AAS 21-762

SENSITIVITY ANALYSIS OF ADAPTIVE GUIDANCE VIA DEEP REINFORCEMENT LEARNING FOR UNCOOPERATIVE SPACE OBJECTS IMAGING

Andrea Brandonisio[*] and Michèle Lavagna[†]

In this work a Proximal Policy Optimization (PPO) algorithm is explored to setup an agent that, via imaging, plans sub-optimal paths to fly-around an uncooperative and unknown orbiting object with shape reconstruction purposes. The proximity dynamics between the spacecraft and the target object of the chaser agent is linearized taking into account the target orbital eccentricity; rotation of the target object is included only. The target geometry is rendered as a polyhedron shaped model with a triangular mesh. The PPO algorithms is deeply analysed to improve the stability and reduce the sensitivity of the algorithm with respect to the particular conditions in which the problem is solved. Both policy and state-value functions are approximated using Artificial Neural Networks (ANN) and trained according to RL principles. To improve the stability and robustness of the agent in different scenario conditions, a formulation exploiting recurrent layers is proposed and studied. Random initial environment conditions are adopted to train the agent to react to largely different operational scenarios. A large database of training tests has been collected, exploiting the network configurations already designed in our previous work. The proposed method and its results are critically presented to contribute to the RL techniques applicability in the field of proximity guidance synthesis for image based chasers fly-around uncooperative objects, possibly expanding the baseline research for future works on autonomous guidance in space engineering.

INTRODUCTION

In the last decades, leading space agencies are increasingly investing in the gradual automation of the space missions and the rapid developments of Artificial Intelligence is everyday more strongly influencing the aerospace researches. Autonomous flight operations may be crucial for large-scale sustainable on-orbit servicing missions, leading to several benefits, including flexibility, reactivity and robustness. Up to now, on-orbit servicing (OOS) activities are mostly performed and studied by government agencies, e.g. NASA or DARPA. The demand of orbital servicing has increased in the recent years, pushing more companies towards the study of complete autonomous satellite services.[1]

According to NASA and American Institute of Aeronautics and Astronautics (AIAA), the term *on-orbit servicing* is referred to all on-orbit activities that a space vehicle intentionally conducts on another resident space object (RSO). Such definition includes several activities: non-contact support, relocation and orbit modification, maintenance, refueling, commodities replenishment, upgrade, repair, assembly, and debris mitigation. The vehicle able to perform one or more of these

[*]PhD Student, Aerospace Department, Politecnico di Milano, andrea.brandonisio@polimi.it
[†]Associate Professor, Aerospace Department, Politecnico di Milano, michelle.lavagna@polimi.it

activities is called *servicer*, while the vehicle that receives the service is called *client*. The client is called *cooperative client* if it transfers useful information for acquisition, tracking, rendezvous or other servicing activities to the servicer via direct communication link or ground link; otherwise, it is defined *uncooperative* or *non-cooperative client*. In most of the cases, a strong distinction is not possible.

Within the spectrum of proximity operations, this work focuses on autonomous path-planning of a chaser satellite devoted to the reconstruction of geometry and inertia properties of an uncooperative and unknown target object by flying around. The objective is to optimize the acquisition of all the information necessary for an effective target object mapping.

The autonomous exploration of an unknown environment has been studied especially in the field of robotics. The most popular formulation is called Simultaneous Localization and Mapping (SLAM) problem, that has been firstly developed in 1987. In SLAM problems an agent continuously estimates the map of an unknown environment while simultaneously localizing itself in it.[2] After the full characterization of the SLAM problem, in the late '90s, the task of exploration planning has been added generating the *active* SLAM problem. In active SLAM mapping, localization and planning are strictly coupled.[3] Active SLAM applications aim to automate data collection that is needed to create the highest quality map with the least time and cost possible. SLAM may be formulated as a Partially Observable Markov Decision Process (POMDP) and therefore solved with Deep Reinforcement Learning (DRL) methods. Pose (position and attitude), motion and inertia matrix estimation is the first step to develop a complete autonomous system for proximity operations with uncooperative clients. Several works have been developed to understand problems such as the reconstruction of geometric properties, inertia or dynamics of a space object or a small body, including asteroids and comets. This work starts from the foundations we have already introduced exploiting a reinforcement learning agent that maps a shape reconstruction task directly to a thrust value that controls the spacecraft motion.[4]

Reinforcement Learning (RL) and Deep Reinforcement Learning techniques have been widely used to design agent's policies in planning problems.[5] In the robotic field, the first authors to propose RL to optimize a trajectory for map exploration have been Kollar *et al.*[6] Tools such as RL and DRL are very useful especially when coupled to the generalizing capabilities of neural networks: they allow to deal with complex planning problems that present wide and continuous state spaces.

Previous works highlighted the potential in adopting reinforcement learning techniques for guidance design on astrodynamics: Brandonisio *et al.* studied the autonomous mapping of uncooperative space objects,[4] Gaudet *et al.* and Linares *et al.* explored the autonomous landing and asteroid close-proximity operations,[7,8] Pesce *et al.*, Piccinin *et al.* and Chan *et al.* analysed the autonomous mapping of asteroids,[9–11] Indaco *et. al* studied autonomous path-planning for asteroid gravimetry.[12]

Starting from our previous work, a Proximal Policy Optimization (PPO) algorithm is here explored to setup an agent that, via imaging, plans sub-optimal paths to fly-around an uncooperative orbiting objects with shape reconstruction purposes. The proximity dynamics between the spacecraft and the target object is linearized taking into account the target object orbital eccentricity; rotation of the target object is included only to the equations of motion. The target geometry is rendered as a polyhedron shaped model with a triangular mesh. A batch formulation of PPO is proposed and explored. Both policy and state-value functions are approximated using deep fully-connected Artificial Neural Networks (ANN) and trained according to RL principles. Random initial environment conditions are adopted to train the agent to react to largely different operational scenarios. To achieve satisfactory results, the agent has to learn how to survive in the environment without prematurely escaping or colliding with the target and to limit the cost at the most. In addition, the agent - vision based - shall monitor the Sun orientation and the target relative orientation, which are essential to achieve a good quality of the map. The map is computed as the total number of successful observations for each face of the polyhedron mesh with respect to a fixed value that the agent should achieve. A large database of training tests has been collected, exploiting the network configurations already designed.

Moreover, in order to improve the stability and robustness of the agent in different scenario conditions, a formulation of PPO exploiting recurrent neural networks (RNN) is proposed and studied. The recurrent layers capabilities to store past states information may strongly affect the agent safe trajectories planning and faster getting to the mission goals. In addition, training an RNN is beneficial to refine the agent's environmental conditions sensitivity, which works in favour of the agent's robustness, regardless the specific operational environment. Therefore, both methodologies - fully-connected and recurrent - are analysed to improve the stability and reduce the sensitivity of the algorithms with respect to the particular conditions in which the problem is solved.

The proposed method and its results are critically presented to contribute to the RL techniques applicability in the field of proximity guidance synthesis for image based chasers fly-around uncooperative objects, possibly expanding the baseline research for future works on this innovative field of the space engineering.

PROBLEM SCENARIO

The dynamics models characterizing the relative motion between the spacecraft and a target object are described in this section. Subsequently, the visibility relation between the two, here named *visibility model*, is defined and described.

