Hybrid reinforcement learning enhanced trajectory planning for an ADR scenario for combined control of a satellite with a 7-axis robotic arm using Modelica/FMI

Matthias J. Reiner[1]

[1]DLR, Institute of Robotics and Mechatronics, Oberpfaffenhofen, Germany, Matthias.Reiner@dlr.de

Abstract

This work presents a novel hybrid trajectory planning algorithm that leverages reinforcement learning (RL) to address the challenges of active debris removal (ADR) in space, specifically for the combined control of a satellite and its 7-axis robotic arm. The proposed approach integrates a lightweight RL policy with correction algorithms and classical trajectory planning, enabling robust and collision-free maneuvering of the chaser satellite and precise placement of the robotic gripper near the target grasp point. This hybrid method is designed to handle uncertainties in target dynamics and sensor measurements, while remaining feasible for implementation on space-grade hardware. The effectiveness of the algorithm is demonstrated through comprehensive simulation studies using Modelica/FMI, validating its capability to generate safe and reliable trajectories in complex ADR scenarios.

Keywords: Reinforcement Learning, Trajectory Planning, Active Debris Removal, Combined Control, Robotics, Modelica, FMI

1 INTRODUCTION

Inactive satellites and debris, especially in the most used Low Earth Orbits (LEO), have been known to be a potential problem for a long time. The number of satellites increases every year, which increases the chance of a collision. Active Debris Removal (ADR) using satellites (chaser) equipped with a robotic arm is a possible technical solution to reduce this risk (Liou and Johnson 2006). ADR missions with robotic arms are very challenging, because they involve complex systems consisting of a satellite with a robotic arm with many degrees of freedom, while the information about a debris target is not always perfect (Weiss and Sarli 2014). Recent advances in the field of reinforcement learning (RL) could help solve problems in the field of ADR, especially when large uncertainties are involved. RL algorithms can adapt to changing environments, making them suitable for such tasks (Arulkumaran, Deisenroth, Miles Brundage, et al. 2017). However, the onboard hardware of satellites is usually very limited in its computational capabilities, and questions about the robustness of RL algorithms are still not completely solved (Kober, Bagnell, and Peters 2013; Han et al. 2023; Arulkumaran, Deisenroth, Michael Brundage, et al. 2017). In this paper, a hybrid trajectory planning algorithm is presented, which tries to combine conventional path planning methods and control strategies with an RL algorithm to address uncertainties in the ADR scenario.

Reinforcement learning (RL) is particularly attractive for ADR because it can learn adaptive strategies for collision avoidance and trajectory planning in the presence of nonlinear dynamics and uncertainty, where classical methods often face limitations.

The field of reinforcement learning (RL) for satellite on-orbit servicing (OOS) has witnessed notable advancements, particularly in the area of robotic arms on satellites. RL-based approaches have been developed to autonomously detect potential collisions, rendezvous with target satellites, and execute optimal collision avoidance maneuvers (CAMs) (Patnala and Abdin 2024). In addition, deep reinforcement learning (DRL) has been employed for autonomous guidance of redundant space manipulators. These approaches focus on solving path planning during the motion-synchronization phase with the mission target, using algorithms like Proximal Policy Optimization (PPO) to optimize the manipulator's guidance law (Ambrosio et al. 2024).

Furthermore, reinforcement learning algorithms with fast convergence have been proposed for routing in Low Earth Orbit (LEO) satellite networks. These algorithms address the dynamic topology changes and transmission requirements of LEO networks, enabling satellites to quickly adapt their routing strategies based on network link status updates (Ding et al. 2023).

Additionally, reinforcement learning techniques have been applied to path-planning for smart imaging of uncooperative space objects, enhancing the precision and efficiency of on-orbit servicing missions (Brandonisio, Lavagna, and Guzzetti 2021).

2 MODELLING AND SCENARIO

This paper focuses on a simulation study of an ADR scenario that is based on the inactive (target) satellite Envisat and a (chaser) satellite equipped with a robotic arm. The scenario starts with the chaser already in close proximity with variable initial conditions with respect to the starting distance and target spinning rate to Envisat. The angular

