# Autonomous Imaging and Mapping of Small Bodies Using Deep Reinforcement Learning

**David M. Chan**
**University of California, Berkeley**
**253 Cory Hall**
**Berkeley, CA 94720**
**davidchan@berkeley.edu**

**Ali-akbar Agha-mohammadi**
**Jet Propulsion Laboratory,**
**California Institute of Technology**
**4800 Oak Grove Dr.**
**Pasadena, CA 91109**
**aliakbar.aghamohammadi@jpl.nasa.gov**

***Abstract*—The mapping and navigation around small unknown bodies continues to be an extremely interesting and exciting problem in the area of space exploration. Traditionally, the spacecraft trajectory for mapping missions is designed by human experts using hundreds of hours of human time to supervise the navigation and orbit selection process. While the current methodology has performed adequately for previous missions (such as Rosetta, Hayabusa and Deep Space), as the demands for mapping missions expand, additional autonomy during the mapping and navigation process will become necessary for mapping spacecraft. In this work we provide the framework for optimizing the autonomous imaging and mapping problem as a Partially Observable Markov Decision Process (POMDP). In addition, we introduce a new simulation environment which simulates the orbital mapping of small bodies and demonstrate that policies trained with our POMDP formulation are able to maximize map quality while autonomously selecting orbits and supervising imaging tasks. We conclude with a discussion of integrating Deep Reinforcement Learning modules with classic flight software systems, and some of the challenges that could be encountered when using Deep RL in flight-ready systems.**

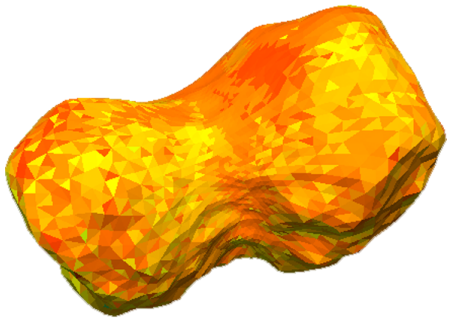

**Figure 1**. Example of the generated map quality of the Mithra asteroid using the REINFORCE algorithm under the constrained spacecraft model. Red patches correspond to high map quality, while green corresponds to low map quality.

## TABLE OF CONTENTS



## 1. INTRODUCTION

In the last decade, a number of space agencies and industries have focused on enhancing the automation capabilities of spacecraft. One particular area of interest is the autonomous mapping of small bodies. The current process for planning information-gathering "mapping missions" around small bodies relies heavily on a human-in-the-loop system. By analyzing data and using complex models, trained analysis can construct trajectories that simultaneously obtain map quality improvements, as well as respect spacecraft flight constraints and competing science objectives.

While the current methods have performed well for current missions (such as Rosetta, Hayabusa and Deep Space), with increased interest in small body mapping, as well as the increasing number and scale of ongoing mapping missions, human-in-the-loop planning is rapidly becoming an unattractive approach. While such methods have been successful in the past (such as in the Rosetta mission), the process is time-consuming and may be impossible for more complex or wide-reaching future mission involving multiple spacecraft exploring several bodies simultaneously.

Thus, it is of great interest to explore options for planning trajectories around small bodies, which is less reliant on humans, or is even fully autonomous. Modern end-to-end deep learning techniques stand out as a possible solution to alleviate the requirements of human-in-the-loop systems, particularly for the small body mapping problem. Deep reinforcement learning (Deep RL), an approach pioneered by Minh *et al.*[1], has recently become a widely active and interesting area of research. The aim of a deep reinforcement learner is to generate an end-to-end policy for the control of an agent, learning from sensory precepts, and attempting to maximize the obtained reward over a horizon. Recent successes in deep reinforcement learning for control in autonomous driving [2–4] are inspiring, as they show that it is possible to learn complex interactions end-to-end to maximize reward functions.

The focus of this paper is to explore deep reinforcement learning methods for increasing the autonomy of mapping and allowing autonomous mapping systems to scale beyond the current human-limited processes. Particularly, we focus on the trajectory planning problem as it relates to maximizing information gain in an Active Simultaneous Localization and

978-1-5386-6854-2/19/$31.00 

Mapping (Active SLAM) approach. In addition, we keep an eye towards the implementation of such tools in real-world flight frameworks and discuss the implementational challenges which can arise from using deep reinforcement learning in practice. The major contributions of this work are as follows:

1. We frame the spacecraft-constrained Active SLAM trajectory planning problem as a Partially Observable Markov Decision Process. (Section 3)
2. We introduce a simulation environment which can be used to model the above POMDP, and demonstrate improved performance over the baseline using state of the art reinforcement learning techniques. We also show that the learned policy for trajectory planning and map quality optimization generalizes well to new scenarios. (Sections 4, 5)
3. We discuss the implementation of the mapping module as a part of a larger spacecraft autonomy framework. We also outline some of the challenges of implementing such trajectory planning in highly-constrained real-world environments. (Section 6)

## 2. BACKGROUND & RELATED WORK

The work presented here is closely related to the Active Simultaneous Localization and Mapping (Active SLAM) problem in robotics and classical control [5]. In the Active SLAM scenario, an agent must concurrently construct a map, while using that map to navigate. This is a particularly hard problem for spacecraft, so we restrict our definition of the problem to constructing a map given the initial geometry of the body, and planning a trajectory to improve the quality of that map, or to perform science mapping experiments, such as those in [6] or [7]. While this is a restriction of the problem, it falls well into the phases of the typical mapping mission.