Linearized Eccentric Model

The linearized relative dynamic model is based on the assumption that the relative distance between the spacecraft and the target object is much lower than the distance between the target and the main attractor. Common use is to express the relative equations of motion in the Local Vertical Local Horizontal (LVLH) reference frame centred in the target object center of mass.[13] The resulting system is shown in Eq. (1).

$$\ddot{x} = \left(\frac{2\mu}{R^3} + \frac{h^2}{R^4}\right) x - \frac{2(\mathbf{V}\cdot\mathbf{R})h}{R^4} y + \frac{2h}{R^2}\dot{y} \tag{1a}$$

$$\ddot{y} = \left(\frac{h^2}{R^4} - \frac{\mu}{R^3}\right) y + \frac{2(\mathbf{V}\cdot\mathbf{R})h}{R^4} x - \frac{2h}{R^2}\dot{x} \tag{1b}$$

$$\ddot{z} = -\frac{\mu}{R^3} z \tag{1c}$$

R, **V**, R and h are the position vector, the velocity vector, the position magnitude distance and the angular momentum of the target object in its Earth-centered inertial reference frame respectively; x, y and z are the Cartesian components of the relative position of the spacecraft with respect to the target object in LVLH frame.

Rotational Model

To maintain the same reference frame of the translational motion, we express the rotational dynamics again in the LVLH frame centered in the target object center of mass, through a transformation of the Euler's equations of motion written in body-reference frame.[14] Applying the small angle approximation and the transpose theorem to the Euler's equations, the differential equations for the rotational dynamics of the target in LVLH frame can be written as in Eq. (2). Afterwards, the products between infinitesimal terms can be neglected, leading to the final model derived in Eq. (3).

$$\begin{cases} I_x(\ddot{\alpha}_x - \dot{\alpha}_y n) + (I_z - I_y)(\dot{\alpha}_z + n)(\dot{\alpha}_y + \alpha_x n) = 0 \\ I_y(\ddot{\alpha}_y + \dot{\alpha}_x n) + (I_x - I_z)(\dot{\alpha}_z + n)(\dot{\alpha}_x - \alpha_y n) = 0 \\ I_y\ddot{\alpha}_z + (I_y - I_x)(\dot{\alpha}_x - \alpha_y n)(\dot{\alpha}_y + \alpha_x n) = 0 \end{cases} \tag{2}$$

$$\begin{cases} I_x\ddot{\alpha}_x + n(I_z - I_y - I_x)\dot{\alpha}_y + n^2(I_z - I_y)\alpha_x = 0 \\ I_y\ddot{\alpha}_y + n(I_x + I_y - I_z)\dot{\alpha}_x + n^2(I_z - I_x)\alpha_x = 0 \\ I_z\ddot{\alpha}_z = 0 \end{cases} \tag{3}$$

I_x, I_y and I_z are the principal moments of inertia of the target object, α_x, α_y and α_z are the rotational angles in LVLH frame, n is the target mean motion. Similarly to other studies, we assume to neglect the spacecraft attitude dynamics and control considering the spacecraft camera always pointed towards the target object center of mass.[4,10,11] Although this is a strong assumption, it helps to formulate a simpler problem, that at this level of development is necessary to asses the feasibility of deep reinforcement learning applications to spacecraft guidance.

Visibility Model

As in our previous work, a *visibility model* generates the portion of the target object that the spacecraft can observe at each time step.[4] The model of the target object is rendered with a set of faces; for this reason the *visibility* model locates the faces that are in direct line of sight (LoS) with the spacecraft camera and under favorable light conditions. The quality of the image is determined in function of the the incidence angles between the face normal direction and the Sun and camera directions. These variables are important for the formulation of the reward model.

Image Processing. To correctly formulate the reward model the image processing method selected is the *Stereophotoclinometry* (SPC). SPC descents from shape-from-shading approaches and links photometry to stereoscopy and is a state of the art method that guarantees high resolution models given a sufficient number of observations.[15] The requirements to obtain high-quality SPC models derive from the studies done by Gaskell *et al.*[16] Such requirements inform the definition of the reward model for our RL agent:

- A minimum of three images per face (typically > 40).
- Emission angle, defined as the angle between the camera direction and the normal direction to the face, should be maintained around $\sim 45°$ (acceptable 35°-48°, limit 5°-60°).
- Incidence angle, defined as the angle between the Sun direction and the normal direction to the face, should be maintained around $\sim 45°$ (acceptable 30°-50°, limit 0°-70°).

- Variation in illumination conditions (acceptable 40°-90°, limit 10°-120°), that is the variation of the Sun incidence angle during the observation period.

AUTONOMOUS NAVIGATION AND REINFORCEMENT LEARNING FRAMEWORK

The geometry reconstruction of a target body task, that requires the spacecraft to plan trajectories in a uncertain environment, can be formulated as an Simultaneous Localization and Mapping (SLAM) problem. SLAM problem is a mathematical formulation of a map construction problem of an unknown object or environment while simultaneously tracking the agent location[17, 18] in it. The SLAM problem may also include trajectory planning, in this case it is known as active SLAM.[19] One of the possible methodology to solve active SLAM problems is the Partially Observable Markov Decision Process (POMDP).

Partially Observable Markov Decision Process

POMDP derives from Markov Decision Process (MDP), when part of the information characterizing the agent state is unknown (that is why *partially observable*).[20] In this case, the decision maker is only aware of partial information about the environment. A POMDP problem can be described as a six-tuple, $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \Omega, \mathcal{O})$, where: $\mathcal{S}$ is the space of all possible states in the environment, $\mathcal{A}$ is the space of all possible actions that can be taken in all the states of the environment, $\mathcal{R}_{\mathcal{A}}(s, s')$ is the reward space, function of all the pairs between two following states, $\mathcal{T}_{\mathcal{A}}(s, s')$ is the transition probability related to all the state-action pairs, Ω is the space of possible observations and $\mathcal{O}(o' \mid a, s')$ is the probability of making a particular observation, taking an action that leads to a particular new state.

To obtain an exact solution to a POMDP problem, it is necessary to select the optimal action for each possible observation over the world states. The optimal action maximizes the expected reward of the agent over a possibly infinite horizon. Exact solutions exist only for a very thin class of POMDP problems. This complexity leads to an approximation of the general problem to make it more tractable: indeed, a POMDP problem can be reduced to a MDP problem exploiting a new formulation called *belief-space* MDP. This new formulation is characterized by a four-tuple $(\mathcal{B}, \mathcal{A}, \mathcal{R}_{\mathcal{A}}, \mathcal{T}_{\mathcal{A}})$.

- $\mathcal{B}$ is the belief space, where the *belief* is defined as $b = p(s \mid h)$, that is the probability of being in a state s after the history h of all the previous states.
- $\mathcal{A}$ is the original action space.
- $\mathcal{R}_{\mathcal{A}}(b, b')$ is the reward space in the new formulation.
- $\mathcal{T}_{\mathcal{A}}(b, b')$ is the belief transition function, that is the probability of reaching a new belief b' from a belief b taking an action a.