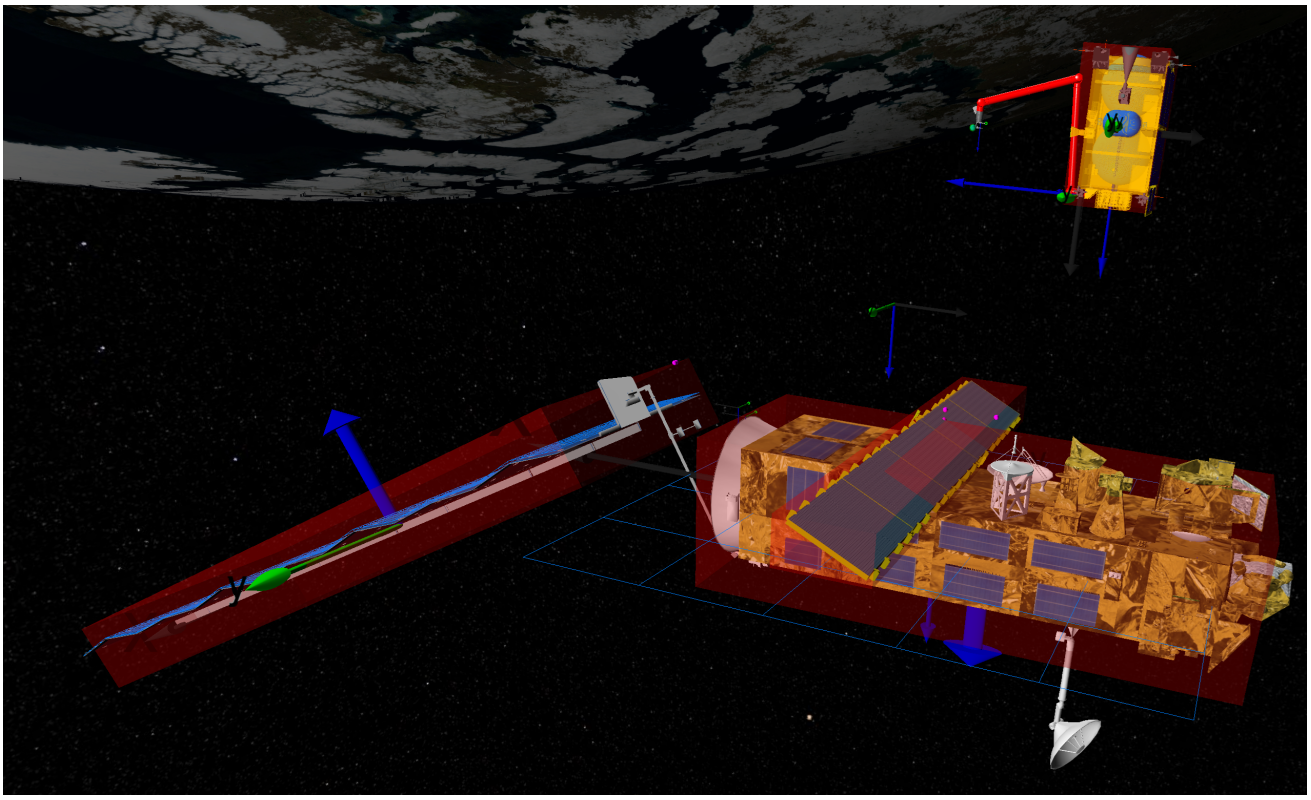


Figure 1. Visualization of the ADR scenario with Envisat and the chaser satellite with its 7-axis robotic arm and gripper. The four red bounding boxes around the target and chaser satellite are used for the collision avoidance of the RL algorithm. The computed critical closest points on the bounding boxes are shown as small pink spheres.

rates between target and chaser are assumed to be synchronized at the start of the simulation.

The scenario consists of multiple phases with different control modes. The first phase, which is the main focus of this paper, is the collision-free approach of the chaser satellite above Envisat's adapter ring and placing the gripper (also collision free) very close to the grasping point on the adapter ring.

This phase is handled by the hybrid RL algorithm which will be described in section 3. After this phase classical trajectory planning is used to stay synchronized with the spinning target using the combined control and grasping the adapter ring. After the gripper at the robot's tool center point (TCP) is closed, a detumbling mode is activated to reduce the angular rate of the target. After the detumbling is completed the deorbiting of the target would be started, but these phases are not considered in this work.

The models and scenario used are based on previous simulation studies, which were performed together with ESA (M. Reiner 2021; M. Reiner, Fernandez, and Ortega 2017; M. Reiner 2016), but were extended by additional uncertainty parameters for this scenario with respect to the tumbling (spinning) rate of the target and the starting distance for the trajectory planning.

The simulation models are created using the equation-based modeling language Modelica and dedicated Modelica-based libraries, the DLR Space Systems Library (M. Reiner and Bals 2014) and the DLR Environment Library (Briese, Klöckner, and M. Reiner 2017) and consider the following aspects (more details are given in the previous publications):

- Orbit dynamics for Envisat's sun synchronous orbit (LEO).
- First order approximation for the elasticity of Envisat's large solar array.
- Chaser satellite with 3D-pendulum models to approximate the fuel and oxidizer sloshing of the large tanks.
- Detailed control thruster array with 24 individual 22 N thrusters. The thruster firing pulse sequencing is implemented using a Pulse Width Pulse Frequency (PWPF) control in addition to a constrained least square (CLS) force and torque allocation.
- 7-Axis robotic arm mounted on top of the chaser with a fourth order approximation for each of the seven joints which approximate the flexibility of the gearboxes and friction as well as the internal high frequency torque control.
- Coupling and reaction forces between all elements are automatically considered by the multibody im-

DOI
10.3384/ecp218489

plementation supported by the Modelica Standard Library.

