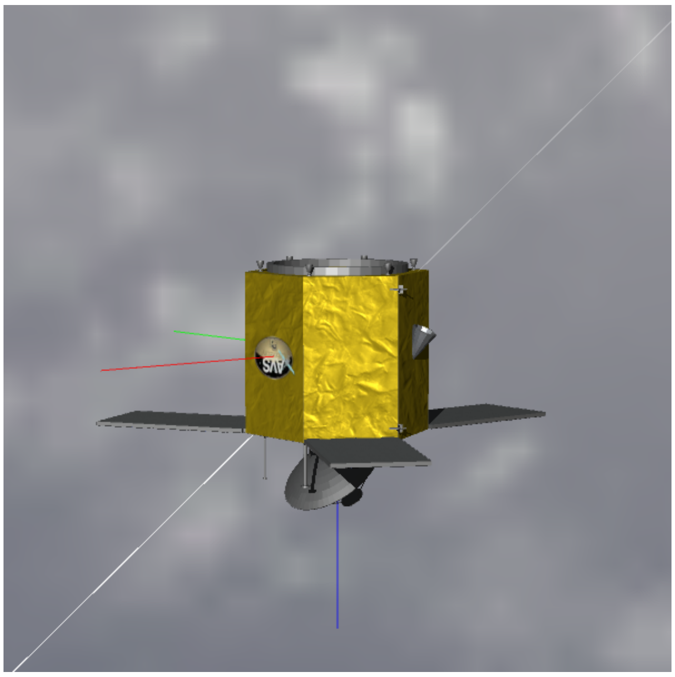
**$\omega^i_{\mathcal{B}/\mathcal{N}\text{final}}$ :** All final attitude rate vectors are transformed into one list by concatenating their components. Then the mean and standard deviation of that list is calculated.

The results of the 100 episodes from the discussed metrics is listed in table 3.3.

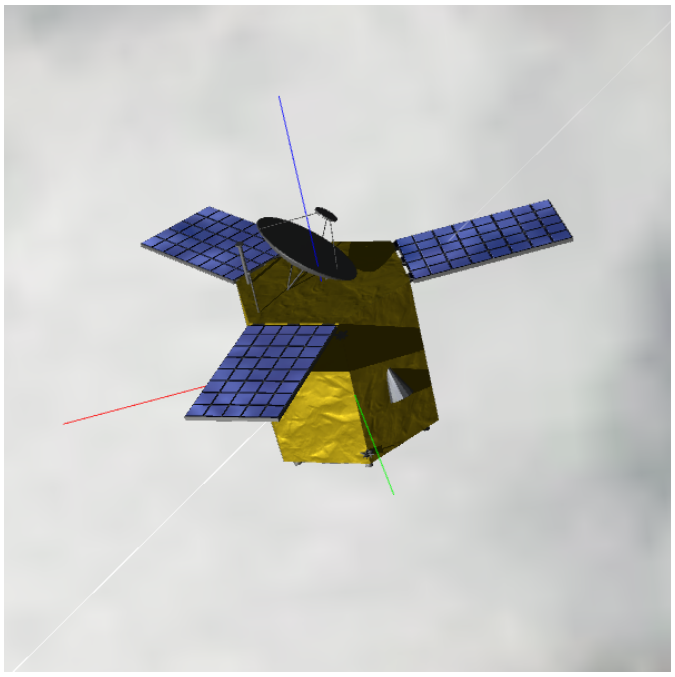
Table 3.3: The evaluation of the trained Agent Constable tested for 100 episodes. The agent reaches a consistent high return and the final attitude error $\sigma_{\mathcal{B}/\mathcal{N}\text{final}}$ lies between $-3.3°$ and $3.1°$ with 95% confidence. The attitude rate is also very stable at the end of the episode.

| Return | $\lvert\sigma_{\mathcal{B}/\mathcal{N}\text{final}}\rvert$ [°] | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | CI for $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | $\omega^i_{\mathcal{B}/\mathcal{N}\text{final}}[\frac{\text{rad}}{\text{s}}]$ |
|---|---|---|---|---|
| $3.730 \pm 0.024$ | $4.63 \pm 0.21$ | $0.6 \pm 2.6$ | $[-3.3, 3.1]$ | $(3 \pm 30) \cdot 10^{-5}$ |

The trained Agent Constable shows very good results for the constant attitude scenario. The final attitude error norm is below $5°$, which demonstrates the agents ability to orient the satellite correctly. The final attitude error for each component $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ has a maximum deviation between $[3.3, 3.1]°$ with 95% confidence. In comparison to the value itself, the final attitude rate has a large standard deviation. It's possible that the agent finds the correct position, but is still rotating slowly around one axis. The value is very small though, showing a stable and controlled satellite at the end of the simulation. To further evaluate and visualize the trained agent, the learned model is tested in only one episode in the environment. The return reached here is $R = 3.71$. In the accompanying visualization through the Vizard software the satellites behavior can be observed. The initial and final position of the spacecraft can be seen in Fig. 3.3.



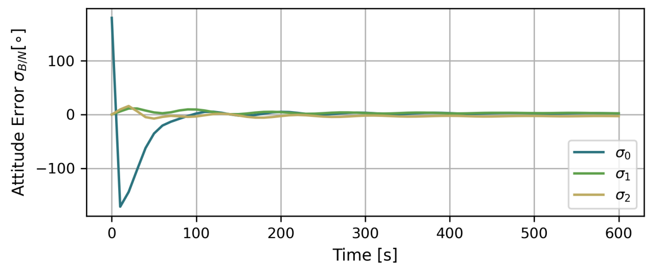
(a) Initial spacecraft attitude



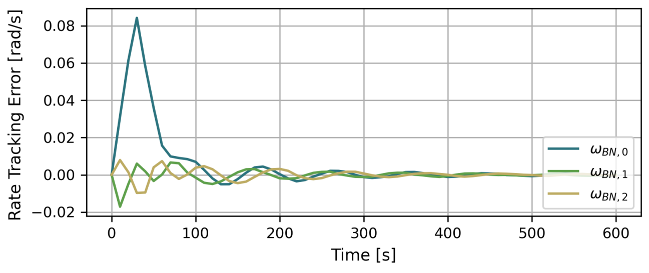
(b) Spacecraft attitude after 10 minutes

Figure 3.3: The spacecraft at the beginning of the simulation and after a 10 minute episode. The final attitude error is $\sigma_{\mathcal{B}/\mathcal{N}\text{final}} = [2.2, 2.4, -2.5]^{\circ}$

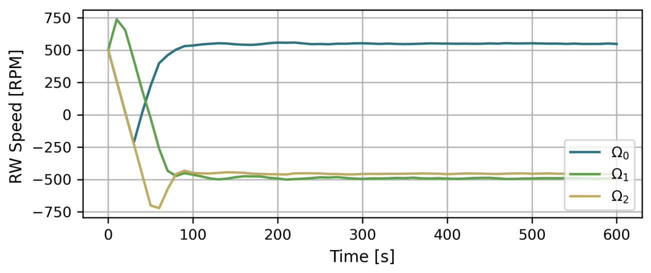
The satellite quickly turns around and stabilizes at the desired position. The attitude rate is stable. At the end the satellite is shifted almost $180^{\circ}$ from it's initial position, as can be seen in Fig. 3.3. The attitude error, the attitude rate, the RW speeds and the applied motor torque in this single 10 minute episode can be seen in Fig. 3.4.



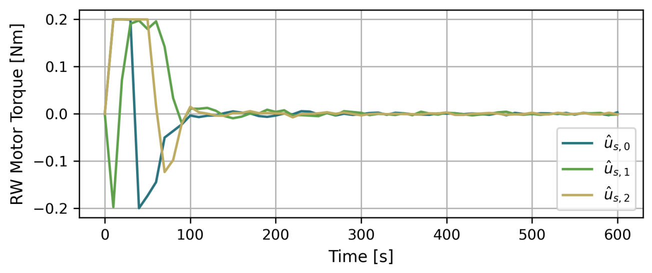
(a) Attitude error



(b) Attitude rate error



(c) RW speeds



(d) RW motor torques

Figure 3.4: The spacecraft attitude state information. The attitude error (a) describes the difference between the desired and current attitude. In the beginning the agent overshoots the correction into the other direction, but then corrects itself. After 100 $s$ the final attitude is reached. In (b) the corresponding attitude rate error can be seen. (c) shows that the reaction wheels are not saturated and their speed stays centered around the initial speed. The torques the agent applies can be seen in (d). The agent applies a large torque at the beginning, which shifts the agent almost 180° in the other direction, after which the agent slowly brings the satellite into the desired position.

Agent Constable shows good performance in this one episode. The agent overcorrects at the beginning, but is able to bring the satellite to the desired orientation after around 100 s. The agent does not saturate the reaction wheels, or cause the satellite to spin uncontrollably. The final attitude error is $\sigma_{\mathcal{B}/\mathcal{N}\text{final}} = [2.2, 2.4, -2.5]^\circ$. The final attitude error is small and shows only a slight deviation from the desired orientation in each coordinate.

This agent was only trained for one constant initial attitude. The next complication is to vary the initial attitudes and attitude rates. As a first step, Agent Constable is tested in 100 episodes again, but this time the initial attitudes and attitude rates are sampled in each axis according to $\boldsymbol{\sigma}^i_{\mathcal{B}/\mathcal{N}\text{Init}} = [0, 1], \quad \boldsymbol{\omega}^i_{\mathcal{B}/\mathcal{N}\text{max}} = 0.1\frac{\text{rad}}{\text{s}}$. The results are displayed in table 3.4.

