Acta Astronautica 172 (2020) 90–99

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Six degree-of-freedom body-fixed hovering over unmapped asteroids via LIDAR altimetry and reinforcement meta-learning

Brian Gaudet[a], Richard Linares[b], Roberto Furfaro[c,*]

[a] Deep AnalytX LLC, Tucson, AZ, USA
[b] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, USA
[c] Department of Systems and Industrial Engineering, Department of Aerospace and Mechanical Engineering, University of Arizona, USA

ARTICLE INFO

Presented at the AIAA Scitech 2020 Forum

Keywords:
Reinforcement learning
Asteroid missions
Hovering artificial intelligence
Autonomous maneuvers

ABSTRACT

We optimize a six degrees of freedom hovering policy using reinforcement meta-learning. The policy maps flash LIDAR measurements directly to on/off spacecraft body-frame thrust commands, allowing hovering at a fixed position and attitude in the asteroid body-fixed reference frame. Importantly, the policy does not require position and velocity estimates, and can operate in environments with unknown dynamics, and without an asteroid shape model or navigation aids. Indeed, during optimization the agent is confronted with a new randomly generated asteroid for each episode, insuring that it does not learn an asteroid's shape, texture, or environmental dynamics. This allows the deployed policy to generalize well to novel asteroid characteristics, which we demonstrate in our experiments. Moreover, our experiments show that the optimized policy adapts to actuator failure and sensor noise. Although the policy is optimized using randomly generated synthetic asteroids, it is tested on two shape models from actual asteroids: Bennu and Itokawa. We find that the policy generalizes well to these shape models. The hovering controller has the potential to simplify mission planning by allowing asteroid body-fixed hovering immediately upon the spacecraft's arrival to an asteroid. This in turn simplifies shape model generation and allows resource mapping via remote sensing immediately upon arrival at the target asteroid.

1. Introduction

Recently there has been increased interest in robotic missions to near Earth asteroids, for both scientific and commercial purposes. The prevalent concept of operations requires complete characterization of the asteroid's shape and dynamics prior to a sample return maneuver. Before a shape model can even be generated, the environmental dynamics must be characterized to a high degree of accuracy in order to allow calculation of stable orbits from which shape model generation takes place [1]. Moreover, these stable orbits are in general only possible over a limited range of latitudes [2]. Asteroid body-fixed hovering at arbitrary locations in proximity to the asteroid has the potential to simplify mission planning, allowing high resolution sensor measurements at arbitrary locations [3]. Hovering in the inertial frame with the asteroid rotating below the spacecraft is possible at arbitrary altitudes (within the limits of terrain hazards) and in general will require less fuel than hovering in the asteroid body-fixed frame. Although hovering in the asteroid body-fixed frame requires more fuel expenditure and cannot be performed at large distances from the asteroid, body frame hovering has the advantage of allowing multiple sensor measurements from a fixed position with respect to the asteroid. Finally, body fixed hovering close to the surface would allow a spacecraft to drill or collect surface samples while compensating for the force induced by the manipulators. Clearly, both types of hovering would be useful for asteroid missions.

Previous work in hovering in close proximity to asteroids includes [4], where the authors develop a hovering controller that uses altimetry measurements to hover in the asteroid body-fixed frame. Their work uses a single altimeter and thrusting direction, but assumes the sensor is aligned with the gravitational acceleration at the hovering point, the altitude is below the resonance radius (the altitude where gravitational and centrifugal forces cancel), and that the centrifugal force components perpendicular to thrust direction are known. Furfaro develops a 3-DOF hovering controller using sliding mode control theory [5]. In other work [6] Gaudet and Furfaro demonstrate both hovering and TAG maneuvers using a Rao-Blackwellized particle filter to infer the spacecraft's position and velocity using altimetry measurements and an asteroid shape model. Lee et al. demonstrates 6-DOF hovering using a control law developed in the Lie group SE(3) [7], but their method requires an estimate of the environmental dynamics. Gaudet and

* Corresponding author.
E-mail address: robertof@email.arizona.edu (R. Furfaro).

https://doi.org/10.1016/j.actaastro.2020.03.026
Received 21 February 2020; Accepted 17 March 2020
Available online 29 March 2020

Furfaro developed a 3-DOF hovering controller using reinforcement learning [8] that showed improved transient response as compared to an LQR controller. Importantly, previous work does not cover the case where the spacecraft arrives at an asteroid and we want the spacecraft to be able to immediately hover in the body-fixed frame in the case where 1.) there is no knowledge of the environmental dynamics and 2.) there is not an existing shape model that can be used by a navigation system to infer the spacecraft's position and velocity.

Inertial hovering has been successfully executed in both Hayabusa missions. The most recent, Hayabusa 2 [9] arrived at the asteroid Ryugu in June 2018 and, after a sequence of close proximity operations including asteroid mapping and surface's touchdown and sample collection, departed the celestial body in November 2019. It is expected to deliver the sample to Earth in late 2020. One of the major modes of operation included the ability of the spacecraft to hover at different altitudes for either surface mapping and/or in preparation for the touchdown sequence. The spacecraft employs a combination of Reaction Wheels (RW) and RCS thrusters to control attitude and position. A wide angle camera called ONC-W [10] is employed for navigation purposes. The camera is coupled with a dedicated image processor. Navigation has two major modes: the Asteroid Image Tracking (AIT) mode which calculates the image center of the asteroid Ryugu when in the Field of View (FOV). Conversely, in the Target Marker Mode (TMT) mode, the ONC-W tracks a target marker previously deployed on the asteroid surface [11]. A LIDAR system is employed to measure the spacecraft altitude [12]. The latter is generally used at distance larger than 50 m. For lower altitudes (5–50 m), a Laser Range Finder (LRF) is employed. In a home position of about 20 km, hovering is executed by a Ground Control Point Navigation (GCP-NAV) which employs the AIT mode [11]. Indeed the ONC-W sends images to the ground every 10 min. A ground operator manually overlay the asteroids estimated shape and GCPs to the image to estimate the spacecraft position. Subsequently, the spacecraft position is propagated forward to account for the communication time delay. Eventually, the required delta-V is uploaded to the spacecraft for timed execution. Once the spacecraft is hovering below 50 m, the TMT mode is executed by a combination of ground and on-board operations. In this phase, the position of a predeployed surface marker (reflector) is autonomously computed onboard. At this stage, hovering is controlled in a 6-DOF fashion using attitude and navigation information. Hovering generally occurred above the marker. For the final descent, although the team had originally planned to hover at 25 m altitude, flight data showed that the hovering occurred at an altitude of 8.5 m [13].

In this work we focus on the body-fixed hovering problem where neither a shape model nor information about the environmental dynamics are available. Without a shape model, which allows a navigation system to infer the spacecraft's position in the asteroid body-fixed frame, body-fixed hovering is a challenging problem that to our knowledge has not yet been solved. The chief difficulty is that as the asteroid rotates it induces hovering position errors, and the hovering policy must learn how to correct for these errors by observing the changing LIDAR altimetry readings and use its recollection of these changing sensor readings to correct the hovering position error. The problem is further complicated by pulsed thrusters, which will likely cause an overshoot with corrective thrust commands.

The goal is to remain at a constant asteroid body-fixed position and attitude from the commencement of the hovering maneuver. We will assume that the spacecraft is equipped with a flash LIDAR system, gyroscopes that can measure the change in the spacecraft's attitude from the initiation of the hovering maneuver, and rate gyros that measure rotational velocity. We further assume that these sensors can provide measurements every 6s. At the start of the hovering maneuver, the spacecraft is pointed in the general direction of the asteroid, and consequently at least some of the flash LIDAR elements can return valid altimeter readings. A possible concept of operations would be for the spacecraft to slowly approach the asteroid using a navigation system that keeps the asteroid centered in a camera's field of view, and then commence hovering when the mean range of the flash LIDAR elements indicates an acceptable hovering altitude. What happens next is mission specific, potential low altitude scenarios include the spacecraft hovering close to the surface to release a beacon or rover, or collect samples. Potential high altitude hovering scenarios include shape model generation (where the ability to take multiple readings from the same position should simplify simultaneous location and mapping), remote sensing, as well as tagging the landing site with a targeting laser to facilitate a precision landing by a separate lander, as described in Ref. [14].