- The grasping of the robotic gripper of the adapter ring of Envisat is approximated by a switchable force and torque constraint with Baumgarte-like stabilization (Acquatella B. and M. J. Reiner 2014)
- Approximated IMU uncertainty using randomized simulated measuring noise and a simplified Exponentially Correlated Random Variable (ECRV) based camera performance model for the on-board camera and Light Detection and Ranging (LIDAR) system uncertainty.
- Robust H-infinity-based controller with outer Quaternion feedback loop for the combined control of the chaser and its 7-axis robotic arm.

To account for the uncertainty of the target's spinning rate, the initial angular velocity is allowed to vary within the following range:

$$\omega_{\mathrm{ta},0} = (\omega_{\mathrm{ta},0,x}, \omega_{\mathrm{ta},0,y}, \omega_{\mathrm{ta},0,z})^T \tag{1a}$$

$$\text{with: } -1^\circ/s <= \omega_{\mathrm{ta},0,x} <= 1^\circ/s, \tag{1b}$$

$$-1^\circ/s <= \omega_{\mathrm{ta},0,y} <= 1^\circ/s, \tag{1c}$$

$$-5^\circ/s <= \omega_{\mathrm{ta},0,z} <= 5^\circ/s \tag{1d}$$

Similarly, the (relative) starting distance of the chaser to the target (with respect to the target's CoM) is allowed to vary within the following range:

$$r_{\mathrm{ta},0} = (r_{\mathrm{ta},0,x}, r_{\mathrm{ta},0,y}, r_{\mathrm{ta},0,z})^T \tag{2a}$$

$$\text{with: } -2.0\,\mathrm{m} \leq r_{\mathrm{ta},0,x} \leq 2.0\,\mathrm{m} \tag{2b}$$

$$-2.0\,\mathrm{m} \leq r_{\mathrm{ta},0,y} \leq 2.0\,\mathrm{m}, -7.0\,\mathrm{m} \leq r_{\mathrm{ta},0,z} \leq -3.0\,\mathrm{m} \tag{2c}$$

For the RL algorithm, which requires many simulation steps, this model is too complex, and the training of the RL-algorithm would take an extremely long time on the available hardware (a single laptop with an Nvidia GeForce RTX 4090 GPU and Intel i9-13950HX CPU). Therefore, a simplified model is derived from the complex model for the training phase of the RL algorithm. However, the full model is later used for verification and validation (V&V). The simplified model only considers the kinematics for the chaser satellite and robotic gripper (at TCP). The spinning target is approximated as a rigid body with given initial conditions in a perfect zero gravity environment. The inputs to the model are direct kinematic translations for the chaser and robot TCP (without controller or actuator/sensor models). The outputs of the model are the observations and reward functions for the RL algorithm, which will be described in detail in sec. 3. This model can be computed very quickly, and is suitable for the time-consuming training process of the RL algorithm.

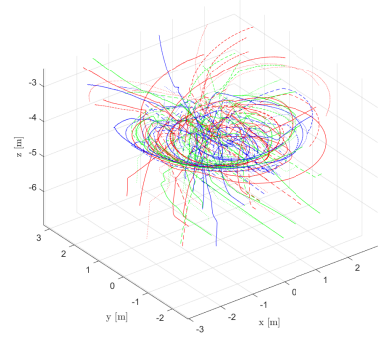


Figure 2. 3D-Visualization for the generated reference trajectories for the chaser satellite $r_{\mathrm{CH,ref}}$ (in relative target coordinate system) for the 100 random scenarios. From a random start location the trajectories convert to the desired position close to the grasping point on the target.

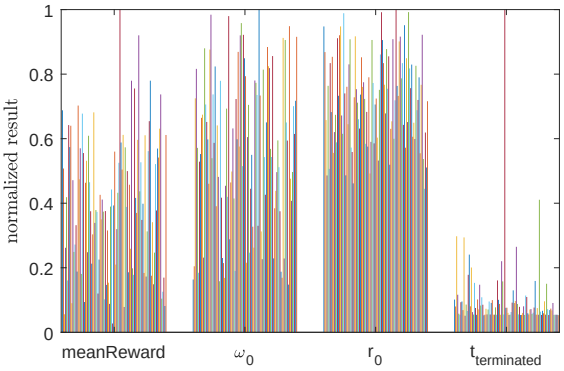


Figure 3. Visualization of the result of 100 random scenarios for different start values for the initial target angular velocity (vector length ω_0) and distance to target (vector length r_0).

