Available online at www.sciencedirect.com

ScienceDirect

Advances in Space Research xxx (xxxx) xxx

ADVANCES IN SPACE RESEARCH (a COSPAR publication)

www.elsevier.com/locate/asr

Deep reinforcement learning spacecraft guidance with state uncertainty for autonomous shape reconstruction of uncooperative target

Andrea Brandonsio *, Lorenzo Capra [1], Michèle Lavagna

Aerospace Department, Politecnico di Milano, 20157 Milan, Italy

Received 15 November 2022; received in revised form 26 June 2023; accepted 3 July 2023

Abstract

In recent years, space research has shifted heavily its focus towards enhanced autonomy on-board spacecrafts for on-orbit servicing activities (OOS). OOS and proximity operations include a variety of activities: the focal point of this work is the autonomous guidance of a chaser spacecraft for the shape reconstruction of an artificial uncooperative object. Adaptive guidance depends on the ability of the system to build a map of the uncertain environment, figuring out its location inside of it and accordingly determining the control law. Thus, autonomous navigation is framed as an active Simultaneous Localization and Mapping (SLAM) problem and modeled as a Partially Observable Markov Decision Process (POMDP). A state-of-the-art Deep Reinforcement Learning (DRL) method, Proximal Policy Optimization (PPO), is investigated to develop an agent capable of cleverly planning the shape reconstruction of the target. Starting from previous research on the topic, this work develops further proposing a continuous action space, such that the agent is no more forced to choose between a predefined set of possible discrete actions, fixed both in magnitude and direction. In this way any combination of the three-dimensional thrust vector components is available. The chaser spacecraft is a small satellite mounting an electric propulsion engine defining the action space range, in linearized eccentric relative motion with the selected uncooperative object. Through a rendered triangular mesh, the agent capabilities of geometry reconstruction and mapping are evaluated, considering the number of quality pictures made for each face. Extensive training tests are performed with random initial conditions to verify the generalizing capability of the DRL agent. The results are then validated in a comprehensive testing campaign, whose primary focus is the introduction of noisy measurements coming from navigation, affecting pose estimation. The sensitivity of the proposed method to this condition is analyzed and the efficiency of a retraining procedure is examined. The applicability of DRL methods and neural networks to support autonomous guidance in a close proximity scenario is corroborated and the technique employed is vastly tested and verified.

Keywords: On-orbit servicing; Relative dynamics; Reinforcement learning; State uncertainty; Shape reconstruction

1. Introduction

Enhanced autonomy is driving the research of leading space agencies, as spacecraft independence would allow for reliable, cost-effective, lower risk services, and for much more flexibility in future missions planning of a space vehicle. Concerning the application of these studies, on-orbit servicing (OOS) activities (Joshua et al., 2019) have recently gained a lot of interest: maintenance, refueling,

* Corresponding author.
E-mail addresses: andrea.brandonisio@polimi.it (A. Brandonsio), lorenzo.capra@polimi.it (L. Capra), michelle.lavagna@polimi.it (M. Lavagna).
[1] Equal contribution.

https://doi.org/10.1016/j.asr.2023.07.007
0273-1177/

debris mitigation, upgrade, repair, assembly, relocation, orbit modification and non-contact support are some of the operations included in this definition. An increased level of autonomy while carrying out these activities would lead to major benefits. These operations share a similar scenario, in which two objects are in relative motion and interacts in some kind of form, generally with the first one, referred to as the *servicer*, performing some action on the second, called *client*. The work here presented develops an innovative adaptive guidance algorithm for the path-planning of a chaser spacecraft trajectory around an uncooperative artificial object, aiming to the shape and map reconstruction of the latter. As specified, in this scenario the client is non-cooperative, so it does not interact directly with the servicer, and it is not able to exchange information with it. In this context, the spacecraft autonomously explores the surrounding environment, understanding where it is located inside of it, and consequently choosing the actions to take according to a certain objective function to be defined. Thus, the problem falls in the Simultaneous Localization and Mapping (SLAM) (Durrant-Whyte and Bailey, 2006) framework, and since the planning operations is also performed, it is called *active*. This problem formulation has been widely researched in the last few years, particularly in the field of robotics. An overview on active SLAM state-of-the-art is given by Placed et al. (2023). SLAM may be phrased as a Partially Observable Markov Decision Process (POMDP) (Sutton and Barto, 2018), which entails an agent interacting with the environment and exchanging information with it. POMDP derives from Markov Decision Process (MDP), which is a stochastic control process, that does not have any memory. Nowadays, according to (Silvestrini et al., 2023), Deep Reinforcement Learning (DRL) is the most common method to solve both MDP and POMDP, whose goal is to solve for the decision-making policy of the agent. Reinforcement Learning (RL) algorithms are a powerful tool when dealing with decision-making problems and the combination with Neural Networks (NN) in DRL allows to improve the generalizing capabilities of the resulting policy, and to solve more complex problems characterized by high-dimensional and continuous state and action spaces (Sutton and Barto, 2018). Even if POMDP represents the first step in applying policy models in the real world scenarios, it is worth to underline how it still remains a problem in RL since directly viewing partial observations as full states violates the Markov property of state transitions: this is particularly problematic when the system is strongly affected by historical information or large state/action spaces. The first problem is treated in (Xie et al., 2023), where the addition of long memory ability is proposed; while the second is handled with alternative Actor-Critic formulation for a multi-component systems, as in (Andriotis and Papakonstantinou, 2019).

Different applications of machine learning techniques have been studied in the last few years in the context of guidance, navigation and control. Starting from relative motion scenarios, in Silvestrini and Lavagna (2021b) and Silvestrini and Lavagna (2021a), neural networks are exploited to help a model predictive control (MPC) guidance and control system in learning the dynamics of the environment and be able to safely maneuver the spacecraft or reconfigure distributed systems. In Silvestrini et al. (2022a) and Silvestrini et al. (2022b), a step further is accomplished using supervised learning trained convolutional networks as embedded navigation system for lunar lending and relative motion rendezvous scenarios. However, all these works focus on well-established GNC guidance and control algorithms. Instead, the first formulation as a POMDP for a spacecraft guidance design problem was proposed by Pesce et al. (2018) and then developed further in Chan and Agha-mohammadi (2019) and Piccinin et al. (2022), who adopted Reinforcement Learning (RL) to plan the trajectory of a chaser spacecraft for small-bodies imaging. These works adopted an active SLAM formulation based on RL algorithms, such as NFQ (Neural Fitted Q) and DQN (Deep Q Network), which nowadays have been outperformed by other methods more suitable for continuous state-action space problems, like PPO. Major contributions to the application of RL advanced techniques, as the PPO algorithm, are given by Gaudet et al. (2020a) and Gaudet et al. (2020b), who investigates, among others, planetary landing and close proximity operations. This particular RL approach has been proved effective in different environments as the circular restricted three-body problem (CR3BP) framework (Scorsoglio et al., 2023) and compared to cloning techniques, as done by Federici et al. (2021).

This work takes shape from our previous studies (Brandonisio et al., 2021; Capra et al., 2023), where, for the first time, these tools were exploited for the guidance and control of a chaser spacecraft aimed at reconstructing the shape of an artificial object. The selected target is VESPA (Vega Secondary Payload Adapter), and the chaser spacecraft is a small satellite equipped with both visible and thermal infrared cameras and a single thruster with electric propulsion. A state-of-the-art Deep Reinforcement Learning algorithm, Proximal Policy Optimization (PPO) (Schulman et al., 2017), is investigated to solve for the chaser decision-making policy, mapping the input observations to the action to take. An extensive testing campaign is performed, to verify the results and analyze the model robustness and sensitivity to several different conditions. In particular, a detailed analysis of the performance in presence of state uncertainty is presented. Therefore, this paper, starting from this background, develops the formulation further, removing some limiting assumptions and testing a more realistic, flexible, and robust architecture. In particular, we want to analyze two different aspects: the improvement of the action space from a discrete space to a continuous one, and the study of how the agent performance are affected by state uncertainty. The aim is to take a step forward in the autonomous mapping of uncooperative artificial space objects, thus confirming the

applicability of these DRL techniques to support a potential exploratory mission around VESPA, as currently explored by the ESA e.Inspector project (Silvestrini et al., 2020).

2. Problem statement

The development of an autonomous guidance algorithm depends on the overall architecture, which is described in Fig. 1. The part of the block diagram that is highlighted represent what has been developed in this work, while the other blocks are reported with the intention of giving the overall picture of the close proximity GNC scenario.

The inputs to the guidance block are the relative motion between the chaser and the target object (ex. VESPA), and the relative attitude between the two. These information may come from different sensors sources. In this work, the inputs have been considered as generated by image processing and pose estimation tools. A vision-based navigation system may employ both visible (VIS) and thermal infrared (TIR) imaging. In Brandonisio et al. (2021), only a VIS is considered to generate the input of the autonomous agent, and to define the reward model necessary to train it under the mission objectives. A coupled approach (VIS + TIR) has been proposed by Civardi et al. (2023), where it demonstrated the effectiveness of combining imagery from different bands to space navigation purpose in proximity operation scenarios. This allows to avoid problems of illumination condition typical of VIS-only systems, as underlined by Brandonisio and Lavagna (2021).