The goal is always the maximization of the long-term reward. Two different optimal policies can be defined depending on whether the problem episode is finite or infinite. The problem formulation treated in this work falls in the category of the *infinite horizon* problems, whose optimal policy is defined exploiting a discount factor $\gamma \in [0, 1]$, as in Eq. (4).

$$\pi_\star = \underset{\pi}{\operatorname{argmax}} \, \mathbb{E}_\pi \Big[\sum_{k=0}^{\infty} \gamma^k \mathcal{R}_{\mathcal{A}}(b_k, b_{k+1}) \Big] \tag{4}$$

Proximal Policy Optimization

In this work, we implemented a formulation of the proximal policy optimization (PPO) algorithm.[21] The PPO algorithm is a policy gradient algorithm representing the state of the art in terms of performance of many of the typical DRL benchmark problems. It is a derivation of the A2C method and it is developed from the Thrust Region Policy Optimization (TRPO) method.[22] TRPO formulates the optimization problem in a way that the gradient step is restricted by some constraints. Starting from that, the PPO method approximates TRPO process by using a clipped objective function for the policy optimization step. The objective function used in PPO algorithm is expressed defining the probability ratio $p_k(\theta)$ as in Eq. (5).

$$p_k(\theta) = \frac{\pi_{\theta,k}(u_k|x_k)}{\pi_{\theta,k-1}(u_k|x_k)} \tag{5}$$

Exploiting the probability ratio the PPO clipped objective function is defined in Eq. (6). The introduction of the clipping term help the convergence by ensuring that the policy does not change drastically when updated.

$$L_{\text{CLIP}}(\theta) = \mathbb{E}_{p(\tau)}[\min[p_k(\theta), \text{clip}(p_k(\theta), 1-\epsilon, 1+\epsilon)]A_w^\pi(u_k, x_k)] \tag{6}$$

Moreover, common practice in PPO algorithms is the addition of an entropy regularization term to the clipping objective function to ensure a sufficient exploration level during training.[21,23] In this way Eq. (6) becomes:

$$L_p = L_{\text{CLIP}}(\theta) + c_2 S[\pi_\theta](x_k) \tag{7}$$

The term $A_w^\pi(u_k, x_k)$ represents the advantage function, defined as the difference between the discounted reward and the state value function baseline, as defined in Eq. (8). It gives information about how much better an action is with respect to the average action that can be chosen by the agent; its value also depends on a discount factor $\gamma \in [0, 1)$.

$$A_w^\pi(x_k, u_k) = \Big[\sum_{j=k}^{T} \gamma^{j-k} r(x_j, u_j)\Big] - V_w^\pi(x_k) \tag{8}$$

In Eq. (8), the term $V_w^\pi(x_k)$ defines the value function of the algorithm, that is computed by the critic network at each step of the simulation. The optimization of the value function is learned by the the cost function defined in Eq. (9).

$$L_w = \sum_{i=1}^{N} \Big(V_w^\pi(x_k^i) - \Big[\sum_{j=k}^{T} \gamma^{j-k} r(x_j^i, u_j^i)\Big]\Big)^2 \tag{9}$$

Here N represents the number of trajectories used to update the network: in practice, the policy gradient algorithm back-propagation updates the policy using a batch of state observations collected by interaction with the scenario environment. Each trajectory is considered as a single episode, with $x_k, u_k, r(x_k, u_k)$ representing a sample of observation, action and reward of the trajectory N at the k-th time step. In the Appendix A the pseudo-code of the used PPO algorithm is presented.

PROBLEM ARCHITECTURE

In Figure 1, the on-board planning architecture of our work is displayed. The core of the architecture is the *Autonomous Decision Process*; it is preceded by a pre-processing phase, based on the relative pose, the camera characteristics and the external environment conditions, i.e. the Sun direction. The pre-processing output feeds the autonomous decision block, characterized by the neural network trained by PPO to optimize the control policy aimed to maximize the mapping of the target.

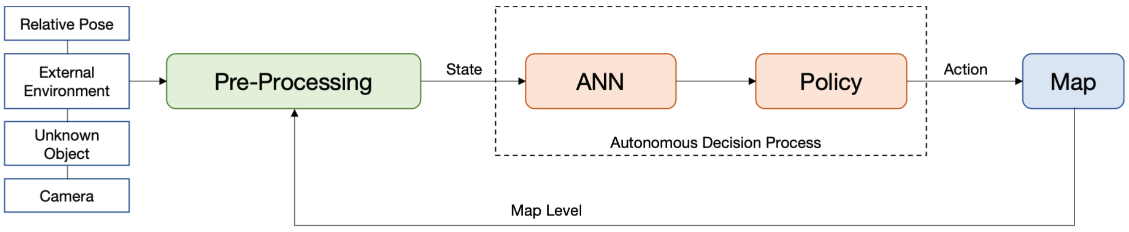


Figure 1. On-Board Planning Architecture Scheme

DRL Formulation

In our work, the environment is characterized by three main components: the spacecraft, the unknown and uncooperative target object and the Sun. The Sun relative dynamics depends only on the initial epoch; instead, the spacecraft-object relative dynamics is directly influenced by the actions taken by the agent. The spacecraft and the target object orbit around the Earth in orbits that may be eccentric. Two main assumptions are introduced: the first one is that the spacecraft camera constantly points toward the target center of mass and the second neglects Earth influence on the illumination condition, therefore eclipses and reflections are ignored. All the environment information are employed to create the state space, action space and reward models needed to define the autonomous decision process algorithm.

Reward Model. The most significant part of a DRL architecture is the definition of the reward model. RL agents typically learn a policy with the goal of maximizing the cumulative reward. Multiple objectives may be present in the reward model. In general terms, our goal is defining an agent that achieves an high quality map of the target object together with a fast and safe process. Quality of the map depends on the adopted mapping technique. In the present work, reward is defined to ease SPC, in terms of Sun and camera exposition. In addition, the agent has to perform mapping operations in the shortest time possible, avoiding regions of the space that are considered dangerous and may end the episode.

Our reward defines the total reward as a function of different scores that combined together try to solve and optimize one or more specific tasks.

- **Map level score.** The faces that have both good Sun and camera exposition are the ones that generate an improvement in the level of the map. The maximum level defined for the map consists in having each faces photographed $N_{accuracy}$ times. Therefore, at each time step the map level, $Ml_{\%}$, can be computed considering how many good photos of each face have been taken until that moment. The corresponding reward score is defined in Eq. (10) and at each time step k rewards the agent for increasing the map level over the current value, $Ml_{\%,k-1}$.

$$R_m = \begin{cases} 1 & \text{if } Ml_{\%,k} > Ml_{\%,k-1} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The improvement in the map depends on the Sun and camera incidence angles:

- **Sun incidence score.** The Sun incidence angle η is the angle between the Sun direction relative to the target object and the normal to the face considered. The Sun incidence angle should be between 0° - 70°, to avoid shadows or excessive brightness. Values outside that interval may correspond to conditions that degrade the quality of the image. If the angle exceeds this range, the photo can not be considered enough good to make a real improvement of the map.
- **Camera incidence score.** The camera incidence angle ε is intended as the angle between the normal to the face and the camera direction. This angle should be maintained between 5° - 60°. Also in this case, if the angle exceeds this range, the photo can not be considered enough good to make a real improvement of the map.