*Mission Architectures*

Typical mapping missions are divided into a number of phases; For example, the Rosetta mission was broken into several operational phases, characterized by the distance to the body [8]. The first phase of the Rosetta mission upon arrival was the characterization phase, occurring approximately 90km to 52km from the body. During this phase, images were taken every hour, and features in the image were manually identified. This gave a rough geometric map of the body. The next phase is the global mapping phase (50-20km), where a more detailed global map is defined (similarly by hand). This map, in the case of Rosetta, was used to construct estimates of the gravitational potential and rotational state, as well as to estimate the topography. Finally, during the close observation phase (20-10km), a few potential areas were studied for the selection of landing sites. Our proposed mapping technique, while restricted to the case where we know the rough geometry already, falls well into the global mapping and close observation phases.

*SLAM Methods*

Traditional mapping of small bodies is done using a number of techniques, including stereophotoclinometry (SPC) [6] and stereophotogrammetry (SPG) [7]. In this work, we choose to focus on SPC-based techniques, as they lend themselves to image quality enhancement, and are largely the most popular. In general, the trajectories that a spacecraft takes are chosen by hand [6, 7, 9] from a number of possible orbits created by ground-based planning software. Thus, the orbit selection and generation process are often decoupled, and cannot interact in complex ways. In addition, these orbit selection processes rely on robust localization, to maintain the selected orbit. Traditionally, the AutoNav [10] approach is used to localize the spacecraft (which relies on images taken of global features), however, additional localization methods have been explored [11–14].

Under the SPC method, the surface map $M$ of the body is often represented by a set of smaller *maplets* with their associated information. Thus, we can represent $M$ as a set of faces $f_i^M$ on a mesh. The $i^0th$ face is characterized by a triangle (as is classic in graphics applications) with its corresponding vertices and a normal vector describing the general direction of the face.

*Planning Under Uncertainty*

In this paper, we explore autonomous mapping and planning around small bodies. This is, generally, a subset of the larger problem of planning in an uncertain environment. This Active SLAM problem is often posed as a Partially Observable Markov Decision Process (POMDP) [15–17]. We need some general notation from POMDP literature to discuss our problem, which we present here. Formally, a POMDP is a tuple $(S; A; T; R; \ ; O; \ )$ where $S$ is a set of states, $A$ a set of actions, $T$ is a set of conditional transition probabilities between states (often implemented as a noisy transition function), $R : S \quad A \ ! \ R$ is a reward function, is a set of observations, $O$ is a set of conditional observational probabilities, and is the discount factor. A solution to a POMDP is a policy $: S \ ! \ A$ which for any state, chooses the action to take in the environment which maximizes the expected sum of discounted rewards:

$$= \arg\max \ \mathbf{E}\left[\sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{T}} \ ^{\mathbf{k}}\mathbf{R}(\mathbf{b_k}; \quad (\mathbf{b_k}))\right] \qquad (1)$$

where is the set of all policies, $k$ is the time, $T$ is the horizon of the problem, is the discount factor and $b_k$ is the "belief" at time $k$.

Because we are planning under uncertainty, we do not directly observe $x_k \ 2 \ S$ from the model. We instead obtain observations $z_k \ 2$ based on the probabilities $O(z_k j s_k \ 2 \ S)$. Thus, to handle this, we maintain a belief $b$ over the state space - a probability distribution across the presence of the agent in any given state. Traditionally, the belief is updated using a belief propagation algorithm (forward message passing), see [18] for more details.

One major contribution of this work is to outline a POMDP formulation of the small body mapping problem. Such a formulation was previously explored in [19], however, they focused primarily on the reward design with simplified transition dynamics and a fully observable environment. In this paper, we relax a large number of their planning constraints and explore solving the relaxed problem.

*Solving POMDPs*

In general, the POMDP solution is NP-Hard, and thus, can be intractable to compute optimally. Simple POMDPs can be solved optimally using the policy-iteration or value-iteration algorithms [20]. [19] reduced the small-body autonomous mapping problem sufficiently so that it could be solved using such methods. In this work, however, we relax a wide number of assumptions made by [19], and thus, our problem becomes intractable to optimally solve. There has been a wide body

of research devoted to approximately solving POMDPs - for a survey, consult [21]. In this work, we focus on using reinforcement learning, particularly deep reinforcement learning, to tackle this problem.

Deep reinforcement learning is a recent area of research popularized by [1], and focuses on using the reward signal from numerous *rollouts* or "simulations" of the game to adjust the parameters of a parametric policy which, in the case of deep reinforcement learning, is a deep neural network. In this work, we use the policy-gradient algorithm [22] (also called REINFORCE), a simple and effective method for learning strong policies in POMDPs. In REINFORCE, we write the goal of the learner to learn :

$$= \arg\max \mathbf{E} \quad _{\theta(\ )}[\mathbf{r}(\ )] \tag{2}$$

where is a trajectory, is a policy with parameters and $r(\ )$ is the expected reward over that trajectory. Thus, we attempt to learn the optimal parameters for the policy. For convenience, we let

$$J(\ ) = \mathbf{E} \quad _{\theta(\ )}[\mathbf{r}(\ )] \tag{3}$$

Then, we can solve the problem given in Equation 2, by finding $r \ J(\ )$. Instead of computing this analytically (which would be rather difficult), we use an estimate of the gradient by sampling $i$ times from the policy, then computing:

$$r \ J(\ ) \quad \sum_i \left[ \left( \sum_t r \ \log \quad (a_t^i | s_t^i) \right) \left( \sum_t r(s_t^i, a_t^i) \right) \right] \tag{4}$$

We then update the parameters using gradient descent:

$$^{i+1} = \ ^i + \quad r \ _i J(\ ^i) \tag{5}$$

Additional methods for reinforcement learning have been explored, including deep Q-Learning [1], and the A3C algorithm [23] - and it would be interesting further research to expand the ideas here to modern reinforcement learning algorithms.

## 3. FRAMING SMALL BODY MAPPING AS A POMDP

The first contribution of our work comes as a formalization of the Active SLAM problem for autonomous small-body mapping as a POMDP. In this section we discuss the individual components of a potential POMDP formulation, and discuss some of the necessary modeling choices.

