Multi-Agent Spacecraft Docking with Reinforcement Learning

Selim Olgu Pilav
sop24@cantab.ac.uk

***Abstract*—This research explores the application of Multi-Agent Reinforcement Learning (MARL) to a cooperative spacecraft docking problem with three chaser spacecraft and a lightly tumbling target. The study extends Proximal Policy Optimisation (PPO) based single-agent reinforcement learning (RL) docking to multi-agent spacecraft using Multi-Agent Proximal Policy Optimisation (MAPPO), targeting a simple 3-DOF planar setup. The primary goal is to develop a safe, decentralised docking policy that accounts for low-thrust constraints, fuel efficiency, and inter-agent communication limitations. Agents are only given access to the noisy measurement of the nearest agent to mimic large constellations. The policy is trained with a reward function that penalises position, velocity, fuel consumption, and angular errors while encouraging successful docking and collision avoidance. Experimental validation is conducted via Monte Carlo simulations. Results demonstrate the feasibility of applying MARL to spacecraft docking tasks, achieving a 99.1% docking success rate in simulation. The research highlights the potential of reinforcement learning approaches for future distributed multi-agent space missions. However, further work is needed to address robustness concerns and optimise the policy for more complex scenarios and a large number of agents.**

I. Introduction

Spacecraft docking and proximity operations are essential manoeuvres in space missions. Successful docking requires the spacecraft to execute highly accurate and precise manoeuvres to ensure proper alignment and safety. With the advent of complex missions, such as in-orbit servicing and debris removal, robust autonomous docking capabilities are becoming increasingly important. In such missions, guidance, navigation, and control (GNC) algorithms for docking will require higher levels of autonomy to address challenges such as varying constraints, target motion, and uncertain dynamics [1]. Additionally, the use of multiple smaller satellites is emerging as a cheap solution to various space exploration missions. In particular, the NASA On-Orbit Satellite Servicing Study describes mission concepts in which a central servicer will be sent for on-orbit servicing and deploy smaller satellites, which will later dock back [2]. There are further mission concepts that discuss similar ideas to increase flexibility, reduce cost and improve success rate through the redundancy of multiple spacecraft. These missions will require improvements in coordinated GNC algorithms.

Docking is a precision manoeuvre that needs to be performed under many constraints. These include low-thrust restrictions, no-go zones, limited on-board computer (OBC) resources, and fuel limitations. These restrictions are often increasingly restrictive for smaller spacecraft. Traditional methods for docking control are restrictive and inflexible when faced with scenarios where model identification is difficult, such as in-orbit servicing, where the dynamics of the docking spacecraft are likely to be uncertain due to attached servicing parts. Various traditional control methods have been developed to address this issue. Boyarko et al. [3] developed minimum-time and fuel rendezvous trajectories with a tumbling object. Similarly, Jewison et al. [4] successfully generated probabilistically optimal trajectories under significant uncertainties. However, the high computational cost of both these methods renders them unimplementable in on-board real-time applications. Model predictive control has also been implemented with limited success for challenging docking scenarios, as demonstrated by Weiss et al. [5]. However, such MPC methods are less flexible when dealing with model uncertainties and incur high computational costs. Consequently, RL has been suggested as a potential improvement to traditional methods. RL involves learning a policy to update actions following observations. Due to its model-free framework, it is better suited to handle uncertainty. The motivation for using RL for docking is to generate a policy implementable as a feedback control law. This control law should require minimal computational effort and memory and be robust to initial conditions. In contrast, standard trajectory generation techniques must be recalculated when deviations occur from the standard scenario. Recently, RL has been successfully implemented for spacecraft docking manoeuvres in simulation. Broida and Linares [6] accomplished 3-degree-of-freedom (DOF) rendezvous trajectories using RL with proximal policy optimisation (PPO). This was successfully extended to 6-DOF by Oestreich et al. [7]. Similarly, Hovell et al. developed a deep RL solution for spacecraft proximity operations and docking [8].

However, the use of RL for simultaneous docking of multiple spacecraft is underexplored. Given the advantages of small spacecraft for servicing, debris removal, and exploration, it is likely that future missions will require multi-agent docking capabilities. Due to communication and resource constraints in space, spacecraft swarms will often have to implement decentralised policies with limited information on other agents. The purpose of this research is to extend RL docking applications to the decentralised, multi-agent, cooperative

docking problem under uncertainties in agent dynamics, initial conditions, and inter-agent communication. A simple planar 3-DOF setup with three docking agents will be considered. The primary constraints will be the computational resources, low-thrust, inter-agent communication limitations and strict docking requirements. It is expected that the developed RL policy will achieve a fuel-conscious and, importantly, safe trajectory for the three docking agents.

It should be noted that, although motivated by difficult guidance tasks likely involving many spacecraft operating simultaneously and very large uncertainties, this paper considers a proof-of-concept task. Demonstrating multi-agent reinforcement learning in this simple case will highlight its potential for extension to more realistic 6-DOF orbital scenarios.