• **Position score.** Negative scores are given when the spacecraft escapes from the region defined by a minimum and maximum distance, D_{min} and D_{max}.

$$R_d = \begin{cases} -100 & \text{if } d \leq D_{min} \text{ or } d \geq D_{max} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

• **Time of flight score.** This score punishes the agent for exceeding a defined time window, identified by bounding the time at T_{bound}.

$$R_t = \begin{cases} 0 & \text{if } \Delta t < T_{bound} \\ -1 & \text{otherwise} \end{cases} \tag{12}$$

• **Thrust score.** This score considers the number of times that thrusters are fired, n_f. Two thresholds are assigned: the first is a medium level threshold, l_{mid}, beyond which is still possible to fire the thrusters; the second threshold, l_{max}, defines the maximum number of firings, beyond which the thrusters can not be used again. The score is defined in Eq. (13).

$$R_f = \begin{cases} -0.1 & \text{if } l_{mid} \leq n_f < l_{max} \\ -10 & \text{if } n_f \geq l_{max} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Action Model. The action space is the space among which the agent must choice the action to take when required. The *control interval*, Δt, is the time interval between two state transitions. The agent interacts with the environment directly controlling the spacecraft thrusters. The action space here modelled assumes that the spacecraft can thrust in each of the six reference cartesian frame directions, namely $x, -x, y, -y, z, -z$. The option of null action is also available to the agent. This formulation renders a direct and continuous control on the trajectory, consistent with an active SLAM problem. The action space is defined in Eq. (14).

$$\mathcal{A} = \left[T_{x+}, T_{x-}, T_{y+}, T_{y-}, T_{z+}, T_{z-}, 0\right] \tag{14}$$

The control action values are fixed in terms of acceleration, $\bar{a} = 0.001\text{m/s}^2$; therefore, the actions directly affect the translational equations of motion of the spacecraft defined in Eq. (1). For example, if the agent selects the following action, $a_k = [0, 0, 1, 0, 0, 0, 0]$, the relative translational equations of motion will add a positive acceleration to the second equation of the system.

State Model. The state space is designed to synthesize essential information needed by the agent to decide what is the best action to take. The state space is shown in Eq. (15) and comprises $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ which are the relative position and velocity between the spacecraft and the target object, and $\boldsymbol{\alpha}$ and $\dot{\boldsymbol{\alpha}}$ that are the relative angular position and velocity.

$$\mathcal{S} = \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \\ \boldsymbol{\alpha} \\ \dot{\boldsymbol{\alpha}} \end{bmatrix} \tag{15}$$

PPO Implementation. As usual for all the reinforcement learning algorithms, a set of hyperparameters must be defined to correctly assure the convergence of the method. In our PPO implementation, both the policy and the value function (actor and critic network) are learned concurrently. In order to select the action at each time step, the policy network exploits the softmax activation function to generate the elements of the actor vector. The output of the softmax activation function is a multi-categorical distribution, among which the policy samples the action to take during the optimization process. Concerning the parameters related to the loss functions, the reward discount factor γ and the terminal reward discount factor λ are set to 0.99 and 0.94 respectively; the selected clipping parameter ϵ is 0.2. The policy and value networks are implemented in two different cases: a fully-connected ANN architecture and a RNN architecture. The first case uses three linear layers with tanh and Leaky-ReLU as activation functions for the policy and value functions. The architecture is described in Table 1, where dim_obs is the observation state dimension and act_dim is the action space dimension. In order to improve the convergence and avoid saturation problem the tanh-layers are initialized as semi-orthogonal matrices.[24]

Table 1. Policy and Value Networks Architecture: Linear Case

	Policy Network		Value Network	
Layer	Elements	Activation	Elements	Activation
1^{st} Hidden Layer	10*dim_obs	tanh	10*dim_obs	tanh
2^{nd} Hidden Layer	$\sqrt{n_{h1} * n_{n3}}$	tanh	$\sqrt{n_{h1} * n_{n3}}$	tanh
3^{rd} Hidden Layer	10*dim_act	Leaky-ReLU	10*dim_act	Leaky-ReLU
output	dim_act	softmax	dim_act	linear

The RNN architecture case, instead, exploits a sequence of one Long Short-Term Memory (LSTM) recurrent layer and two drop-out linear layers.[25] This architecture is shown in Table 2. It is worth to underline that the activation functions of the policy and value output layers are the same for both the cases.

Table 2. Policy and Value Networks Architecture: Recurrent Case

	Policy Network		Value Network	
Layer	Elements	Activation	Elements	Activation
LSTM Layer	24	-	24	-
1^{st} Hidden Layer	64	ReLU	64	ReLU
2^{nd} Hidden Layer	32	ReLU	32	ReLU
output	dim_act	softmax	dim_act	linear

The policy and value functions are periodically updated during optimization after accumulating trajectories for 10 episodes. Afterwords, the two networks are optimized using the simulated results divided in batch of dimension 32. An episode terminates when the target object map is completely acquired or if the spacecraft escapes the region defined by the minimum and maximum distance from the target. Another possible termination is related to overcoming the mission time, even if it is very unlikely to occur.

RESULTS

As shown in Fig. 1 in this work we developed a problem that depends on the chaser-target relative pose, the external environment, the unknown object geometry and the camera model. In order to bound the problem some characteristics have been maintained constant during the overall training procedure. In particular the camera field of view (FOV) is fixed as $10°$, that can be considered as a common FOV for space optical cameras; the integration time is fixed at 30s and the accuracy level for the map is fixed at 25 correct photos per face. Instead, considering the chaser and target initial conditions, two levels of randomness have been considered:

- The first case considers a random Sun initial phase, a random target initial conditions, in terms of orbital true anomaly and rotational dynamics (angle position and velocity). Here, the chaser-target initial relative position is considered fixed. For simplicity we will name this case: ***random case A***.
- The second case has a random Sun initial phase, a random target initial conditions, in terms of orbital true anomaly and rotational dynamics (angle position and velocity) and the chaser-target initial relative position. Their relative position is however constrained to have both the x, y and z coordinates positive. This assumption comes from two reasons in particular: firstly the will of containing the complexity of the problem in order not to saturate the neural network learning capabilities and also simulate a possible real scenario, in which the initial condition is constrained in a specific space without knowing a priori the correct engagement position. As before we will name this case as ***random case B***.

To better understand the DRL agent behaviour, the neural network was trained with two different reward models. In the first one, there was no care about the thrust level of each simulated episode, hence the total reward is given by:

$$R_{k,1} = R_m + R_d + R_t \tag{16}$$

Where R_m, R_d and R_t are the scores defined before. Then, we decided to make the mission goal more complex and we introduced the thrust level related objective. In this case the total reward is given by:

$$R_{k,2} = R_m + R_d + R_t + R_f \tag{17}$$

Random Case A. In Figure 2, the results obtained with random target initial condition, random Sun phase and fixed relative position are shown. Here the $R_{k,1}$ reward model is exploited to make the agent learn to reach the best mapping quality possible. In the plot the average map level trends of the Linear and Recurrent policy architecture are compared in a simulation of 15000 episodes length.