The implementation of the navigation system is out of the scope of the presented work, that focuses on the development of the guidance algorithm; therefore, the information regarding relative motion and target attitude are assumed to be known at each step during the nominal case. As anticipated, state uncertainty will be then introduced and the model will be tested against it to assess its robustness. The output of the guidance algorithm is the trajectory that the spacecraft should follow to achieve the mission objectives, consisting in the target object shape reconstruction map, which can be retrieved via stereophotoclinometry (SPC) methodology developed by Gaskell (2001).

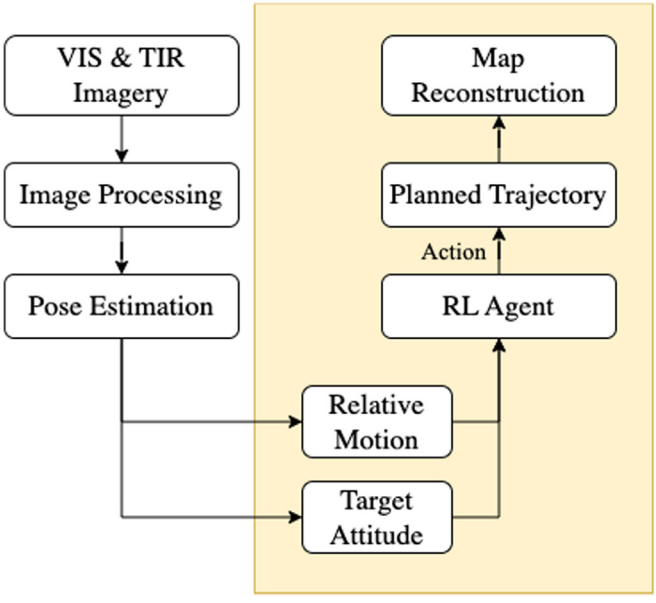

Fig. 1. Fly-around planning architecture.

2.1. Problem dynamics

This section introduces the dynamics models that express the relative motion between the chaser and the target. The equations of motion can be linearized under the assumption of a very low chaser-target relative distance with respect to the main attractor distance. The linearized eccentric model here exploited is the one proposed by Inalhan et al. (2002), selected as a result of a trade-off between dynamics accuracy and computational efficiency. The equations are reported in Eq. 1, considering the Local Vertical Local Horizontal (LVLH) reference frame centered in the target object center of mass:

$$\begin{cases} \ddot{x} = \frac{2\mu}{r^3}x + 2\omega\dot{y} + \omega^2 x \\ \ddot{y} = \frac{-\mu}{r^3}y - 2\omega\dot{x} - \omega^2 y \\ \ddot{z} = \frac{-\mu z}{r^3} \end{cases} \tag{1}$$

where r is the radius of the target orbit, μ is the primary attractor gravitational parameter, and $\omega = \dot{f}$, defined in Eq. 2, is the time derivative of the target true anomaly, expressed as follows:

$$\omega = \dot{f} = \frac{n(1 + e\cos f)^2}{(1 + e^2)^{\frac{3}{2}}} \tag{2}$$

with f being the target true anomaly, e its orbit eccentricity, and $n = \sqrt{\frac{\mu}{r^3}}$ the mean motion.

Concerning the relative target attitude motion, this work follows the two main assumptions defined in Brandonisio et al. (2021), exploited to simplify the already quite complex problem: target small angles rotation and chaser camera always pointed toward the target center. In this way, the spacecraft attitude dynamics control is completely neglected and the Euler equations for the target in the LVLH reference frame can be approximated as expressed in Eq. 3.

$$\begin{cases} I_x\ddot{\theta}_x + n\left(I_z - I_y - I_x\right)\dot{\theta}_y + n^2(I_z - I_x)\theta_x = 0 \\ I_y\ddot{\theta}_y + n\left(I_x + I_y - I_z\right)\dot{\theta}_x + n^2(I_z - I_x)\theta_y = 0 \\ I_z\ddot{\theta}_z = 0 \end{cases} \tag{3}$$

with I_x, I_y, I_z being the target object principal components of inertia, $\theta_x, \theta_y, \theta_z$ being the Euler angles around the three directions of the LVLH reference frame, and $\dot{\theta}_i, \ddot{\theta}_i$ the corresponding first and second time derivative.

2.2. Visibility model

At each step of the simulation, a *visibility* model is necessary to evaluate which parts of the target are observable by the VIS and/or TIR cameras. The model is based on the target object shape, that is designed with a triangular mesh. This makes the definition of other possible shapes easier, without changing the visibility framework. As already introduced, in this work the analysis have been carried out considering VESPA as target object, and its triangular mesh is shown in Fig. 2.

Two main parameters affects the visibility model:

- The cameras field of view (FOV);
- The incidence angle between the cameras and each of the mesh faces.

If only the VIS camera is present on the chaser spacecraft, also the Sun incidence angle becomes greatly important to define the visibility of the object.Knowing the relative pose between chaser and target, the normal vector of each faces can be computed, and consequently also the incidence angle. All the faces inside the camera FOV are considered visible. The reconstruction of the map is ideally performed during the image processing step, which, as already mentioned, is out of the scope of the presented work. Anyway, the selection of the shape reconstruction method is important for the formulation of the reward model. Following the definition presented in Brandonisio et al. (2021), the main requirements for a good shape reconstruction with SPC can be expressed mathematically by the follow conditions:

- minimum of 3 images per face;
- incidence angle of 45° (with acceptable range between $10^\circ - 50^\circ$ and limits between $5^\circ - 60^\circ$)

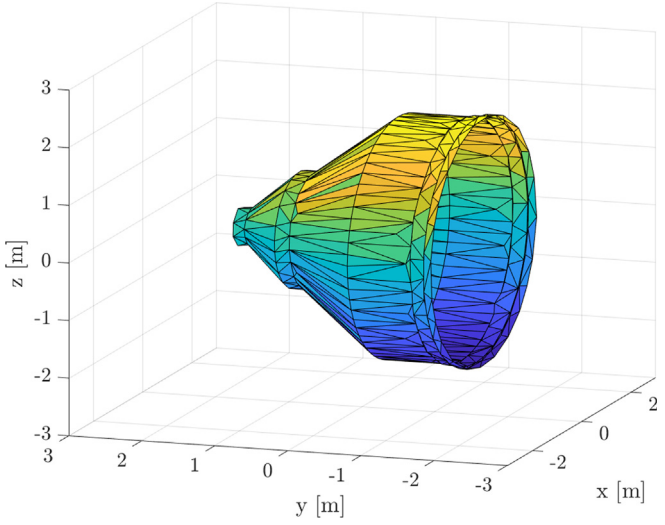

Fig. 2. Triangular mesh of VESPA.

There should be also few additional recommendations on the sun incidence angle conditions, negligible when the architecture also adopts thermal infrared cameras, as in this case.

3. Autonomous guidance

The autonomous exploration and trajectory planning in an unknown environment can be formulated as an active Simultaneous Localization and Mapping (SLAM) problem, in which an agent builds a map of its surroundings while concurrently estimating its positions and planning the next action to take. This problem can be phrased as a Partially Observable Markov Decision Process (POMDP). The next section aims at developing the mathematical tools necessary to understand the problem and how it is solved.

3.1. Partially observable Markov Decision Process

A POMDP is a Markov Decision Process (MDP) with state uncertainty, meaning the agent cannot know the *true* state, but only a *belief* state using observations. This formulation is valid whenever the agent senses the environment via sensors, which inherently introduce errors in their measurements, or when it may not be able to observe all the state variables describing the environment.

A POMDP is characterized by a 6-tuple. (**S**, **A**, **R**, **T**, Ω, **O**):

- **S** is the space of all possible states in the environment;
- **A** is the space of all possible actions that can be taken in all the states of the environment;
- **R** is the reward function, guiding the action selection to maximize it;
- **T** (s, s') is the transition function governing the probability of moving from one state to the next, given the current state and action;
- Ω is the space of possible observations;
- **O** $(o'|a, s')$ is the probability of making a particular observation, taking an action that leads to a particular new state.

This type of problems is quite complex to solve and may become computationally intractable if not reduced to a simpler MDP. This can be done including the history h, that plays the role of an archive of past actions and observations. The new formulation, known as belief-space MDP, is described by a 4-tuple (**B**, **A**, **R**, **T**):

- **B** is the belief space, where the *belief* is defined as $b = p(s|h)$, so it is the probability of being in a certain state s after the history h.