Table 3.4: The trained agent tested for 100 episodes with the initial attitude and attitude rate sampled. The agent reaches a low return and the final attitude error $\sigma_{\mathcal{B}/\mathcal{N}\text{final}}$ has a very high error. The CI for the attitude error shows a large deviation from the desired attitude. The attitude rate is relatively stable at the end of the episode, indicating that the satellite manages to detumble, but not reach the desired attitude.

| Return | $\lvert\sigma_{\mathcal{B}/\mathcal{N}\text{final}}\rvert$ [°] | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | CI for $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | $\omega^i_{\mathcal{B}/\mathcal{N}\text{final}}[\frac{\text{rad}}{\text{s}}]$ |
|---|---|---|---|---|
| $-25.7 \pm 8.5$ | $122.8 \pm 41.9$ | $4.3 \pm 74.9$ | $[-151.7, 136.1]$ | $-0.0045 \pm 0.076$ |

As demonstrated the agent cannot handle the random initial attitudes. The agent has only learned to orient the satellite from one position. Even though the agent was able to collect a little experience of other attitudes through applying exploration techniques and trying out random actions, catapulting the satellite to unknown attitudes, this was not enough experience to be able to handle all of these different cases. The sampling of the attitude and the attitude rate has to be included in the training already, to increase the amount of experience the agent gains in training. This will be done in the next step.

## 3.4 Scenario: Vary Initial Attitudes

In this scenario different initial attitudes $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}\text{Init}}$ and attitude rates $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}\text{Init}}$ are sampled through the *initial-Conditions* module already during the training. The goal state is $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}\text{Desired}} = [0, 0, 0]$. The initial attitude and attitude rate are sampled according to $\boldsymbol{\sigma}^i_{\mathcal{B}/\mathcal{N}\text{Init}} = [0, 1], \quad \boldsymbol{\omega}^i_{\mathcal{B}/\mathcal{N}\text{max}} = 0.1\,\frac{\text{rad}}{\text{s}}$. The trained agent with random initial attitude conditions is referred to as *Agent Spinny*.

### 3.4.1 Training of Agent Spinny

Agent Spinny is trained with a total of 101805 episodes. The return during the training is depicted in Fig. 3.5. The return starts low at around -25 and then slowly rises. After the first checkpoint the return is at around -10, but dropping shortly due to the overfitting to the existing limited data. After that initial drop the return is approaching 0, but still shows a large drop after episode 40000. This large drop indicates that the reaction wheels were saturated. Several more, smaller drops occur during the rest of training, but the drops get smaller, until near the end the return is in a positive region.

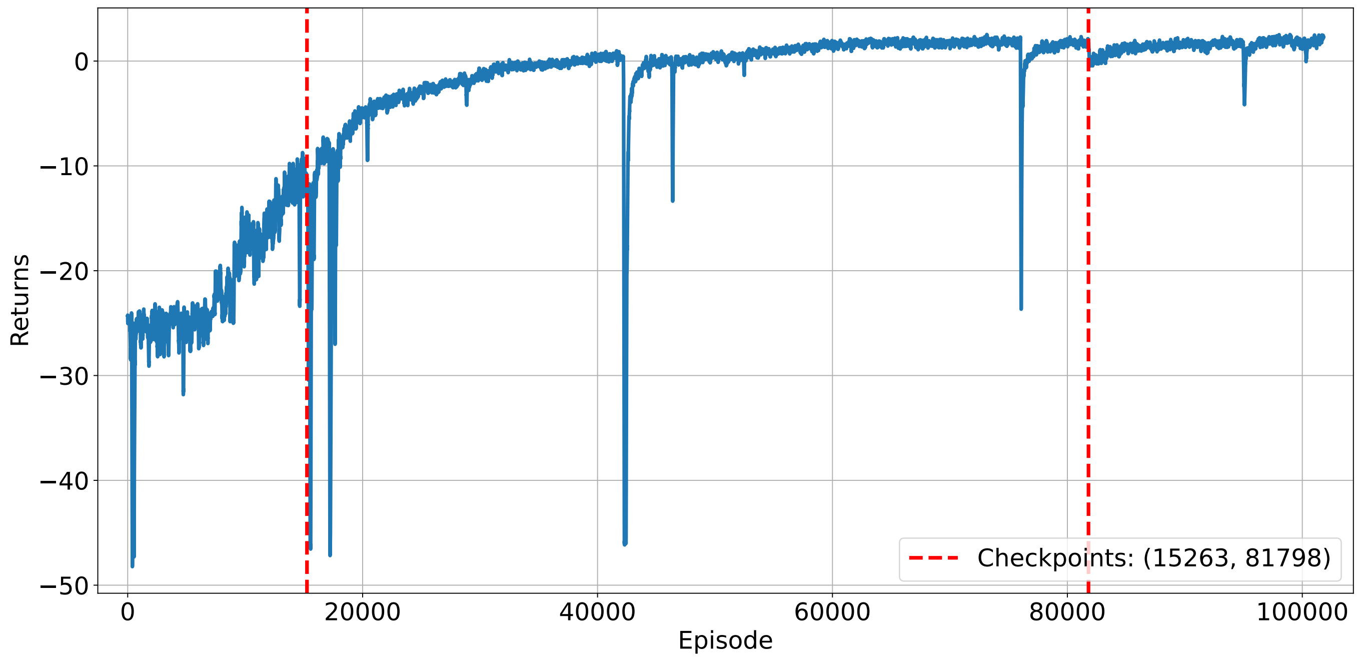
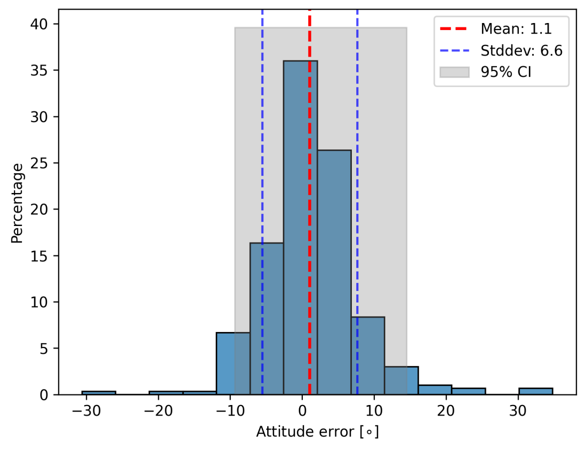


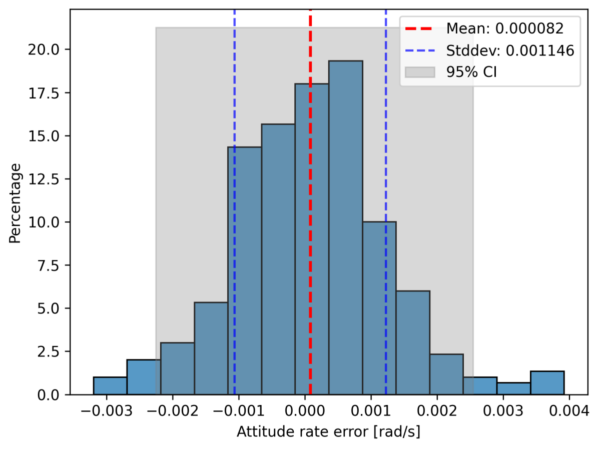
Figure 3.5: The return over the course of the 101805 training episodes. Due to the memory leak the training was conducted in 3 sets: The first checkpoint is after 15263 episodes of training, the second checkpoint after 81798 episodes. The total training episodes are 101805.

### 3.4.2 Evaluation of Trained Agent Spinny

As before, the trained agent is tested in the environment. To test the robustness of the agent the initial conditions are Monte Carlo sampled for 100 samples and the results are evaluated. As a first measure the attitude error and attitude rate error for all components $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$, $\omega^i_{\mathcal{B}/\mathcal{N}\text{final}}$ at the end of the episode is investigated for the 100 samples. In the ideal case the attitude error would be zero, when the episode is finished, since the desired position is reached.



(a) Attitude error



(b) Attitude rate error

Figure 3.6: The distribution of the final attitude error (a) and the final attitude rate (b) over 100 samples for the vary init atts scenario. The mean and standard deviation $\sigma^i_{\mathcal{B}/\mathcal{N}} = (1.1 \pm 6.6)°$, $\omega_{\mathcal{B}/\mathcal{N}} = (8.17 \pm 110) \cdot 10^{-5} \frac{\text{rad}}{\text{s}}$ are sketched in red and blue, respectively. The confidence interval is sketched in grey. With 95% confidence the final attitude error $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ lies between $[-9.4, 14.5]°$.

This distribution shows that the attitude error is centered around zero, which is the desired behavior. The CI still shows a relatively large deviation, lying between $[-9.4, 14.5]°$. The final attitude rate shows a very stable behavior. The satellite manages to detumble itself. As with Agent Constable, the standard deviation is very large compared to the value itself. This indicates that the satellite still has a spin at the end of the episode.

After evaluating these environment metrics, the RL metric of the return is evaluated. As discussed before, the maximum return reachable is $R_{\text{max}} = r_{\text{max}} \cdot \#\text{steps} = 0.1 \cdot 60 = 6$. The maximum return reached in the 100 samples is $R = 5.5$. The minimum return reached is $R = -4$. The mean return including the standard deviation is $R = 2.5 \pm 1.9$.