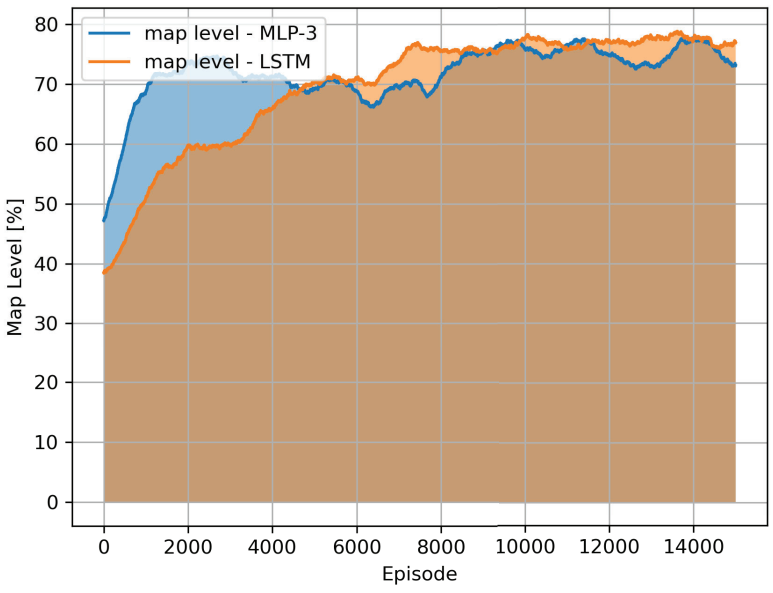


Figure 2. Random Case A. Map Level trend for reward model $R_{k,1}$ with position, map and time scores. Comparison between the two Policy Architectures: linear/MLP-3 and recurrent/LSTM.

Analysing the results some remarks can be derived:

- The linear policy, MLP-3, seems to learn and converge faster then the recurrent policy, LSTM. Nevertheless, the MLP-3 curve presents more oscillations and an overall lower stability with respect to the LSTM curve policy behaviour.
- Concerning the final result of the simulation, the recurrent policy converges to a slightly higher map level in the same training length of the linear policy. In average the map level reached by the two policies is around 70%-80%.
- The fact that the learning curve of the LSTM policy grows gradually shows that having part of the network composed by recurrent layers makes the learning process slower, as expected,

but safer and more stable. Indeed, the potential robustness of a recurrent network was one of the main reason that drove this kind of analysis.

In Figure 3, the same comparative analysis shown before was performed. In this case the reward model, $R_{k,2}$, includes also the thrust level minimization objective. As before, the same general characteristics are present also in this kind of simulation. MLP-3 is faster in learning but unstable in keeping on growing with respect to LSTM. The overall results is around 60%-70%. Not surprisingly, the percentage is slightly less than the one obtained with the $R_{k,1}$ reward model; indeed, the fact that the reward adds a new task to the agent determines a shift in the learning process priorities and therefore an automatic reduction of the map level if considering the same amount of episodes.

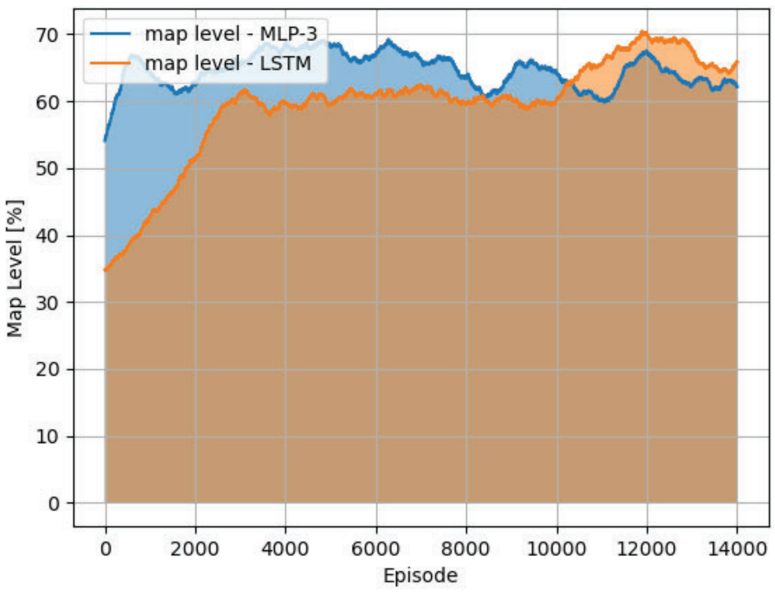
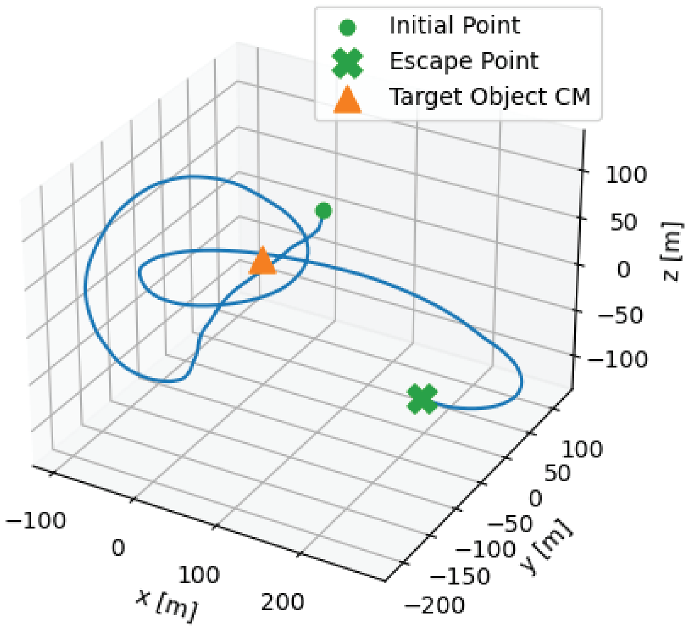


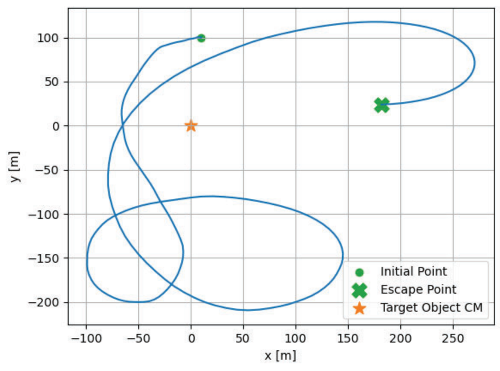
Figure 3. Random Case A. Map Level trend for reward model $R_{k,2}$ with position, map, time and thrust scores. Comparison between the two Policy Architectures: linear/MLP-3 and recurrent/LSTM.

It is worth to notice how the current method outperforms the one adopted in our previous work,[4] where A2C algorithm was exploited. Indeed here with both the linear or the recurrent policy architecture the agent is capable to achieve an average map level of 70%-80%; with A2C algorithm, with equal characteristics in terms of random initial conditions, the results were quite lower, around 40%. In Figure 4, a resulting trajectory is shown. The agent, characterized by a recurrent policy and trained with $R_{k,1}$ reward model, starting from the fixed initial relative position of r_0 = [10m, 100m, 10m] reaches a 100% map level in about 3.67h around the target object before escaping.