II. Reinforcement Learning Overview

RL is a subset of machine learning in which an intelligent agent learns to map observations to actions in a dynamic environment to maximise reward across experienced trajectories. RL is modelled as a Markov decision process with a state space S, action space A, and state transition distribution.

$$P(x_{t+1} \mid x_t, u_t),$$

and reward function

$$r(x_t, u_t),$$

where state $x \in S$, control input $u \in A$, and t is the discrete time-step index. The agent learns a policy by repeatedly interacting with the environment through numerous trajectories. This can be in a simulated or a real environment where the agent receives rewards based on the action taken at each time step. The learning process results in a policy

$$\pi_\theta(u_t \mid x_t)$$

Which is a conditional probability distribution dependent on the parameter vector θ.

One episode with T steps results in a trajectory of state-action pairs, denoted as

$$\tau = [\, x_0, u_0, \ldots, x_T, u_T \,].$$

Rewards received at successive time steps are discounted to accommodate infinite-horizon problems. The discounted sum of rewards over the trajectory is

$$r(\tau) \;=\; \sum_{t=0}^{T} \gamma^t \, r(x_t, u_t),$$

where $\gamma \in (0, 1)$ is the discount factor. RL aims to maximise the expected discounted sum of rewards for the agent across all trajectories:

$$\mathbb{E}_{p_\theta(\tau)}\big[r(\tau)\big] \;=\; \int_\tau r(\tau)\, p_\theta(\tau)\, \mathrm{d}\tau.$$

The policy's conditional distribution has a variance associated with it, which is reduced as the learning progresses to encourage exploitation. After the learning process is complete, a deterministic action is chosen during implementation by setting the variance to zero.

III. Proximal Policy Optimisation

The Proximal Policy Optimisation (PPO) algorithm is used for this reinforcement learning problem [9]. PPO is a model-free, actor-critic method that utilises a clipped surrogate objective function, effectively implementing a trust region. This approach reduces the risk of learning divergence, enhancing the policy update stability. The policy that selects actions (the actor) and an advantage function that evaluates the selected actions (the critic) are learned concurrently. PPO utilises the state-value function $V_w^\pi(x_t)$, estimating the sum of future discounted rewards starting from the current state and using the current policy. The parameter vector w of this function is unknown and is learned simultaneously with the policy parameter vector θ. The advantage function is defined as the difference between the empirical rewards received during the learning process and the state-value function's estimate:

$$V_w^\pi(x_t) \;=\; \mathbb{E}_\pi\Big[\sum_{k=t}^{T} \gamma^{\,k-t}\, r_k(x_k, u_k) \;\Big|\; x_t\Big].$$

$$A_w^\pi(x_t, u_t) \;=\; \sum_{k=t}^{T} \gamma^{\,k-t}\, r_k(x_k, u_k) \;-\; V_w^\pi(x_t).$$

The policy probability ratio is used to prevent overly large policy updates and is defined as:

$$p_t(\theta) \;=\; \frac{\pi_\theta(u_t \mid x_t)}{\hat{\pi}_\theta(u_t \mid x_t)},$$

This compares the policy's probability of selecting a particular action before and after an update. This enters into the PPO objective:

$$J(\theta) = \mathbb{E}_{p(\tau)}\big[\min\big(p_t(\theta)\, A_w^\pi(x_t, u_t),\; \mathrm{clip}[p_t(\theta), \epsilon]\, A_w^\pi(x_t, u_t)\big)\big].$$

The clip operator uses a parameter $\epsilon \in (0, 1)$ to bound the ratio:

$$\mathrm{clip}[p_t(\theta), \epsilon] = \begin{cases} 1 - \epsilon, & \text{if } p_t(\theta) < 1 - \epsilon, \\ 1 + \epsilon, & \text{if } p_t(\theta) > 1 + \epsilon, \\ p_t(\theta), & \text{otherwise.} \end{cases}$$

To learn the state-value function, a cost function $L(w)$ is minimised, defined as the squared error between the value estimate and the empirical return. Together with $J(\theta)$, one performs gradient ascent/descent on θ and w. The following updates are used:

$$\theta^+ = \theta^- \;+\; \beta_\theta\, \nabla_\theta J(\theta)\big|_{\theta=\theta^-},$$
$$w^+ = w^- \;-\; \beta_w\, \nabla_w L(w)\big|_{w=w^-},$$

where β_θ and β_w are the policy and value-function learning rates, respectively.