Our hovering controller is optimized using reinforcement learning (RL), which learns a policy that maps sensor measurements directly to on/off thrust commands, and that can adapt both to unknown environmental dynamics and novel asteroid shapes and textures. The policy is learned through simulated interaction between an environment and an agent instantiating the policy. Adaptability is achieved through RL-Meta Learning (Meta-RL) [15–17], where different asteroid shapes and environmental dynamics are treated as a range of partially observable Markov decision processes (POMDP). In each POMDP, the policy's recurrent network hidden state evolves over the course of an episode based off of the history of observations and actions, capturing information about hidden variables that are useful in minimizing the cost function; these include asteroid shape, texture, environmental dynamics, and changes in the spacecraft's internal dynamics. By optimizing the policy over this range of POMDPs, the trained policy will be able to adapt in real time to novel POMDPs encountered during deployment. Specifically, even though the policy's parameters are fixed after optimization, the policy's hidden state will evolve based off the history of observations and actions experienced in the current POMDP, thus adapting to the environment. We have demonstrated the effectiveness of RL meta-learning to create adaptive policies for aerospace applications in previous work [14,18,19]. In this work our goal is for the agent to hover at a position within 2 m of its position at the start of the hovering maneuver, with constant attitude, and fuel expenditure minimized during hovering. Importantly, the optimized policy will be general in that it will allow hovering over *any* asteroid with arbitrary shape, rotation, and density, provided the size is reasonably close to that of the synthetic asteroids used for optimization, and within the limits of thruster capability. To achieve this, the agent learns the policy in an environment that generates a new random asteroid for each episode.

The optimized policy serves as an integrated guidance, navigation, and control system for the purposes of a hovering maneuver, and interfaces with peripheral spacecraft systems as shown below in Fig. 1.

2. Problem formulation

2.1. Spacecraft configuration

The spacecraft is modeled as a uniform density cube with height $h = 2$, width $w = 2$, and depth $d = 2$, with inertia matrix given in Eq. (1). The spacecraft has a wet mass ranging from 450 to 500 kg. The thruster configuration is shown in Table 1, where x, y, and z are the body frame axes. Roll is about the x-axis, yaw is about the z-axis, and pitch is about the y-axis. Firing both thrusters on a face give translational thrust without rotation, while firing a single thruster on a given face induces a torque. The navigation system provides updates to the guidance system every 6 s, and we integrate the equations of motion using fourth order Runge-Kutta integration with a time step of 2 s. Thrusters have a specific impulse of 210s.

$$\mathbf{J} = \frac{m}{12}\begin{bmatrix} h^2 + d^2 & 0 & 0 \\ 0 & w^2 + d^2 & 0 \\ 0 & 0 & w^2 + h^2 \end{bmatrix} \tag{1}$$

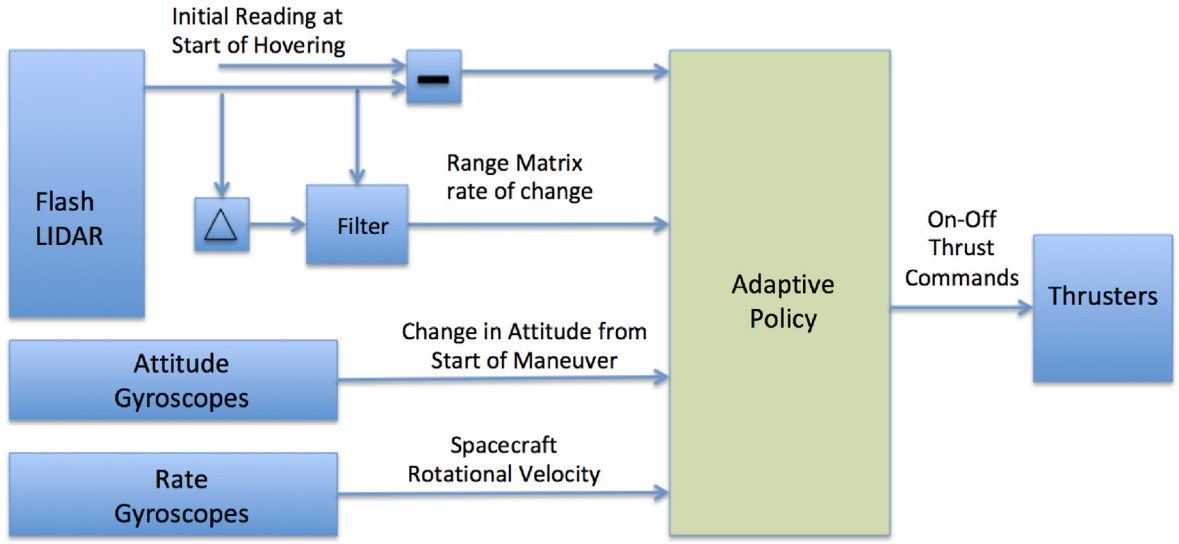

Fig. 1. Deployed policy interface with peripheral systems.

Table 1
Body frame thruster locations.

Thruster	x (m)	y (m)	z (m)	Thrust
1	−1.0	0.0	0.4	1.0
2	−1.0	0.0	−0.4	1.0
3	1.0	0.0	0.4	1.0
4	1.0	0.0	−0.4	1.0
5	−0.4	−1.0	0.0	1.0
6	0.4	−1.0	0.0	1.0
7	−0.4	1.0	0.0	1.0
8	0.4	1.0	0.0	1.0
9	0.0	−0.4	−1.0	1.0
10	0.0	0.4	−1.0	1.0
11	0.0	−0.4	1.0	1.0
12	0.0	0.4	1.0	1.0

m is the spacecraft's mass, which is updated as shown in Eq. (6c).

2.2. Asteroid and sensor models

Since our goal is for the agent to hover above an asteroid with unknown shape, we need to insure that the agent does not learn the asteroid's shape during optimization. To this end, we randomly generate a new asteroid for each episode. Each asteroid starts as an icosahedron based on the unit sphere, after which we recursively (twice) expand each face into four equal triangles, with the new vertices projected onto the unit sphere. In the following, we will refer to this object as an "isosphere". Next, we randomly perturb each vertex of the unit isosphere by adding a value $\mathbf{p} \in \mathbb{R}^3$, where each element of $\mathbf{p}$ is uniformly drawn over the range $[-p, p]$, with p uniformly drawn at the start of each episode from the range $[0.005, 0.05]$. Thus, different episodes will feature asteroids with different textures. We then randomly generate the asteroid's positive and negative a, b, and c axes over the range 300–600 m, and then scale the vertices appropriately. Since the positive and negative axes values are independently generated, this creates asymmetric a, b, and c axes. A sample randomly generated asteroid with 1280 faces and 642 vertices is shown below in Fig. 2.

For the modeling of environmental dynamics, we model the asteroid as an ellipsoid with uniform density. We assume that the asteroid is in general not rotating about a principal axis, and therefore to calculate the angular velocity vector we must specify the spin rate, the nutation angle (angle between the asteroid's z-axis and the axis of rotation), and moments of inertia [20]. The moments of inertia in turn depend on the asteroid's density and dimensions. The dimensions are specified by the

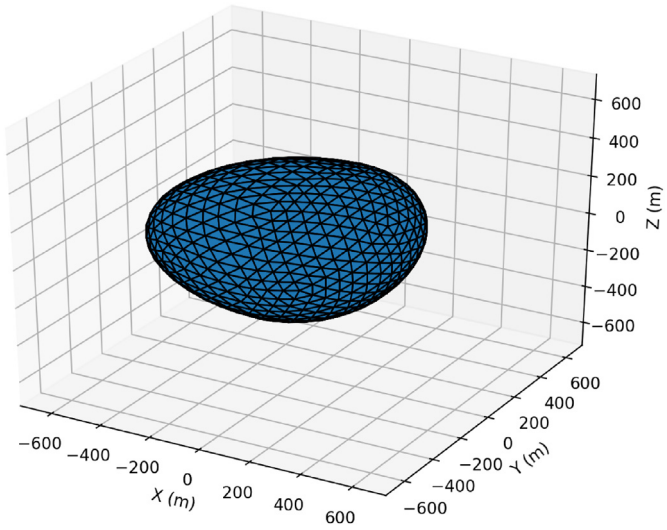

Fig. 2. Sample random asteroid.

ellipsoid axes $a = b \neq c$, where the axis constraints significantly simplifies the equations of motion. Since for the random asteroid the a and b axes are asymmetric and not in general equal, we average them for purposes of calculating the asteroid's rotational velocity components. We use a gravity model that assumes a uniformly distributed sphere. Although an ellipsoid model would have been more accurate, it would also be more computationally expensive, and with the asteroid sizes considered in this work, rotational forces dominate the dynamics.