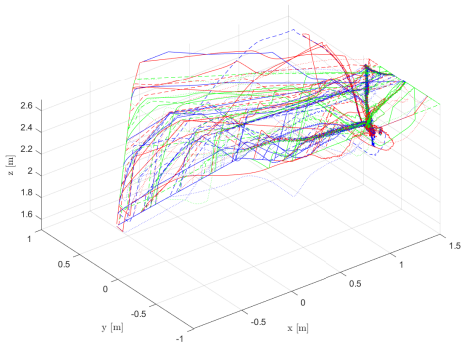
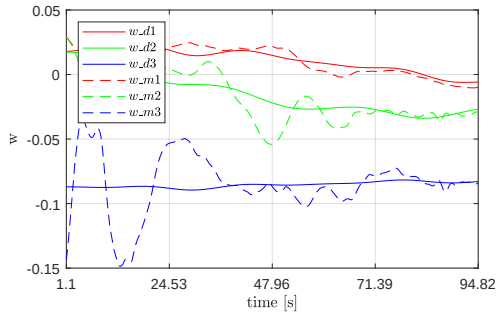
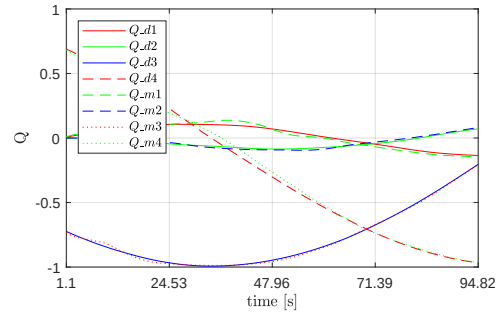


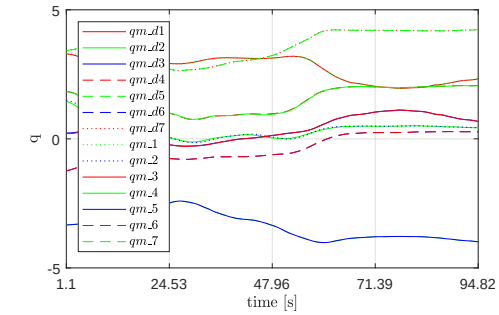
Figure 4. 3D-Visualization for the generated reference trajectory for the robot TCP $r_{\mathrm{TCP,ref}}$ (in local robot coordinate system) for the 100 random scenarios. The start location for all cases is $(-0.99, -0.01, 2.01)$m. The limitations for the TCP position given by $r_{\mathrm{TCP},min}$ and $r_{\mathrm{TCP},max}$ can be seen for some trajectories, that move along the given limits.



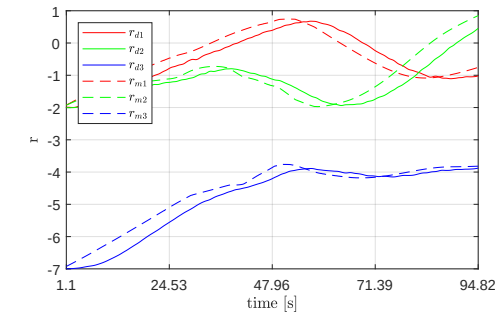
(a) Comparison of the commanded (ω_d) and measured (ω_m) angular velocity (including correction) to the combined controller in rad/s.



(b) Comparison of the commanded (Q_d) and measured (Q_m) Quaternion (including correction) to the combined controller.

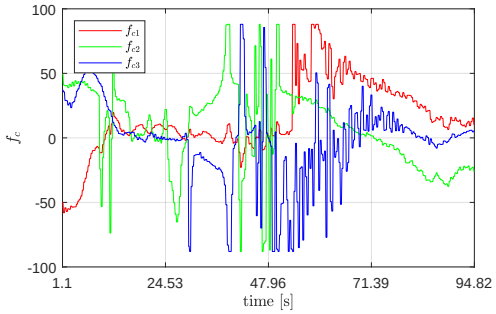


(c) Comparison of the commanded ($q_{m,d}$) and measured (q_m) robot motor position (including correction) to the combined controller in rad.

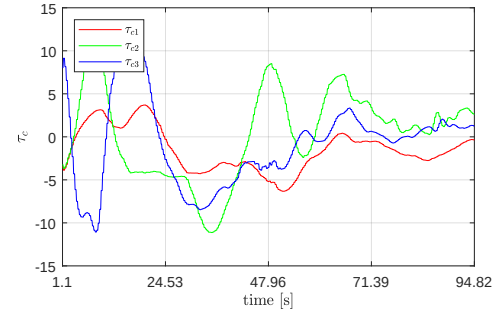


(d) Comparison of the commanded (r_d) and measured (r_m) chaser relative distance to the target in m.

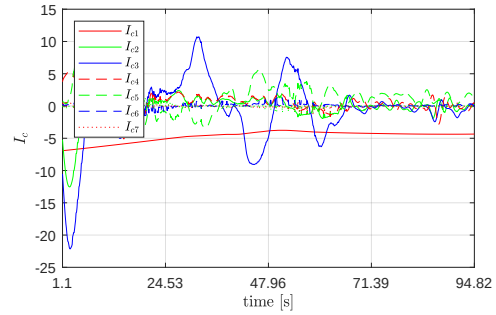
Figure 5. Simulation results for the generated command inputs and measured results for the combined controller for an example simulation case with high starting target angular velocities $\omega_{\text{ta},0} = (-1,-1,-5)^T\,^\circ/s$ and starting distance $r_{\text{ta},0} = (-2,-2,-7)^T\,m$.