A, **R**, and **T** have the same meaning described before. The solution to the belief-space MDP is a *policy* π, which represents the mapping function from states to actions that the agent is employing. This decision-making feature is

optimal if the agent concurrently maximizes the reward function, thus reaching the goal effectively. In case of an infinite horizon problem, the optimal policy is defined as in Eq. 4:

$$\pi_* = \underset{\pi}{\text{argmax}} \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R(a_k, b_k) \right] \tag{4}$$

where $\gamma \in [0, 1]$ is the discount factor, introduced as a mechanism to control how myopic or short-sided is the agent, exponentially decaying the effect of rewards far away in time. R represents indeed the reward signal, depending on the action a and belief b at step k.

4. Deep reinforcement learning

Reinforcement Learning is a widely employed tool for solving MDPs (Sutton and Barto, 2018), and its combination with Neural Networks for function approximation, allows to solve many complex problems characterized by high-dimensionality and partial observability. A state-of-the-art Deep Reinforcement Learning algorithm for continuous state and action space problems, called Proximal Policy Optimization (PPO) (Schulman et al., 2017), is considered for the optimization of the spacecraft decision-making policy.

4.1. Proximal policy optimization

PPO is a policy-gradient method, belonging to the Actor-Critic family (Mnih et al., 2016). It outclasses most of the other DRL algorithms in many typical benchmark problems, because of its improved training stability. It builds up from the Trust Region Policy Optimization (TRPO) method (Schulman et al., 2015), retaining its reliability and data efficiency, but handling the loss function in a much simpler and well-planned fashion. Starting from the TRPO loss function, which exploits the probability ratio, defined in Eq. 5, between the policy at subsequent timesteps, PPO increases training robustness by clipping the objective function and limiting the possible update, so that the policy does not change drastically. This simple expression is reported in Eq. 6.

$$p_k(\theta) = \frac{\pi_w(a_k|s_k)}{\pi_w(a_{k-1}|s_{k-1})} \tag{5}$$

$$L^{CLIP}(w) = \widehat{\mathbb{E}}_k \left[min(p_k(w), clip(p_k(w), 1-\epsilon, 1+\epsilon)) \right] A_k \tag{6}$$

In both Eq. 5 and Eq. 6, w refers to the networks parameters (i.e. their weights and biases). The parameter ϵ indicates the clipping factor (generally selected in the range $\epsilon \in [0, 0.2]$), while A_k is the advantage function at step k.

Common practice in PPO algorithms is the addition of an entropy regularization term to the clipping objective function, as in Eq. 7, to ensure a sufficient exploration level during training:

$$L^{PPO}(w) = L^{CLIP}(w) + c_2 S(\pi_w) s_k \tag{7}$$

where $S(\pi_w)$ is the entropy bonus term, that is function of the current policy, and c_2 a scalar multiplying factor that determines the influence of the entropy term on the overall loss function.

From the A2C formulation (Mnih et al., 2016), the advantage function A is retained, and represents how better a selected action is compared to all the others at a given state. It is simply computed as the difference between the discounted reward r and the state value function V baseline, computed by the critic network. This concept is described in Eq. 8.

$$A(s_k, a_k) = \left[\sum_{j=k}^{T} \gamma^{j-k} r(s_k, a_k) \right] - V(s_k) \tag{8}$$

The agent stores a number of transitions specified by the batch dimension of the experience buffer, by interacting with the environment, and then exploits them to update the networks via backpropagation. This way the networks adjust their parameters to better fit the problem objectives.

5. Planning architecture

This work proposes an innovative decision-making process to autonomously plan the pseudo-optimal guidance around the uncooperative space object VESPA, with Deep Reinforcement Learning. This is coupled with a pre-processing phase, in which information coming from the external environment, sensors and the object conditions are elaborated to estimate the state. Once again, it is highlighted that this part is out of the scope of this work, and as such, the developed analysis starts with this information given for granted as input. The state is then fed to the autonomous guidance agent which crafts the control policy to maximize the reward, affecting the environment and all the others information providers. In the next sections, a detailed and critical description of all the architecture components is presented, according to the DRL framework.

Environment model. It represents everything surrounding the agent and, in this context, it is the relative dynamics model governing the reciprocal motion between the chaser and the target, the attitude motion of the object and the visibility model. In particular, the relative motion between the two objects is directly affected by the agent actions, that influence the set of equations. With respect to Brandonisio et al. (2021), the implementation of a vision-based system, exploiting also thermal infrared imagery, allows to reduce the applicability constraints and removes a few assumptions: unfavorable illumination conditions are no more an issue, like an inadequate incidence angle between the Sun direction and the normal of the mesh faces, or Earth eclipses and reflections. The main part of this work selects VESPA as target object. VESPA is a space debris (the upper stage of the VEGA launcher), currently orbiting around the Earth, and it is of huge interest for several projects, like the ClearSpace-1 mission or the e.Inspector project (Silvestrini et al., 2020), funded by the European Space

Table 1
Orbital elements of VESPA.

$a\,[km]$	e	$i\,[^\circ]$	$\Omega\,[^\circ]$	$\omega\,[^\circ]$	$f\,[^\circ]$
6878	0.009633	98	0	0	0

Agency. To simulate both its orbital and rotational motion, the inertial properties and keplerian parameters are approximated as reported in Eq. 9 and Table 1.

$$I = \begin{bmatrix} 0.5208 & 0 & 0 \\ 0 & 0.5208 & 0 \\ 0 & 0 & 0.6667 \end{bmatrix} kgm^2 \tag{9}$$

State model. The state space models all the information coming from the environment. Taking inspiration from Brandonisio et al. (2021), at first it is assumed that the agent has the knowledge of the correct state variables at each timestep. Obviously, in practice all the input data derive from sensor measurements, which inherently introduce errors. This uncertainty is studied in the second part of the paper, where the sensitivity analysis tests have been performed. In general, the input vector, with which the agent is trained, should be tailored in such a way that it contains only essential information for the decision-making process, to build a policy capable of selecting the appropriate action in every condition the agent may find itself. In this case, the variables are the ones that properly identify the close proximity scenario and facilitate the agent learning. Moreover, a key factor to be considered is the possibility of estimating these quantities by means of on-board instruments, which is of fundamental importance for an autonomous spacecraft.

$$\mathbf{S} = \begin{Bmatrix} p \\ v \\ \theta \\ \dot{\theta} \end{Bmatrix} = \begin{Bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \theta_x \\ \theta_y \\ \theta_z \\ \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{Bmatrix} \tag{10}$$

The state in Eq. 10 contains the relative motion (position p and velocity v) between the chaser and the target, together with the attitude information about VESPA, in terms of Euler angles and angular velocities $(\theta, \dot{\theta})$.

Action model. The action space represents all the possible decisions that the agent could take at each timestep with its policy. In this work, the agent can interact with the surrounding environment by means of the acceleration vector a, representative of the control action, that directly affects the equations of motion as expressed in Eq. 11. This kind of action control can easily represent a thrust vector coming from the thruster.

$$\begin{cases} \ddot{x} = \frac{2\mu}{r^3}x + 2\omega\dot{y} + \omega^2 x + a_x \\ \ddot{y} = \frac{-\mu}{r^3}y - 2\omega\dot{x} - \omega^2 y + a_y \\ \ddot{z} = \frac{-\mu z}{r^3} + a_z \end{cases} \tag{11}$$

$[a_x, a_y, a_z]$ is the control vector given by the thruster.

This section encapsulates one of the main advancement developed by this work. Current research on the topic developed in Brandonisio et al. (2021); Brandonisio and Lavagna (2021) adopts a *discrete* action space, so that the action is selected between a set of predefined thrust impulses fixed both in direction and magnitude, as in Eq. 12.

$$A = \left[+T_x, -T_x, +T_y, -T_y, +T_z, -T_z, 0\right] \tag{12}$$

where, for example, $+T_x$ means a thrust action aligned with the +x direction of the chaser's body. With a *continuous* action space, instead, the control action becomes a tridimensional vector, that ideally is free to point towards the entire 3D space, without being constrained by some specific direction. Moreover, since it is continuous, the magnitude of the thrust vector can vary inside the limits specified by the propulsion system on-board. To define the action space, an electric propulsion system with a single thruster (Martínez et al., 2019) is employed. The most notable attribute affecting the formulation is the maximum thrust and the minimum impulse bit (MIB), since they define the range inside which the decision-making policy can select the magnitude of the action. Another important parameter for the action model is the control interval Δt, that defines the time elapsing between one control action and the other. The setting of these parameters entails a trade-off between fidelity of the control frequency and computational burden. In the *training* section, the resulting analysis on the continuous action space model will be presented. In the *testing* section, instead, a sensitivity analysis comparison between the discrete and continuous models will be described in terms of state uncertainty.