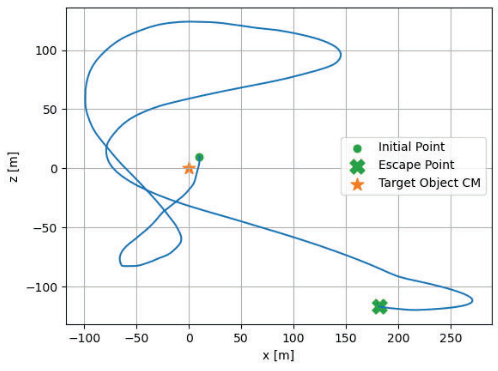
In Figure 5, a sensitivity analysis about the thrust level has been performed. The analysis exploited the already trained linear, MLP-3, networks for the two different reward models. The aim of the analysis is to understand if the additional score, related to the thrust level, actually helps the agent to reduce the number of firings. In particular, it can be noticed that, for the same type of policy architecture the average thrust level is lower for the $R_{k,2}$ reward model. This result is expected be-



(a) 3D Trajectory



(b) Trajectory XY-projection

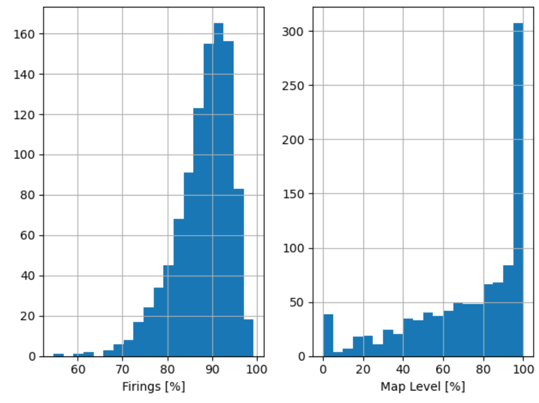


(c) Trajectory XZ-projection

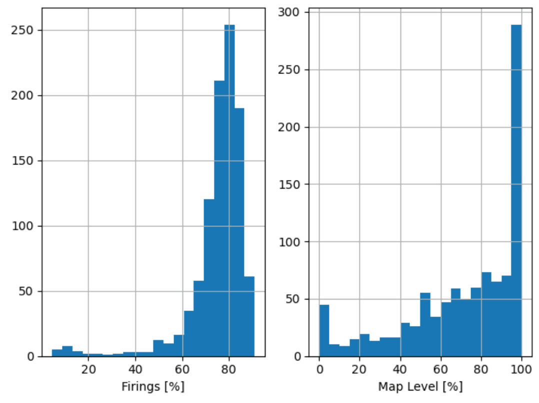
Figure 4. Random Case A. Trajectory obtained with recurrent policy.

cause the reward model considers also the score relative to the containment of the number of firings. The value shown in the plot, that is the percentage of the firing is computed as the percentage of the firings selected by the agent with respect to all the possible firings that the agent could have chosen. As observable, in the $R_{k,1}$ model the average number of firings is around 90%, while in $R_{k,2}$ model the average number is around 80%. Despite the results do not show a strong reduction in firing percentage, they demonstrates how the use of a specific thrust score is effective to accomplish the task requested. The same result is obtained also considering the recurrent policy.

In Figure 6, as example, a trajectory obtained with the $R_{k,2}$ reward model by the linear policy is shown. In this plot it is possible to observe the control action profile during the episode. In particular

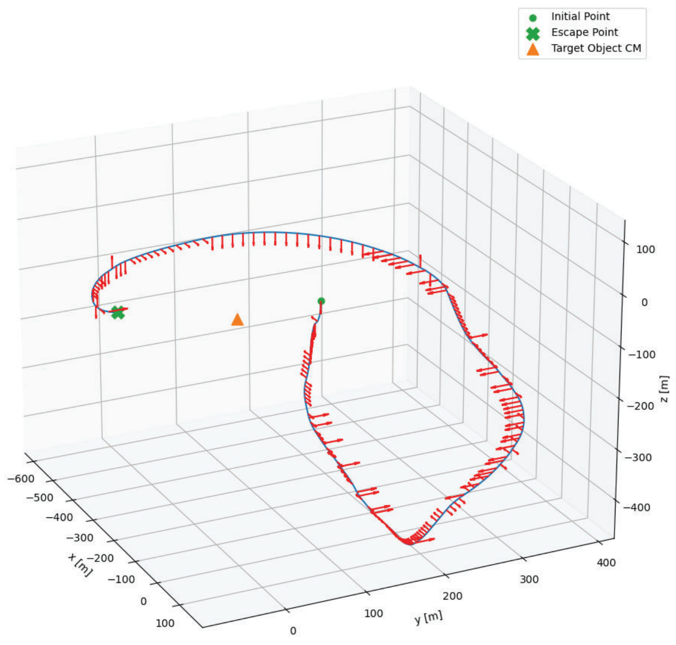


(a) Reward model $R_{k,1}$



(b) Reward model $R_{k,2}$

Figure 5. Random Case A. Thrust level comparison between the two reward models $R_{k,1}$ and $R_{k,2}$, with policy architectures: linear/MLP-3.



(a) 3D Trajectory

Figure 6. Random Case A. Trajectory obtained with linear policy.

the agent obtained this result reaching a map level of 94%, in 2.1h with a 75.4% of firings.

Random Case B. In Figure 7, the results obtained with random target initial condition, random Sun phase and random initial relative position are shown. Here the $R_{k,1}$ reward model is exploited to make the agent learn to reach the best mapping quality possible. In the plot the average trends of the Linear and Recurrent policy architecture are compared for 15000 episodes simulation. The level reached by the two policies is comparable, around 55%. Also here, as in the first two analyses shown, the characteristics of the different architectures hold. As expected, the average map level is lower than the one achieved in the *Random Case A*; this outcome was expected considering the fact that the state space is quite bigger now, due to the randomness in the initial relative position. As before, it is worth to underline that the results obtained with a PPO learning process greatly overcome the ones obtained by A2C in our previous work.[4]

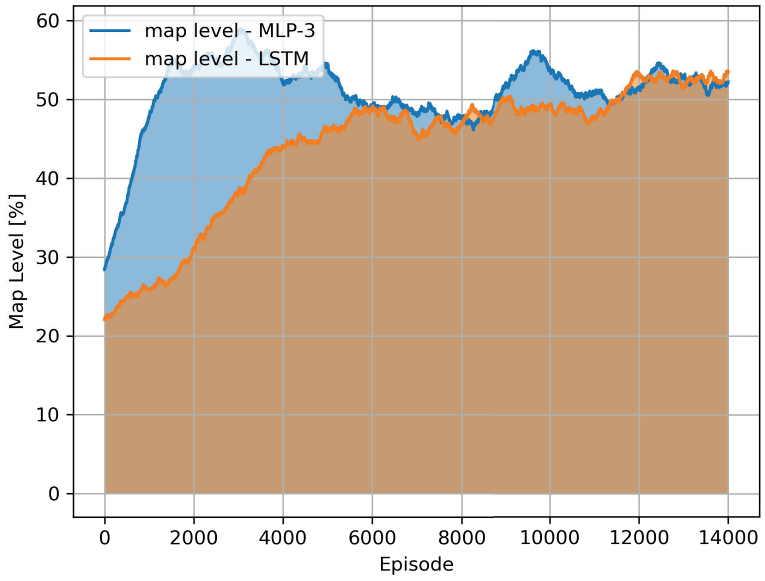


Figure 7. Random Case B. Map Level trend for reward model $R_{k,1}$ with position, map and time scores. Comparison between the two Policy Architectures: linear/MLP-3 and recurrent/LSTM.