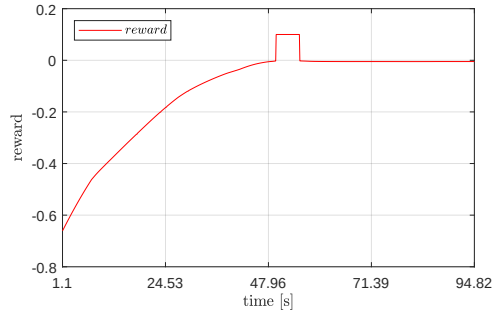
(a) Resulting thrust forces f_c commanded by the combined controller which are input for the force allocation to assign them to individual thrusters.



(b) Resulting thrust forces τ_c commanded by the combined controller which are input for the force allocation.



(c) Commanded motor torques for the 7-axis robot by the combined controller.



(d) PPO reward for the simulation case. The peak in the reward function (48 s) triggers the terminal condition C_{ter} and the classical trajectory planing takes over from the PPO algorithm.

Figure 6. Resulting thrust forces and torques for the chaser satellite and motor currents for the robot arm for the simulation case from Figure 5. Also plotted is the resulting reward function for the PPO algorithm.

DOI 10.3384/ecp218489

3 HYBRID REINFORCEMENT LEARNING ENHANCED TRAJECTORY PLANNING

The basic idea behind the hybrid reinforcement learning enhanced trajectory planning is to combine the advantages of the flexibility of an RL algorithm with classical trajectory planning for the chaser including an optimization-based inverse kinematics for the robot arm and correction terms for the simplified RL model.

Since the computational power of today's satellites is still limited, the algorithm is designed such that a simple RL part with relatively small neural network sizes is sufficient. It is then combined with correction terms and classical trajectory planning to achieve good accuracy and robustness with respect to the initial conditions and spinning behavior of the target satellite, sensor noise, as well as actuator and control limitations.

The RL base algorithm is a Proximal Policy Optimization (PPO) (Schulman et al. 2017) as implemented in DLR's Stable Baseline 3 (SB3) Python software package (Raffin, Hill, et al. 2021). PPO is widely used and has proven its capabilities in many applications, especially in the field of robotics (Arulkumaran, Deisenroth, Miles Brundage, et al. 2017; Kober, Bagnell, and Peters 2013; Han et al. 2023; Arulkumaran, Deisenroth, Michael Brundage, et al. 2017).

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}\hat{A}_t,\right.\right.$$
$$\left.\left.\mathrm{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right] \tag{3}$$

The objective function for the Proximal Policy Optimization (PPO) algorithm can be expressed by Equation 3, were θ represents the parameters of the current policy, $\pi_\theta(a_t|s_t)$ is the probability of taking action a_t given state s_t under the current policy, $\pi_{\theta_{\mathrm{old}}}(a_t|s_t)$ is the probability of taking action a_t given state s_t under the old policy, $\hat{A}_t$ is the estimated advantage at time t and ε is a hyperparameter that controls the extent of clipping (which limits extreme changes). The policies are implemented as neural networks.

CATIA-Systems FMPy Python software package is used to simulate Functional Mockup Units (FMUs) in Python, which are generated by Dymola from Modelica Code for the models described in sec. 2. SB3 allows many models (environments) to be computed in parallel. This feature was extended to use FMPy and underlying Modelica models. For a fast and efficient implementation, Modelica/Dymola's feature of inline integration (Elmqvist, Otter, and Cellier 1995) is used, so that the numerical solver is directly part of the Modelica model inside the FMU. As a result, from the outside (SB3), the model can be used as a model with discrete step size and no additional integrator in SB3 is needed.

The basic idea for the RL part of the hybrid algorithm is to train a network to find a collision free trajectory for both the chaser satellite and the gripper, mounted at the TCP of the 7-axis robot arm, from a varying initial distance to the target, which can spin with different rates as defined in Equation 1 and Equation 2.

To simplify the collision avoidance problem, the target satellite is approximated by bounding boxes for the collision algorithm. For the target three different collision boxes are used. One for the main satellite bus and two for the solar arrays. Figure 1 shows the bounding boxes in transparent red. It is assumed, that the onboard cameras can accurately (with the modeled measurement noise) track these boxes for the target. The chaser itself is approximated by using an additional safety offset for the distance calculation (with respect to the chaser's CoM) according to its largest dimension, in addition, for the TCP a smaller offset is used to account for the size of the gripper.

For both the chaser and the robot's TCP the closest points on these collision boxes are computed (some are visible as small pink spheres in Figure 1) as well as the distance of these critical points to the TCP and chaser including the safety offsets for chaser and TCP.

Since the rotation of the chaser is controlled such that it tries to stay synchronized to the spinning target, only the translational part is of interest for the PPO algorithm (a correction described later will handle occurring deviations).

As observation inputs for the PPO algorithm, a series of normalized vectors are used. All are given as relative vectors with respect to the robot's mounting frame.

- The actual position of the TCP.
- The vector between the TCP and the target grasp point (with safety offset) shown in Figure 1 as small visual coordinate system (cosys) above the grasp point on the adapter ring.
- The vector between the chaser CoM and the chaser goal position close to the grasp point shown in Figure 1 as a local coordinate system above the central solar array.
- A set of $n_{vec} = 6$ vectors $d_{\mathrm{vec},i}$ describes the distance between the computed closest critical points for the three bounding boxes relative to the TCP and chaser center (with safety offsets).