Reward model. The reward function is one of the main characters, if not the main one, when talking about Reinforcement Learning. It drives the agent policy, aiming at maximizing it by means of positive and negative scores, which should incentivize a specific agent behavior. The reward here defined phrases the objectives and constraints of the problem in mathematical form with different scores' formulations:

- *distance score*: in the case of proximity operations, a general and intuitive idea is that the chaser spacecraft shall not crash onto the target, nor escape far away from it. This constraint is formulated adopting a lower and upper limit in terms of relative distance between the two objects. In this way it is intrinsically introduced

safety in the operations, by incentivizing the agent to avoid dangerous regions of space, in which the mission would completely fail. Note that this bounded region represents also a possible terminal state for the agent: if the lower or the upper limit are overcome, then the episode ends and the agent is given a negative reward signal. The associated mathematical expression and score are reported in Eq. 13:

$$r_d = \begin{cases} -100 & \text{if } d \leqslant D_{min} \text{ or } d \geqslant D_{max} \\ 1 & \text{otherwise} \end{cases} \tag{13}$$

where $d = \|p\|$ is the distance between chaser and target, with $D_{min} = 50m$ and $D_{max} = 500m$.

- *incidence angle score*: regarding the main goal of the presented work, the agent should maximize a reward function that enables it to better map the target. This is strictly connected to the adopted mapping technique, that, being SPC, requires to assert some specific conditions in terms of incidence angle and number of quality images per mesh face, as discussed before. At each time-step the agent keeps track of the target rotation and recomputes the normal direction for each of the mesh faces. First, a screening of the faces that are in the field of view of the spacecraft's cameras is performed. Then, the angle, ε, between each of the normal directions of the faces in visibility and the camera vector, assumed as continuously pointing the target center, is calculated. A score, reported in Eq. 14, is formulated on the base of the visibility model defined in Section 2.2:

$$r_\varepsilon = \begin{cases} 1 & \text{if } 10^\circ \leqslant \varepsilon \leqslant 50^\circ \\ \frac{1}{5}\varepsilon - 1 & \text{if } 5^\circ \leqslant \varepsilon \leqslant 10^\circ \\ 6 - \frac{1}{10}\varepsilon & \text{if } 50^\circ \leqslant \varepsilon \leqslant 60^\circ \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The scores retrieved from each mesh face are then summed together and an average reward signal is obtained dividing by the number of faces n_{faces}, as reported in Eq. 15.

$$r_{\varepsilon_{avg}} = \frac{\sum r_\varepsilon}{n_{faces}} \tag{15}$$

- *map percentage score*: to better reconstruct the target geometry and shape, a reward on the current level of the map is necessary. The overall map is fragmented into a number N_p of quality photos for each face constituting the mesh, where quality is to be intended with respect to the incidence angle ε between the camera and the face. At each time step, the map percentage can be computed counting the number of quality pictures ($r_\varepsilon \neq 0$) available for each face N_q up to that moment and dividing this quantity by N_p times the number of mesh faces n_{faces}, as in Eq. 16. At each time step, the algorithm checks which faces of the mesh are in visibility of the camera, and a picture of one of these faces is said to be of "good quality" if the reward signal r_ε associated to that single face is greater than zero.

$$M_{\%,k} = \frac{N_q}{N_p * n_{faces}} \tag{16}$$

$$r_m = \begin{cases} 1 & \text{if } M_{\%,k} > M_{\%,k-1} \\ 100 & \text{if } M_{\%,k} = 100 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

In Eq. 17, note how the agent is rewarded for improving the map level and it is also given a big bonus for completing the map reconstruction. The condition in which the agent has successfully reconstructed the total target shape is intentionally not defined as a stopping condition. In this way the agent is rewarded more if it rapidly completes the map. In general an episode will end only if the spacecraft escapes the bounded region of space defined or if the defined simulation time runs out.

After all the reward signals are described, the total reward received by the agent at each time step is simply the sum of the three components:

$$R = r_d + r_{\varepsilon_{avg}} + r_m \tag{18}$$

As anticipated, the following result analysis is differentiated into two sections: the first describes the process adopted to train a continuous action space agent to perform the map reconstruction of VESPA. The second, instead, compares the already trained discrete (derived from Brandonisio and Lavagna (2021)) and continuous action space agent, testing them on the robustness with respect to the state space uncertainty.

6. Training

In order to perform the training of the DRL agent, the major algorithm characteristics and some important variables relative to the problem at hand are critically commented in the following paragraphs.

Initial conditions. Generalization is a key property in this work because the goal is to obtain an agent that can consistently perform well, whatever the initial conditions of the state variables are. For this reason, at the start of each episodic simulation run, they are randomly generated inside a certain range, reported in Table 2. The relative ini-

Table 2
Initial condition ranges for all the input state variables.

Variable	Range
d	$2D_{min} < d < 0.5D_{max}$
α	$0^\circ < \alpha < 360^\circ$
δ	$-90^\circ < \delta < 90^\circ$
v	$0\,m/s$
θ_i	0°
$\dot{\theta}_i$	$-0.001 rad/s < \dot{\theta}_1 < 0.001 rad/s$

tial position is formulated with spherical coordinates, where α and δ represent the azimuth and the elevation angle respectively. The other variables instead have already been introduced.

Neural Networks design. A simple Multi-Layer Perceptron (MLP) neural network is employed for both the actor and the critic. The size and depth of the network for both characters are reported in Table 3; they have been defined from literature review (Gaudet et al. (2020a)), and previous trade-off analysis carried out in Brandonisio et al. (2021). The learning rate and the hidden layers specifications, instead, come from a simple grid search analysis (Yang and Shami, 2020).

Scenario parameters. Some more general parameters (related to the scenario formulation) that affects the performance level obtained are here discussed.

- The cameras field of view is selected in accordance with typical values for VIS and TIR cameras: FOV= 10°;
- The *control* timestep determines how frequently the thruster is fired: $\Delta t = 1s$;
- The *picture* timestep, Δt_p, represents the time interval between two subsequent images: $\Delta t_p = 10s$. This feature is important because neither a memory or camera state is inserted in the overall state vector, and therefore it is needed to limit the smearing effect due to a too short time elapsing between two subsequent pictures.
- The number of quality pictures for each face of the mesh is set at $N_p = 20$. Recall that in Section 2.2, the minimum number of images requirement was set to 3, so N_p is designed with a large enough margin to have a more robust method.

Starting from these parameters, the agent is trained for 30000 episodes on an objective function that rewards the decision-making policy when it improves the covered map of the uncooperative target VESPA, and remains in the safe region of space. The results of this training process are presented in Fig. 3, Fig. 4.

Few remarks can be highlighted from the plots:

- the learning process can be considered successful, as the curve of the score is stable, and, for the most part, increases with the number of episodes;

Table 3
Actor & Critic Network characteristics.

Layer	Actor Neurons	Critic Neurons
Input	12	12
1^{st} hidden layer	256	256
2^{nd} hidden layer	256	256
Output	3	1
Learning rate	10^{-5}	5×10^{-5}
Activation function	Tanh	Tanh

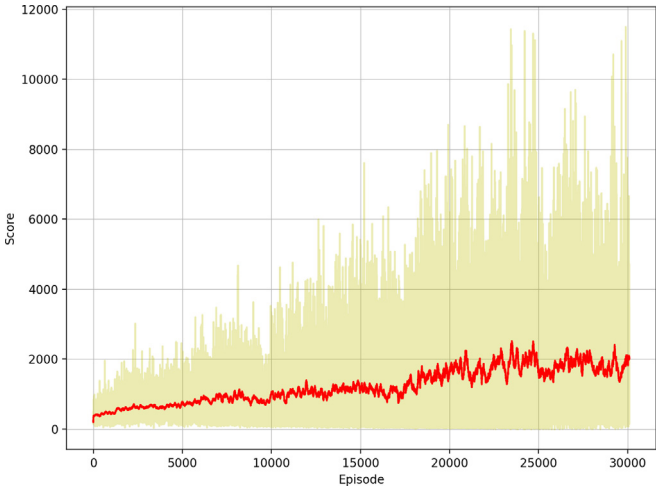

Fig. 3. Average score profile during model training.

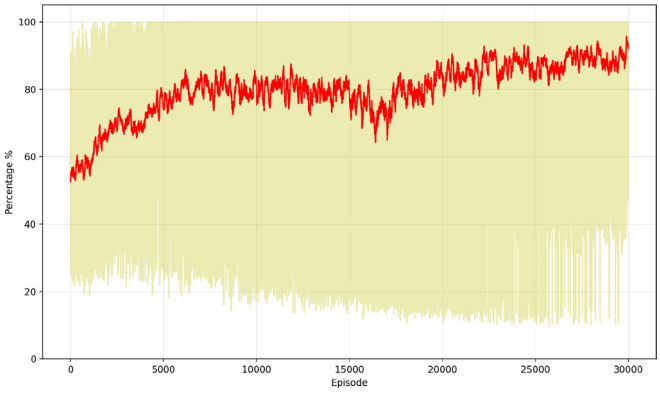

Fig. 4. Average map percentage profile during model training.

- the trend of the two profiles (score and map) is quite similar and they both grows over time. This can be thought as a check on the reward function formulation: the agent improves the average score when it concurrently reaches higher level of map reconstruction, meaning that the agent is learning correctly;
- at the very beginning of the training, the autonomous guidance agent seems to behave randomly, that is because it still needs to learn the most convenient mapping between state and action. As highlighted in Capra et al. (2023), the average map percentage settles down around 50% with a random acting agent. Therefore this value can be considered as a benchmark to evaluate the effectiveness of the agent training;
- the average map level achieved at the end of the training step is around 95%.