IV. Multi-Agent Reinforcement Learning (MARL)

For multi-agent spacecraft docking, the MARL can take three forms: centralised training and execution (CTE), centralised training for decentralised execution (CTDE), and decentralised training and execution (DTE). CTE performs well but becomes less scalable as the action and observation spaces grow. This makes centralised execution unsuitable for spacecraft swarms. In space, communication is limited, and a decentralised approach is preferred. CTDE promises to be effective. During training, agents leverage a centralised critic to learn coordinated policies using full system information. However, independent policies are developed for each agent, and actions are executed solely based on each agent's local observations. [10] This suits the communication constraints in space and enhances cooperation and scalability. Recent research on CTDE shows that this approach efficiently handles cooperative multi-agent tasks, making it well-suited for autonomous multi-spacecraft docking. For example, Hua-Ching Chen et al. [11] developed a MARL model for path finding, a similar application to multi-agent docking.

A. Multi-Agent Proximal Policy Optimisation (MAPPO)

The effectiveness and feasibility of MAPPO on cooperative multi-agent tasks were demonstrated by Yu *et al.* [12]. It was concluded that, with minimal modifications and tuning from PPO, MAPPO can achieve strong results with a comparable number of samples to those of off-policy methods across a variety of cooperative multi-agent tasks. Given the success of PPO in spacecraft docking problems, MAPPO is the MARL algorithm of choice for this study, as it extends PPO. The implementation developed in the work of Yu *et al.* is used for this study.

MAPPO uses a centralised value function and independent policy networks for each agent, following the CTDE structure. Two separate neural networks are trained: $\pi_\theta(a_i \mid o_i)$ for the policy, and $V_w(s)$ for the value function. $V_w(s)$ is used only during training for variance reduction, allowing it to incorporate extra global information not present in the agent's local observation, a key feature of CTDE.

The overall system is modelled as a decentralised partially observable Markov decision process (DEC-POMDP) with shared rewards [13]. A DEC-POMDP is defined by

$$\langle S, A, O, R, P, n, \gamma \rangle.$$

S is the state space. A is the shared action space for each agent i. $o_i = O(s; i)$ is the local observation for agent i at global state s. $P(s' \mid s, A)$ denotes the transition probability from s to s' given the joint action

$$A = \big(a_1, \ldots, a_n\big)$$

for all n agents. $R(s, A)$ denotes the shared reward function. γ is the discount factor. Agents use a policy

$$\pi_\theta(a_i \mid o_i)$$

parameterized by θ to produce an action a_i from the local observation o_i, and jointly optimize the discounted accumulated reward

$$J(\theta) = \mathbb{E}_{A^t, s^t}\Big[\sum_t \gamma^t \, R(s^t, A^t)\Big]$$

where

$$A^t = \big(a_1^t, \ldots, a_n^t\big)$$

is the joint action at time step t. [12].

The agents in the problem are homogeneous, meaning that they have identical observation and action spaces. A centralised critic is used while actors are not shared, and each agent has its own policy for specialised learning. In preliminary experiments, it was found that docking success improves with separate actors, given each agent's unique initial conditions, despite slower learning.

V. Problem Definition

A 3-DOF planar multi-agent scenario with three small chaser spacecraft and a large central target is investigated. Although the MAPPO algorithm can handle a more complex scenario, this simple task is considered in this research as a proof of concept. A double-integrator planar dynamics model is used:

$$\dot{\mathbf{r}} = \mathbf{v}, \qquad \dot{\mathbf{v}} = \frac{\mathbf{F}}{m} \tag{1}$$

$$\dot{\theta} = \omega, \qquad \dot{\omega} = \frac{\tau}{J} \tag{2}$$

This model provides a reasonable approximation of relative motion over the short time and distance scales considered in this study [14]. Future work should replace this simplified model with Clohessy–Wiltshire (CW) relative-motion dynamics and extend it to full 6-DOF motion [15].

The dynamics and kinematics are simulated by the Vectorized Multi-Agent Simulator (VMAS) [16]. It is comprised of a fully-differentiable vectorised 2D physics engine using a semi-implicit Euler method for integration with customisable scenarios. Friction, damping and gravity are all set to zero in the simulator to replicate space conditions. Additionally, VMAS is integrated with BenchMARL [17], which contains the MAPPO implementation discussed above with flexible parameter settings.

The target is modelled as a circular body rotating at a constant angular velocity ω_{spin} to represent a lightly tumbling object. An inertial frame $\mathcal{I}$ is defined whose origin is fixed at the target center with fixed axes in inertial space. Three chaser-centred body frames $\mathcal{B}_i$ are used, where i is the agent index. Each agent has an action space a_i and a local observation space o_i at a global state s. These form the joint observation and action spaces used in MAPPO discussed in the previous

section. The local state of the agent is denoted s_i. The states and the action spaces are in the $\mathcal{I}$ frame and are as follows:

$$\mathbf{s}_i = \begin{bmatrix} x_i \\ y_i \\ \dot{x}_i \\ \dot{y}_i \\ \theta_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \mathbf{r}_i \\ \dot{\mathbf{r}}_i \\ \theta_i \\ \dot{\theta}_i \end{bmatrix}, \quad \mathbf{a}_i = \begin{bmatrix} f_{x,i} \\ f_{y,i} \\ \tau_i \end{bmatrix}$$