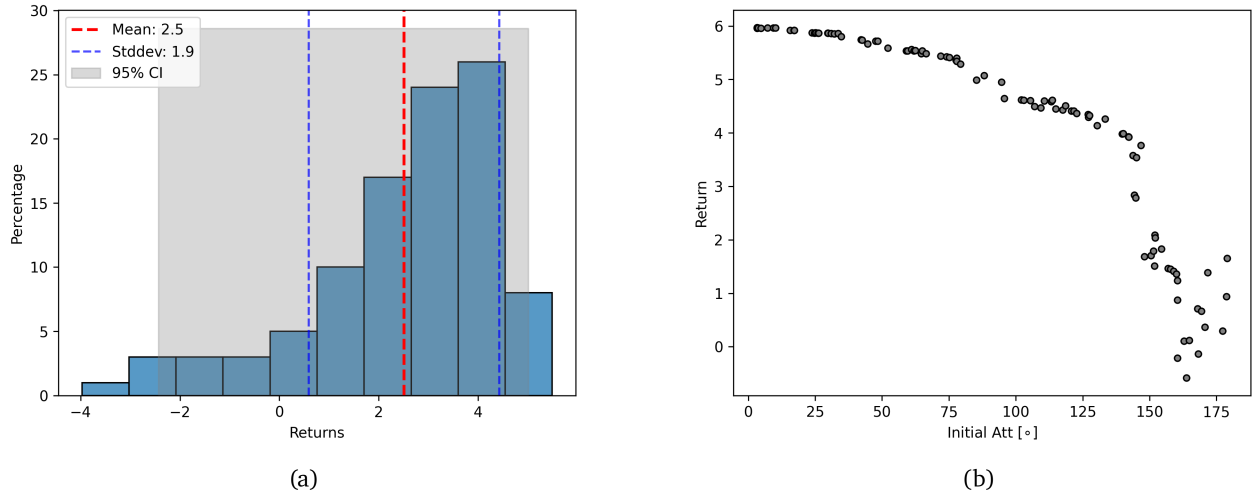


Figure 3.7: In (a) the distribution of the return over 100 samples for the vary init atts scenario is shown. The mean and standard deviation $R = 2.5 \pm 1.9$ are sketched in red and blue, respectively. The confidence interval is sketched in grey. With 95% confidence the return lies between $[-2.4, 5.0]$. In (b) the return over the initial attitude is displayed. The rotation is constrained to the x-axis. It shows the relationship between the initial attitude and the return. The further away the satellite is from the desired position at the beginning of the simulation, the lower the return.

In this scenario it is to be expected that the maximum return of $R = 6$ is not reached every time and that the return varies depending on the starting position. The maximum return could be reached, if the start position would be equal to the desired end position. It is expected that the further away from the desired position the satellite is at the beginning, the lower the return, since time is needed for the control maneuver. This phenomenon can be demonstrated by investigating the one-dimensional case. Instead of sampling the attitude and attitude rate between any values, the rotation is constrained to the x-axis for simplicity. The desired attitude is still $\sigma_{\mathcal{B}/\mathcal{N}\text{Desired}} = [0, 0, 0]$. Considering this simpler case will help to gain a better understanding of the influence of the initial attitude on the return. In Fig. 3.7 (b) it is clearly visible, that initial conditions that are close to the desired state yield the highest return. The further away the initial state is from the desired, the lower the return. This is due to the fact that the return is a cumulative property of all rewards. If the agent is in an unoptimal position in the beginning, the reward will be lower until the agent gets close to the desired position. This is to be expected.

## 3.5 Comparison: Agent Spinny vs. Agent Constable

In this section, the Agents Spinny and Constable (section 3.3) are compared for two different scenarios: In the first scenario the initial conditions are set to $\sigma_{\mathcal{B}/\mathcal{N}\text{Init}} = [1, 0, 0]$ and the satellite has no initial attitude rate. In the second scenario the attitude is sampled the same way as in section 3.4.

Table 3.5: Comparison of Agent Spinny and Agent Constable in 100 samples in the environment for each scenario.

| Scenario | Agent | Return | $\lvert\sigma_{\mathcal{B}/\mathcal{N}\text{final}}\rvert$ [°] | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | CI for $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | $\omega^i_{\mathcal{B}/\mathcal{N}\text{final}}[\frac{\text{rad}}{\text{s}}]$ |
|---|---|---|---|---|---|---|
| Static Att | Constable | $3.730 \pm 0.024$ | $4.63 \pm 0.21$ | $0.6 \pm 2.6$ | $[-3.3, 3.1]$ | $(3 \pm 30) \cdot 10^{-5}$ |
| | Spinny | $0.03 \pm 0.20$ | $17.4 \pm 0.9$ | $-2.9 \pm 9.7$ | $[-11.1, 11.8]$ | $(5.12 \pm 110) \cdot 10^{-5}$ |
| Vary Att | Constable | $-25.7 \pm 8.5$ | $122.8 \pm 41.9$ | $4.3 \pm 74.9$ | $[-151.7, 136.1]$ | $(-4.5 \pm 76) \cdot 10^{-3}$ |
| | Spinny | $2.5 \pm 1.9$ | $9.3 \pm 6.9$ | $1.1 \pm 6.6$ | $[-9.4, 14.5]$ | $(8.17 \pm 110) \cdot 10^{-5}$ |

Agent Spinny shows little spread among the return in both scenarios, while Agent Constable demonstrates a large spread of the return in the "Vary Att" scenario, while being very consistent in the "Static Att" scenario. The return distribution of the Agents performing in this scenario is plotted in Fig. 3.8. Agent Constable shows the best CI overall, in the scenario with a static initial attitude. The CI for Agent Spinny is bigger, but still the final attitude deviation is still in a tolerable area. The strength of Agent Spinny is in generalizing and being able to handle a wide array of initial conditions.

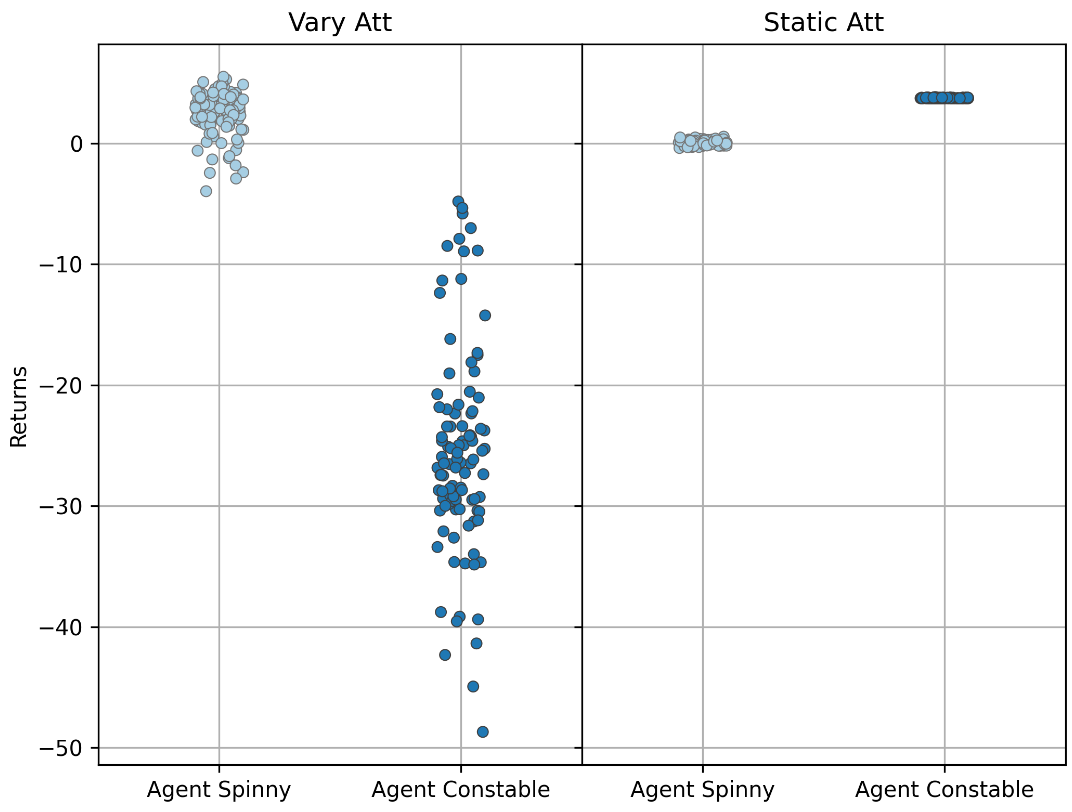


Figure 3.8: The spread of the return (shown on the y-axis) for the "Vary Att" and "Static Att" scenarios. Both scenarios are tested with Agent Spinny and Agent Constable. Agent Spinny performs better overall, but for the specific initial attitude of $\sigma_{\mathcal{B}/\mathcal{N}\text{Init}} = [1, 0, 0]$, Agent Constable demonstrates a slightly better performance.

For the specific initial attitude of $\sigma_{\mathcal{B}/\mathcal{N}\text{Init}} = [1, 0, 0]$ Agent Constable is able to perform better than Agent Spinny. This can be explained by several factors: The specific attitude of $\sigma_{\mathcal{B}/\mathcal{N}\text{Init}} = [1, 0, 0]$ is very unlikely to be picked if every coordinate is sampled between $0$ and $1$. This is also demonstrated on the left hand side of the plot: The return for Agent Constable starts very low, indicating that in the 100 samples used for evaluation the state of $\sigma_{\mathcal{B}/\mathcal{N}\text{Init}} = [1, 0, 0]$ and $\omega_{\mathcal{B}/\mathcal{N}\text{Init}} = [0, 0, 0]$ was not present. The same thing likely occured in the training of Agent Spinny, Agent Spinny never saw the specific state that Agent Constable was trained with. With increased training time, Agent Spinny might achieve results close to Agent Constable. Still, Agent Spinny achieves a good return and a CI of [-9.4,14.5]°. To improve this performance it might also help to increase the network capacity. This can be achieved by increasing the size of neurons in the hidden layers. When varying both the attitude and the attitude rate, the space state is very large, since there are many different combinations. Increasing the network capacity can help to better approximate the complicated functions. If the network capacity is too low, overgeneralization can happen, which decreases performance. Overall, Agent Spinny is able to handle this specific attitude, but performs slightly worse than Agent Constable. Agent Spinny's ability to deal with changing attitude conditions is also demonstrated in the next scenario that will be discussed.