An example of trajectory obtained with the trained agent and completing 100% of the target object map is reported in Fig. 5.

The model is tested on 5000 episodes to retrieve some interesting statistical data on its performance, summarized in Table 4.

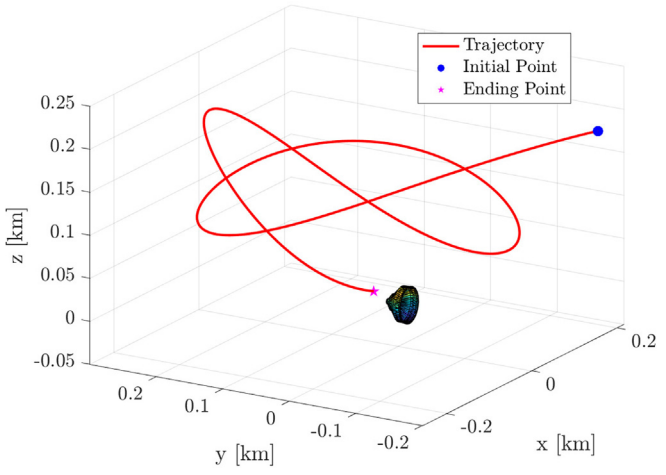

Fig. 5. Example fly-around trajectory.

Table 4
Principal model statistics.

$n_{\%}$	$t_{100\%}\,[s]$	$m_{prop}\,[kg]$
$\sim 60\%$	1595	4.56×10^{-5}

These indicators can be interpreted as follows:

- $n_{\%}$ is the percentage of episodes in which the agent reconstructs the entire 100% of the map;
- $t_{100\%}$ is the average time it takes the agent to complete the map;
- m_{prop} is the average propellant mass consumed by the chaser spacecraft, computed with the Tsiolkovsky's rocket equation.

7. Testing

An extensive testing campaign is carried out to assess the model performance, its robustness to non-nominal conditions, and its sensitivity to some of the parameters previously selected. Particular emphasis is given to the noise affecting on board sensors, which in turn causes issues in the perfect reconstruction of the pose. In the following sections a detailed analysis of the performance of two different models of the agent is described, together with possible improvements in the design.

7.1. Navigation uncertainty

A faulty assumption employed during both training and testing the proposed guidance algorithm is the correct estimation of the relative pose with the target at each time step. In reality, these estimation is, in most cases, the output of an estimation filter, which takes as input the model of the dynamics and the sensor measurements, that are inherently affected by noise and imperfections. This means that the assumption of perfectly known state is not satisfied a priori and, as such, a deeper analysis is required to verify the applicability of the algorithm in higher fidelity scenarios.

Here the autonomous guidance performance is evaluated in presence of noise, which, as just said, can be a result of both errors in the sensors measurement and in the mathematical formulation of the problem, that is not able to represent all its characteristics. Noise is added at each time-step between the estimated value coming from navigation and the guidance block. In regard to the problem at hand, noise is assumed to affect the state input vector in terms of chaser-target relative position and velocity, as well of the attitude angles describing the relative target rotation. As specified in the previous chapters, the incoming state vector is considered resulting from an image-based navigation system. Therefore, much attention has been paid in order to properly define the type of errors affecting the navigation outputs. Indeed, most of the time, image-based navigation systems are not affected by deterministic errors, but by stochastic errors. For this reason, concerning the relative position and velocity, a uniformly random error constrained in predefined ranges is taken into account. Differently happen, for the angular position error, where a Gaussian noise is applied to the input state, as motivated in Piazza et al. (2022). In Eq. 19 and Eq. 20, the position and velocity error ranges are defined. These are typical values for VIS image-based navigation tool, as the one proposed in Silvestrini et al. (2020). In Eq. 21 the standard deviation considered to describe the angular error is defined, and it is valid for all the three directions.

$$e_{pos} = [-10, +10]m \tag{19}$$

$$e_{vel} = [-0.1, +0.1]m/s \tag{20}$$

$$\sigma_{\alpha} = 2^{\circ} \tag{21}$$

7.1.1. Discrete agent

In this sub-section the results of the sensitivity analysis are generated starting from a discrete action space agent, as the one defined in Brandonisio and Lavagna (2021). The motivation that drives the comparison between the discrete and continuous agent on this particular analysis, as will be later shown, is twofold: indeed, both the control and model robustness performances can be compared. The following results are focused on a target object shaped as a simple rectangular, exactly as the one defined in Brandonisio et al. (2021), and shown in Fig. 17.

The first analysis consisted in comparing the performance level (in terms of mapping level) of the trained agent with nominal input state and with noisy input state, testing it for 5000 episodes. In Fig. 6, the mapping level of the trained agent with noisy input in terms of relative position is shown. The histogram shows that the overall agent performance lowers down, but not dramatically, how, instead, it happens when the navigation errors are applied also to the relative velocity and attitude position, as shown in Fig. 7 and Fig. 8 respectively. In Fig. 9, the result derived

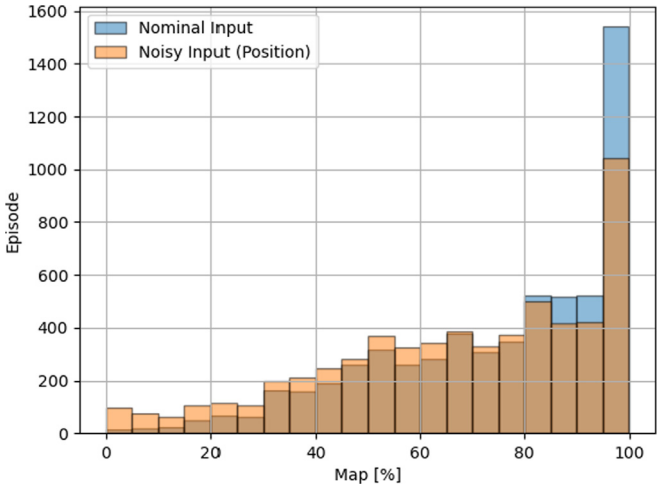

Fig. 6. Map level comparison between trained discrete agent with nominal input and with noisy chaser-target position input (rectangular object).

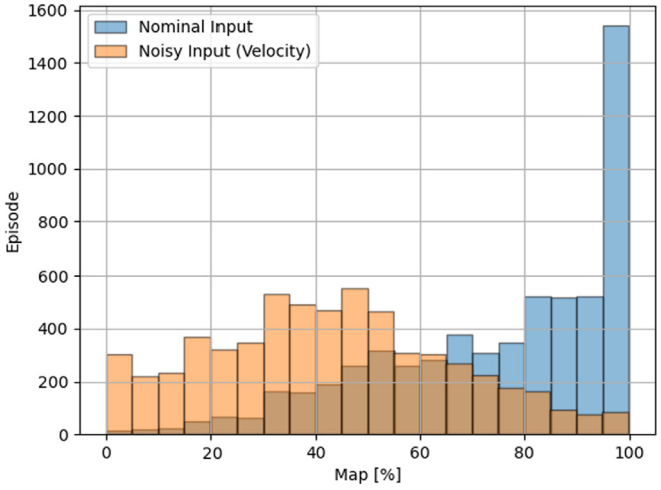

Fig. 7. Map level comparison between trained discrete agent with nominal input and with noisy chaser-target velocity input (rectangular object).

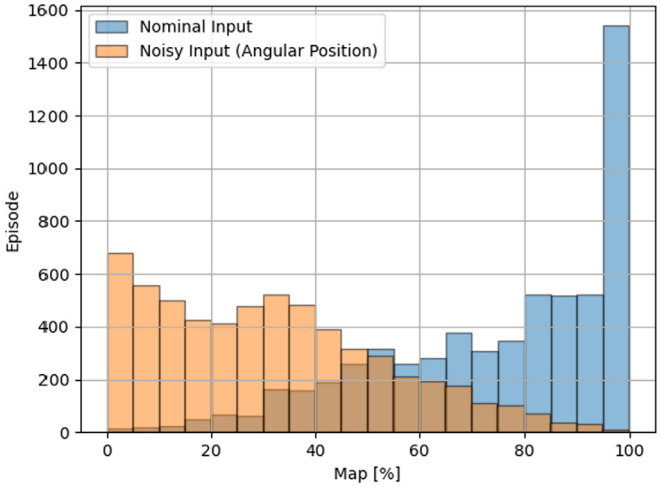

Fig. 8. Map level comparison between trained discrete agent with nominal input and with noisy chaser-target angular position input (rectangular object).