The termination condition C_{ter} is achieved when both the TCP and the chaser CoM are within a given distance limit to the target positions as given in Equation 5, where $k_{\mathrm{safe}} < 1$ is just a safety factor to ensure the TCP is not very close to a bounding box and $k_{\mathrm{col,TCP}}$ is a scaling factor, to ensure the TCP is allowed to be closer to a bounding box than the chaser itself. $d_{TCP,Gr}$ is the distance between the TCP and the grasping end position (with safety offset), $d_{TCP,\max}$ is the maximum allowed deviation, $d_{\mathrm{CH},Gr}$ and $d_{\mathrm{CH,max}}$ are similar terms for the chaser. C_{col} is a criterion

for collision avoidance as defined in Equation 4. The reward r_t^{PPO} for the PPO is computed by Equation 6 if the termination is not reached yet. If the termination criterion is reached it is set to a constant high termination reward $r_t^{\text{PPO}} = R_{PPO,ter}$, which helps the PPO optimization to quickly find and keep the termination condition C_{ter}.

$$C_{\text{col}} = \sum_{i=1}^{n_{vec}} \frac{1}{\max(\varepsilon_d, \|d_{\text{vec},i}\|)} \tag{4}$$

$$\begin{aligned} C_{ter} \implies & d_{\text{TCP},Gr} < d_{\text{TCP,max}} \\ & \wedge\ d_{\text{CH},Gr} < d_{\text{CH,max}} \\ & \wedge\ C_{\text{col}}\, k_{\text{col,TCP}} < C_{\text{max,col}}\, k_{\text{safe}} \end{aligned} \tag{5}$$

$$\begin{aligned} r_t^{\text{PPO}} = \\ -\left(d_{\text{TCP,Gr}} + d_{\text{CH,Gr}} + \min\left(C_{\text{max,col}}, C_{\text{col}} k_{\text{col,TCP}}\right)\right) \end{aligned} \tag{6}$$

Since the optimization of the neural networks inside the PPO algorithm works better with scaled values, the reward and the observation vectors are normalized element-wise between zero and one using a scaling function $\text{f}_s()$ with predefined minimum and maximum values which scales an input vector using a combination of linear and sigmoid functions (near the limits), with adjustable bounds and smoothing.

The PPO neural networks consist of two networks (actor and critic value-function), each with two layers and 128 neurons per layer. The hyper-parameters for the PPO tuning were found using a hyper-parameter optimization implemented in Optuna (Akiba et al. 2019). In addition, generalized State Dependent Exploration (gSDE) (Raffin, Kober, and Stulp 2022) is used instead of action noise exploration.

The resulting action a_t of the PPO algorithm (Equation 7) is also a normalized vector with six elements (with each element between -1 and 1). The first three elements are the translation command for the gripper at the robot's TCP. The last three elements are the translation vector for the chaser satellite for the next time step. Both vectors are given with respect to the robot's mounting frame. A predefined scaling vector k_s is used to account for the faster dynamics of the robot arm compared to the chaser satellite (Equation 8). k_s is chosen such, that the robot and chaser should be able to handle the resulting translation movement in most cases. The actual agility depends on the current pose of the robot arm and the satellite, and is limited by the required thrust to achieve rotational synchronization, so k_s is an approximation. In addition, $a_{t,s,rob}$ is set to zero if the sum over all actions would move the relative TCP location outside the allowed bounds $r_{\text{TCP,min}} = (-1.5, -1.0, 1.0)\text{m}$ and $r_{\text{TCP,max}} = (1.5, 1.0, 2.7)\text{m}$. These bounds ensure that the robot arm is able to reach the location and that no self-collision with the chaser can occur.

$$a_t = \left(a_{t,\text{rob}}, a_{t,\text{ch}}\right)^T \tag{7}$$

$$a_{t,s} = a_t \circ k_s = \left(a_{t,s,\text{rob}}, a_{t,s,\text{ch}}\right)^T \tag{8}$$

For the PPO algorithm, the only measured sensor inputs are the target rotation matrix in each time step relative to the chaser, which is assumed to be measurable with the onboard sensors. Also available for the PPO algorithm is the relative distance vector between chaser and target at the start of the first phase. With this information and the geometric data for the bounding boxes, the PPO algorithm can compute a trajectory (and internally also the required observations and reward). The actual states of the chaser and robot arm are not fed back to the PPO algorithm. This has the advantage that the PPO algorithm works much more stably, since no instability due to feedback can occur; in some sense, it works as a feed-forward control for trajectory planning.