In Figure 8, the same comparative analysis is performed. In this case the reward model, $R_{k,2}$, includes also the thrust level minimization objective. In this analysis, differently from the previous ones, the linear policy performs better than the recurrent policy with an equal learning process length. Some features are equal, like the different grown rate, especially for the recurrent policy. The reason behind the fact that the linear policy reaches ah higher level after 15000 episodes is related to the combination of the reward model, $R_{k,2}$ and the randomness level. Indeed, here, the state space is bigger and also the objective are more complex and wider, that determines the grown of the recurrent policy to be slower than in the other analysis. However, for both the architectures the level reached is around the 50%; in terms of map level, this result is not good as the ones obtained in $R_{k,1}$. However, considering the fact that for the current reward model the state space is greatly wider, the result is high enough to be considered good.

In Figure 9, the sensitivity analysis about the thrust level is shown. In this case, the analysis exploited the already trained recurrent, LSTM, network for the two different reward models. Also here, it can be noticed that, for the same type of policy architecture the average thrust level is lower for the $R_{k,2}$ reward model. As observable, for $R_{k,1}$ model the average number of firings is around 95%-100%, while for the $R_{k,2}$ model the average number is around 90%. The values are higher than the ones in *Random Case A* always because the state space is wider and therefore the agent

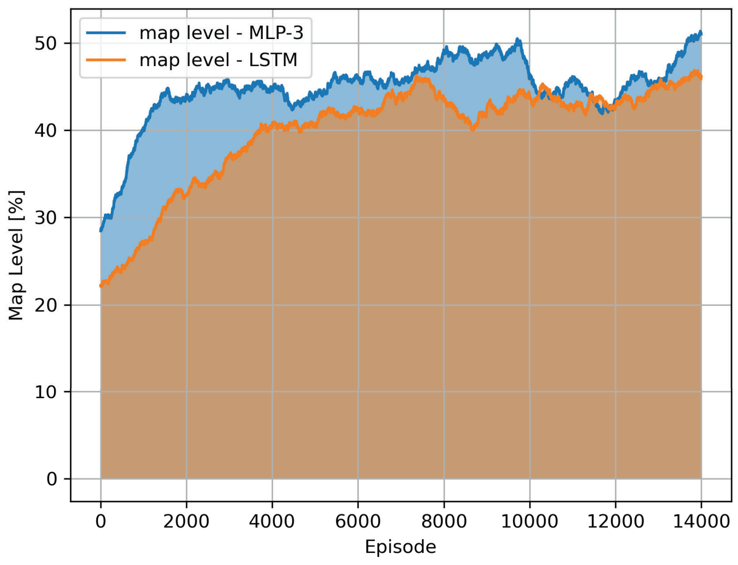
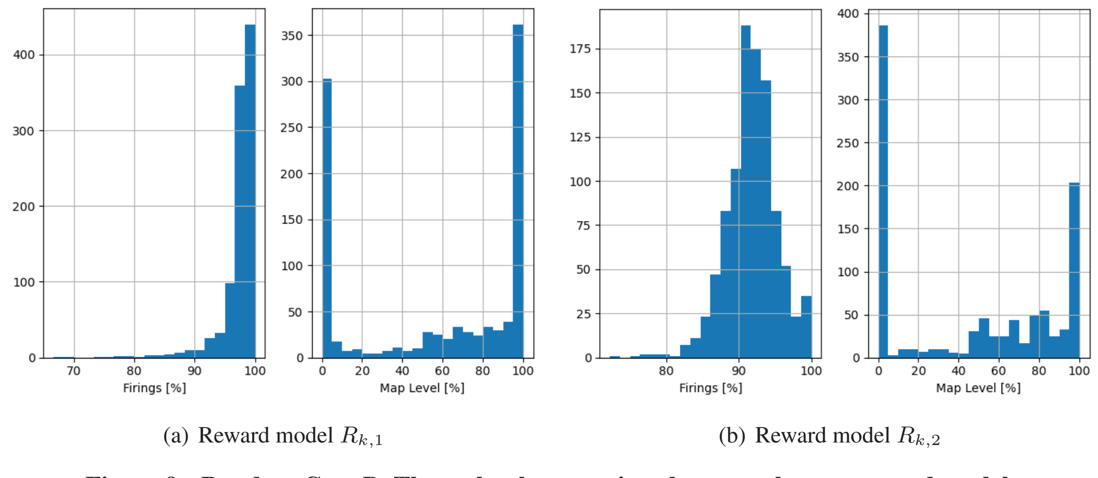


Figure 8. Random Case B. Map Level trend for reward model $R_{k,2}$ with position, map, time and thrust scores. Comparison between the two Policy Architectures: linear/MLP-3 and recurrent/LSTM.

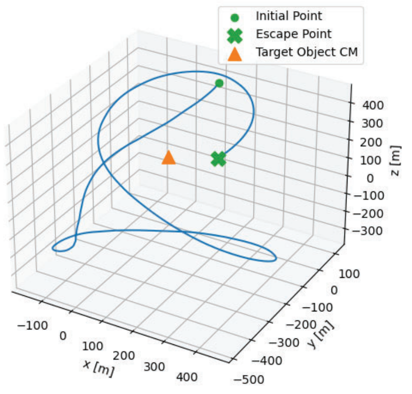
needs stronger control to follow the objectives from different initial relative positions.



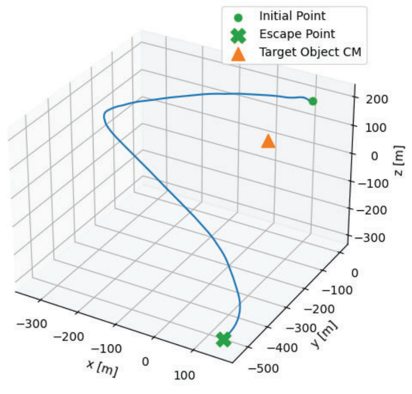
(a) Reward model $R_{k,1}$ (b) Reward model $R_{k,2}$

Figure 9. Random Case B. Thrust level comparison between the two reward models $R_{k,1}$ and $R_{k,2}$, with policy architectures: recurrent/LSTM.

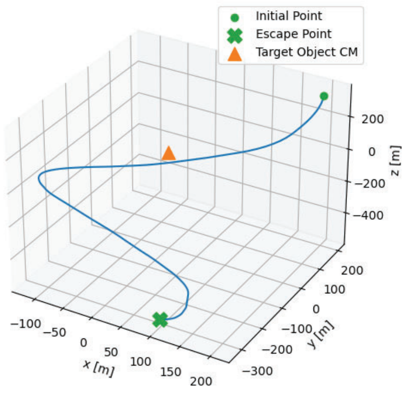
In Figure 10 some trajectories obtained with the recurrent policy are shown. As observable, the initial relative position is different for each of them.



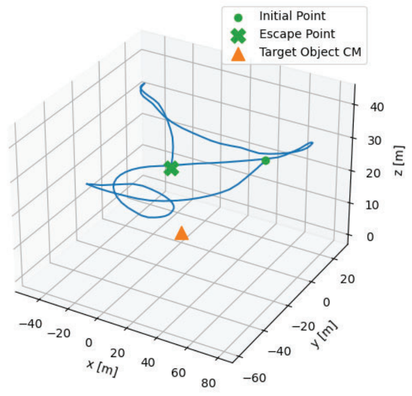
(a) 3D Trajectory - Map: 76.1% - Time: 2.21h



(b) 3D Trajectory - Map: 84.3% - Time: 1.27h



(c) 3D Trajectory - Map: 50.7% - Time: 1.06h



(d) 3D Trajectory - Map: 39.0% - Time: 1.08h

Figure 10. Random Case B. Trajectories obtained with recurrent policy.