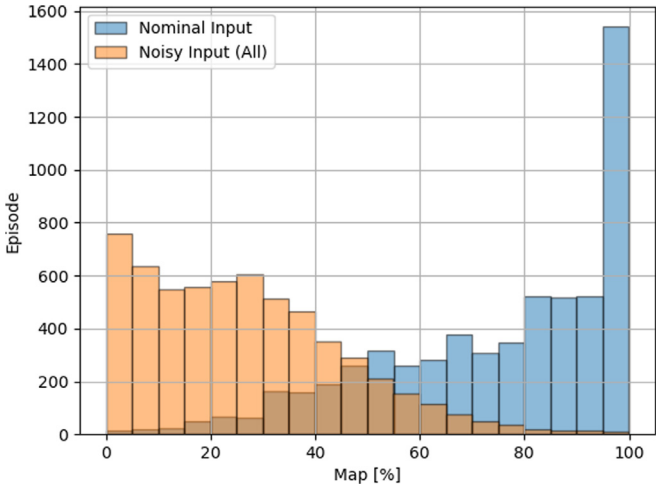

Fig. 9. Map level comparison between trained discrete agent with nominal input and with overall noisy input (rectangular object).

by testing the trained agent with all the noisy input state traces the trend of the worst error due to angular position uncertainty. As it is, the agent is completely useless for a real image-based navigation system.

To overcome this problem, the agent is re-trained again, but this time the erratic input information becomes part of the training simulation. The results can be appreciated in Fig. 10, where the training architecture model is confirmed robust to environment changes. Despite the problem becoming more complex, the training properties remained unchanged due to the higher and uncertain information regarding the state space.

To better understand the robustness properties of this discrete agent model, this analysis has been performed also for a different target object shape, namely the VESPA debris. In this way a direct line between the discrete and continuous action space agents is created.

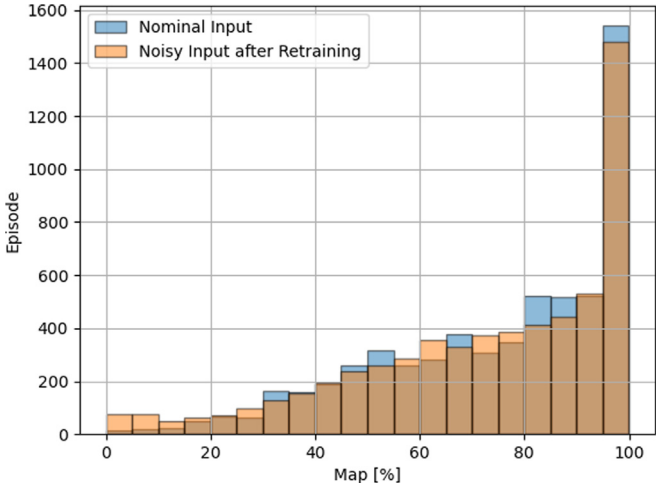

Fig. 10. Map level comparison between discrete agent with nominal input and re-trained agent with overall noisy input (rectangular object).

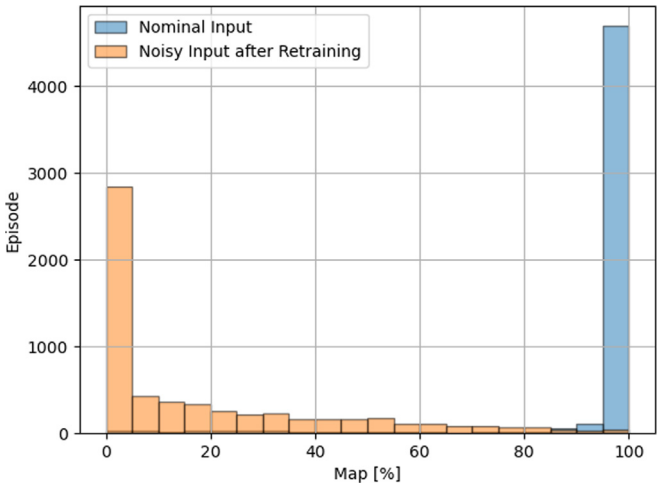

Fig. 11. Map level comparison between discrete agent with nominal input and and with overall noisy input (VESPA object).

Similarly to before, in Fig. 11 the comparison between the trained agent (this time on VESPA) fed by nominal and noisy inputs is shown. It is notable that, with this object, the performance really drops down, underlining how the efficiency of agent model is different in terms of objects or state errors sensitivity. Indeed, even if the trained agent performs better around VESPA than with the rectangular object, it is greatly more sensitive to noisy input. As observable in Fig. 12, the agent has been re-trained exactly as the one for the rectangular object, but in this case, the performance recovery is poorer. This demonstrates that at least for a discrete space agent the coupling between object and state uncertainty can not be ignored. A possible solution to this problem is briefly proposed in the *Future developments* section.

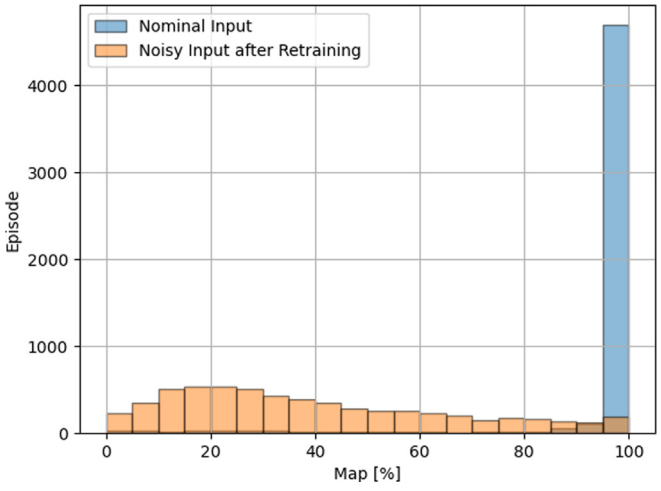

Fig. 12. Map level comparison between discrete agent with nominal input and re-trained agent with overall noisy input (VESPA object).

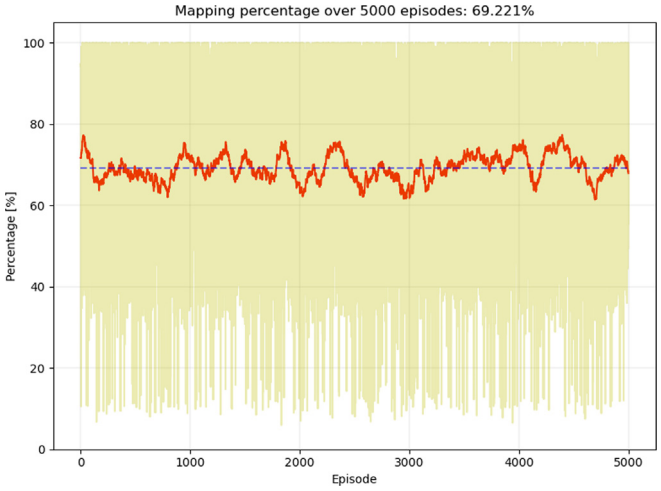

Fig. 13. Average map percentage obtained adding noise to the relative position and velocity, and angles measurements to continuous agent.

7.1.2. Continuous agent

In this sub-section, the state uncertainty sensitivity analysis for a continuous space agent is presented. Referencing the results obtained after the training shown in Section 6, the 95% average map level achieved by the agent is compared to the level reached when the input is affected by noise. In this case, the performance drops fairly, settling down at about 69% of average map level, as reported in

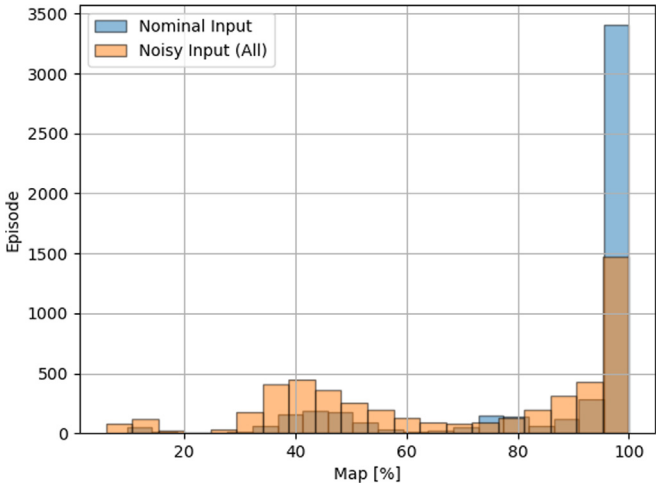

Fig. 14. Map level comparison between continuous agent with nominal input and with overall noisy input (VESPA object).

Table 5
Map percentage comparison applying differently navigation uncertainty.

Noisy state	Map level
Position + Velocity + Angles	69.22%
Position	84.07%
Velocity	87.58%
Angles	81.71%

Fig. 13. Analogously, in Fig. 14, the results are shown comparing the episodic map level in histogram.

The same test is performed applying distinctively the three uncertainties, first on the relative position, then on the velocity, and finally on the target attitude angles. Table 5 summarizes the results of all tests.