Since the actual states of the chaser and the robot arm can deviate from the internal state of the PPO (mainly because of actuation limitations and sensor noise and disturbance effects), a correction is needed. Thus, using the measured relative orientation $R_{\text{rob,base},m}$ and measured (relative) position of the chaser robot mounting frame $r_{\text{base},0,m}$, a corrected action $a_{t,\text{cor,rob}}$ is computed and transformed in the actual measured robot mounting frame Equation 9.

$$a_{t,\text{cor,rob}} = R_{\text{rob,base},m}\left(a_{t,s,\text{rob}} - r_{\text{base},0,m}\right) \tag{9}$$

For the position of the chaser, no such correction is made, since the TCP can be moved faster and errors in the rotation have only a small effect on the actual CoM position. The chaser directly follows the trajectory given by $a_{t,s,ch}$. Since $a_{t,cor,rob}$ is a Cartesian translation, an inverse kinematics algorithm is used to transform the trajectory in the robot joint space. As desired orientation for the TCP the orientation of the target grip point is used. A discrete implementation using a least squares algorithm, instead of the original constrained least squares (CLS) implementation from (Bellmann 2014), with additional limitations is used for performance reasons.

The PPO trajectory is followed until the condition C_{ter} is true. In this state the gripper is very close to the grasp point and the chaser close at a defined location near the grasp point.

From there classical trajectory panning algorithms are used for the grasping, detumbling and de-orbiting, as described in (M. Reiner 2021; M. Reiner, Fernandez, and Ortega 2017; M. Reiner 2016).

4 SIMULATION EXPERIMENT RESULTS

To test the proposed hybrid algorithm, a series of 100 simulations of the ADR scenario were performed, using the full complex model (instead of the simplified model used for training the PPO algorithm) as described in sec. 2.

Figure 2 shows the trajectories for the chaser satellite generated by $a_{t,s,\text{rob}}$. For the 100 simulations, the start values for the initial target angular velocity and distance to target were randomized within their respective bounds

DOI 10.3384/ecp218489

as defined in Equation 1 and Equation 2. For all 100 simulation the algorithm was successful and collision free.

Figure 3 shows the results for the mean achieved reward which is normalized with the minimal reward achieved (meanReward in the bar plot). The plot also shows the vector length of the initial target angular velocity and distance to target (normalized to the maximal values). The last bar shows the time it took for the termination, normalized to the longest time. The mean time for a termination was $40.99s$. The large visible outlier was a case were the algorithm was able to move both gripper and chaser very close to the target, but not close enough to achieve the terminal condition C_{ter}, only after $404.35s$ the condition C_{ter} was achieved. For such cases the algorithm could be improved in the future with a more adaptive terminal condition C_{ter} or an abort criterion if the phase takes too long.

Figure 4 shows the trajectory for the gripper at the TCP, which result from $a_{t,s,\text{rob}}$ for the 100 simulations.

Figure 5 and Figure 6 show some detailed results for one simulation case which is at the limit for the allowed initial target angular velocity. The trajectory, generated by the hybrid algorithm, is used as input for the combined controller which is used to control the chaser satellite and robot arm. The plots show the approach with the PPO algorithm and the transition to the grasping (using a classical approach). The de-tumbling and de-orbiting is not shown. The combined controller and classical trajectory planning used are described in more detail in the previous publications (M. Reiner 2021; M. Reiner, Fernandez, and Ortega 2017; M. Reiner 2016).

5 CONCLUSION AND OUTLOOK

This work describes a novel hybrid reinforcement learning enhanced trajectory planning for an ADR scenario for combined control of a satellite with a 7-axis robotic arm. A PPO algorithm is combined with a correction algorithm and classical trajectory planning to handle the collision free approach of a chaser satellite to a target and placing the gripper at the robots TCP near the grasping point for use with a combined controller, which commands the satellite and its robotic arm simultaneously.

The algorithm was tested in a complex simulation ADR scenario with 100 randomized initial conditions, and was able to generate collision free trajectories for all cases.

The PPO part of the algorithm is trained using a simplified model of the scenario, which allows fast training times and leads to small neural networks, which could be run on weaker space computing hardware. It acts as a feed-forward component to a correction algorithm, which helps to reduce instability caused by uncertain measurement feed-back and model mismatch.

In the future the algorithm could be extended to use a more complex model for the training if more computing power would be available and larger neural networks on board of the satellite.

The space industry is very restrictive in regards to use neural networks as part of the control but a hybrid algorithm, as presented here, which can be more easily supervised and extended with a fall-back solution could help the acceptance.

As potential directions for further work, systematic comparisons with other RL algorithms and classical control baselines could be explored, along with more detailed statistical analyses and ablation studies. It may also be valuable to investigate advanced domain adaptation and sim-to-real transfer techniques, as well as formal safety and stability guarantees, to further broaden the applicability of the proposed approach.

References

Acquatella B., Paul and Matthias J. Reiner (2014). "Modelica stage separation dynamics modeling for End-to-End launch vehicle trajectory simulations". In: *Proceedings of the 10th International Modelica Conference - Lund 2014*. Ed. by Hubertus Tummescheit and Karl-Erik Arzen. Vol. 96. Linköping Electronic Conference Proceedings. LiU Electronic Press, pp. 589–598. URL: https://elib.dlr.de/92218/.