$$\mathbf{o}_i = \begin{bmatrix} \mathbf{r}_{i,\text{target}} - \mathbf{r}_i^{(\text{self})} \\ \dot{\mathbf{r}}_{i,\text{target}} - \dot{\mathbf{r}}_i^{(\text{self})} \\ \theta_{\text{target}} - \theta_i \\ \dot{\theta}_i \\ \left(\mathbf{r}_i^{(\text{nearest})} - \mathbf{r}_i^{(\text{self})}\right) \cdot \epsilon \\ \left(\dot{\mathbf{r}}_i^{(\text{nearest})} - \dot{\mathbf{r}}_i^{(\text{self})}\right) \cdot \epsilon \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{i,\text{rel}}^{(\text{self})} \\ \dot{\mathbf{r}}_{i,\text{rel}}^{(\text{self})} \\ \theta_{i,\text{rel}} \\ \dot{\theta}_i \\ \mathbf{r}_{i,\text{rel}}^{(\text{nearest})} \cdot \epsilon \\ \dot{\mathbf{r}}_{i,\text{rel}}^{(\text{nearest})} \cdot \epsilon \end{bmatrix}$$

Where x_i, y_i describe the Cartesian coordinate position, θ_i the rotation angle from the horizontal, f the thrust, and τ the torque. Each agent has a docking target defined as the target location for the center of the agents with Cartesian coordinates $\mathbf{r}_{i,\text{rel}}$.

In larger constellations, each spacecraft may have access only to the state information of nearby agents, with significant uncertainty. To replicate the communication constraints in space, the observation space of an agent consists of the local state concatenated with the Cartesian state of the nearest agent with a multiplicative random noise vector $\epsilon \sim \mathcal{U}(0.95, 1.05)$. For each agent, the Euclidean distance from its center to the centers of all other agents is calculated, and the nearest agent is defined to have the smallest distance. This is recomputed at every control step and updated in the agent's observation.

The control actions assume full continuous and decoupled actuation of the spacecraft in all 3-DOFs to simplify the problem, as is common in many theoretical RL studies discussed earlier. This thrust and torques reasonably approximate the time-averaged effect of pulsed reaction control systems and are both limited by max/min constraints. The states are discretised at a sample period of 1 second, and actions are updated. This large sample period ensures computational efficiency. The dynamics are calculated accurately with 10-sub steps per control update.

For simplicity, the chasers are modelled as squares of side length 0.1 meters docking to a much larger circular target with radius 0.97 meters centred at the origin. Therefore, the target radius for the agents is 1.02 meters, and any further incursions without satisfying the docking criteria are considered collisions. The moment of inertia of the square agents was tied to the mass by the following equation: $J = m\frac{0.1^2}{6}$. This problem is inspired by future mission concepts where multiple smallsats dock to a central mother spacecraft or to an uncooperative target for servicing. The problem is formulated so that the docking ports of the chaser and the target meet under given constraints for successful docking, given in table I.

The docking tolerances in Table I are representative of cubesat docking operations but are not tied to a specific mission. They are chosen to be somewhat looser than the values reported in the literature. These typically require $< 1cm$ lateral alignment and attitude misalignments of $< 2^\circ$ [18] [19].

Docking State	Goal	Acceptable Deviation	Units
$\mathbf{r}_{\text{rel}}$	0	± 0.02	m
Approach Velocity	0	≤ 0.1	m/s
Lateral Velocity	0	± 0.02	m/s
θ_{rel}	0	± 5	deg

TABLE I: Criteria for Successful Docking Relative to Target

A. Reward Function

The reward function is defined as follows:

$$\begin{aligned} r_i(\mathbf{x_t}, \mathbf{u_t}) = &- \left(\mathbf{r_{i,rel}}\right)^\top K \left(\mathbf{r_{i,rel}}\right) - k\theta_{i,rel}^2 \\ &- \left(\dot{\mathbf{r}}_{\mathbf{i,rel}}\right)^\top Q \left(\dot{\mathbf{r}}_{\mathbf{i,rel}}\right) - \left(\mathbf{a_i}\right)^\top P(\mathbf{a_i}) \\ &+ \text{Docking Bonus} + \text{Collision Penalty} \end{aligned} \tag{3}$$

where K, k, Q and P are weighing matrices for the rewards and angles are in radians. The position and angle errors incentivise the agents to move towards the target at the correct attitude. The velocity error reward penalises fast movement. Additionally, it is aimed to achieve fuel-conscious control through the "fuel penalty", which penalises the agent for each action. A docking bonus is introduced to an agent when docked successfully. If all agents dock, the episode is terminated. To encourage docking, two further incentives are given to the agents:

$$\text{Incentives} = \begin{cases} 0, & \text{if } \|\mathbf{r}_{i,rel}\| > 1 \\ I_1, & \text{if } 0.2 < \|\mathbf{r}_{i,rel}\| \leq 1 \\ (I_1 + I_2), & \text{if } \|\mathbf{r}_{i,rel}\| < 0.2 \end{cases} \tag{4}$$