The flash LIDAR is modeled as an 8 × 8 array of range sensors. Current commercial flash LIDAR units typically have a 100 × 100 sensor array, but the smaller sensor array allows much faster computation during optimization, and it seems intuitive that if hovering performance is satisfactory with an 8 × 8 array, it would likely be even better with a 100 × 100 array. In the unlikely event that this is not the case, an 8 × 8 array can be derived from a 100 × 100 array by downsampling. We use the Moller-Trumbore ray casting algorithm [21] to compute the intersection of each LIDAR beam with a triangle in the randomly generated asteroid. Our implementation only returns a range if a triangle is intersected on the correct side, and the intersection point is not occluded by another triangle. If a beam fails to intersect a triangle, a max range reading of 2000 m is returned.

We also assume the spacecraft is equipped with a gyroscope that can

measure the spacecraft's change in attitude (measured from the initiation of hovering), and a rate gyroscope to measure rotational velocity. At the beginning of an episode, the spacecraft's flash LIDAR is pointed in the same direction as the spacecraft's Z-axis. For the remainder of the episode, the LIDAR system is stabilized, i.e., we keep the LIDAR system's attitude constant during the maneuver as the spacecraft's attitude changes. Stabilization helps the hovering policy differentiate between changes in altimetry readings caused by rotation and changes caused by translation. In an actual implementation, this stabilization could be achieved by physically rotating the LIDAR platform to account for the spacecraft's change in attitude, similar to missile seeker stabilization [22]. We used a similar stabilization scheme for our work with seeker based guidance for asteroid close proximity operations [14].

2.3. Equations of motion

The force $\mathbf{F}_B$ and torque $\mathbf{L}_B$ in the lander's body frame for a given commanded thrust depends on the placement of the thrusters in the lander structure. We can describe the placement of each thruster through a body-frame direction vector $\mathbf{d}$ and position vector $\mathbf{r}$, both in $\mathbb{R}^3$. The direction vector is a unit vector giving the direction of the body frame force that results when the thruster is fired. The position vector gives the body frame location with respect to the center of mass, where the force resulting from the thruster firing is applied for purposes of computing torque, and in general the center of mass varies with time as fuel is consumed. For a lander with k thrusters, the body frame force and torque associated with one or more thrusters firing is then as shown in Equations (2a) and (2b), where $T_{cmd_i} \in [T_{min}, T_{max}]$ is the commanded thrust for thruster i, T_{min} and T_{max} are a thruster's minimum and maximum thrust, $\mathbf{d}^{(i)}$ the direction vector for thruster i, and $\mathbf{r}^{(i)}$ the position of thruster i. The total body frame force and torque are calculated by summing the individual forces and torques.

$$\mathbf{F}_B = \sum_{i=1}^{k} \mathbf{d}^{(i)} T_{cmd}^{(i)} \tag{2a}$$

$$\mathbf{L}_B = \sum_{i=1}^{k} (\mathbf{r}^{(i)} - \mathbf{r}_{\text{com}}) \times \mathbf{F}_B^{(i)} \tag{2b}$$

The dynamics model uses the lander's current attitude $\mathbf{q}$ to convert the body frame thrust vector to the inertial frame as shown in Equation (3) where $[\mathbf{BN}](\mathbf{q})$ is the direction cosine matrix mapping the inertial frame to body frame obtained from the current attitude parameter $\mathbf{q}$.

$$\mathbf{F}_N = [[\mathbf{BN}](\mathbf{q})]^T \mathbf{F}_B \tag{3}$$

The rotational velocities $\boldsymbol{\omega}_{B/N}$ are then obtained by integrating the Euler rotational equations of motion, as shown in Equation (4), where $\mathbf{L}_B$ is the body frame torque as given in Equation (2b), $\mathbf{L}_{env}$ is the body frame torque from external disturbances, and $\mathbf{J}$ is the lander's inertia tensor. Note we have included a term that models a rotation induced by a changing inertia tensor.

$$\mathbf{J}\dot{\boldsymbol{\omega}}_B = -\tilde{\boldsymbol{\omega}}_B \mathbf{J}\boldsymbol{\omega}_B - \dot{\mathbf{J}}\boldsymbol{\omega} + \mathbf{L}_B + \mathbf{L}_{B_{env}} \tag{4}$$

The lander's attitude is then updated by integrating the differential kinematic equations shown in Equation (5), where the lander's attitude is parameterized using the quaternion representation and ω_i denotes the i^{th} component of the rotational velocity vector $\boldsymbol{\omega}_B$.

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ \omega_0 \\ \omega_1 \\ \omega_2 \end{bmatrix} \tag{5}$$

The translational motion is modeled as shown in 6a through 6c.

$$\dot{\mathbf{r}} = \mathbf{v} \tag{6a}$$

$$\dot{\mathbf{v}} = \frac{\mathbf{F}_N}{m} + \mathbf{a}_{\text{env}} - g(\mathbf{r}, M) + 2\dot{\mathbf{r}} \times \boldsymbol{\omega}_a + (\boldsymbol{\omega}_a \times \mathbf{r}) \times \boldsymbol{\omega}_a \tag{6b}$$

$$\dot{m} = -\frac{\sum_i^k ||\mathbf{F}_B^{(i)}||}{I_{\text{sp}} g_{\text{ref}}} \tag{6c}$$

Here $\mathbf{F}_N^{(i)}$ is the inertial frame force as given in Eq. (3), k is the number of thrusters, $g_{\text{ref}} = 9.8$ m/s^2, $\mathbf{r}$ is the spacecraft's position in the asteroid centered reference frame, $g(\mathbf{r}, M)$ is a spherical gravity model and M is the asteroid's mass, $I_{\text{sp}} = 225$ s, and the spacecraft's mass is m. $\mathbf{a}_{\text{env}}$ is a vector representing solar radiation pressure. $\boldsymbol{\omega}_a$ is the asteroid's rotational velocity vector, which we compute as shown in Equation (7a) through (7f), which uses the simplifying assumption that $J_x = J_y$ [4]. Here ω_o is the asteroid's spin rate and θ the nutation angle between the asteroid's spin axis and z-axis. We modified the equations from Reference ([4]) to add the phase term φ to handle the case where the spacecraft starts the maneuver at an arbitrary point in the asteroid's rotational cycle.

$$\omega_{a_x} = \omega_o \sin\theta \cos(\omega_n t + \varphi) \tag{7a}$$

$$\omega_{a_y} = \omega_o \sin\theta \sin(\omega_n t + \varphi) \tag{7b}$$

$$\omega_{a_z} = \omega_o \cos\theta \tag{7c}$$

$$\omega_n = \sigma\omega_o \cos\theta \tag{7d}$$

$$J_{x,y}/J_z = (b^2 + c^2)/(a^2 + b^2) \tag{7e}$$

$$\sigma = \frac{(J_z - J_x)}{J_x} \tag{7f}$$

3. Guidance law development

3.1. RL overview

In the RL framework, an agent learns through episodic interaction with an environment how to successfully complete a task by learning a policy that maps observations to actions. The environment initializes an episode by randomly generating a ground truth state, mapping this state to an observation, and passing the observation to the agent. These observations could be a corrupted version of the ground truth state (to model sensor noise) or could be raw sensor outputs such as Doppler radar altimeter readings, a multi-channel pixel map from an electro-optical sensor, or in our case, a flash LIDAR range matrix. The agent's policy uses this observation to generate an action that is sent to the environment; the environment then uses the action and the current ground truth state to generate the next state and a scalar reward signal. The reward and the observation corresponding to the next state are then passed back to the agent. The process repeats until the environment terminates the episode, with the termination signaled to the agent via a done signal. Possible termination conditions include the agent completing the task, satisfying some condition on the ground truth state (such as altitude falling below zero), or violating a constraint.

A Markov Decision Process (MDP) is an abstraction of the environment, which in a continuous state and action space, can be represented by a state space $\mathscr{S}$, an action space $\mathscr{A}$, a state transition distribution $\mathscr{P}(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, and a reward function $r = \mathscr{R}(\mathbf{x}, \mathbf{u})$), where $\mathbf{x} \in \mathscr{S}$ and $\mathbf{u} \in \mathscr{A}$, and r is a scalar reward signal. We can also define a partially observable MDP (POMDP), where the state $\mathbf{x}$ becomes a hidden state, generating an observation $\mathbf{o}$ using an observation function $\mathscr{O}(\mathbf{x})$ that maps states to observations. The POMDP formulation is useful when the observation consists of raw sensor outputs, as is the case in this work. In the following, we will refer to both fully observable and partially observable environments as POMDPs, as an MDP can be considered a POMDP with an identity function mapping states to observations.