## 3.6 Scenario: Location Pointing with Two Spacecrafts

In this scenario there are two spacecrafts, where one spacecraft simulates the servicer and the other spacecraft simulates a non-functioning debris object. The servicer is in orbit around the spacecraft, with the goal to get more information about the debris object before attempting the capture. For this purpose there is a camera attached to the servicer, which should be pointed at the debris objects during the orbit.

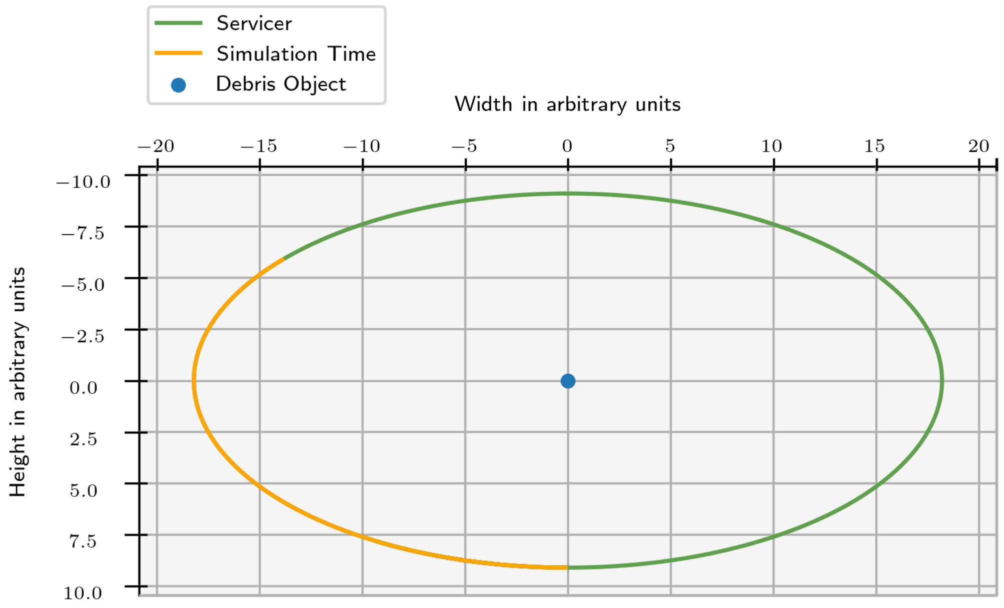


Figure 3.9: The location pointing scenario involving two spacecrafts. The debris object is being orbited by a servicer. The simulation time is marked in orange. During the entire simulation time the servicer should keep the debris object in its field of view.

### 3.6.1 Simulation Architecture

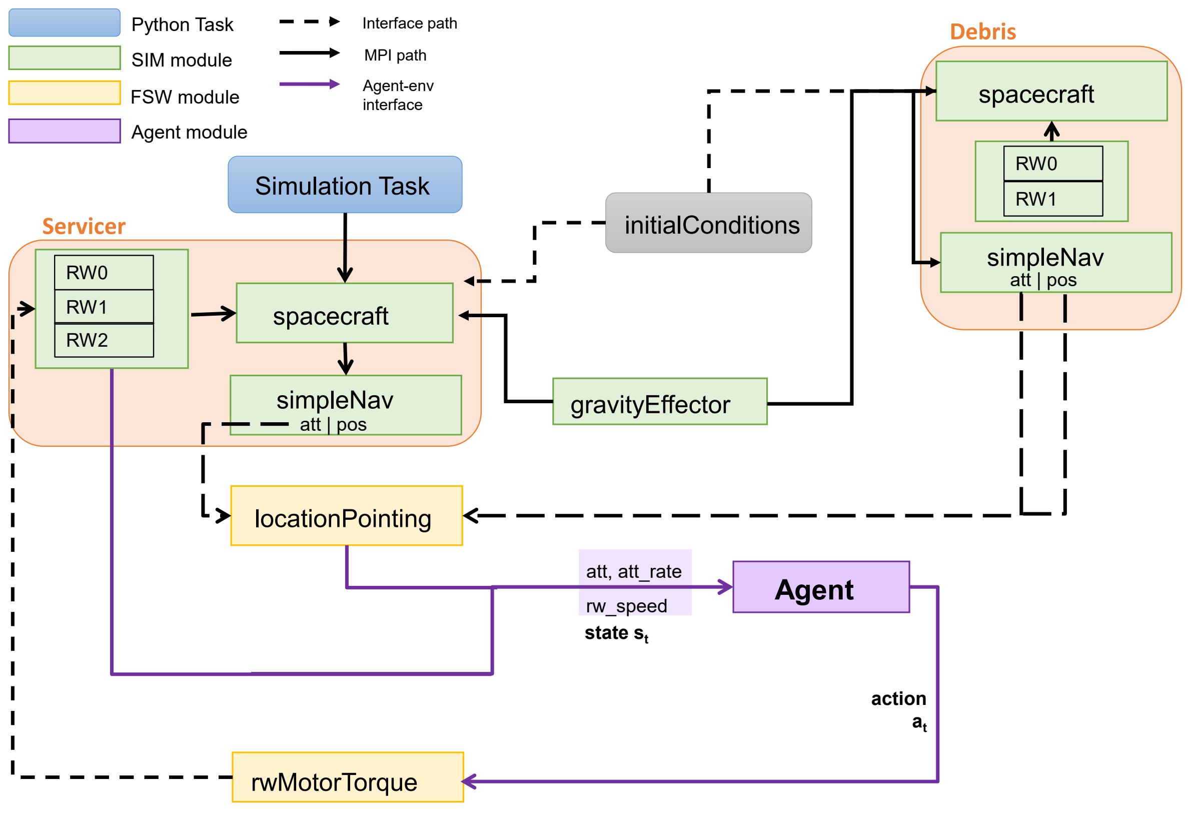


Figure 3.10: The simulation architecture and agent interface. The simulation (SIM) and flight software (FSW) modules communicate through the message passing interface (MPI). The agent receives the state $s_t$ from the environment. The state contains the attitude error $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}}$, the attitude rate $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$ and the wheel speed $\boldsymbol{\Omega}$. The attitude error is calculated by the locationPointing module, which takes the attitude from the servicer and the debris object into account, to calculate the difference. The agent then gives an action $a_t$ in the form of a torque vector into the environment. The communication between agent and environment is based on an OpenAI Gym interface.

In the simulation the theoretical scenario is realized through adding a second debris spacecraft with spinning reaction wheels, causing the spacecraft to spin freely, and a navigation module (*simpleNav*). The *attTrackingError* module and the *inertial3D* module from the previous simulation (see Fig. 3.1) is replaced with a *locationPointing* module. The *locationPointing* module takes both the navigational data from the servicer and the debris spacecraft as input and translates the difference directly into an attitude error, which is fed to the agent in the state. The debris spacecrafts location in the inertial frame is given by $r_{L/N}$. The vector $r_{L/S}$ pointing from the satellite location $r_{S/N}$ to the debris object is then

$$r_{L/S} = r_{L/N} - r_{S/N}. \tag{3.9}$$

Let $\hat{r}_{L/S}$ be the normalized heading vector to this location. $\hat{p}$ denotes a body-fixed vector, which is to be pointed towards the debris spacecraft. Thus, this module performs a 2-degree of freedom attitude guidance and control solution. The eigen-axis $\hat{e}$ to rotate $\hat{p}$ towards $\hat{r}_{L/S}$ is then given by

$$\hat{e} = \frac{\hat{p} \times \hat{r}_{L/S}}{|\hat{p} \cdot \hat{r}_{L/S}|}. \tag{3.10}$$

The attitude tracking error is then given by

$$\sigma_{B/N} = -\tan(\phi/4)\hat{e}, \text{ with} \tag{3.11}$$

$$\phi = \arccos(\hat{p} \cdot \hat{r}_{L/S}). \tag{3.12}$$

In the previous scenarios, the desired attitude was always set to $\sigma_{\mathcal{B}/\mathcal{N}\text{Desired}} = [0, 0, 0]$. In this scenario the desired attitude depends on the location of the debris spacecraft.

Table 3.6: The "Location Pointing" scenario in comparison to the "Vary Init Pos" scenario. Instead of a varying initial attitude, the "Location Pointing" scenario has a time dependent desired attitude.

| Scenario | Agent | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{Init}}$ | $\sigma_{\mathcal{B}/\mathcal{N}\text{Desired}}$ | $\omega^i_{\mathcal{B}/\mathcal{N}\text{Init}}[\frac{\text{rad}}{\text{s}}]$ | Mass[kg] |
|---|---|---|---|---|---|
| Vary Init Pos | Spinny | $0 - 1$ | $[0, 0, 0]$ | $0 - 0.1$ | 330 |
| Location Pointing | Spinny | $[1, 0, 0]$ | Time Dependent | $[0, 0, 0]$ | 330 |

The *locationPointing* module is further set up in such a way, that the side, where the camera is attached to, should point at the debris object the entire time. For this purpose the body fixed vector $\hat{p}$ that is to be aimed at the location is set to the vector that the camera is attached to. The agent then calculates an action in the form of a three dimensional motor torque [24].

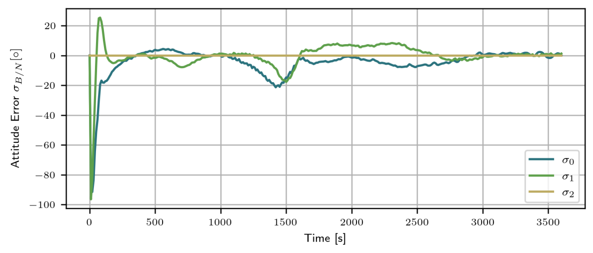
### 3.6.2 Evaluation of Agent Spinny

It is tested if Agent Spinny can handle this new scenario without any further training. On the one hand, this problem is more complicated than the scenario in section 3.4, since the attitude error is now time dependent. On the other hand, the problem simplifies to a two dimensional case. Instead of having to reach the desired attitude at the end of the simulation, the agent has to keep adjusting the satellite to have the debris object in view at all times. Furthermore, the change in attitude is much more gradual, therefore the simulation time is increased to $3600\,s = 1$ hour. This impacts the minimum and maximum return, which now lies between $-360 \leq R \leq 36$. Agent Spinny is evaluated for this problem with 100 samples. The results are listed in table 3.7.