Finally, a scalar collision penalty is applied to inter-agent and agent-target collisions of the form:

$$f(r_{col}; r_h, r_s) = \text{clip}_{[0,1]}\left(\frac{r_s - r_{col}}{r_s - r_h}\right),$$

where $\text{clip}_{[0,1]}(x) = \min\{1, \max\{0, x\}\}$.

$$\Gamma_i^{\text{target}} = -C_{target} \sin\Big(\tfrac{\pi}{2} f(\|\mathbf{r}_{i,rel}\|; r_{h,target}, r_{s,target})\Big)$$

$$\Gamma_i^{\text{inter}} = -C_{inter} \sum_{j \neq i} \sin\Big(\tfrac{\pi}{2} f(\|\mathbf{r}_i - \mathbf{r}_j\|; r_{h,inter}, r_{s,inter})\Big)$$

r_h and r_s are the hard and soft collision distances, respectively. This reward prevents sharp gradient penalties, stabilising the algorithm. The agents can gain valuable experience outside the hard collision distance. If breached, the episode is terminated by default for collisions with the target. For inter-agent collisions, episode termination is linked to a separate variable $r_{h,ep}$. This is set to -1 during the initial phase of training, disabling hard inter-agent collisions and preventing premature episode termination, helping agents learn to dock.

VI. Experimental Set-Up

The central target was set to have much larger dimensions than the chasers, and the initial location of agent targets was defined at the edge of the circle:

Parameter	Value	Units
Collision radius	1.02	m
$\mathbf{r_{target,1}}$	$[0.2, -1.0]$	m
$\mathbf{r_{target,2}}$	$[-0.2, -1.0]$	m
$\mathbf{r_{target,3}}$	$[0.5692, -0.846]$	m

TABLE II: Collision Geometry and Target Positions

$r_{s,target}$ is initially set slightly smaller than the boundary of collision to prevent early penalties for near-docks, and $r_{h,target}$ is a 0.12-meter penetration into the collision zone. The thrust constraints were set to be representative of smallsats and are a key factor in increasing the problem's difficulty.

Action Command	Value	Units
a_i	$\pm[0.005, 0.005, 0.00005]$	[N, N, Nm]

TABLE III: Force Constraints

Initial conditions were set to test the cooperative collision avoidance and docking capabilities of the policies. The agents begin close to each other with significant uncertainty. The agents' independent near-optimal paths to docking under most starting conditions would result in inter-agent collisions, forcing them to learn collision avoidance.

Parameter	Training Range	Units
$\mathbf{r_1}$	$[0.6, -3.3] \pm[0.2, 0.1]$	m
$\mathbf{r_2}$	$[0.8, -2.9] \pm[0.2, 0.1]$	m
$\mathbf{r_3}$	$[0.2, -2.9] \pm[0.2, 0.1]$	m
$\dot{\mathbf{r}}_{\mathbf{all}}$	$[0, 0] \pm[0.01, 0.01]$	m/s
$\theta_{\mathbf{all}}$	0 ± 30	deg
$\dot{\theta}_{\mathbf{all}}$	0 ± 2	deg/s
Chaser Mass	1 ± 0.1	kg
ω_{spin}	0.143 ± 0.143	deg/s

TABLE IV: Initial Conditions

The following tables demonstrate the reward function weights, learning hyperparameters, and the neural network architecture. These parameters were initially based on previous docking RL studies and then further refined.

Parameter	Value	Parameter	Value
K	diag([1, 1])	k	0.5
Q	diag([200, 200])	P	100
Docking Bonus	600	C_{target}	25
I_1	0.5	I_2	1

TABLE V: Reward Function Parameters

Description	Value	Description	Value
Frames per update	51,200	Discount factor	$\gamma = 0.99$
GAE parameter	$\lambda = 0.9$	Clip Range	$\epsilon = 0.14$
Entropy Coefficient	0.001	Max Episode Steps	120
Critic Coefficient	1	Environment No.	400

TABLE VI: Training Hyperparameters

	Policy Network		Value Network	
Layer	**Neurons**	**Activation**	**Neurons**	**Activation**
1st hidden	256	tanh	256	tanh
2nd hidden	256	tanh	256	tanh

TABLE VII: Neural Network Structural Parameters

Curriculum learning was employed in the training. Due to tight thrust constraints and short inter-agent distances, the agents struggled to learn collision avoidance alongside docking when episodes were terminated at inter-agent collisions. Therefore, initially, hard inter-agent collisions were disabled by setting $r_{h,ep} = -1$. The following table displays the curriculum where $\alpha =$ learning rate:

Stage	Frames	α	$\dot{\mathbf{r}}_{\mathbf{all}}$ init range (m/s)
1	4.2×10^6	3×10^{-4}	$[-0.01, 0.01]$
2	5.0×10^6	3×10^{-5}	$[-0.01, 0.01]$
3	6.0×10^6	3×10^{-5}	$[-0.01, 0.01]$
4	7.5×10^6	1×10^{-5}	$[-0.02, 0.02]$

TABLE VIII: Curriculum schedule in cumulative number of environment frames.