*State Space*

We represent the state space $S$ of our POMDP as the tuple:

$$\begin{aligned}(&p_{body}, v_{body}, a_{body}, d_{body}, p_{craft}, v_{craft}, a_{craft}, \\ &d_{craft}, s_{camera}, s_{memory}, s_{antenna}, \\ &s_{map}, \ _{body}, p_{star})\end{aligned} \tag{6}$$

In general, we restrict the simulation needs of our problem by only keeping the state of two objects, the body to be imaged and the spacecraft. Thus, the state of our world consists of the position ($p_{body}$), attitude ($d_{body}$), velocity ($v_{body}$), and acceleration ($a_{body}$) of the body, as well as the position ($p_{craft}$), attitude ($d_{craft}$), velocity ($v_{craft}$) and acceleration ($a_{craft}$) of the spacecraft. In addition to maintaining the world/positional state, we also maintain the state of a number of on-board spacecraft components. We keep track of the camera state (on/off, $s_{camera}$), the memory state (number of MB used, $s_{memory}$) and the state of the antenna (is the antenna currently downlinking ($s_{antenna}$).

In addition to this positional state, we support a number of global state variables, such as the trajectory of the body through space ( $_{body}$) and the position of the nearest star ($p_{star}$) which we use to obtain the global lighting conditions. Finally, we also keep track of the current state of the map ($s_{map}$). The spacecraft needs to know what it has already imaged in order to determine the most efficient use of its time for future images and to determine the best trajectories to explore.

The state of the world represents a distinct enlargement of the state examined in [19]. Our explored state has the ability to model lighting conditions in an advanced way, as well as allowing for changes in attitude and rotation of the body and spacecraft. Because in our model, the spacecraft need not be pointed at the body (and in fact, may have to be pointed elsewhere during downlinking), we are able to model a much wider set of conditions.

*Action Space*

In [19], the possible action space was restricted to a set of preselected orbits around the body, and a "per-orbit" reward was measured. We relax that assumption by giving the algorithm the ability to act directly on the acceleration of the spacecraft through the use of a simplified set of thrusters. Our action space $A$ consists of the tuple:

$$(t_{+X}, t \ _X, t_{+Y}, t \ _Y, t_{+Z}, t \ _Z, c_{image}, a_{downlink}) \tag{7}$$

In the above, each of $t_{+X}$, $t \ _X$, $t_{+Y}$, $t \ _Y$, $t_{+Z}$, $t \ _Z$, $c_{image}$, $a_{downlink} \ 2 \ f0; 1g$ with 0 being the "off" state, and "1" being the on state. While in the real world, thrusters are often graded, and can have a range of impulse powers, learning in continuous action-space POMDPs is orders of magnitude more difficult than doing so in discrete space [24, 25]. Thus, we restrict our thruster controls to "on" and "off" conditions.

In addition to modeling the behavior of the thrusters, we also include actions for downlinking and taking images. In [19], no attention was paid to the fact that only a certain number of images can be held in memory at any given time, and that transmission of these images needs to occur. Thus, we restrict the agent to have to explicitly image the body and explicitly perform downlinking of images to regain memory space onboard the craft.

We also employ several constraints on the actions of taking images and downlinking. First, we ensure that neither of these actions can happen at the same time - thus, the algorithm must make trade-offs and learn to downlink images when it can. In addition, we restrict downlinking so that it can only happen during certain time intervals, simulating the effect of a DSN (Deep Space Network) pass. Thus, the spacecraft must operate with an awareness of these passes in order to generate high-quality maps.

It is certainly interesting future work to explore additions to this state space such as including on-board spacecraft power,

or thruster fuel.

*Observation Space & Model*

Because we are planning in an uncertain space, the spacecraft does not receive information with certainty about the state of the world, and instead receives observations $z$ 2 . Because the focus of this paper was not on modeling uncertainty in orbital scenarios, but about generating orbit trajectories to maximize image quality, we represent this uncertainty by adding isotropic Gaussian noise to the state variables involving the body and the spacecraft's position, velocity, and acceleration. I.e. = $S$, and

$$z_{position} = s_{position} + \quad ; \qquad N(0; \quad )$$

(and similarly for other variables). We represent the rest of the state faithfully, as the spacecraft should be able to understand its own memory with a relatively high degree of accuracy.

*Transition Model*

Recall that the transition model is a set of conditional probabilities $P(s_{k+1} j s_k; a_k)$. In [19] - the model was over a single orbit, and transitions to the next orbit could be made at the end of an *orbital period*. We relax this requirement and allow for actions to be made at $_t$ interval time steps (a parameter of the simulation).

The trajectory of the body is specified as an input parameter to the environment, and in our experiments, is considered to be a Keplerian orbit around the "star" celestial body. We specify the orbit as a tuple:

$$(e; a; i; \quad ; !) \tag{8}$$

where $e$ is the eccentricity of the elliptical orbit, $a$ is the semi-major axis of the orbit, $i$ is the inclination of the orbit, is the longitude of the ascending node, and $!$ is the argument of periapsis. Figure 2 gives a detailed description of these parameters. We evolve the orbit of the body around an imaginary star by first computing the mean motion of the body:

$$n = \sqrt{\frac{}{a^3}} \tag{9}$$

where $n$ is the mean motion, is the gravitational system constant, and $a$ is the semi-major axis of the orbit. We then evolve the anomaly of the orbit (the distance in radians around the elliptical orbit from the 0 anomaly) as $_{t+1} = \ _t + n \ _t$.