The agent operates within an environment defined by the POMDP,

generating some action $\mathbf{u}_t$ based off of the observation $\mathbf{o}_t$, and receiving reward r_{t+1} and next observation $\mathbf{o}_{t+1}$. Optimization involves maximizing the sum of (potentially discounted) rewards over the trajectories induced by the interaction between the agent and environment. Constraints such as minimum and maximum thrust, glide slope, attitude compatible with sensor field of view, maximum rotational velocity, and terrain feature avoidance (such as targeting the bottom of a deep crater) can be included in the reward function, and will be accounted for when the policy is optimized. Note that there is no guarantee on the optimality of trajectories induced by the policy, although in practice it is possible to get close to optimal performance by tuning the reward function [23].

Reinforcement meta-learning differs from generic reinforcement learning in that the agent learns to quickly adapt to novel POMDPs by learning over a wide range of POMDPs. These POMDPs can include different environmental dynamics, actuator failure scenarios, mass and inertia tensor variation, and varying amounts of sensor distortion. Learning within the RL meta-learning framework results in an agent that can quickly adapt to novel POMDPs, often with just a few steps of interaction with the environment. There are multiple approaches to implementing meta-RL. In Ref. [24], the authors design the objective function to explicitly make the model parameters transfer well to new tasks, whereas in Ref. [15] the authors demonstrate state of the art performance using temporal convolutions with soft attention. And in Refs. [16], the authors use a hierarchy of policies to achieve meta-RL. In this proposal, we use a different approach [17] using a recurrent policy and value function. Note that it is possible to train over a wide range of POMDPs using a non-meta RL algorithm [23,25]. Although such an approach typically results in a robust policy, the policy cannot adapt in real time to novel environments.

In this work, we implement metal-RL using proximal policy optimization (PPO) [26] with both the policy and value function implementing recurrent layers in their networks. To understand how recurrent layers result in an adaptive agent, consider that given some ground truth agent state $\mathbf{x}_t$ and action vector $\mathbf{u}_t$ output by the agent's policy, the next state $\mathbf{x}_{t+1}$ and observation $\mathbf{o}_{t+1}$ depends not only on $\mathbf{x}_t$ and $\mathbf{u}_t$, but also on the ground truth agent mass, inertia tensor, and external forces acting on the agent, as well as the asteroid's shape. Specifically, during optimization, the hidden state of a network's recurrent network evolves differently depending on the observed sequence of observations from the environment and actions output by the policy, with the state evolution capturing unobserved and potentially time-varying information, such as external forces, that are useful in minimizing the cost function. In contrast, a non-recurrent policy, which does not maintain a persistent hidden state vector, can only optimize using a set of current observations, actions, and advantages, and will tend to under-perform a recurrent policy on tasks with randomized dynamics [18]. After training, although the recurrent policy's network weights are frozen, the hidden state will continue to evolve in response to a sequence of observations and actions, thus making the policy adaptive. In contrast, a policy without a recurrent network layer has behavior that is fixed by the network parameters at test time.

The PPO algorithm used in this work is a policy gradient algorithm which has demonstrated state-of-the-art performance for many RL benchmark problems. PPO approximates the Trust Region Policy Optimization (TRPO) process [27] by accounting for the policy adjustment constraint with a clipped objective function. The objective function used with PPO can be expressed in terms of the probability ratio $p_k(\theta)$ given by Eq. (8), where π_θ is the policy parameterized by parameter vector θ.

$$p_k(\theta) = \frac{\pi_\theta(\mathbf{u}_k|\mathbf{o}_k)}{\pi_{\theta_{\text{old}}}(\mathbf{u}_k|\mathbf{o}_k)} \tag{8}$$

The PPO objective function is shown in Equations (9a) through (9c). The general idea is to create two surrogate objectives, the first being the probability ratio $p_k(\theta)$ multiplied by the advantages $A^\pi_{\mathbf{w}}(\mathbf{o}_k, \mathbf{u}_k)$ (see Eq. (10)), and the second a clipped (using clipping parameter ε) version of $p_k(\theta)$ multiplied by $A^\pi_{\mathbf{w}}(\mathbf{o}_k, \mathbf{u}_k)$. The objective to be maximized $J(\theta)$ is then the expectation under the trajectories induced by the policy of the lesser of these two surrogate objectives.

$$\text{obj1} = p_k(\theta) A^\pi_{\mathbf{w}}(\mathbf{o}_k, \mathbf{u}_k) \tag{9a}$$

$$\text{obj2} = \text{clip}(p_k(\theta) A^\pi_{\mathbf{w}}(\mathbf{o}_k, \mathbf{u}_k), 1-\varepsilon, 1+\varepsilon) \tag{9b}$$

$$J(\theta) = \mathbb{E}_{p(\tau)}[\min(\text{obj1}, \text{obj2})] \tag{9c}$$

This clipped objective function has been shown to maintain a bounded KL divergence with respect to the policy distributions between updates, which aids convergence by insuring that the policy does not change drastically between updates. Our implementation of PPO uses an approximation to the advantage function that is the difference between the empirical return and a state value function baseline, as shown in Equation (10):

$$A^\pi_{\mathbf{w}}(\mathbf{o}_k, \mathbf{u}_k) = \left[\sum_{\ell=k}^{T} \gamma^{\ell-k} r(\mathbf{o}_\ell, \mathbf{u}_\ell) \right] - V^\pi_{\mathbf{w}}(\mathbf{o}_k) \tag{10}$$

Here the value function $V^\pi_{\mathbf{w}}$ parameterized by vector $\mathbf{w}$ is learned using the cost function given by Eq. (11), where γ is a discount rate applied to rewards generated by reward function $\mathscr{R}(\mathbf{o}, \mathbf{u})$. The discounting of rewards improves optimization performance by improving temporal credit assignment.

$$L(\mathbf{w}) = \sum_{i=1}^{M} \left(V^\pi_{\mathbf{w}}(\mathbf{o}^i_k) - \left[\sum_{\ell=k}^{T} \gamma^{\ell-k} \mathscr{R}(\mathbf{u}^i_\ell, \mathbf{o}^i_\ell) \right] \right)^2 \tag{11}$$

In practice, policy gradient algorithms update the policy using a batch of trajectories (roll-outs) collected by interaction with the environment. Each trajectory is associated with a single episode, with a sample from a trajectory collected at step k consisting of observation $\mathbf{o}_k$, action $\mathbf{u}_k$, and reward $r_k = \mathscr{R}(\mathbf{o}_k, \mathbf{u}_k)$. Finally, gradient ascent is performed on θ and gradient descent on $\mathbf{w}$ and update equations are given by

$$\mathbf{w}^+ = \mathbf{w}^- - \beta_{\mathbf{w}} \nabla_{\mathbf{w}} L(\mathbf{w})|_{\mathbf{w}=\mathbf{w}^-} \tag{12}$$

$$\theta^+ = \theta^- + \beta_\theta \nabla_\theta J(\theta)|_{\theta=\theta^-} \tag{13}$$

where $\beta_{\mathbf{w}}$ and β_θ are the learning rates for the value function, $V^\pi_{\mathbf{w}}(\mathbf{o}_k)$, and policy, $\pi_\theta(\mathbf{u}_k|\mathbf{o}_k)$, respectively.

In our implementation, we dynamically adjust the clipping parameter ε to target a KL divergence between policy updates of 0.001. The policy and value function are learned concurrently, as the estimated value of a state is policy dependent. The policy uses a multi-categorical policy distribution, where a separate observation conditional categorical distribution is maintained for each element of the action vector. Note that exploration in this case is conditioned on the observation, with the two logits associated with each element of the action vector determining how peaked the softmax distribution becomes for each action. Because the log probabilities are calculated using the logits, the degree of exploration automatically adapts during learning such that the objective function is maximized. Finally, note that a full categorical distribution would be impractical, as the number of labels would be 2^{12}, as opposed to 2×12 for the multi-categorical distribution.