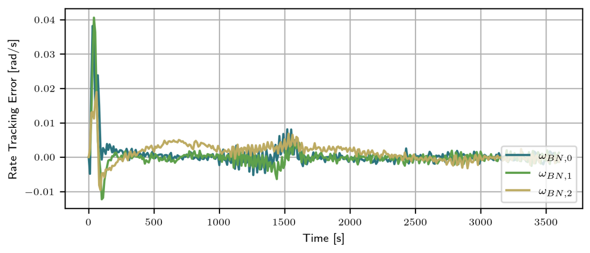
Table 3.7: Evaluation of Agent Spinny in the scenario with two spacecrafts.

| Return | $\lvert\sigma_{\mathcal{B}/\mathcal{N}\text{final}}\rvert$ [°] | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | CI for $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | $\omega^i_{\mathcal{B}/\mathcal{N}\text{final}}[\frac{\text{rad}}{\text{s}}]$ |
|---|---|---|---|---|
| $31.02 \pm 0.11$ | $1.3 \pm 0.6$ | $0.3 \pm 0.7$ | $[-1, 1.9]$ | $(-1.89 \pm 1.51) \cdot 10^{-3}$ |

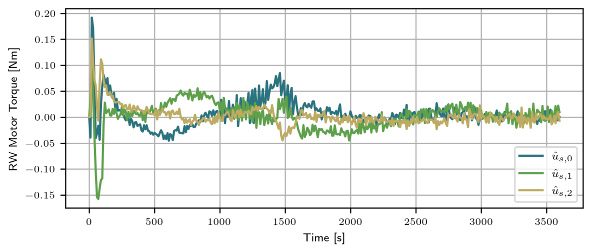
Agent Spinny performs very well in the scenario with two spacecrafts. The CI for the final attitude error is even smaller than in the previous scenarios. The final attitude error also shows a small value with a small standard deviation. The attitude rate is higher than in previous scenarios, but the satellite also needs to be constantly adjusted, since the target changes. For this reason it is vital to not only look at the final attitude error, but to also investigate how the agent performs during one episode. For this purpose the Vizard software is used. One episode is visualized and the results are depicted in Fig. 3.12. Additionally, the state information is evaluated in Fig. 3.11.



(a) Attitude error



(b) Attitude rate error



(c) RW motor torques

Figure 3.11: The attitude error (a), attitude rate error (b) and RW motor torques (c). Since the problem simplifies to two dimensions, $\sigma_2$ is zero throughout the entire episode. The other two coordinates are in a range between -20° and 20°. Even though the problem does not require to control the third axis of the satellite, in (b) it can be seen that the attitude rate error is around zero and the satellite is very stable. The RW motor torques are shown in (c). The agent continuously applies small torques to follow the debris object.

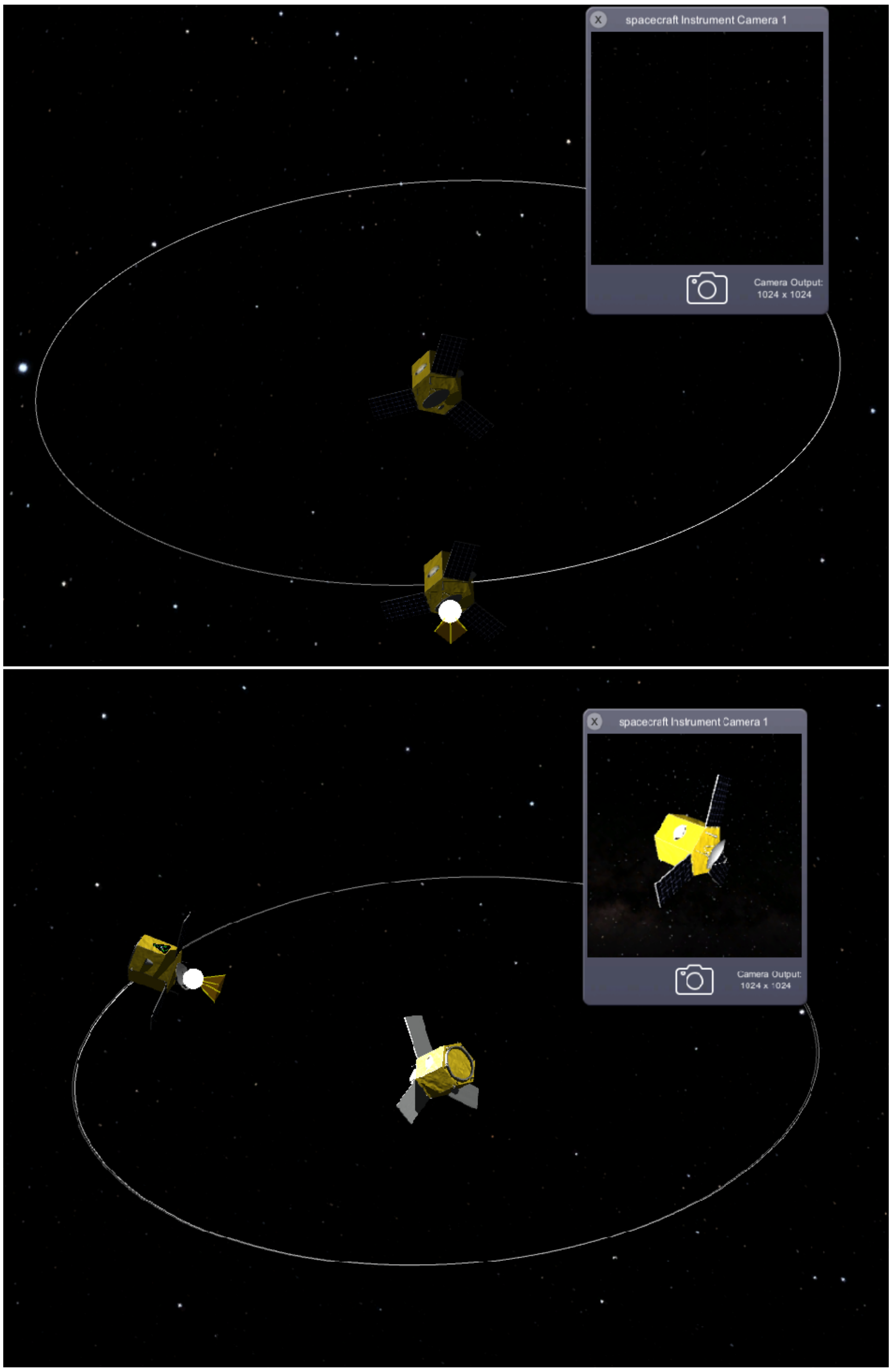

spacecraft Instrument Camera 1
Camera Output:
1024 x 1024
spacecraft Instrument Camera 1
Camera Output:
1024 x 1024

Figure 3.12: The visualization of the beginning of the episode is shown in the top panel. The debris object is in the middle, being orbited by the servicer. The camera is marked with a white dot and a yellow cone around it, demonstrating the field of view. In the "spacecraft Instrument Camera1" panel the view of the camera is visible. In the second panel the end of the episode is shown. In the beginning of the episode the camera is pointed away from the debris object, at the end of the episode the camera is pointed toward the debris object.

At the beginning of the episode the camera is pointed away from the debris object towards empty space. At the end of the episode the camera is pointed at the debris object and the debris object is clearly visible in the camera panel. In conclusion, Agent Spinny is able to handle the time-dependent attitude error without any further training. This demonstrates its capability to generalize and handle new unknown scenarios.

## 3.7 Scenario: Vary Initial Mass

In this scenario a fixed initial attitude $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}\text{Init}} = [1, 0, 0]$ (rotation by 180°) and attitude rate $\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}\text{Init}} = [0, 0, 0]\,\frac{\text{rad}}{\text{s}}$ are set. The goal state is $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{N}\text{Desired}} = [0, 0, 0]$. For future references the agent in this scenario will be referred to as *Agent Massive*. The mass is sampled uniformly between 10 kg and 1000 kg.

### 3.7.1 Training of Agent Massive

The agent is trained for 17593 episodes in total, in two sets. The return is depicted in Fig. 3.13. Before the first checkpoint the return stays at around -20, even showing several dips. After the second checkpoint, the return rises, only showing one dip. After episode 10000 the return converges. The agent has learned how to maximize the return.