We model the transition of the spacecraft using basic Newtonian physics. First, from the second law we get that:

$$a_t = \frac{GM}{jj\flat_t \quad p_t jj}(\flat_t \quad p_t) \tag{10}$$

where $p_k$ is the spacecraft position at time $k$, $\flat_k$ is the position of the body at time $k$, $G$ is the gravitational constant, $M$ is the mass of the body, and $a_k$ is the acceleration of the spacecraft at time $t$ due to the mass of the body. To this acceleration we add the acceleration due to the thrusters (based on the action). We use a simple double integration to compute the change in position of the spacecraft at time $k$ (as we are stepping in discrete time intervals).

To evolve the spacecraft state outside of position - if the camera is on, we attempt to take a picture. If memory is not full (there is enough space in memory for the picture), we add the captured image to the observation record and increment the memory. If the spacecraft is in the process of downlinking (as specified by the downlinking action), then we decrease the amount of used memory at a slow rate. The spacecraft cannot capture an image and downlink at the same time (if both actions are attempted, the camera action takes priority in the simulator).

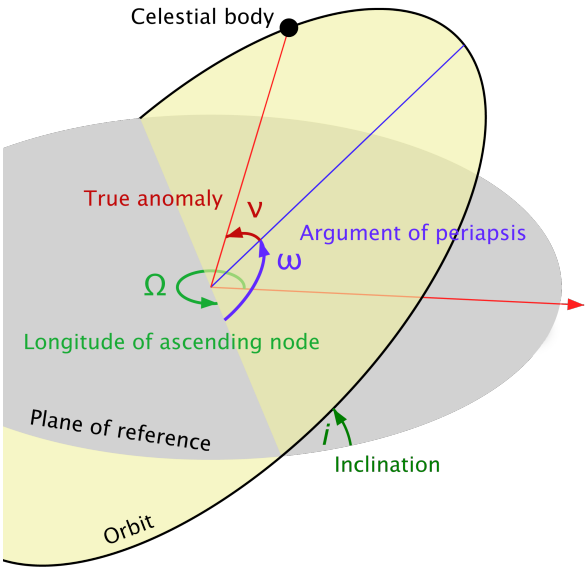


**Figure 2**. Figure showing the different orbital elements. We evolve the true anomaly over time, and specify the other elements of our celestial body. Figure modified from [26].

In our experiments we do not generate scenarios when the spacecraft is not pointing NADIR, Thus, we do not evolve the rotation of the camera. It is the case, however, that the orbital environment that we constructed can handle such situations, and it remains interesting future work.

*Reward Model*

Recall that the reward function $R : S \quad A \ ! \ \mathrm{R}$ specifies the reward that is achieved by the craft. Because we want to plan trajectories which increase the quality of the map, we attempt to represent the map quality (under SPC) using such a function. The model that we use for reward is similar to the model used in [7], however, we extend their model to take into account the dynamics of our camera and spacecraft operation. our reward consists of a weighted sum of the following subcomponents:

$$\mathbf{R} = (w_0 R_{emis} + w_1 R_{inc} + w_2 R_{azi}) \quad S_{camera} + w_3 R_{thrust} + w_4 R_{dist} \tag{11}$$

We discuss each of these components below, however, notice that the reward coming from the images depends on the state of the camera. While the (potential) reward is calculated at each step, we only give the spacecraft the reward if it decided (and was able to) take a picture, increasing the difficulty of the control problem.

*Emission Reward ($R_{emis}$)*—The emission angle, the angle between the normal of the facet of the body and the spacecraft (i.e. the angle at which the light reaches the spacecraft), is an important component of the quality of the image map.

Because SPC relies on a least-squares solver, a large variance in emission angle is desirable. Thus, we record for each facet the variance in the emission angles by breaking the possible angles $e \in [0, \pi]$ into $u$ bins. For each bin, if the facet does not have an image already with the emission angle in those bins, then we add 1 to the reward - that is:

$$R_{emission} = \sum_{facets} \sum_{u} I_u \tag{12}$$

where:

$$I_u = \begin{cases} 1 & \text{No image exists with emission angle } u \text{ in the map} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

*Incidence Reward ($R_{inc}$)*—The angle of incidence, the angle between the incoming light and the normal of the facet in the map is also important in SPC. This measurement is an indication of the condition of illumination of the facet and can be indicative of very dark or very bright patches of illumination. The angle of incidence should be between $-\pi/2$ and $\pi/2$ radians. If the image is not within this range, then there is no reward, otherwise, we add 1 reward to the score. Thus, $R_{incidence}$ is an indicator for the individual facet as to whether or not the image is in the correct illumination.

*Azimuth Reward ($R_{azi}$)*—The goal of mapping is to construct a good view of the topology of the surface of the body. The azimuth of the body (the angle between the projection of the vector from the surface to the star, and local north) is an important part of maintaining that variance. A high variance in body azimuth angles is desired, so similar to emission angle, we reward this for falling into one of many different bins.

On the other hand, variation in the azimuth of the spacecraft can be detrimental to the quality of the images, as high variation in the azimuth angle of the spacecraft can lead to difficulty when matching features during the mapping process. Thus, we penalize variation in azimuth, based on the distance from the mean azimuth of the already assembled images.

*Thruster Reward ($R_{thrust}$)*—To avoid orbits which require large numbers of thruster commands, we discourage the use of thrusters by providing a small negative reward for each active thruster in an action. In more complex simulation environments, this negative reward could be replaced by a state measuring the fuel/power remaining, however, we do not model this, so instead provide a negative reward for the use of the thrusters.

*Distance Reward ($R_{dist}$)*—In addition to the thruster reward, we provide a large negative reward if the spacecraft becomes greater than distance $_{max}$, or closer than distance $_{min}$ to the body. This is in order to greatly penalize the spacecraft from crashing into the body, or from leaving the body entirely.

*Reward Trade-offs*—In many cases, the weights $w_0, w_1, \ldots$ for the rewards are not known or could vary between mapping parameters. By leaving the weights in the reward as hyper-parameters we can tune the behavior of the spacecraft to match particular imaging sessions and scenarios.