Stage	C_{inter}	$r_{h,ep}$	$r_{h,\text{inter}}$	$r_{s,\text{inter}}$	$r_{h,\text{target}}$	$r_{s,\text{target}}$
1	1.0	-1	0.15	0.20	0.90	0.98
2	1.0	0.15	0.15	0.20	0.90	0.98
3	5.0	0.16	0.15	0.20	0.90	0.98
4	5.0	0.17	0.15	0.20	0.90	1.02

TABLE IX: Inter-agent and target collision penalty radii.

In step 2 of the curriculum, hard inter-agent collisions were enabled, and $r_{h,\text{ep}}$ and C_{inter} slowly increased to further disincentivise near passes. Simultaneously, the learning rate was reduced, allowing the agents to learn to strictly avoid collision while not deviating from the previous policy quickly. The final step involved increasing the initial velocity of the agents beyond the desired evaluation range, as most inter-agent collisions occurred at large initial relative velocities. The $r_{h,\text{target}}$ was set to 0.9 and $r_{s,\text{target}}$ to 0.98 to initially allow the agents to gain useful experience from minor collisions. $r_{s,\text{target}}$ was increased to the actual collision radius at the last step of the curriculum for further safety, although target-collisions were not occurring.

VII. Results and Discussion

Learning occurred over 7.5 million frames in total training three independent policies without actor parameter sharing. Each episode was limited to 120 frames, or 120 seconds, which was plenty of time for the agents to dock under the given conditions. The policy was updated every 51 thousand frames, and evaluations with 200 episodes ran every 102,400 frames. The initial conditions were randomly sampled from Table IV and adjusted based on the curriculum. All results are reported for a single trained policy, initialised with random seed 0. Preliminary experiments with alternative seeds showed noticeable variance in convergence speed, and performing a seed-averaged study is left for future work.

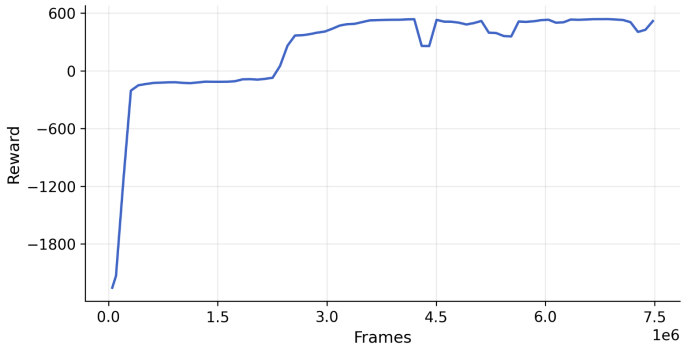

Fig. 1: Mean episode reward during evaluation

Figure 1 displays the mean episode reward during evaluation. There is rapid improvement in rewards early on as the agents learn to aim for the target. Following this, the reward plateaus as the agents attempt to balance approaching the target while avoiding significant inter-agent collision penalties, despite the lack of a hard collision at this stage. Eventually, the docking bonus is discovered, and the reward sharply rises. Following this point, the curriculum is applied to further disincentivise inter-agent collisions, and minor policy adjustments are optimised for fuel use.

A. Monte Carlo Evaluation

To assess generalisation beyond the training environment, the trained policy was exported and evaluated for 10,000 episodes under randomised initial conditions in a separate double-integrator dynamics simulator. To better reflect real-world communication constraints, measurement noise and drift were injected into inter-agent communication. The communication is implemented as a finite buffer, so each agent receives delayed neighbour measurements. The bias drift fraction sets the size of the random-walk step. The per-step bias change has a standard deviation equal to the bias drift fraction times the nominal noise standard deviation.

Description	Value
Position noise (1σ)	0.02 m
Velocity noise (1σ)	0.005 m/s
Position quantization	0.01 m
Velocity quantization	0.001 m/s
Bias drift fraction per step	0.02
Link delay	200 ms
Signal dropout probability per step	0.001

TABLE X: Inter-agent communication constraint parameters used in evaluation.

The final policy was evaluated in 10000 episodes with randomly sampled initial conditions from table IV and achieved a multi-agent docking success rate of $\mathbf{99.1}\%$ with 95% Wilson confidence interval [0.989, 0.993] [20]. The failures were all due to inter-agent collisions during transience when initial velocities were particularly unfavourable. When initial velocities were limited to [-0.001,0.001], the observed success rate was $\mathbf{100}\%$. Nevertheless, the policy's success clearly highlights its potential, given the large uncertainties in the initial conditions and realistic noise. In comparison, the same training resulted in 90.7% success and 1.3% inter-agent collisions when actors were shared, indicating the policy didn't generalise across agents well.