The results show how the model performance is deprecated, specifically in the case in which all three measurements are noisy. In particular, errors in the estimation of the target Euler angles seem to be the ones affecting the most the average map level, similarly to what happened for the discrete agent. This could be linked to the way the algorithm tries to reconstruct the shape of the target: indeed it needs to know very well how it is oriented to maneuver towards the faces that it still has to see. The same workflow used in the discrete case is followed also for this analysis, re-training the agent to investigate whether accounting for these errors during the training step leads to benefits in the overall autonomous guidance performance. As such, at each step of every episodic simulation, noise errors are added to the state with the characteristics previously defined in Eq. 19, Eq. 20 and Eq. 21. The result of the new training is reported in Fig. 15 and compared with the nominal input in Fig. 16.

Some key takeaways from the obtained result are here described:

- when the training starts, the average map level is around 50%, so practically the same as in the previous training in Fig. 4. Again, at the beginning, the agent selects its action in a random way;
- the profile of the average map percentage is similar to the one in the previous training, suggesting that, over the episodes, the learning pattern of an agent is repeated;
- the number of training episodes is increased to account for the additional complexity introduced by the presence of errors in the state. So, for this case, 40000 episodes have been simulated during the training;
- the average map level achieved at the end of the training step is around 85%. As expected, the performance level is lower with respect to the nominal case, but it is of great interest the fact that it outperformed the nominal model by about 16% in map percentage. The comparison is explicitly reported in Table 6.

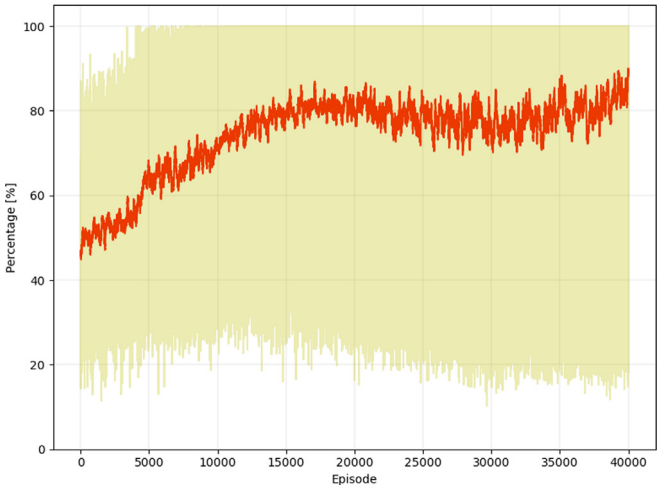

Fig. 15. Average map percentage profile during model retraining the continuous agent with noisy state.

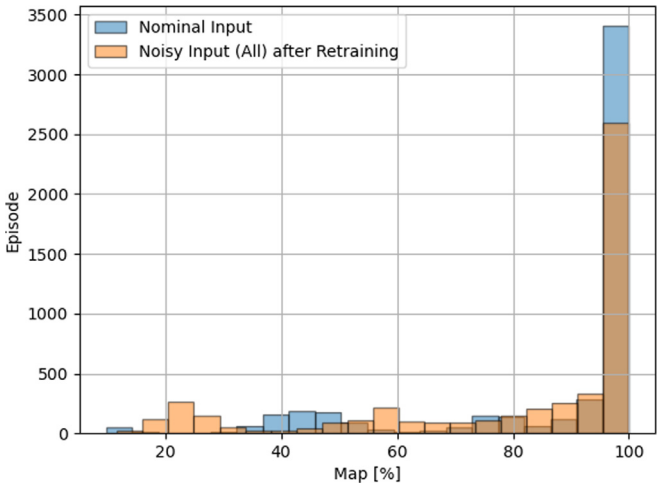

Fig. 16. Map level comparison between continuous agent with nominal input and re-trained agent with overall noisy input (VESPA object).

Table 6
Map percentage comparison between nominal and retrained model with state uncertainty.

Nominal training	Retraining
69.22%	~ 85%

Therefore, in general, it seems that the continuous action space agent is less sensitive to state input uncertainty or at least more prone to re-training on a different environment to improve its behavior. This is also confirmed by the

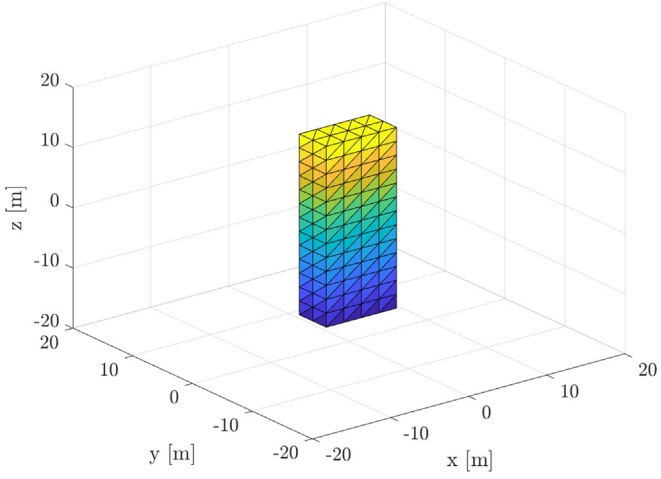

Fig. 17. Triangular mesh of parallelepiped object.

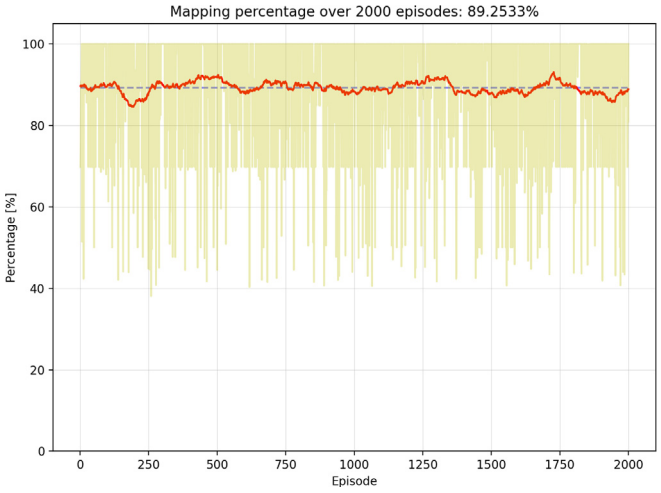

Fig. 18. Average map level with a rectangular object.

object sensitivity analysis where a simple test is performed changing the target object and checking how the agent responds, to evaluate its adaptability. The new target is a plain parallelepiped, as in the previous discrete case test, depicted in Fig. 17.

Indeed, the results are quite promising, reaching nearly 90% of average map level, as in Fig. 18. These results confirm the feasibility of having a continuous action space, representing a strong advancement with respect to previous works. Indeed, the advancement should not be focused on the achievement of an higher performance level, but instead, on the achievement of at least the same performance level of a problem that is much more simple, as the discrete action space's one, while retaining higher fidelity with respect to control thrusting authority.

7.1.3. Failure event

To conclude the testing phase on the agent performance, a further interesting case has been studied, focused on the robustness of the model with respect to a failure event. The test is structured as follows: at a random time step at the beginning of an episode, one of the 3D thrust vector components becomes suddenly unavailable and it is set to zero. For the sake of the simulation, a random time step in the range $0 - 100s$ is selected, and from there on the spacecraft is no more able to provide an action in every possible direction. Testing is performed on all directions simultaneously and on each axis, to see if one of them may be more sensitive than the others. The results are reported in Table 7.

On average the agent seems to lose $\sim 7\%$ in mapping percentage, which can be considered a satisfactory outcome, considering the performance level is still quite high. However, note that the largest drop occurs in correspondence of a failure event along the x-axis. This is not a casualty, since by propagating the free dynamics from random initial conditions, one discovers that the x-direction is the most sensitive one.

Table 7
Map percentage comparison considering a failure event in each directions.

All directions	X	Y	Z
88.46%	86.01%	89.37%	88.63%

8. Conclusion

In conclusion, the presented work deals with an adaptive guidance algorithm to autonomously reconstruct the geometry and shape of the selected uncooperative artificial object in space. The fly-around and mapping problem has been mathematically formulated as a Partially Observable Markov Decision Process and framed as an autonomous exploration active SLAM task. Relative dynamics and visibility models dictate the outcome of the simulations, describing how the chaser moves around the target and which surfaces are visible from the cameras, under the assumption that they are always pointing towards the target. A vision-based architecture is assumed, employing images both in the visible and infrared band, and processing them to obtain the object map via stereophotoclinometry. This step generates the essential inputs for the following guidance block, that have been implemented with a state-of-the-art Deep Reinforcement Learning algorithm to design the decision-making policy of the spacecraft agent. Proximal Policy Optimization in continuous action space is implemented, advancing from the previous related works, which exploited a discrete action space. Therefore, the agent can select any value for the action, under the form of a thrust vector, which can ideally point in any direction. The training of the PPO agent is performed, followed by an extensive testing campaign to assess the model robustness and sensitivity. A planning policy is obtained, capable of reconstructing on average 95% of the map, starting from random initial conditions. The testing campaign is then carried out, specifically analyzing the case of uncertainty in the state and how it affects the performance of the algorithm. A retraining procedure is proposed as a possible solution to improve the capabilities of the autonomous guidance agent in terms of average target mapping.