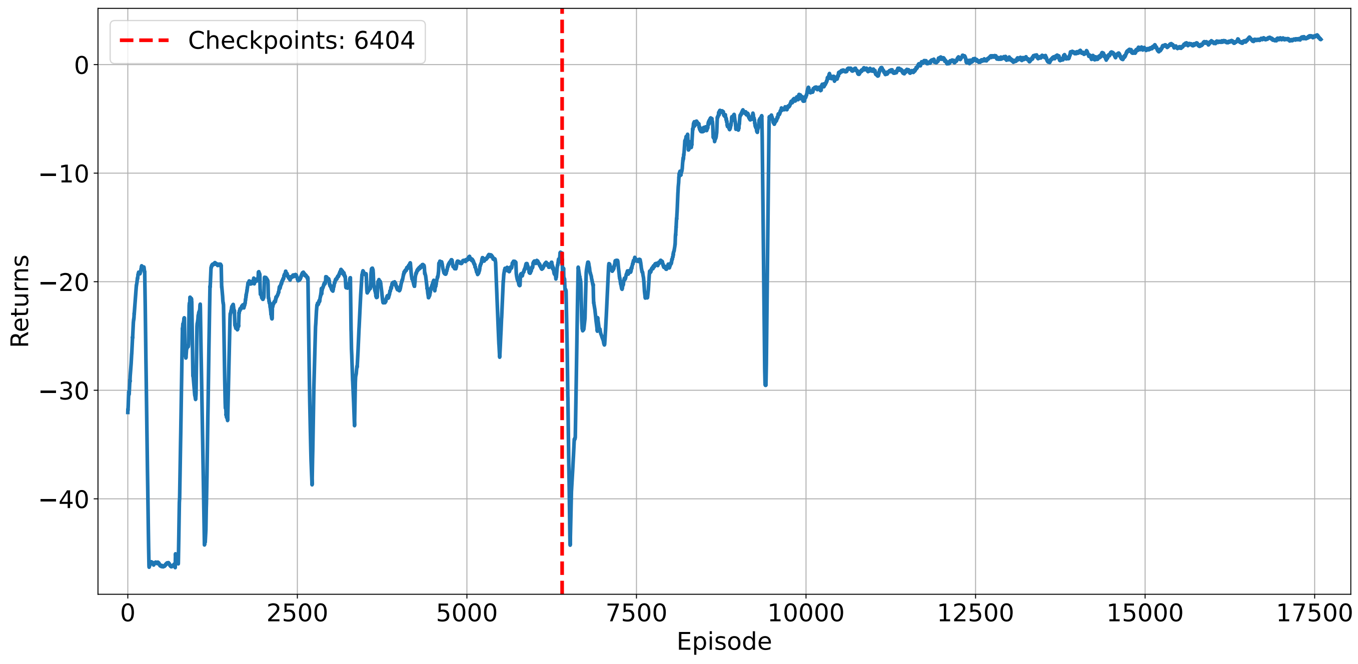


Figure 3.13: The return over the course of the 17593 training episodes. The training is conducted in two sets: The checkpoint is after 6404 training episodes. After this checkpoint the agent is able to maximize the return.

### 3.7.2 Evaluation of Agent Massive

The trained Agent is evaluated in 100 episodes. In these 100 episodes the mass is uniformly sampled in the same mass range, the agent was trained with, i.e. 10-1000 kg.

The results of this evaluation is listed in table 3.8.

Table 3.8: The evaluation metrics for Agent Massive. The return shows a small standard deviation, the attitude error for each component is small. As in the scenarios before the attitude rate shows a large uncertainty.

| Mass [kg] | Return | $\lvert\sigma_{\mathcal{B}/\mathcal{N}\text{final}}\rvert$ [°] | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | CI for $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | $\omega^i_{\mathcal{B}/\mathcal{N}\text{final}}[\frac{\text{rad}}{\text{s}}]$ |
|---|---|---|---|---|---|
| 10-1000 | $1.0 \pm 0.7$ | $4.9 \pm 1.1$ | $-2.0 \pm 2.0$ | $[-5.3, 1.2]$ | $(9.83 \pm 740) \cdot 10^{-5}$ |

The agent achieves good results, but a slight decrease in the return can be seen for higher masses, as can be seen in Fig. 3.14. The confidence interval of the final attitude error is small, lying between [-5.40, 1.45]°.

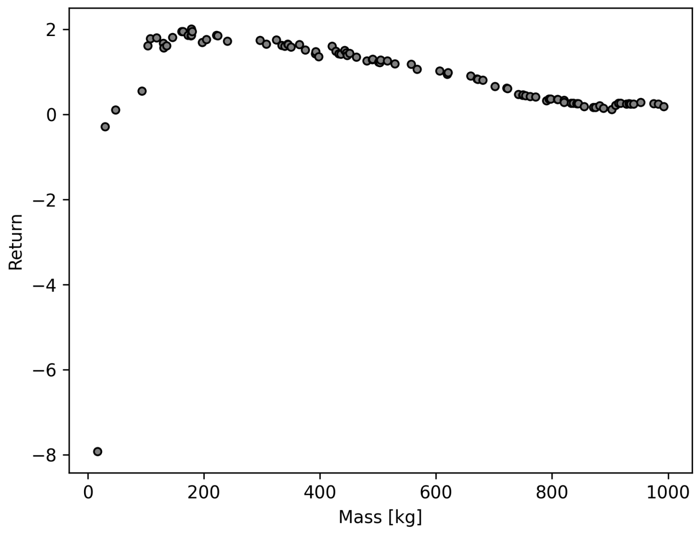


Figure 3.14: The return over the initial mass. For very low masses the agent shows a negative return, possibly the agent is applying a force too strong for the small mass. The return is consistent for masses between 150-500 kg. After that the return decreases slightly.

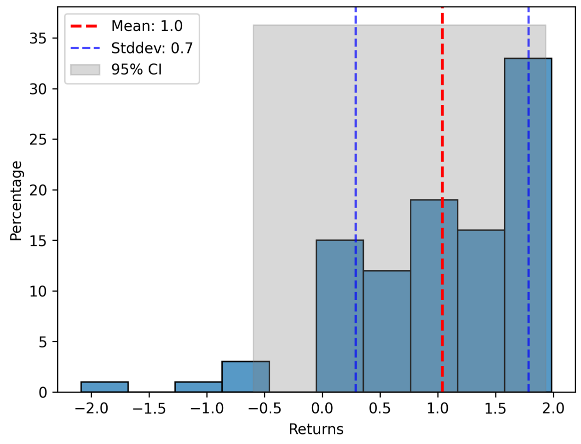
Agent Massive achieves consistent results for a mass range between 150-500 kg. Noticeably, the agent is not able to handle masses between 10-100 kg. Since the agent has no way to extract the mass from the observations, the agent tries to generalize and to optimize the actions for all masses. By doing this, the agent has to find a balance between applying a force that is too small to move higher masses and applying a force too strong for the low masses. In the end, the actions are slightly more optimized towards lower masses, since the negative effect of wrong actions is stronger here. That is why in Fig. 3.14 the return drops slightly for higher masses. For masses below 150 kg the force is still too strong, causing the satellite to oscillate around the desired orientation. The effect on the higher masses will be discussed in more detail in the next section.

### 3.7.3 Testing Agent Massive with Masses > 1000 kg

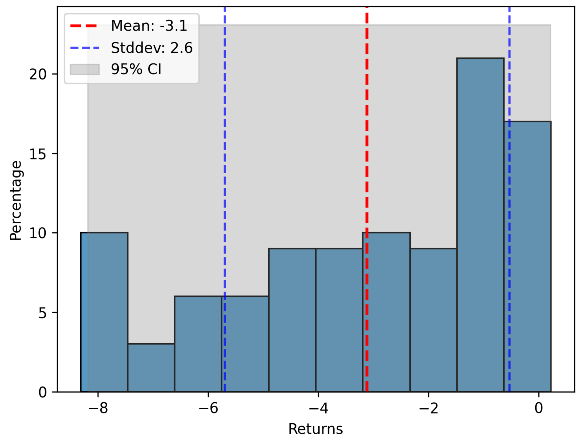
To investigate the phenomenon of the decreasing return with higher masses and the agents ability to handle unknown masses, Agent Massive is also tested with masses higher than those the agent was trained with. Now, the initial mass is varied between 1000-5000 kg. The results are displayed in table 3.9. The distributions are also plotted in Fig. 3.15.

Table 3.9: The results of the evaluation of Agent Massive for two mass ranges. The return is significantly lower for the mass range between 1000-5000 kg. The CI is also larger.

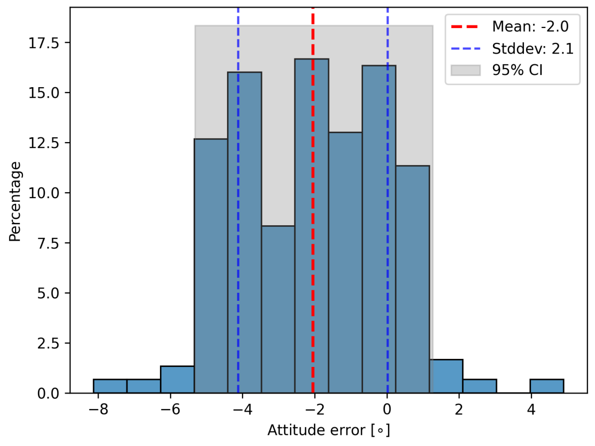
| Mass [kg] | Return | $\lvert\sigma_{\mathcal{B}/\mathcal{N}\text{final}}\rvert$ [°] | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | CI for $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | $\omega^i_{\mathcal{B}/\mathcal{N}\text{final}}[\frac{\text{rad}}{\text{s}}]$ |
|---|---|---|---|---|---|
| 10-1000 | $1.0 \pm 0.7$ | $4.9 \pm 1.1$ | $-2.0 \pm 2.0$ | $[-5.3, 1.2]$ | $(9.83 \pm 740) \cdot 10^{-5}$ |
| 1000-5000 | $-3.1 \pm 2.6$ | $5.4 \pm 3.5$ | $-1.0 \pm 3.6$ | $[-9.6, 7.3]$ | $(5.26 \pm 66) \cdot 10^{-5}$ |