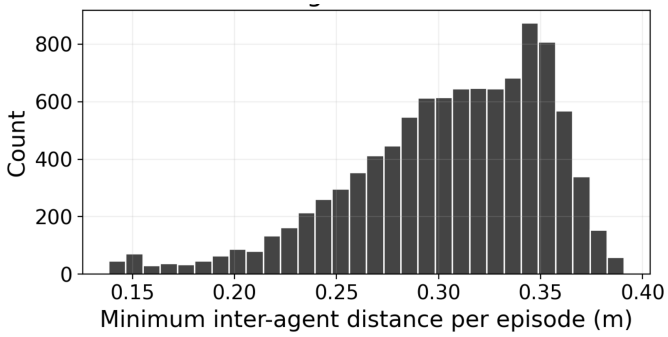

Fig. 2: Minimum inter-agent distance per episode in the Monte-Carlo simulation

The average minimum inter-agent distance across the simulation was 0.303 m, a safe distance. Figure 2 shows the distribution of minimum distances. Only 3.32% of scenarios resulted in inter-agent distances below 0.2, which may be considered the danger zone, highlighting the policy's efficacy not only in preventing collisions but also in maintaining a safe distance.

The quality of communication constraints was critical to the docking success. Under perfect inter-agent communication, a success rate of $\mathbf{99.2}\%$ was observed. This slight but measurable increase in success rate underscores the safety-critical role of resilient inter-spacecraft communication. The success was particularly dependent on the position noise, and the following table highlights this effect:

Position noise (1σ)	Success %
0.02	99.1
0.1	97.9
0.2	80.8

TABLE XI: Inter-agent communication position noise effect on success.

Equally, the uncertainty in mass was challenging. In real operation, an on-board estimate of mass from acceleration measurements following thrust would be valuable and could significantly increase the docking success rate.

Fig. 3 displays the 20 randomly chosen trajectories from the evaluation at a constant ω. Although not seen in this figure, the attitude correction was within the bounds in each evaluation episode.

For a baseline for fuel consumption, a discrete-time LQR controller is designed on the planar double-integrator model with state $x = s_i$ and input $u = a_i$, the same as the RL environment state. The quadratic cost is

$$J = \sum_{k=0}^{N-1} (x_k^\top Q x_k + u_k^\top R u_k),$$

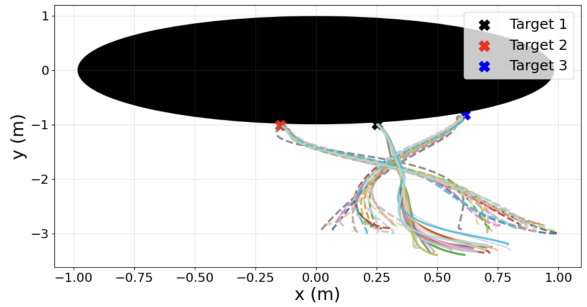

Fig. 3: Trajectories from 20 random initial conditions and mass within the training range at ω_{spin} = 0.057 deg/s

with

$$Q = \text{diag}(q_x, q_y, q_{\dot{x}_i}, q_{\dot{y}_i}, q_\theta, q_{\dot{\theta}}), \quad R = \text{diag}(r_{f_{x,i}}, r_{f_{y,i}}, r_{\tau_i}).$$

Q and R are initially selected according to Bryson's rule using docking conditions and force bounds, and then the weights are scaled to prioritise alignment and relative velocity [21]. The final values are:

$$Q = \text{diag}(10,\ 1,\ 200,\ 20,\ 2,\ 0.1)$$

$$R = \text{diag}(60000,\ 100000,\ 1000).$$

The LQR trajectories are collision-agnostic and result in inter-agent collision in 86.8% in the same Monte-Carlo simulation. The following table compares the RL policy's fuel consumption with the collision-agnostic LQR policy's fuel consumption when hard inter-agent collisions are disabled for the latter.

Agent	RL	LQR reference	Relative change [%]
Chaser 1	0.174	0.146	+19.2
Chaser 2	0.188	0.182	+3.3
Chaser 3	0.289	0.133	+117.3
Mean over agents	0.217	0.154	+40.9

TABLE XII: Mean fuel consumption per agent over Monte Carlo episodes at $\omega = 0$. Fuel is measured as the time integral of the sum of the magnitude of the commanded translational force, $\int_0^T \|f_{x,i}(t)\| + \|f_{y,i}(t)\|\,\mathrm{d}t$.

For the system, the RL policy's translational fuel consumption is $+40.9\%$ higher. However, this enables the collision avoidance and is a moderate increase given the extra manoeuvres required. Attitude fuel is not taken into account here as it is decoupled from the trajectory.