Akiba, Takuya et al. (2019). "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Ambrosio, Matteo et al. (2024). "Redundant Space Manipulator Autonomous Guidance for In-Orbit Servicing via Deep Reinforcement Learning". In: *Aerospace* 11.5, p. 341. DOI: 10.3390/aerospace11050341.

Arulkumaran, Kai, Marc Peter Deisenroth, Michael Brundage, et al. (2017). "Deep reinforcement learning: A brief survey". In: *IEEE Signal Processing Magazine* 34.6, pp. 26–38. DOI: 10.1109/MSP.2017.2743240. URL: https://ieeexplore.ieee.org/document/8099596.

Arulkumaran, Kai, Marc Peter Deisenroth, Miles Brundage, et al. (2017). "A brief survey of deep reinforcement learning". In: *IEEE signal processing magazine* 34.6, pp. 26–38.

Bellmann, Tobias (2014). "Optimierungsbasierte Bahnplanung für interaktive robotische Bewegungssimulatoren". PhD thesis. Universität der Bundeswehr München. URL: https://elib.dlr.de/93031/.

Brandonisio, Andrea, Michele Lavagna, and Davide Guzzetti (2021). "Reinforcement Learning for Uncooperative Space Objects Smart Imaging Path-Planning". In: *The Journal of the Astronautical Sciences* 68, pp. 1145–1169. DOI: 10.1007/s40295-021-00288-7.

Briese, Lale Evrim, Andreas Klöckner, and Matthias Reiner (2017-05). "The DLR Environment Library for Multi-Disciplinary Aerospace Applications". In: *12th International Modelica Conference*. URL: https://elib.dlr.de/112971/.

Ding, Zhaolong et al. (2023). "Fast-Convergence Reinforcement Learning for Routing in LEO Satellite Networks". In: *Sensors* 23.11, p. 5180. DOI: 10.3390/s23115180.

Elmqvist, H., M. Otter, and F. Cellier (1995). "Inline Integration: A New Mixed Symbolic/ Numeric Approach for Solving Differential-Algebraic Equation Systems". In: *Proceedings of ESM, SCS European Simulation MultiConference*.

Han, Dong et al. (2023). "A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation". In: *Sensors* 23.7, p. 3762. DOI: 10.3390/s23073762. URL: https://www.mdpi.com/1424-8220/23/7/3762.

Kober, Jens, J Andrew Bagnell, and Jan Peters (2013). "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11, pp. 1238–1274.

Liou, J-C and Nicholas L Johnson (2006). "Understanding the full impact of space debris on the Earth environment". In: *Science* 311.5759, pp. 340–341.

Patnala, Susmitha and Adam Abdin (2024). "On-orbit Servicing for Spacecraft Collision Avoidance With Autonomous Decision Making". In: *Papers With Code*. URL: https://paperswithcode.com/paper/on-orbit-servicing-for-spacecraft-collision-avoidance-with-autonomous-decision-making.

Raffin, Antonin, Ashley Hill, et al. (2021). "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268, pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

Raffin, Antonin, Jens Kober, and Freek Stulp (2022-08–11 Nov). "Smooth Exploration for Robotic Reinforcement Learning". In: *Proceedings of the 5th Conference on Robot Learning*. Ed. by Aleksandra Faust, David Hsu, and Gerhard Neumann. Vol. 164. Proceedings of Machine Learning Research. PMLR, pp. 1634–1644. URL: https://proceedings.mlr.press/v164/raffin22a.html.

Reiner, Matthias (2016). "GNC Simulation Tool for active Debris Removal with a Robot Arm". In: *Proceedings for the 6th International Conference on Astrodynamics Tools and Techniques (ICATT)*. URL: https://elib.dlr.de/104850/.

Reiner, Matthias (2021). "Real-time combined control for active debris removal for a satellite with a robot arm implemented on a SBC connected to a detailed multi-physics and VR simulation". In: *GNC 2021 ABSTRACT BOOK*. ESA. URL: https://elib.dlr.de/144621/.

Reiner, Matthias and Johann Bals (2014). "Nonlinear inverse models for the control of satellites with flexible structures". In: *Proceedings of the 10th International Modelica Conference - Lund, Sweden - Mar 10-12, 2014*. Ed. by Hubertus Tummescheit and Karl-Erik Arzen. Linköping Electronic Conference Proceedings. LiU Electronic Press, pp. 577–587. URL: https://elib.dlr.de/92164/.

Reiner, Matthias, Jesus G. Fernandez, and Guillermo Ortega (2017). "Combined control for active debris removal using a satellite equipped with a robot arm". In: *Proceedings for the GNC 2017, 10th International ESA Conferece on Guidance, Navigation and Control Systems*. URL: https://elib.dlr.de/115866/.

Schulman, John et al. (2017). "Proximal Policy Optimization Algorithms." In: *CoRR* abs/1707.06347. URL: http://dblp.uni-trier.de/db/journals/corr/corr1707.html#SchulmanWDRK17.

Weiss, Benjamin and Bruno M Sarli (2014). "Autonomous on-orbit servicing of satellites". In: *IEEE Access* 2, pp. 614–626.

 DOI 10.3384/ecp218489