*Planning Horizon*

While in principle, the problem could be phrased as an infinite horizon problem, after a time, the spacecraft will run out of energy, or be re-tasked with the mapping complete. We employ a planning horizon of time $T$ to limit the problem to a finite-horizon problem, making the potential policies easier to learn. We use the discount factor $= 0.99$, a much higher discount factor than traditional problems, to encourage long-tailed reward behavior, and planning for the future. While this discount factor increases the difficulty of the finite-horizon problem, planning orbit transfers may require a large number of thruster commands, leading to a highly delayed payoff.

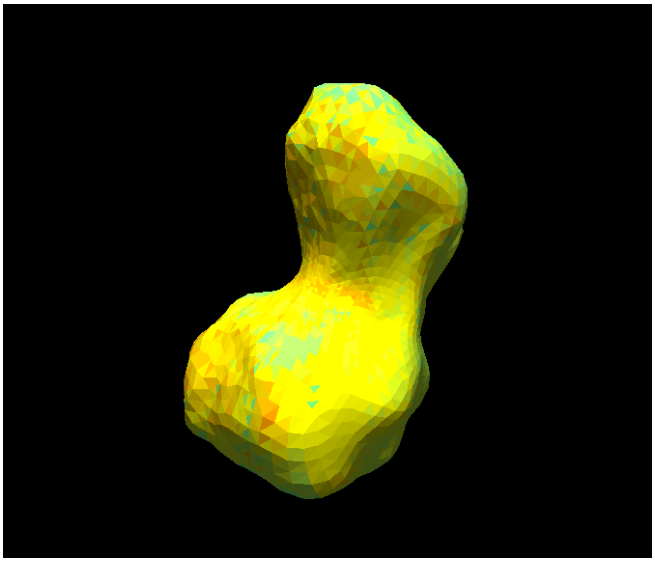

**Figure 3**. Sample image from the OSIM simulator. The images captured correspond to the camera on the satellite. While the dark side of the asteroid appears light in this simulated image, the dark side is actually much darker, and we have boosted the ambient light in the display image to demonstrate the dynamic lighting. In the experiments, the ambient light was considered 0.

*Conclusion*

In this section, we have presented the Active SLAM problem for mapping of small bodies as a complex POMDP. Our formulation heavily deviates from the previous work such as in Pesce *et al.*[19] by introducing a dynamics model, as well as a camera memory-state model. These introductions significantly expand the real-world applications of our formulation, however, also greatly increase the difficulty of solving the problem. Thus, given this formulation, we can no longer use optimal algorithms to explore the problem due to the large state space, and branching factor of the POMDP search trees. Instead, in the next section, we turn to deep reinforcement learning and approximate optimal solutions.

## 4. Simulator and Experimental Design

*Simulator Design*

In order to explore the POMDP proposed in Section 3, we developed a simulator environment, called OSIM (for Orbital Simulation). An example image from the simulator is given in Figure 3. This simulator environment has rich support for all of the necessary variables needed to model the POMDP in Section 3. OSIM is written in C++ and OpenGL, and makes use of GPU computation to compute the reward values for images fixed in the environment.

In addition to the C++ back-end of the simulation, we provide a python-based front end which communicates with the back-end through TCP/IP. The front-end is modeled after the environments in OpenAI Gym [27], which allows for easy transfer of state of the art machine learning algorithms to the OSIM environment. In fact, the environment is compatible with most existing algorithms for deep reinforcement learning - exploring the implementation of such algorithms is important future work. In addition to making the code transportable, the python front end allows us to use popular machine learning libraries such as Tensorflow [28] and PyTorch [29]. The C++ back-end allows us to maintain significant performance while performing the ray-tracing which is required to properly image the non-convex bodies with the associated light sources.

The performance of the simulator was an important consideration when building the OSIM environment. Traditional RL algorithms learn by performing hundreds of thousands of experiments in the environment to gain experience with new situations. This leads to millions of iterations being necessary for the algorithms to achieve an optimal solution. Traditional mapping and imaging simulators such as [30] cannot achieve the performance required for effective learning, as they were not designed to be high-performance environments. One instance of our simulator can achieve roughly 250,000 iterations per hour, which is fast enough for efficient reinforcement learning.

*Camera Design*— By actually simulating the images taken in the 3D environment, we gain the ability to experiment with intrinsic camera parameters, something never previously explored in this line of research. We settled on the use of a camera with a resolution of 400 300, and a field of view of =8 radians (45 ). We model our camera using perspective projection under the pinhole model. While this assumption means that we have no lens distortion (which may influence the quality of images), there are numerous algorithms which have been developed to correct for such aberration. In addition, we model all surfaces as Lambertian surfaces, meaning that the angle of incidence of the light from the sun does not affect the strength of the incoming light. To model specular surfaces would require a specular map of the asteroids, which we were unable to obtain for any of the real-world examples, thus we decided to omit such simulation. In real-world scenarios, however, astral bodies have a wide range of albedos (defining how much light is scattered), ranging from 0.99 (Enceladus) to asteroids with measured albedo less that 0.05 [31, 32]. Generating a more complex model of the reflectance of the astral bodies is important future work - and could greatly influence the design of trajectories and reward systems.

*Example Bodies*— In our paper, we explore the mapping of bodies through two different examples, the Mithra and Toutatis bodies. Both models were obtained from the NASA online 3D model repository [33]. The Mithra model contains 5596 facets, and our simulator can image the body at up to 70FPS (Frames Per Second), allowing for extremely fast data collection. The Toutatis body contains 39996 facets, and we can image it at up to 26FPS.

*REINFORCE Model Design*

The REINFORCE algorithm requires a function $(b)$ which is parameterized by , takes a belief state (a distribution), and returns the action which should be taken. For this function we use a two-layer fully connected neural network with ReLU activations [34]. Each hidden layer consists of 1024 neurons. The architecture is detailed in Figure 4. We avoid using normalization on the inputs (Such as in [35]) as we found that normalizing the input values can make it difficult for the network to accurately process the state of the body and the spacecraft.