VIII. Conclusion

This research has demonstrated the feasibility of a MAPPO-based decentralised reinforcement learning framework for a simple case with three agents and a lightly tumbling object. The algorithm mostly avoids inter-agent collisions while achieving a high docking success rate. This success is achieved at the cost of an increase of approximately 40.9% in total translational fuel consumption compared to the collision-agnostic LQR baseline. This additional fuel is used to perform the collision-avoidance manoeuvres. The uncertainty about mass and communication constraints was found to be the most challenging for the policy. The ease of training, success in inter-agent collision avoidance, and low on-board implementation cost make this an attractive framework for scaling to many more agents in more complex scenarios. Despite this success, this algorithm fails to guarantee stability and robustness. Consequently, it would be wise to implement it with a certified safety filter that offers these formal guarantees. Future work may involve 6-DOF dynamics with CW frame relative motion to address more realistic scenarios. Additionally, hardware-in-the-loop experiments would address the simulation-to-reality gap, particularly in inter-agent communications.

References

[1] NASA Technology Roadmaps TA_4: Robotics and Autonomous Systems, 2015. Available at: https://www.nasa.gov/wp-content/uploads/2016/08/2015_nasa_technology_roadmaps_ta_4_robotics_and_autonomous_systems_final.pdf [Accessed 3 Nov. 2025].

[2] On-orbit satellite servicing study project report. Technical report, National Aeronautics and Space Administration Goddard Space Flight Center, 2010. Available at: https://www.nasa.gov/wp-content/uploads/2023/10/nasa-satellite-servicing-project-report-0511.pdf [Accessed 3 Nov. 2025].

[3] G. Boyarko, Y. Yakimenko, and M. Romano. Optimal rendezvous trajectories of a controlled spacecraft and a tumbling object. *Journal of Guidance, Control, and Dynamics*, 34(4):1239–1252, June 2011.

[4] C. Jewison and D. Miller. Probabilistic trajectory optimization under uncertain path constraints for close proximity operations. *Journal of Guidance, Control, and Dynamics*, 41(9):1843–1858, 2018.

[5] A. Weiss, M. Baldwin, R. S. Erwin, and I. Kolmanovsky. Model predictive control for spacecraft rendezvous and docking: Strategies for handling constraints and case studies. *IEEE Transactions on Control Systems Technology*, 23(4):1638–1647, July 2015.

[6] J. Broida and R. Linares. Spacecraft rendezvous guidance in cluttered environments via reinforcement learning. In *Proceedings of the 29th AAS/AIAA Space Flight Mechanics Meeting*, Ka'anapali, HI, January 2019.

[7] C. E. Oestreich, R. Linares, and R. Gondhalekar. Autonomous six-degree-of-freedom spacecraft docking maneuvers via reinforcement learning. arXiv preprint arXiv:2008.03215, 2020.

[8] Kirk Hovell and Steve Ulrich. Deep reinforcement learning for spacecraft proximity operations guidance. *Journal of Spacecraft and Rockets*, 58(2):254–264, Mar 2021.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[10] C. Amato. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning. arXiv preprint arXiv:2409.03052, 2024.

[11] H.-C. Chen, S.-A. Li, T.-H. Chang, H.-M. Feng, and Y.-C. Chen. Hybrid centralized training and decentralized execution reinforcement learning in multi-agent path-finding simulations. *Applied Sciences*, 14(10):3960, 2024.

[12] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre M. Bayen, and Yi Wu. The surprising effectiveness of MAPPO in cooperative, multi-agent games. *CoRR*, abs/2103.01955, 2021.

[13] Frans A. Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[14] K. Saulnier, D. Pérez, R.C. Huang, D. Gallardo, G. Tilton, and R. Bevilacqua. A six-degree-of-freedom hardware-in-the-loop simulator for small spacecraft. *Acta Astronautica*, 105(2):444–462, 2014.

[15] W. H. Clohessy and R. S. Wiltshire. Terminal guidance system for satellite rendezvous. *Journal of the Aerospace Sciences*, 27(9):653–658, Sep 1960.

[16] Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. VMAS: A vectorized multi-agent simulator for collective robot learning. *The 16th International Symposium on Distributed Autonomous Robotic Systems*, 2022.

[17] Matteo Bettini, Amanda Prorok, and Vincent Moens. BenchMARL: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(217):1–10, 2024.

[18] Fabrizio Stesina. Tracking model predictive control for docking maneuvers of a CubeSat with a big spacecraft. *Aerospace*, 8(8):197, Jul 2021.

[19] Camille Pirat, Finn Ankersen, Roger Walker, and Volker Gass. Vision based navigation for autonomous cooperative docking of cubesats. *Acta Astronautica*, 146:418–434, May 2018.

[20] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209, Jun 1927.

[21] Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere, Washington, DC, 1975.