FINAL REMARKS

The present work proposes a refined autonomous path-planning architecture for uncooperative and unknown target object exploration, aimed to reconstruct the target shape through image processing. The proposed methodology improves the results previously obtained with A2C, exploiting PPO algorithms to design an autonomous policy to control the spacecraft trajectory to obtain the highest map quality of the target. Sun illumination and viewing conditions of the body emerge as the key aspects for the realization of the algorithm to optimally retrieve image information. The sensitivity analysis has underlined the robustness of the trained agents to different Sun illumination and target object initial conditions. The sensitivity to the initial relative position is less robust and probably need for more extensive training to facilitate policy generalization to this large set of environment configurations. DRL confirms, again, to be a valid approach for solving the decision process problem, merging the advantages of reinforcement learning and neural networks.

APPENDIX A: PPO ALGORITHM

In Alg. 1 the pseudo-code for the Proximal Policy Optimization algorithm is presented.

Algorithm 1 Proximal Policy Optimization - PPO-Clip

```
1: Input: initialization policy parameters $\theta_0$, initialization value parameters $\psi_0$
2: Initialize batch
3: for k = 0, 1, 2, ... do
4:     while batch-step $b_i \leq$ batch-size do
5:         Collect set of trajectories $\mathcal{T}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment
6:         Compute Rewards $\hat{\mathcal{R}}_k$
7:         Compute Advantages $\hat{A}_k$ based on the current value function $\hat{V}_{\psi_k}$
               $$A_\pi(x_i, u_i) = \Big[\sum_{j=i}^{T} \gamma^{j-i} r(x_j, u_j)\Big] - V_\psi(x_i)$$
8:     end while
9:     Compute the probability ratio $p_k(\theta_k)$
10:    Update the policy by maximizing the clipping objective function via stochastic gradient
       descend (ADAM optimizer):
               $$L_{\text{CLIP}}(\theta) = \mathbb{E}_{p(\tau)}[\min[p_k(\theta), \text{clip}(p_k(\theta), 1-\epsilon, 1+\epsilon)]A_\pi(u_k, x_k)]$$
11:    Update value function by regression on mean-squared error:
               $$L_V = \sum_{i=1}^{N} \Big(V_\psi(x_k^i) - \Big[\sum_{j=k}^{T} \gamma^{j-k} r(x_j^i, u_j^i)\Big]\Big)^2$$
12: end for
```

REFERENCES

[1] J. P. Davis, J. P. Mayberry, and J. P. Penn, "On-orbit servicing: Inspection repair refuel upgrade and assembly of satellites in space," *The Aerospace Corporation, Report*, 2019.

[2] R. Smith, M. Self, and P. Cheeseman, "Estimating Uncertain Spatial Relationships in Robotics," *Autonomous Robot Vehicles*, Vol. 1, No. 1, 1986, pp. 435–461.

[3] H. J. S. Feder, J. J. Leonard, and C. M. Smith, "Adaptive mobile robot navigation and mapping," *The International Journal of Robotics Research*, Vol. 18, No. 7, 1999, pp. 650–668.

[4] A. Brandonisio, D. Guzzetti, and M. Lavagna, "Deep Reinforcement Learning to Enhance Fly-around Guidance for Uncooperative Space Objects Smart Imaging," *AAS/AIAA Astrodynamics Specialist Conference*, Aug 2020.

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 2015, Cambridge, Massachusetts.

[6] T. Kollar and N. Roy, "Trajectory Optimization using Reinforcement Learning for Map Exploration," *The International Journal of Robotics Research*, Vol. 27, No. 2, 2008, pp. 175–196.

[7] R. Linares, T. Campbell, R. Furfaro, and D. Gaylor, "A Deep Learning Approach for Optical Autonomous Planetary Relative Terrain Navigation," *AAS/AIAA Spaceflight Mechanics Meeting*, Feb 2017, San Antonio, Texas.

[8] B. Gaudet, R. Linares, and R. Furfaro, "Adaptive Guidance and Integrated Navigation with Reinforcement Meta-Learning," *Acta Astronautica*, Vol. 169, Apr 2020.

[9] V. Pesce, A. a. Agha-mohammadi, and M. Lavagna, "Autonomous Navigation and Mapping of Small Bodies," *IEEE Aerospace Conference*, Mar 2018, Big Sky, Montana.

[10] M. Piccinin and M. Lavagna, "Deep Reinforcement Learning approach for Small Bodies Shape Reconstruction Enhancement," *AIAA Scitech 2020 Forum*, Jan 2020, Orlando, Florida.

[11] D. M. Chan and A. a. Agha-mohammadi, "Autonomous Imaging and Mapping of Small Bodies Using Deep Reinforcement Learning," *IEEE Aerospace Conference*, Mar 2019, Big Sky, Montana.

[12] M. Indaco, D. Guzzetti, and M. Lavagna, "Autonomous Small Body Gravimetry via A2C Path-Planning," *AAS/AIAA Astrodynamics Specialist Conference*, Aug 2020.

[13] H. D. Curtis, *Orbital Mechanics for Engineering Students*. Butterworth-Heinemann, 2014, Oxford, UK.

[14] J. R. Wertz, *Spacecraft Attitude Determination and Control*. Springer Science & Business Media, 1978, Torrance, USA.

[15] R. Gaskell, "Automated landmark identification for spacecraft navigation," *Advances in the Astronautical Sciences*, Vol. 109, No. 1, 2002, pp. 1749–1756.

[16] R. W. Gaskell, O. S. Barnhouin-Jha, D. J. Scheeres, A. S. Konopliv, T. Mukai, S. Abe, J. Saito, M. Ishiguro, T. Kubota, T. Hashimoto, J. Kawaguchi, M. Yoshikawa, K. Shirakawa, T. Kominato, N. Hirata, and H. Demura, "Characterizing and navigating small bodies with imaging data," *Meteoritics & Planetary Science*, Vol. 43, No. 6, 2008, pp. 1049–1061.

[17] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): part I," *IEEE Robotics Automation Magazine*, Vol. 13, No. 2, 2006, pp. 99 – 110.

[18] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics Automation Magazine*, Vol. 13, No. 3, 2006, pp. 108 – 117.

[19] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, Vol. 32, No. 6, 2016, pp. 1309–1332.

[20] K. Åström, "Optimal control of Markov processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, Vol. 10, No. 1, 1965, pp. 174 – 205.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *CoRR*, Vol. abs/1707.06347, 2017.

[22] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization," *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), Vol. 37 of *Proceedings of Machine Learning Research*, Lille, France, PMLR, 07–09 Jul 2015, pp. 1889–1897.

[23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), Vol. 80 of *Proceedings of Machine Learning Research*, PMLR, 10–15 Jul 2018, pp. 1861–1870.

[24] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *International Conference on Learning Representations*, 2014.

[25] H. Sak, A. Senior, and F. Beaufays, “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition,” 2014.