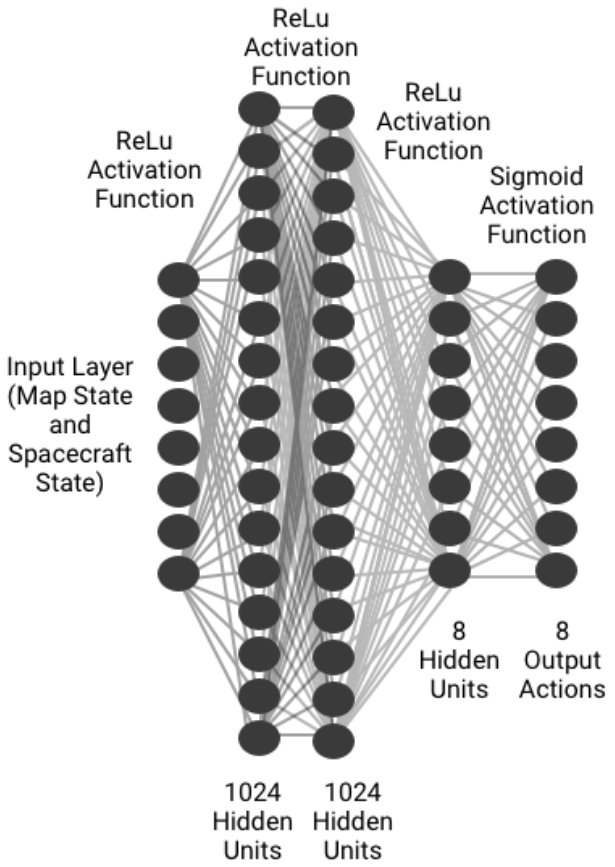


**Figure 4**. The above figure shows the architecture that we used for the neural network. The network is kept small so that it can be updated efficiently, however, it is future work to explore more complex architectures.

*Model Training*

We implemented the algorithm outlined above using the PyTorch library [29]. We then ran the REINFORCE algorithm for 10,000 episodes, each containing a maximum of 500 steps in the environment. We split the target and policy networks (A classic Deep RL trick), updating the target network every 10 episodes. We perform the gradient updates to the network using the RMSProp [36] optimizer. The experiments were run using a GTX Titan Maxwell GPU, and an Intel i7-5820K CPU with 64GB of ram.

*Benchmark Algorithms*

Similar to Pesece *et al.*, we benchmark our method against three other algorithms, the random-action selector algorithm, and the AutoNav fixed inclination orbit selection and random orbit selection algorithms. We do not benchmark against Pesece *et al.*[19] as their code is not publicly available, and thus we were unable to extend their code to our environment in a faithful manner.

The random-action algorithm gives the benchmark random mapping performance, by selecting random actions from the

| Method | Performance | |
|---|---|---|
| Random Action | 1785.47 | 1388.973 |
| AutoNav Random Orbit | 1982.29 | 587.663 |
| AutoNav Fixed Orbit | 1849.01 | 138.128 |
| Proposed Deep-RL method | 3470.29 | 536.3 |

**Table 1**. Results on the Mithra Asteroid. Because the policies were stochastic, we recorded the performance over 100 trial episodes. As we can see, the best reinforcement learning based approach outperforms the other approaches after training on this environment.

action space. This algorithm should be considered a *random level* benchmark for the OSIM environment POMDP.

The AutoNav fixed inclination orbit, on the other hand, is a simple algorithm which maintains a fixed inclination orbit around the asteroid for the duration of the problem. This algorithm takes images at a fixed interval and downlinks between those intervals. This can be considered a *good performance* benchmark for the problem, as AutoNav handles the thruster controls and generates a fixed inclination orbit around the body. In our simulation, we selected a polar orbit for this case.

The AutoNav random orbit selection generalizes the random orbit selection algorithm benchmark in [19]. After each full orbital period (determined by the rotation through 2 of the true anomaly of the spacecraft), AutoNav selects a random orbit and follows that orbit for the next orbital period. Similar to the fixed inclination orbit, the random orbit selection method takes images at fixed intervals and downlinks between those intervals.

## 5. Results and Discussion

Table 1 shows our performance on the Mithra asteroid, while Table 2 shows our performance on the Toutatis asteroid. In both cases, the deep reinforcement learning algorithm outperformed both the random selection methods as well as the AutoNav based methods. This is generally expected, as the deep RL algorithm has the ability to learn from its errors, and can choose to make complex decisions based on the available information. Because the benchmark algorithms are generally agnostic to the current state of the world, we can expect significantly better performance.

What is significantly more interesting are the generalization results shown in Table 3. From these results, we can see that the proposed algorithm is able to outperform the benchmark methods when generalizing to new scenarios. This result highlights significantly the potential for learning methods to increase the amount of autonomy for spacecraft in mapping missions.

Figures 5 and 6 show two exemplar training curves for the REINFORCE algorithm over the episodes of training. As we can see, the performance of the algorithm is relatively poor to begin with and improves through several discrete bumps. These bumps correspond to two major exploration events - the first is when the algorithm learns to orbit the body (by adjusting the velocity of the spacecraft), and the second is when the orbiter learns to properly downlink images, so it can take more than one image.