3.2. Guidance law optimization

A simplified view of the agent and environment are shown in Fig. 3. The environment instantiates the system dynamics model, asteroid shape model, reward function, spacecraft model, and thruster model. Note that when using a policy gradient method such as PPO it suffices to deploy the policy, and it is not necessary to deploy the value function. We can take advantage of this by giving the value function access

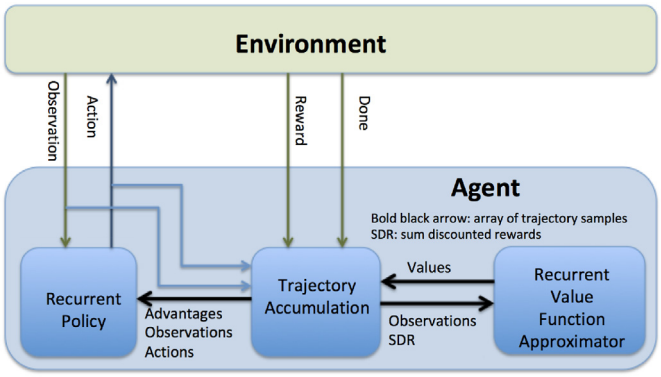

Fig. 3. Agent-environment interface.

to the ground truth state during optimization, whereas the policy only has access to the observations, in this case the flash LIDAR measurements. Specifically, the value function has access to the observation given in Eq. (14), where $\mathbf{r}_{\text{err}}$ and $\mathbf{dq}$ are the changes in the agent's position and attitude since the initiation of the hovering maneuver, $\mathbf{v}$ is the agent's velocity, and $\boldsymbol{\omega}$ the spacecraft's rotational velocity.

$$\text{obs}_{\text{VF}} = [\mathbf{r}_{\text{err}}\mathbf{v}\boldsymbol{dq}\boldsymbol{\omega}] \tag{14}$$

On the other hand, the policy only has access to the difference between the matrix of flash LIDAR readings at the current timestep and the readings at the start of the hovering maneuver $\mathbf{R}_{\text{err}}$, the change in LIDAR readings between consecutive measurements $\mathbf{dR}$, along with the estimated rotational velocity $\boldsymbol{\omega}$ and change in attitude since the start of the hovering maneuver $\boldsymbol{dq}$. Using $\mathbf{R}_{\text{err}}$ as opposed to the actual range matrix R allows the agent to generalize better to different altitude ranges. Note that in an actual implementation, $\mathbf{dR}$ would be smoothed with a Kalman filter. The observation given to the policy is then as shown in Eq. (15).

$$\text{obs}_{\pi} = [\mathbf{R}_{\text{err}}\mathbf{dR}\boldsymbol{dq}\boldsymbol{\omega}] \tag{15}$$

The action space is in $\mathbb{Z}^k$, where k is the number of thrusters. Each element of the agent action $\mathbf{u} \in 0{,}1$ is used to index Table 1, where if the action is 1, it is used to compute the body frame force and torque contributed by that thruster.

The value function is implemented using a four layer neural network with tanh activations on each hidden layer. Layer 2 for the value function network is a recurrent layer implemented as a gated recurrent unit [28]. The network architecture is as shown in Table 3, where n_{hi} is the number of units in layer i and obs_dim is the observation dimension (see Table 4).

The policy has a convolutional [29] front end with an architecture inspired by Ref. [30], where the pooling layer is replaced by a 2-D convolutional layer with stride 2. We have found that this improves performance for RL applications. We use rectified linear activations units for each convolutional layer. The first convolutional layer has 2 channels (one for the range readings, the other for the difference in range readings), 8 filters, a filter size of 3, and stride of 1. The second convolutional layer has 8 channels and 8 filters, a filter size of 4, and a stride of 2. The final layers are fully connected, as shown in 4. The entire policy network is diagrammed in Fig. 4. The policy and value functions are periodically updated during optimization after accumulating trajectory rollouts of 30 simulated episodes.

Table 2
Parameters for randomly generated asteroids.

Parameter	min	max
a-axis (m)	300	600
b-axis (m)	300	600
c-axis (m)	300	600
Mass M (kg)	1×10^{10}	150^{10}
Spin Rate ω_o (rad/s)	5×10^{-4}	1×10^{-6}
Nutation Angle (degrees)	45	90
Acceleration due to SRP m/s^2	-100×10^{-6}	100×10^{-6}

Table 3
Value Function network architecture.

Layer	# units	activation
hidden 1	$10 * \text{obs\_dim}$	tanh
hidden 2	$\sqrt{n_{\text{h1}} * n_{\text{h3}}}$	tanh
hidden 3	5	tanh
output	1	linear

Table 4
Fully Connected Policy network layers.

Layer	# units	activation
FC 1	70	tanh
FC 2	154	tanh
FC 3	120	tanh
FC 4	12	linear

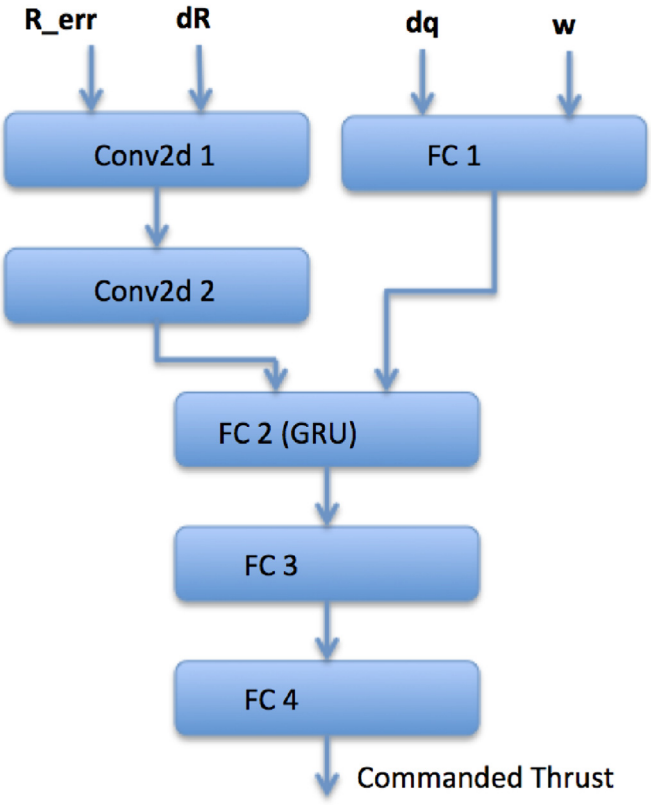

Fig. 4. Policy network.

During optimization, the agent is given negative rewards proportional to the cumulative change in position from the start of the hovering maneuver. Large negative rewards are given for exceeding a maximum rotational velocity of 0.10 rad/s or if the attitude is such that all of the flash LIDAR elements miss the asteroid, which is detected by all elements returning a max range reading of 2000 m. Constraint violation also results in the termination of the current episode. Small negative rewards are given proportional to the control effort at each timestep.

Finally, we provide a terminal reward bonus when the spacecraft executes a good landing (see below). The reward function is then given

by Equation (16), where the various terms are described in the following:

1. α weights a term penalizing the current deviation from desired hovering position.
2. β weights a term penalizing deviation from desired hovering attitude.
3. γ weights a term penalizing control effort.
4. η is a constant positive term that encourages the agent to keep making progress along the trajectory.
5. ζ is a bonus given for satisfying a terminal constraint at the end of the hovering maneuver, where the spacecraft's terminal position and velocity are all within specified limits. The limits are $||\mathbf{r}|| = 2$ m, $||\mathbf{v}|| = 0.1$, m/s, and all components of angular velocity less than 0.025 rad/s
6. κ is a penalty for exceeding any constraint. We impose a rotational velocity constraint of 0.10 rad/s for all three rotational axes. We also constrain the spacecraft's attitude such that at least one LIDAR beam hits the asteroid. If all beams miss the asteroid, we assume the attitude constraint is violated.

$$r = \alpha r_{err} + \beta q_{error} + \gamma||\mathbf{T}|| + \eta + \zeta(\text{terminal constraints satisfied}) + \kappa(\text{constraint violation}) \quad (16)$$

Initial hyperparameter settings are shown in Table 5.

4. Experiments

Code to reproduce these experiments will be made available on our Github page.[1] The spacecraft initial condition limits for these experiments were selected assuming that the spacecraft would start with its sensor pointed in the general direction of the asteroid with minimal residual translational and rotational velocities. The initial conditions used for our experiments are given in Table 6. Note that the initial range is with respect to the asteroid's surface given the line of sight to the asteroid center from the spacecraft's initial position, i.e., a range of 100 m implies an altitude of 100 m with respect to the asteroid's surface, regardless of the asteroid dimensions. Position θ and φ along with the initial range (plus the asteroid radius where it is collinear with the initial line of sight) specify the spacecraft's position in spherical coordinates in the asteroid centered reference frame. The spacecraft has a small uniformly distributed initial velocity.