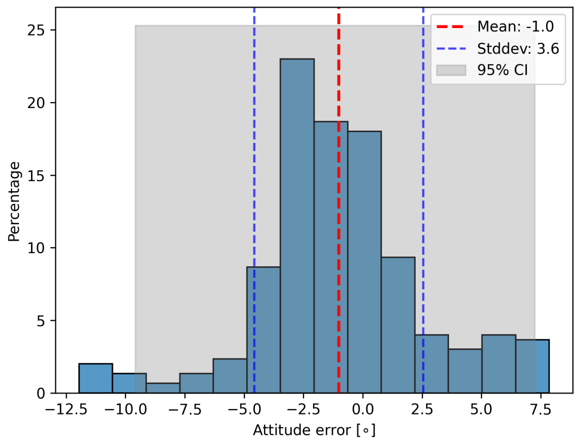
(a) Return distribution 10-1000 kg



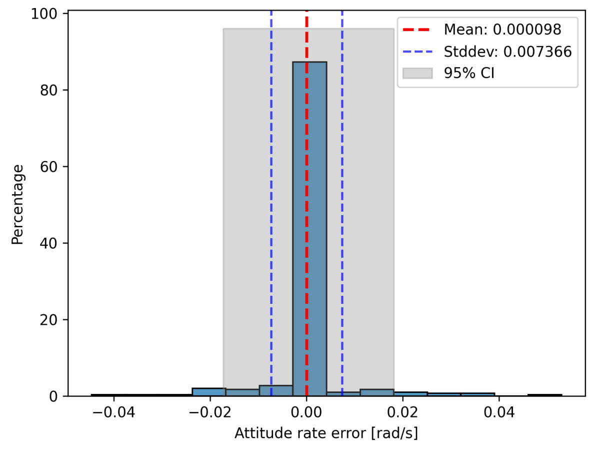
(b) Return distribution 1000-5000 kg



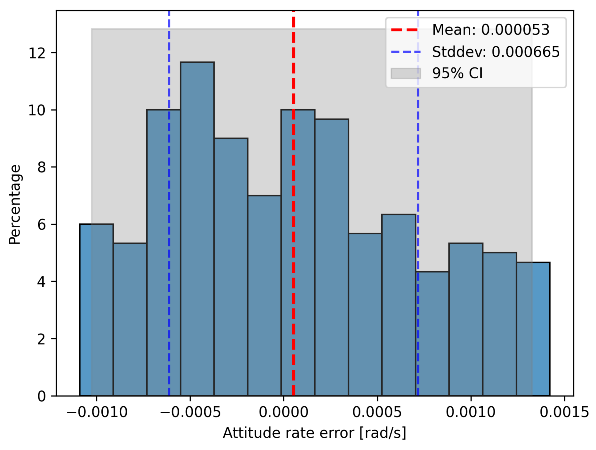
(c) Attitude error distribution 10-1000 kg



(d) Attitude error distribution 1000-5000 kg



(e) Attitude rate error distribution 10-1000 kg



(f) Attitude rate error distribution 1000-5000 kg

Figure 3.15: The distribution of the evaluation metrics for 10-1000 kg and 1000-5000 kg mass ranges. In both cases Agent Massive is tested for 100 samples in the environment.

On the left side of Fig. 3.15, the return, attitude error and attitude rate error distributions are plotted for the mass range between 10-1000 kg. On the right side the same metrics are plotted for a mass range between 1000-5000 kg. At a first glance, the performance on the right hand side seems to be much worse than on the left hand side. The return distribution shows a much larger spread and a lower average in general. But when looking at the attitude error distributions, these are of the same magnitude with the 1000-5000 kg attitude error having a slightly larger CI. The mean attitude error rate is even smaller for the larger masses, though also showing a larger spread. Considering this from a logical standpoint, a higher mass is equal to a higher moment of inertia. The higher moment of inertia translates into a slower acceleration of the object. This explains both the lower attitude rate and the lower return, since the agent needs more time to move the satellite to the correct orientation. This demonstrates the importance of not only looking at the return to evaluate an agent. While the return is much lower for the higher masses, the attitude error is in similar order of magnitude, therefore achieving equally good results, even for higher masses. It is also important to note that the episode length of 10 minutes might not be enough to steer the heavy object to the required attitude. To test this theory the agent is evaluated again, but this time with an episode length of 20 minutes. Therefore, the maximum and minimum return reachable also increases by a factor two to $R_{\text{max}} = 12$ and $R_{\text{min}} = -120$. The results of this evaluation are listed in table 3.10.

Table 3.10: The results of Agent Massive for different mass ranges and episode lengths. Agent Massive is able to achieve a similar final attitude error for the higher mass range, when the episode length is increased.

| Episode | Mass [kg] | Return | $\lvert\sigma_{\mathcal{B}/\mathcal{N}\text{final}}\rvert$ [°] | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | CI for $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | $\omega^i_{\mathcal{B}/\mathcal{N}\text{final}}[\frac{\text{rad}}{\text{s}}]$ |
|---|---|---|---|---|---|---|
| 10 min | 10-1000 | $1.0 \pm 0.7$ | $4.9 \pm 1.1$ | $-2.0 \pm 2.0$ | $[-5.3, 1.2]$ | $(9.83 \pm 740) \cdot 10^{-5}$ |
| 10 min | 1000-5000 | $-3.1 \pm 2.6$ | $5.4 \pm 3.5$ | $-1.0 \pm 3.6$ | $[-9.6, 7.3]$ | $(5.26 \pm 66) \cdot 10^{-5}$ |
| 20 min | 1000-5000 | $2.5 \pm 2.6$ | $5.0 \pm 1.5$ | $-2.0 \pm 2.3$ | $[-6.3, 2.3]$ | $(4.07 \pm 30) \cdot 10^{-5}$ |

When increasing the length of the episode, the final attitude error is almost exactly the same for both mass ranges. The CI is also in the the same range.

This confirms the hypothesis that when looking at higher masses, the agent needs more time to move the satellite due to the high moment of inertia. In conclusion, Agent Massive is able to handle higher masses that the agent was not originally trained with. This demonstrates the agents ability to generalize. For low masses the agent does not achieve good results, since it applies forces that are too strong for the low mass. Currently, the agent has no way to extract information about the mass from the state. To be able to handle lower masses stacked observations could be implemented. Stacked observations describes the process of concatenating several states into one observations. The purpose of this stack is to help the agent understand the temporal dynamics of the environment. It provides the agent with a way to capture motion and can help in getting a sense of the mass for the agent.

## 3.8 Comparison Agent Massive vs. Agent Constable

In this section Agent Massive is compared to Agent Constable (section 3.3). The agents are compared for two scenarios: The first scenario employs a constant mass of 330 kg, while in the second scenario the mass is varied between 10-1000 kg. In both scenarios the initial attitude is set to $\sigma_{\mathcal{B}/\mathcal{N}\text{Init}} = [1, 0, 0]$ and the satellite has no initial attitude rate. The results of the comparison are displayed in table 3.11 and in Fig. 3.16

Table 3.11: Comparison of Agent Massive and Agent Constable in 100 samples in the environment for each scenario. In both cases the initial attitude is static and the agent is supposed to turn the satellite by 180°.

| Scenario | Agent | Return | $\lvert\sigma_{\mathcal{B}/\mathcal{N}\text{final}}\rvert$ [°] | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | CI for $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ | $\omega^i_{\mathcal{B}/\mathcal{N}\text{final}}[\frac{\text{rad}}{\text{s}}]$ |
|---|---|---|---|---|---|---|
| Constant Mass | Constable | $3.730 \pm 0.024$ | $4.63 \pm 0.21$ | $0.6 \pm 2.6$ | $[-3.3, 3.1]^\circ$ | $(3 \pm 30) \cdot 10^{-5}$ |
| | Massive | $1.62 \pm 0.04$ | $5.3 \pm 0.7$ | $-2.2 \pm 2.2$ | $[4.2, 6.7]^\circ$ | $(3.77 \pm 0.46) \cdot 10^{-3}$ |
| Vary Mass | Constable | $-1.5 \pm 4.9$ | $10.7 \pm 23.3$ | $0.4 \pm 14.8$ | $[-16, 13]^\circ$ | $(-9.36 \pm 389) \cdot 10^{-4}$ |
| | Massive | $1.0 \pm 0.7$ | $4.9 \pm 1.1$ | $-2.0 \pm 2.0$ | $[-5.3, 1.2]^\circ$ | $(9.83 \pm 740) \cdot 10^{-5}$ |

In the first scenario "Constant Mass", Agent Constable achieves better results, which is evident in the higher return and the smaller CI for the final attitude error. Agent Massive achieves similar results for both scenarios. In comparison, Agent Constable performs significantly worse in the "Vary Mass" scenario. Here, Agent Constable shows a large CI of $[-16, 13]^\circ$ and a negative return of $R = -1.5 \pm 4.9$.

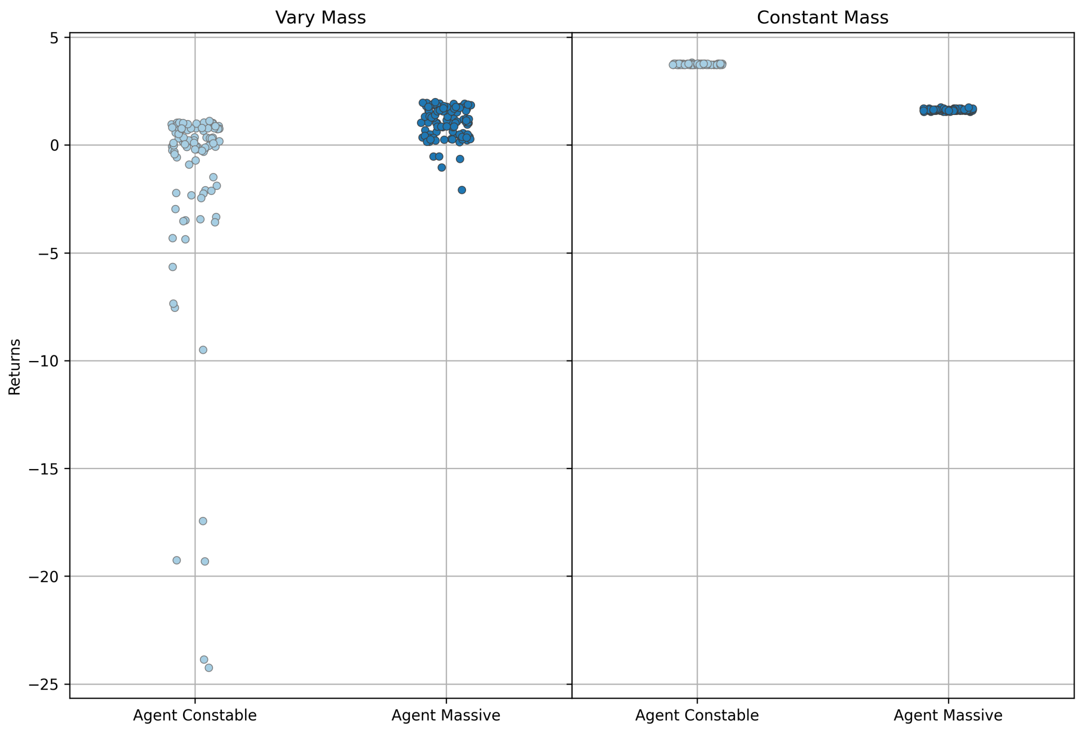


Figure 3.16: Agent Constable vs. Agent Massive in two scenarios: In the "Vary Mass" scenario the mass is varied between 10-1000 kg. In the "Constant Mass" scenario, the mass is set to 330 kg. Agent Constable shows a large spread for the "Vary Mass" scenario, but shows better performance for the "Constant Mass" scenario.