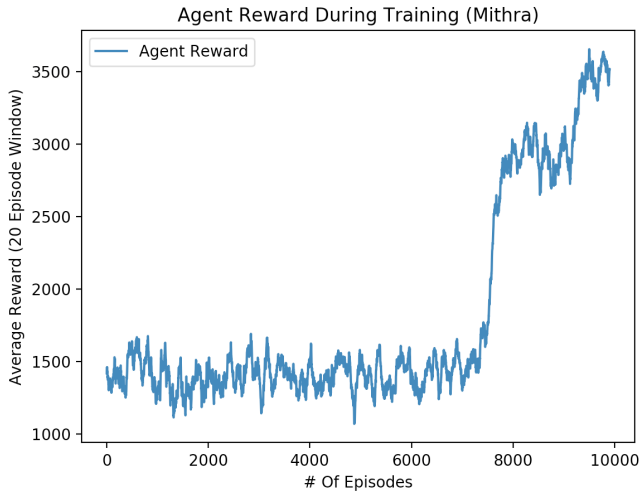


**Figure 5**. Training reward for the Mithra scenario over time. Each point is the average of the last 20 episodes. It is worth noticing the two individual increases corresponding to exploration events. During the first increase the agent learns how to orbit the asteroid while during the second, it learns about complex camera manipulation.

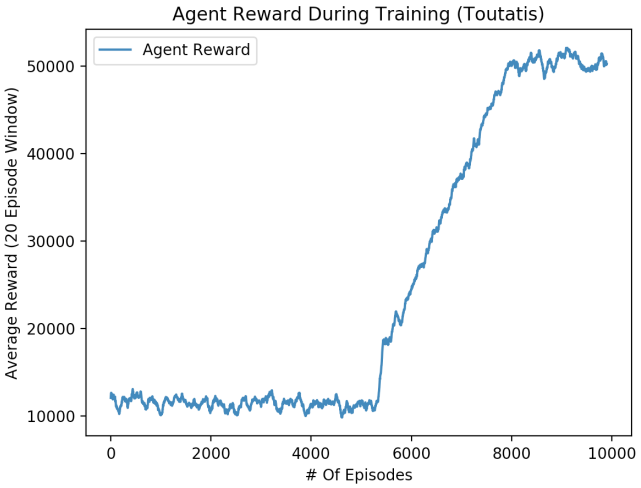


**Figure 6**. Training reward for the Toutatis scenario over time. Each point is the average of the last 20 episodes. Notice that it does not have the same double-bump behavior that Mithra does, suggesting that the camera performance was not necessarily important. This is likely because Toutatis has a more convex shape, and thus, requires less complex planning. In addition, the nearly linear increase in reward is an odd phenomenon, and may suggest that the agent is learning gradually to try a series of action leading to increased reward (such as adjusting angular velocity).

| Method | Performance | |
|---|---|---|
| Random Action | 25873.76 | 8980.833 |
| AutoNav Random Orbit | 38221.93 | 7659.323 |
| AutoNav Fixed Orbit | 43738.28 | 2349.135 |
| Proposed Deep-RL method | 49232.32 | 3873.094 |

**Table 2**. Results on the Toutatis Asteroid. Again, the sample means were collected from 100 trials. The reason that the scores are so much higher is that they are not normalized by the face score, meaning that the number of faces significantly impacts the reward score. Because Toutatis has nearly 7 times as many faces, we see that the scores are higher. It seems, however, that the problem is slightly simpler - as the normalized scores are also higher than the Mithra body.

| Training Asteroid | Test Performance | |
|---|---|---|
| Mithra | 23458.54 | 7004.353 (On Toutatis) |
| Toutatis | 1904.10 | 948.510 (On Mithra) |

**Table 3**. Results when trained on one asteroid, and tested on the other. While we do not outperform the fixed/random orbit performance (likely due to the masses of the bodies being different), we show better than random performance when transferring between the problems.

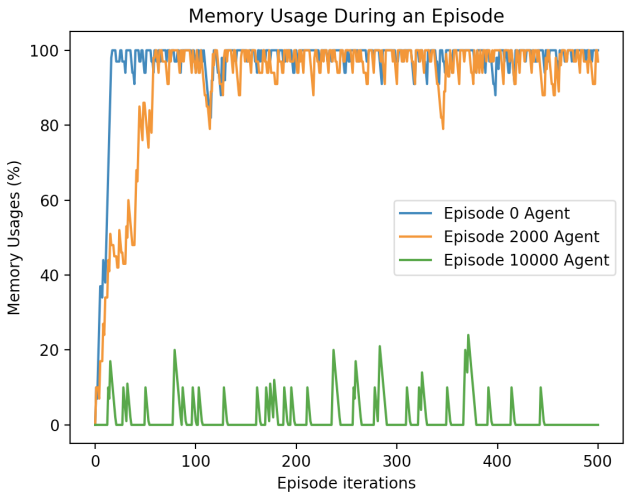


**Figure 7**. Memory usage for different agents during training time. It appears that earlier agents are incapable of memory management while later agents always try to downlink when possible. It's also interesting to notice that the images that the later agent takes are roughly evenly spaced, suggesting that it is learning to space images out, and not take them next to each other (where similar images provide little gain).

While we confirmed the first exploration event visually by examining training samples, the second event was confirmed by examining the memory usage chart given in 7. As we can see from this image, early in the training, the memory uplink/downlink is relatively random, and thus hovers around the full memory mark. In the middle of training, a large score is obtained initially by taking images (the algorithm learns to receive a reward by taking pictures), and then the memory remains used as it does not downlink (except for random behavior). At the end of training, we can see a much more useful memory utilization trend: the spacecraft will take pictures, then immediately start downlinking as much as it can, keeping persistent memory usage low.

## 6. Integration with Modern Flight Software

We have discussed the details of training a policy network on OSIM, however, in general, trusting the control of a spacecraft to a small neural network (a black-box function) seems inadvisable. In this section, we talk about our ongoing work to integrate this mapping-based planner into a modern flight system by discussing how it can interface and interact with MEXEC [37], a flight-autonomy micro-executive running on the spacecraft.