The spacecraft's ideal initial attitude is such that the -Z body-frame axis is aligned with the line of sight to target. This ideal initial attitude is perturbed at the start of each episode such that the angle between the -Z body frame axis and line of sight to target varies uniformly as shown in Table 6. Lower hovering altitudes and larger asteroids both give rise to a scenario where most of the flash LIDAR beams give valid returns, and under these conditions the guidance algorithm can tolerate larger initial attitude errors than that shown in Table 6. Similarly, the guidance system can tolerate higher initial altitudes for smaller initial attitude errors and larger asteroids. At the start of each episode, a slight actuator failure is deemed to occur with probability 0.5. This actuator failure results in the thrust for a randomly chosen thruster to be reduced by a factor of 0.9.

4.1. Optimization results

We optimize using 30 episode rollouts and the initial conditions given in Table 6. To reduce computational requirements, we use an asteroid with only 320 facets for optimization. Each episode attempts to hover for 600s, but early termination is possible in the event of a constraint violation. For each episode, we randomly generate a new asteroid using parameters as given in Table 2. Fig. 5 plots reward statistics and Fig. 6 plots the terminal position error statistics, with statistics for both plots computed over rollout batch of 30 episodes. We see that initially the position error is high as the policy is focusing on satisfying the constraints that at least one element of the flash LIDAR sensor returns a valid reading and the maximum rotational velocity is not exceeded. Once the policy learns to satisfy the constraints, it focuses on minimizing the position error.

Table 5
Hyperparameter settings.

α	β	γ	η	ζ	κ
−0.02	−0.01	−0.05	0.01	10	−50

Table 6
Initial conditions.

Parameter	min	max
Range (m)	100.0	600.0
Position θ (degrees)	0.0	90.0
Position φ (degrees)	$-\pi$	π
x comp of Velocity (cm/s)	−10.0	10.0
y comp of Velocity (cm/s)	−10.0	10.0
z comp of Velocity (cm/s)	−10.0	10.0
Attitude Error (degrees)	0.0	11.0
x comp of Rotational Velocity (mrad/s)	−20.0	20.0
y comp of Rotational Velocity (mrad/s)	−20.0	20.0
z comp of Rotational Velocity (mrad/s)	−20.0	20.0

4.2. Policy testing: synthetic asteroids

We begin by testing the optimized policy on randomly generated synthetic asteroids, using the same initial conditions and asteroid parameters as in optimization. Note that unique scenarios are encountered in testing due to the random selection of asteroid and initial condition parameters. Test results are shown in Table 7, which are computed from 5000 simulated episodes. Note that the rotational velocity row in Table 7 gives the rotational velocity vector element with the worst performance (highest absolute value). The “Good Hover 1” row gives the percentage of episodes where the terminal position error was less than 2 m, terminal speed less than 10 cm/s, and all elements of the terminal rotational velocity less than 0.015 rad/s. The “Good Hover 2” row has the same terminal speed and rotational velocity constraints, but only requires the terminal position error to be less than 5 m. We achieved our terminal performance goals in all 5000 episodes. As a back of envelope calculation for the fuel required to hover, the mean force acting on the spacecraft (rotational and gravitational) was 0.5 N. Plugging in the spacecraft's specific impulse and 300 simulated steps over the 600s hovering duration, we find that 0.08 kg of fuel would be required to cancel the environmental forces. Our actual average fuel consumption of 0.41 kg is considerably larger. Part of the excess fuel consumption can be attributed to using pulsed thrusters, which do not allow exact cancellation of environmental forces. It is also possible we could have increased fuel efficiency by using a higher (absolute) value for the β coefficient in Table 5. A sample trajectory is shown in Fig. 7, where the position error subplot plots the deviation from the spacecraft's initial position. Note that when the environmental dynamics are such that maximum thrust is not required, the policy fires only a single thruster on a given side of the spacecraft, resulting in a 1 N thrust. This saves fuel at the expense of inducing rotation, which is compensated for by firing the opposing thruster on the opposite side at some future time.

To illustrate the ability of the policy to generalize to novel scenarios, we re-ran testing using the cases tabulated in Table 8. Except as noted in this table, the initial conditions and asteroid characteristics were identical to that used for optimization. In each case, performance was similar but slightly worse to that shown in Table 7, with the exception

[1] https://github.com/Aerospace-AI/Aerospace-AI.github.io.

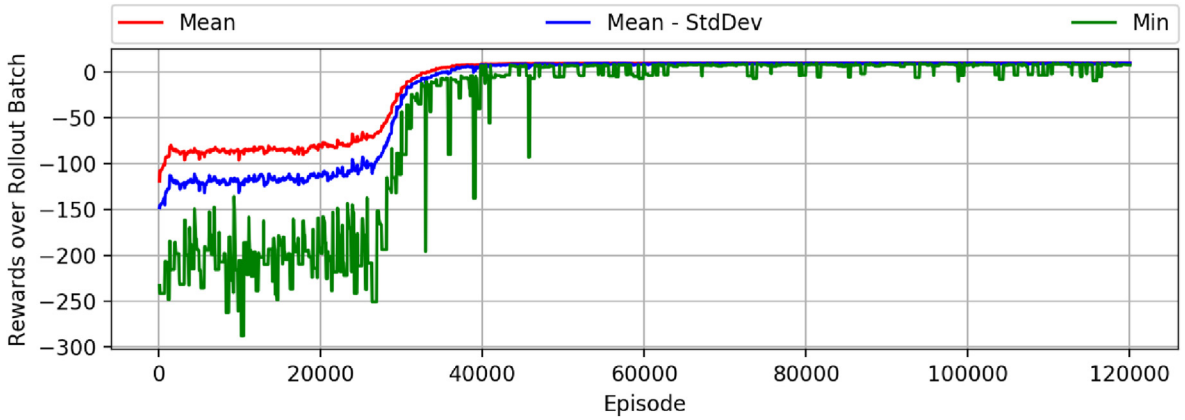

Fig. 5. Optimization rewards learning curves.

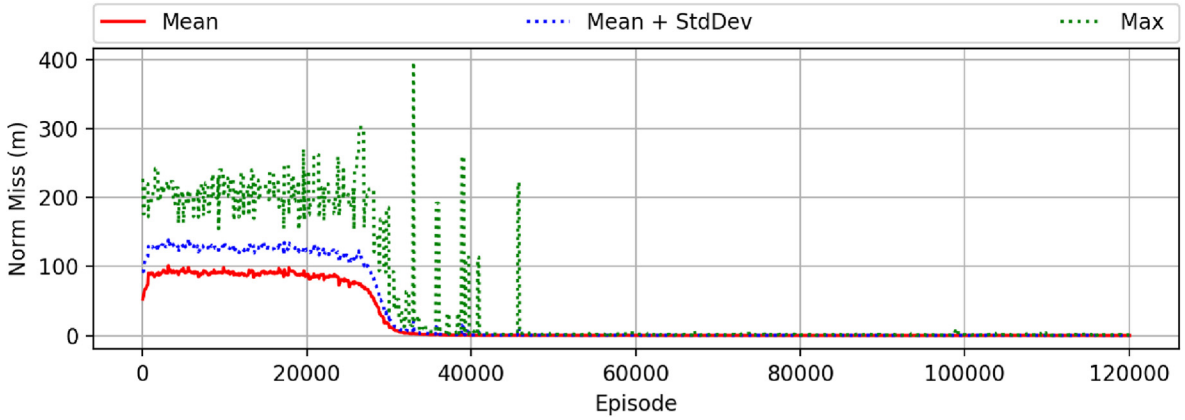

Fig. 6. Optimization terminal position error learning curves.

Table 7
Performance.

Parameter	Mean	Std	Max
Terminal Position (m)	0.21	0.15	1.87
Terminal Velocity (cm/s)	0.9	0.4	3.3
Rotational Velocity (mrad/s)	0.00	0.46	1.47
Good Hover 1 (%)	100.0	N/A	N/A
Good Hover 2 (%)	100.0	N/A	N/A
Fuel (kg)	0.41	0.09	0.63

of increased fuel consumption for the extended hovering duration test, and rare failures to achieve the required hovering performance. Abbreviated results are given in Table 9, where we see that the policy has trouble generalizing to lower altitudes. This may be due to the reduced perceived curvature at lower altitudes, and an obvious remedy that could be explored in future work would be to optimize over positions that cover these lower altitudes. We ran additional experiments with a minimum altitude of 50 m that resulted in performance closer to that of Table 7. The policy generalized fairly well to a more finely grained texture (more facets).