In Fig. 3.16 the spread of the return is evident. Agent Constable is able to handle some of the varying masses, but shows negative returns. In contrast, Agent Massive has similar performance in both scenarios. In the "Vary Mass" scenario Agent Massive achieves a higher return, but also shows a larger spread than the same Agent does in the "Constant Mass" scenario. In the "Constant Mass" scenario Agent Constable clearly performs better, but Agent Massive also shows good and consistent results. Agent Constable achieves better results for the "Constant Mass" scenario, since Agent Constable was trained with this specific mass and thus learned to handle this perfectly. Agent Massive learned a general approach, which will work for a wide mass range, but it is not the perfect solution for the specific mass of 330 kg.

# 4 Conclusion and Outlook

In this thesis the application of deep reinforcement learning to the satellite attitude problem was demonstrated. Furthermore, the basis for the application to active debris removal was set. The satellite was modeled in the Basilisk software and equipped with reaction wheels for the attitude control. An interface between the SAC algorithm and the simulation was established through OpenAI Gym. The SAC algorithm was then trained on different scenarios: Constant initial attitudes, varying initial attitudes and varying initial masses. An overview of the different initial parameters of each scenario can be found in table 3.1. The results of the four scenarios are listed in table 4.1.

Table 4.1: The four scenarios and the results the respective Agents achieved in 100 samples in their environment.

| Scenario | Agent | Return | $\lvert\sigma_{\mathcal{B}/\mathcal{N}\text{final}}\rvert$ [°] | $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ [°] | CI for $\sigma^i_{\mathcal{B}/\mathcal{N}\text{final}}$ |
|---|---|---|---|---|---|
| Constant Init Att | Constable | $3.730 \pm 0.024$ | $4.63 \pm 0.21$ | $0.6 \pm 2.6$ | $[-3.3, 3.1]^\circ$ |
| Vary Init Att | Spinny | $2.5 \pm 1.9$ | $9.3 \pm 6.9$ | $1.1 \pm 6.6$ | $[-9.4, 14.5]^\circ$ |
| Location Pointing | Spinny | $31.02 \pm 0.11$ | $1.3 \pm 0.6$ | $0.3 \pm 0.7$ | $[-1, 1.9]^\circ$ |
| Vary Init Mass | Massive | $1.0 \pm 0.7$ | $4.9 \pm 1.1$ | $-2.0 \pm 2.0$ | $[-5.3, 1.2]^\circ$ |

All agents achieved good results for their respective scenario, but looking at the results in more detail gives insight into the complexity of each scenario. The "Constant Init Att" scenario (section 3.3) served as the starting point. It is the simplest scenario, which is demonstrated both in Agent Constable achieving the highest return and the lowest final attitude error. Agent Spinny was trained with varying initial attitudes (section 3.4), but also achieved good results when the problem was switched to varying desired attitudes, but in two dimensions (section 3.6). While in the "Vary Init Att" scenario Agent Spinny has the highest final attitude error, in the "Location Pointing" scenario Agent Spinny achieves a very low final attitude error. This demonstrates that constraining the problem to two dimensions greatly lowers the complexity. To further improve Agent Spinny, the training duration and the network capacity could be increased, to cover the large state space. In the scenario "Vary Init Mass" (section 3.7), the first step towards the active debris removal was achieved. For active debris removal, the mass varies greatly after capture. This change in mass needs to be handled by the attitude control to detumble the satellite successfully. In this thesis, the masses were varied in the training between $10 - 1000$ kg. Concerning the final attitude error, Agent Massive achieves results similar to Agent Constable. Varying the masses seems to be less complex than varying the initial attitudes. Agent Massive also showed the ability to generalize to higher masses than the Agent was trained with, but performed badly for masses below 150 kg. To mitigate this stacked observations could be implemented. This would help the agent get a sense of the mass of the satellite. In all scenarios, the final attitude error rate $\omega_{\mathcal{B}/\mathcal{N}\text{final}}$ showed a large uncertainty. Currently, only the attitude error is included in the reward modeling. Generally, this also stabilizes the satellite at this position. The large uncertainty of the error rate could indicate that this is not enough to fully stabilize the satellite. To improve this the attitude error rate could be included in the reward modeling.

In conclusion, the deep reinforcement learning approach achieves very good results for satellite attitude control. The trained agent is able to handle a wide array of different initial attitudes, including time dependent goal states. Furthermore, this thesis showed that the agent was able to develop a generalized approach for varying masses. The agent was even able to handle unknown masses it was not trained with. This is exactly what an attitude control system would have to do during an active debris removal. This demonstrates the great promise this approach has for this use case. In the future it would also be interesting to create a scenario that varies both the attitudes and the initial mass, to fully encapsulate the active debris removal scenario. The interplay of two spacecrafts was already investigated in this thesis. To take this one step further the rendezvous maneuver between the chaser and the target can be modeled. The chaser describes the satellite which shall remove the debris fragment, while the target is the debris fragment. The challenge here is to rendezvous with the target without crashing into it. Moreover, the translational motion has to be considered as well as the rotational. Similarly as in the attitude problem, the exact rotational frequency, mass and other parameters of the debris object are usually not known beforehand. Therefore, a DRL approach could prove useful.

# Bibliography

[1] *NASA*. Apr. 2023. URL: https://www.nasa.gov/.

[2] *ESA*. Apr. 2023. URL: https://www.esa.int/.

[3] *SpaceX*. Apr. 2023. URL: https://www.spacex.com/.

[4] *Start-Up Space: Update on investment in commercial space ventures*. Tech. rep. Bryce Tech, 2022. URL: https://brycetech.com/reports.

[5] *Amazon Mega Constellation Project*. Apr. 2023. URL: https://www.aboutamazon.com/news/innovation-at-amazon/what-is-amazon-project-kuiper.

[6] *OneWeb*. Apr. 2023. URL: https://oneweb.net/.

[7] *ClearSpace Company*. Apr. 2023. URL: https://clearspace.today/.

[8] Holger Krag. *Effectiveness of Mitigation Measures*. Space Debris Lecture TU Darmstadt. 2022.

[9] Massimo Tipaldi, Raffaele Iervolino, and Paolo Roberto Massenio. "Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges". In: *Annual Reviews in Control* 54 (2022), pp. 1–23. URL: https://www.sciencedirect.com/science/article/pii/S136757882200089X.

[10] James Allison et al. "Reinforcement Learning for Spacecraft Attitude Control". In: 70th International Astronautical Congress, Oct. 2019.

[11] *Basilisk Software*. Apr. 2023. URL: https://hanspeterschaub.info/basilisk/.

[12] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second Edition. The MIT Press, 2018.

[13] Minghao Han et al. *Reinforcement Learning Control of Constrained Dynamic Systems with Uniformly Ultimate Boundedness Stability Guarantee*. 2020.

[14] Tuomas Haarnoja et al. "Soft Actor-Critic Algorithms and Applications". In: *CoRR* abs/1812.05905 (2018). URL: https://arxiv.org/abs/1812.05905.

[15] *AVS Lab*. Apr. 2023. URL: https://hanspeterschaub.info/AVSlab.html.

[16] *Laboratory for Atmospheric and Space Physics*. Apr. 2023. URL: https://lasp.colorado.edu/.

[17] *Vizard Software*. Apr. 2023. URL: https://hanspeterschaub.info/basilisk/Vizard/Vizard.html.

[18] David A. Vallado and Wayne D. McClain. *Fundamentals of Astrodynamics and Applications*. Fundamentals of Astrodynamics and Applications. Microcosm Press, 2001.

[19] John Alcorn, Cody Allard, and Hanspeter Schaub. "Fully Coupled Reaction Wheel Static and Dynamic Imbalance for Spacecraft Jitter Modeling". In: *Journal of Guidance, Control, and Dynamics* 41 (Jan. 2018), pp. 1–9.

[20] Hanspeter Schaub and John L. Junkins. *Analytical Mechanics of Space Systems*. 4th. Reston, VA: AIAA Education Series, 2018.

[21] Hanspeter Schaub. *Reaction Wheel Scenario*. Apr. 2023. URL: https://hanspeterschaub.info/basilisk/examples/scenarioAttitudeFeedbackRW.html.

[22] *SAC Algorithm*. Oct. 2023. URL: https://docs.ray.io/en/latest/_modules/ray/rllib/algorithms/sac/sac.html.

[23] *Honeywell HR16 Datasheet*. Apr. 2023. URL: https://satcatalog.s3.amazonaws.com/components/223/SatCatalog_-_Honeywell_-_HR16-100_-_Datasheet.pdf.

[24] *Basilisk Location Pointing Module*. URL: https://hanspeterschaub.info/basilisk/Documentation/fswAlgorithms/attGuidance/locationPointing/locationPointing.html.