8.1. Future development

Several improvements can be made to the proposed method, and in this section, some interesting future developments are discussed. As outlined before, it is important to define a stable model capable of facing the state uncertainty problem. Here the retraining procedure is proposed as first step of the analysis, but it is assumed that improving the deep reinforcement learning model, introducing model-based network, may be very beneficial to this kind of issue. Future studies will be carried on in this direction. Moreover, one of the main drawback of the agent, in its current form, is its inability to consider the chaser attitude, assum-

ing that it is always pointing toward the target. A model of the chaser attitude dynamics would bring much closer to reality the discussed method, making sure that the map level during a single episode increases only when the chaser is correctly pointing toward the target, and is able to take pictures of only the faces that are inside the field of view of the cameras. From the RL side, this development could be managed in different ways, as for example:

- implementing the attitude dynamics and controlling it concurrently with the orbital dynamics, so that the same agent outputs the action vector containing both attitude and orbital controls;
- implementing the attitude dynamics and developing a multi-agent reinforcement learning methodology. In this case, one agent would be responsible for the orbital motion and the other for the attitude, meaning that both of them have to learn how to cooperate to reach the end goal of mapping the target object.

Moreover, this research is directed towards the coupling of AI-based navigation tool, which exploits simulated images as input to retrieve the pose, together with the here presented AI-based guidance and trajectory (eventually also attitude) control policy.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

In the person of Andrea Brandonisio, this publication resulted from research grant supported by *Fondazione Fratelli Confalonieri*, Milano.

References

Andriotis, C., Papakonstantinou, K., 2019. Managing engineering systems with large state and action spaces through deep reinforcement learning. Reliab. Eng. Syst. Saf. 191. https://doi.org/10.1016/j.ress.2019.04.036 106483.

Brandonisio, A., Lavagna, M., 2021. Sensitivity analysis of adaptive guidance via deep reinforcement learning for uncooperative space objects imaging. In: 2021 AAS/AIAA Astrodynamics Specialist Conference, Big Sky, Montana, USA, pp. 1–20.

Brandonisio, A., Lavagna, M., Guzzetti, D., 2021. Reinforcement learning for uncooperative space objects smart imaging path-planning. J. Astronaut. Sci. 68 (4), 1145–1169. https://doi.org/10.1007/s40295-021-00288-7.

Capra, L., Brandonisio, A., Lavagna, M., 2023. Network architecture and action space analysis for deep reinforcement learning towards spacecraft autonomous guidance. Adv. Space Res. 71 (9), 3787–3802. https://doi.org/10.1016/j.asr.2022.11.048.

Chan, D.M., Agha-mohammadi, A.-A., 2019. Autonomous imaging and mapping of small bodies using deep reinforcement learning. In: 2019 IEEE Aerospace Conference, Big Sky, Montana, USA, pp. 1–12, https://doi.org/10.1109/AERO.2019.8742147.

Civardi, G.L., Bechini, M., Quirino, M., et al., 2023. Generation of fused visible and thermal-infrared images for uncooperative spacecraft proximity navigation. Adv. Space Res. https://doi.org/10.1016/j.asr.2023.03.022.

Durrant-Whyte, H., Bailey, T., 2006. Simultaneous localization and mapping: part i. IEEE Robot. Automat. Mag. 13 (2), 99–110. https://doi.org/10.1109/MRA.2006.1638022.

Federici, L., Benedikter, B., Zavoli, A., 2021. Deep learning techniques for autonomous spacecraft guidance during proximity operations. J. Spacecraft Rock. 58 (6), 1774–1785. https://doi.org/10.2514/1.A35076.

Gaskell, R.W., 2001. Automated landmark identification for spacecraft navigation. Adv. Astronaut. Sci. 109, 1749–1756.

Gaudet, B., Linares, R., Furfaro, R., 2020a. Deep reinforcement learning for six degree-of-freedom planetary landing. Adv. Space Res. 65 (7), 1723–1741. https://doi.org/10.1016/j.asr.2019.12.030.

Gaudet, B., Linares, R., Furfaro, R., 2020b. Terminal adaptive guidance via reinforcement meta-learning: Applications to autonomous asteroid close-proximity operations. Acta Astronaut. 171, 1–13. https://doi.org/10.1016/j.actaastro.2020.02.036.

Inalhan, G., Tillerson, M., How, J., 2002. Relative dynamics and control of spacecraft formations in eccentric orbits. J. Guidance Control Dyn. 25 (1), 48–59. https://doi.org/10.2514/2.4874.

Joshua P., D., John P., M., Jay P., P., 2019. On-orbit servircing: Inspection, repair, refuel, upgrade, and assembly of satellites in space. Aerospace Corporation, URL: https://aerospace.org/sites/default/files/2019-05/Davis-Mayberry-Penn_OOS_04242019.pdf.

Martínez, J., Rafalskyi, D., Aanesland, A., 2019. Development and testing of the npt30-i2 iodine ion thruster. https://doi.org/10.6084/m9.figshare.11931363.

Mnih, V., Badia, A.P., Mirza, M. et al., 2016. Asynchronous methods for deep reinforcement learning. CoRR, abs/1602.01783. URL: http://arxiv.org/abs/1602.01783.

Pesce, V., Agha-mohammadi, A.-A., Lavagna, M., 2018. Autonomous navigation and mapping of small bodies. In: 2018 IEEE Aerospace Conference, Big Sky, Montana, USA, pp. 1–10, https://doi.org/10.1109/AERO.2018.8396797.

Piazza, M., Maestrini, M., Di Lizia, P., 2022. Monocular relative pose estimation pipeline for uncooperative resident space objects. J. Aerospace Informat. Syst. 19 (9), 613–632. https://doi.org/10.2514/1.I011064.

Piccinin, M., Lunghi, P., Lavagna, M., 2022. Deep reinforcement learning-based policy for autonomous imaging planning of small celestial bodies mapping. Aerosp. Sci. Technol. 120. https://doi.org/10.1016/j.ast.2021.107224 107224.

Placed, J.A., Strader, J., Carrillo, H. et al., 2023. A survey on active simultaneous localization and mapping: State of the art and new frontiers. arXiv:2207.00254.

Schulman, J., Levine, S., Abbeel, P. et al., 2015. Trust region policy optimization. In: Bach, F., Blei, D. (Eds.), Proceedings of the 32nd International Conference on Machine Learning, Lille, France: PMLR vol. 37 of Proceedings of Machine Learning Research, pp. 1889–1897. URL: https://proceedings.mlr.press/v37/schulman15.html.

Schulman, J., Wolski, F., Dhariwal, P. et al., 2017. Proximal policy optimization algorithms. arXiv:1707.06347.

Scorsoglio, A., Furfaro, R., Linares, R., et al., 2023. Relative motion guidance for near-rectilinear lunar orbits with path constraints via actor-critic reinforcement learning. Adv. Space Res. 71 (1), 316–335. https://doi.org/10.1016/j.asr.2022.08.002.

Silvestrini, S., Cassinis, L.P., Hinz, R., et al., 2023. Chapter fifteen - modern spacecraft gnc. In: Pesce, V., Colagrossi, A., Silvestrini, S. (Eds.), Modern Spacecraft Guidance, Navigation, and Control. Elsevier, pp. 819–981. https://doi.org/10.1016/B978-0-323-90916-7.00015-9.

Silvestrini, S., Lavagna, M., 2021a. Neural-aided gnc reconfiguration algorithm for distributed space system: development and pil test. Adv. Space Res. 67 (5), 1490–1505. https://doi.org/10.1016/j.asr.2020.12.014.

Silvestrini, S., Lavagna, M., 2021b. Neural-based predictive control for safe autonomous spacecraft relative maneuvers. J. Guidance Control Dyn. 44, 2303–2310. https://doi.org/10.2514/1.G005481.

Silvestrini, S., Piccinin, M., Zanotti, G., et al., 2022a. Optical navigation for lunar landing based on convolutional neural network crater detector. Aerosp. Sci. Technol. 123. https://doi.org/10.1016/j.ast.2022.107503 107503.

Silvestrini, S., Piccinin, M., Zanotti, G., et al., 2022b. Implicit extended kalman filter for optical terrain relative navigation using delayed measurements. Aerospace 9 (9). https://doi.org/10.3390/aerospace9090503.

Silvestrini, S., Prinetto, J., Zanotti, G. et al., 2020. Design of robust passively safe relative trajectories for uncooperative debris imaging in preparation to removal. In: 2020 AAS/AIAA Astrodynamics Specialist Conference. Lake Tahoe, California, USA.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction, 2nd ed. MIT Press, Cambridge, Massachusetts, USA.

Xie, S., Zhang, Z., Yu, H., et al., 2023. Recurrent prediction model for partially observable mdps. Inf. Sci. 620, 125–141. https://doi.org/10.1016/j.ins.2022.11.065.

Yang, L., Shami, A., 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing 415, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061.