4.3. Policy testing: rq36 and Itokawa

Since a guidance law for hovering over synthetic asteroids would be of limited value, we also test the optimized policy using a shape model of asteroids rq36 and Itokawa. These shape models are shown below in Fig. 8, along with flash LIDAR beams (green for hit, red for miss) from a randomly generated spacecraft initial state. Due to the slightly smaller size of asteroid rq36 as compared to that of the smallest randomly generated asteroids, rare complete misses for the flash LIDAR returns occurred when the altitude was above 500 m. We define a complete miss as none of the flash LIDAR beams intersecting the asteroid. Similarly, the minimum dimension of the peanut shaped asteroid Itokawa also resulted in occasional complete misses at altitudes greater than 250 m when the spacecraft was located close to collinear with the asteroid's x-axis. Since the curvature of Itokawa is quite different from that of the synthetic asteroids, it should be a particularly challenging test case, and we therefore look at two cases. First, we test using the standard Itokawa shape model, but restrict the altitude to below 250 m. Second, we scaled up the dimensions of the Itokawa shape model by a factor of 3, which we refer to in the following as “Itokawa3X”. This scaling resulted in a minimum axis size slightly larger than the smallest experienced during optimization, which allowed testing hovering at higher altitudes. Other than the modified initial altitude shown in Table 10, the initial conditions are identical to that for the random synthetic asteroid testing. A performance summary is given in Table 11, where we see that, similar to the case of the synthetic asteroids, the policy has trouble generalizing to hovering at low altitudes (below 100 m), particularly for the case of Itokawa3X. We ran additional experiments and found that hovering at a minimum altitude of 20 m gives performance close to that observed at 100 m. Also note that the performance of Itokawa3X is a bit worse than the other cases, perhaps due to the policy failing to generalize to the smaller curvature along the major axis.

5. Conclusion

We formulated a particularly difficult problem that to our knowledge has not been solved: precision hovering in an asteroid's body-fixed

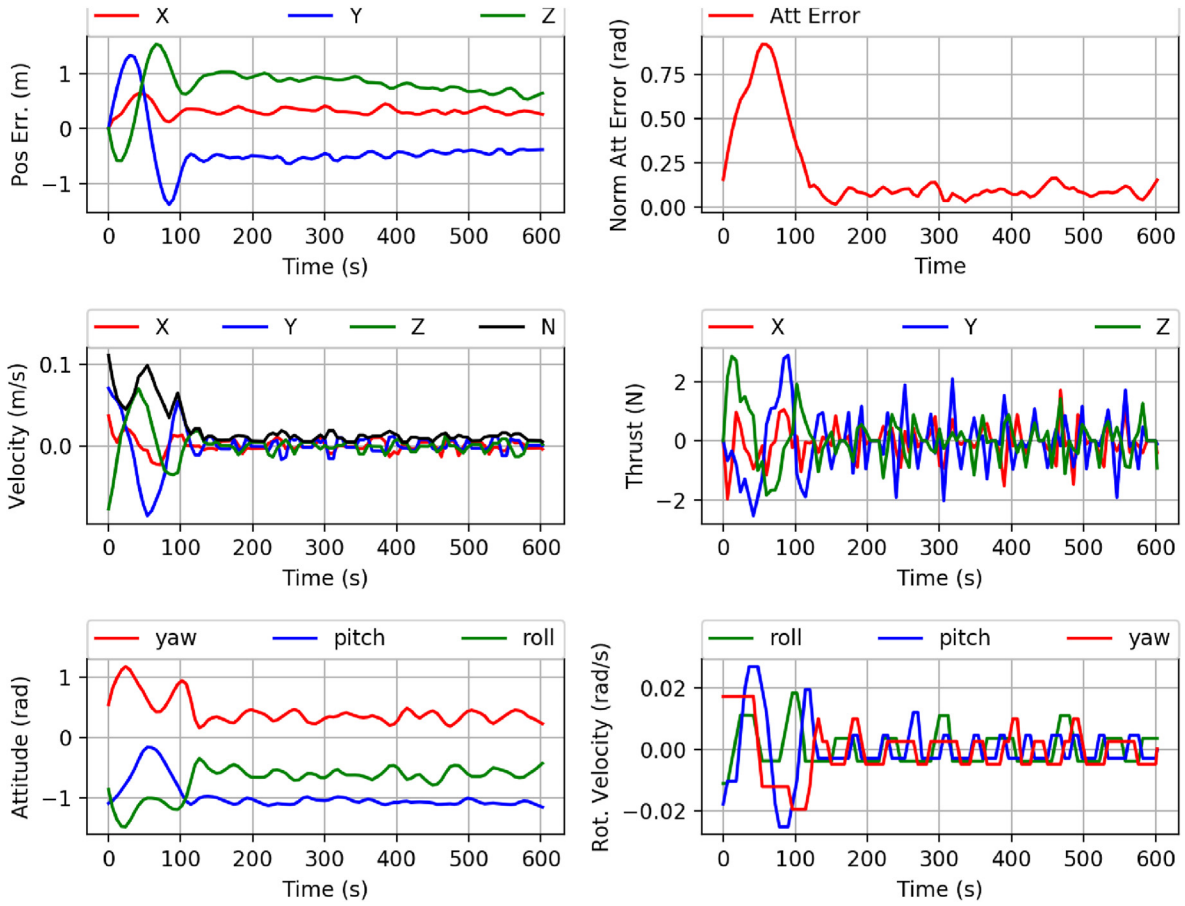

Fig. 7. Sample trajectory.

Table 8
Generalization cases.

Case	Description
Extended Altitude Range	Initial altitude range increased to (10 m, 700 m)
Asteroid Facets	Number of facets on each randomly generated asteroid shape model increased from 320 to 1280
Hovering Duration	We increased hovering duration to 1200s
Actuator Failure	With probability 0.5, random failing thruster has thrust reduced by factor of 0.5
Sensor Noise	Random range bias for each episode uniformly distributed between -5m and 5 m, Gaussian zero mean range noise with 2 m standard deviation at each sample
Environmental Dynamics	Maximum asteroid spin rate increased to 1e-3 rad/s
Center of Mass (COM) Variation	At the start of each episode, the spacecraft's center of mass is randomly set to an initial value between −10cm and 10 cm on each axis.

Table 9
Generalization performance.

Case	Good Hover 1 (%)	Good Hover 2 (%)	Max Pos Err (m)
Extended IC	98.22	99.76	25.2
Facets	100.00	99.94	4.46
Duration	99.96	100.00	3.73
Actuator Fail	100.00	100.00	1.66
Sensor Noise	100.00	98.86	3.67
Env. Dynamics	100.00	100.00	1.78
COM Variation	100.00	100.00	1.37

frame without a shape model or navigation aids, and without knowledge of the asteroid's environmental dynamics. To solve this problem we created a high fidelity 6-DOF simulator that synthesized asteroid models with shapes taking the form of asymmetric ellipsoids. For purposes of computing angular velocity, the asteroid is modeled as a uniform density ellipsoid that in general is not rotating about a principal axis, resulting in time varying dynamics. We then optimized an adaptive policy that maps flash LIDAR sensor measurements directly to actuator commands. The policy was optimized using reinforcement meta-learning, where the policy and value function networks each contained a recurrent hidden layer, and with the policy network using a convolutional front-end. During optimization, the agent was confronted with a new randomly generated asteroid for each episode, with randomized shape, density, rotational speed, and nutation angle. We then tested the policy, and demonstrated that the optimized policy generalizes well to novel hovering altitudes, hovering duration, actuator failure, actuator noise, sensor noise, and asteroid shapes and textures. Finally, we demonstrated hovering using an rq36 shape model and a scaled up Itokawa shape model. Comparing test performance for novel scenarios to that of scenarios used for optimization, we found similar, but slightly worse performance for the novel scenarios. Future work could improve upon this performance by creating synthetic asteroids with varying textures and more complex morphology, optimizing with random actuator failure and sensor noise, and exploring different convolutional network architectures. In addition, robustness to larger

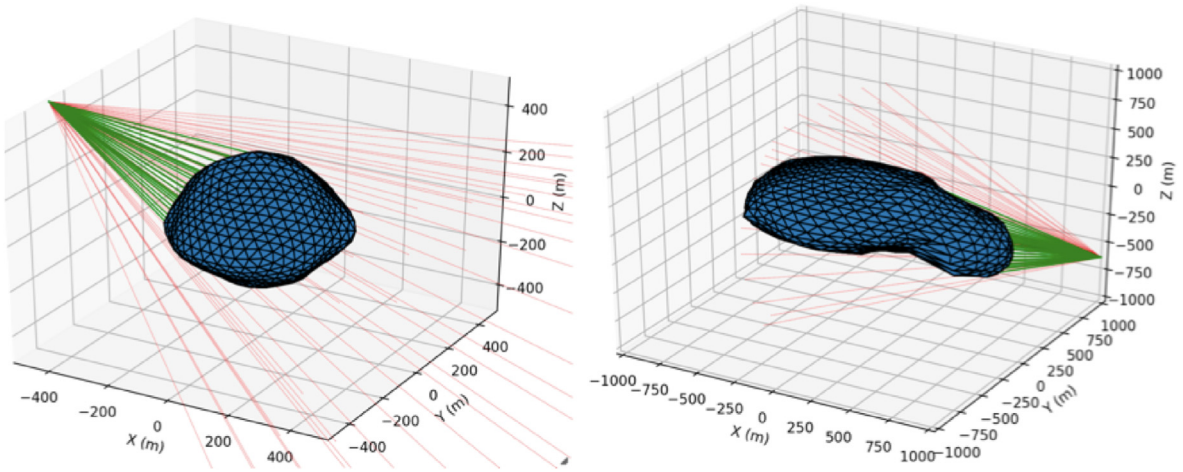

Fig. 8. Asteroids RQ36 (left) and Itokawa 3X (right).

Table 10
Initial Condition altitudes for Real Shape Models.

Asteroid	Min Altitude (m)	Max Altitude (m)
rq36	100	500
rq36 EXT	10	500
Itokawa	100	250
Itokawa EXT	10	250
Itokawa3X	100	600
Itokawa3X EXT	10	600

Table 11
Performance with real shape models.

Asteroid	Good Hover 1 (%)	Good Hover 2 (%)	Max Pos Err (m)
rq36	100.00	100.00	1.63
rq36 EXT	99.30	100.00	4.86
Itokawa	99.84	99.95	8.37
Itokawa EXT	99.80	99.98	9.32
Itokawa3X	99.94	100.00	4.41
Itokawa3X EXT	98.02	99.64	17.52

initial attitude errors and higher hovering altitudes would be enhanced by optimizing a separate policy that, prior to initiation of hovering, rotates the stabilized flash LIDAR seeker in a manner that minimizes the number of beams missing the asteroid. We expect that the ability to hover in the body-fixed frame immediately upon arrival at an asteroid will simplify shape model generation and other aspects of mission planning.

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.

References

[1] B. Schutz, B. Tapley, G.H. Born, Statistical Orbit Determination, Elsevier, 2004.
[2] M. Lara, D.J. Scheeres, Stability bounds for three-dimensional motion close to asteroids, J. Astronaut. Sci. 50 (4) (2002) 389–409.
[3] S.B. Broschart, D.J. Scheeres, Control of hovering spacecraft near small bodies: application to asteroid 25143 itokawa, J. Guid. Contr. Dynam. 28 (2) (2005) 343–354.
[4] S. Sawai, D. Scheeres, S. Broschart, Control of hovering spacecraft using altimetry, J. Guid. Contr. Dynam. 25 (4) (2002) 786–795.
[5] R. Furfaro, Hovering in asteroid dynamical environments using higher-order sliding control, J. Guid. Contr. Dynam. 38 (2) (2014) 263–279.
[6] B. Gaudet, R. Furfaro, Real-time state estimation for asteroid close-proximity operations via lidar altimetry and a particle filter, 2013 AAS/AIAA Astrodynamics Specialist Conference, Astrodynamics 2013, Univelt Inc., 2014.
[7] D. Lee, A.K. Sanyal, E.A. Butcher, D.J. Scheeres, Almost global asymptotic tracking control for spacecraft body-fixed hovering over an asteroid, Aero. Sci. Technol. 38 (2014) 105–115.
[8] B. Gaudet, R. Furfaro, Robust spacecraft hovering near small bodies in environments with unknown dynamics using reinforcement learning, AIAA/AAS Astrodynamics Specialist Conference, 2012, p. 5072.
[9] Y. Tsuda, M. Yoshikawa, M. Abe, H. Minamino, S. Nakazawa, System design of the hayabusa 2—asteroid sample return mission to 1999 ju3, Acta Astronaut. 91 (2013) 356–362.
[10] S. Kameda, H. Suzuki, T. Takamatsu, Y. Cho, T. Yasuda, M. Yamada, H. Sawada, R. Honda, T. Morota, C. Honda, et al., Preflight calibration test results for optical navigation camera telescope (onc-t) onboard the hayabusa2 spacecraft, Space Sci. Rev. 208 (1–4) (2017) 17–31.
[11] S. Yasuda, K. Matsushima, F. Terui, Y. Mimasu, N. Ogawa, G. Ono, K. Yoshikawa, Y. Tsuda, Operational design for hayabusa2 touch-down to ryugu, AIAA Scitech 2020 Forum, 2020, p. 2132.
[12] T. Mizuno, T. Kase, T. Shiina, M. Mita, N. Namiki, H. Senshu, R. Yamada, H. Noda, H. Kunimori, N. Hirata, et al., Development of the laser altimeter (lidar) for hayabusa2, Space Sci. Rev. 208 (1–4) (2017) 33–47.
[13] K. Yoshikawa, S. Kikuchi, H. Sawada, G. Ono, Y. Mimasu, N. Ogawa, F. Terui, T. Saiki, Y. Tsuda, Hayabusa2 spacecraft dynamics and operational design of final descent and touchdown in sampling mission, AIAA Scitech 2020 Forum, 2020, p. 1208.
[14] B. Gaudet, R. Linares, R. Furfaro, Seeker Based Adaptive Guidance via Reinforcement Meta-Learning Applied to Asteroid Close Proximity Operations, arXiv preprint arXiv:1907.06098.
[15] N. Mishra, M. Rohaninejad, X. Chen, P. Abbeel, A Simple Neural Attentive Meta-Learner, arXiv preprint arXiv:1707.03141.
[16] K. Frans, J. Ho, X. Chen, P. Abbeel, J. Schulman, Meta Learning Shared Hierarchies, arXiv preprint arXiv:1710.09767.
[17] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, M. Botvinick, Learning to Reinforcement Learn, arXiv preprint arXiv:1611.05763.
[18] B. Gaudet, R. Linares, Adaptive Guidance with Reinforcement Meta-Learning, arXiv preprint arXiv:1901.04473.
[19] B. Gaudet, R. Furfaro, R. Linares, A Guidance Law for Terminal Phase Exo-Atmospheric Interception against a Maneuvering Target Using Angle-Only Measurements Optimized Using Reinforcement Meta-Learning, arXiv preprint arXiv:1906.02113.
[20] D.J. Scheeres, Orbital Motion in Strongly Perturbed Environments: Applications to Asteroid, Comet and Planetary Satellite Orbiters, Springer, 2016.
[21] T. Möller, B. Trumbore, Fast, minimum storage ray/triangle intersection, ACM SIGGRAPH 2005 Courses, ACM, 2005, p. 7.
[22] G.M. Siouris, Missile Guidance and Control Systems, Springer Science & Business Media, 2004.
[23] B. Gaudet, R. Linares, R. Furfaro, Deep Reinforcement Learning for Six Degree-Of-Freedom Planetary Powered Descent and Landing, arXiv preprint arXiv:1810.08719.
[24] C. Finn, P. Abbeel, S. Levine, Model-agnostic Meta-Learning for Fast Adaptation of Deep Networks, arXiv preprint arXiv:1703.03400.
[25] A. Rajeswaran, S. Ghotra, B. Ravindran, S. Levine, Epopt: Learning Robust Neural Network Policies Using Model Ensembles, arXiv preprint arXiv:1610.01283.
[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms, arXiv preprint arXiv:1707.06347.
[27] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, International Conference on Machine Learning, (2015), pp. 1889–1897.
[28] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Gated feedback recurrent neural networks, International Conference on Machine Learning, 2015, pp. 2067–2075.
[29] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, The handbook of brain theory and neural networks, 3361 (10) (1995) 1995.
[30] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for Simplicity: the All Convolutional Net, arXiv preprint arXiv:1412.6806.