*MEXEC Background*

MEXEC is a flight-autonomy micro-executive, operating in the F-Prime flight software [38], which is responsible for planning the schedule of tasks executed on the spacecraft. The primary computational unit of MEXEC is the *task*, a collection of activities or behaviors, bundled with the execution constraints (such as time, and spacecraft state), the expected nominal execution, and some additional task parameters. The MEXEC planner uses this information to plan the execution of tasks during its planning phase, and then the MEXEC controller issues task execution commands in the context of a larger flight system.

*Towards Integration with MEXEC*

The autonomous imaging and mapping planner (AIM Module) can be implemented as an "Executive" in the MEXEC framework - a module which is responsible for creating and updating tasks which need to be accomplished. An example interaction between the AIM module and MEXEC is given in figure 8.

In this interaction, the AIM module first requests the current state from the AutoNav controller/state database on the spacecraft. It then uses this information to generate, given the current trajectory of the spacecraft, the best times that an image can be taken during the current orbit. This portion of the interaction mirrors heavily the camera/memory problem which we have modeled in the POMDP in section 3. Learning to control the camera, given constraints on uplink/downlink time, spacecraft position/attitude, and possible on-board constraints is a difficult problem on its own. It is interesting future work to explore planning for this problem using traditional optimization techniques, as well as other reinforcement learning algorithms.

In the next portion of the interaction in Figure 8, the MEXEC controller takes images according to the generated tasks from the AIM module and allows the AIM module to process each update. During this phase, AIM updates the generated map and computes the observation scores using SPC or similar mapping tools. This generated map can then be used in the next step.

In the final step, the MEXEC planner calls the update map task, which returns the "Calculate Image Session task" (the first task which calculates the image tasks from the current orbit), as well as attempts to compute the next trajectory (or orbit) for the spacecraft to obtain. This "Attempt to Obtain Orbit" task is sent back to the planner with the requested orbit.

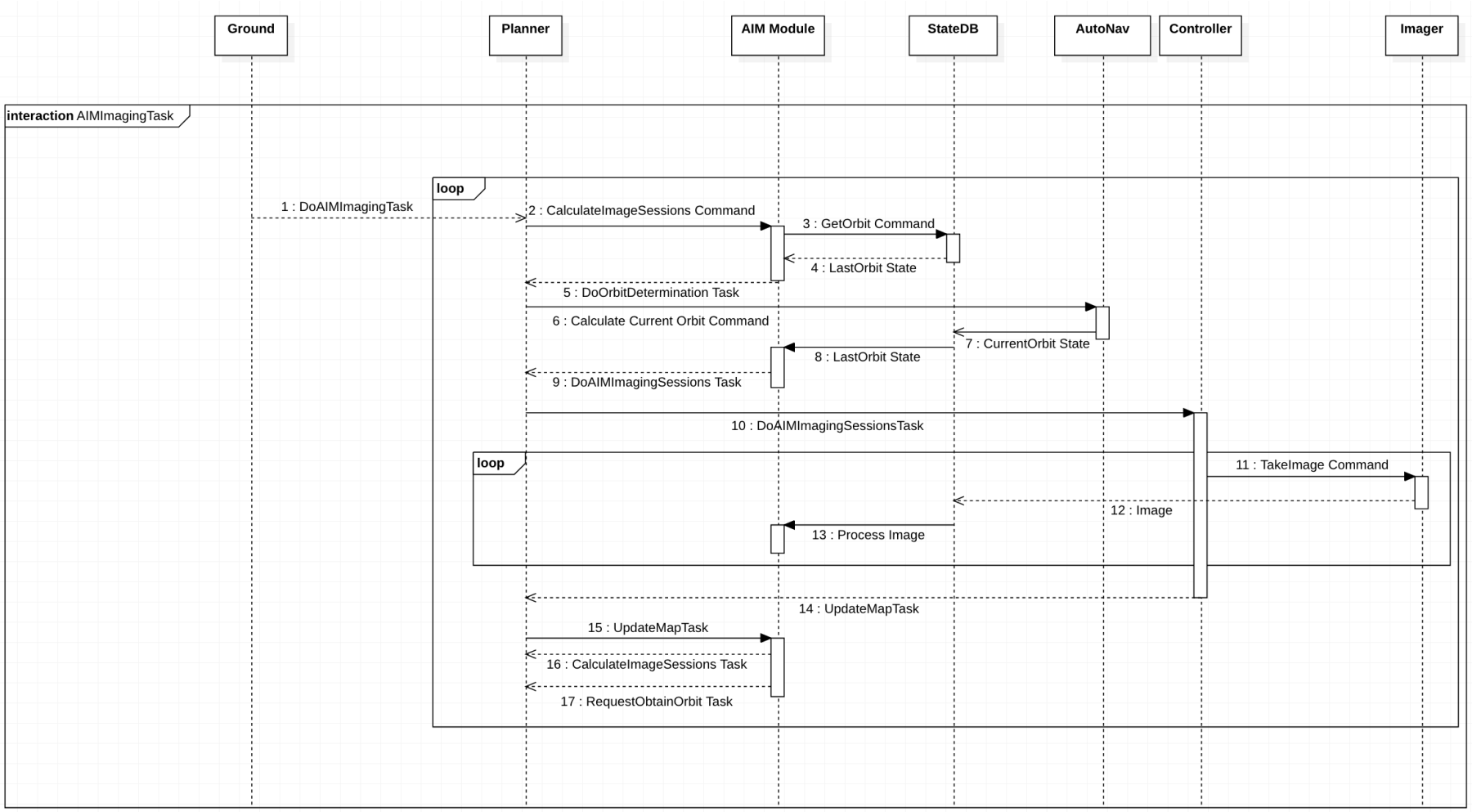


**Figure 8**. Example interaction between MEXEC and the AIM module on a hypothetical autonomous spacecraft.

By separating the control of the spacecraft from the observations, we allow the AIM module to operate well regardless of the selection of trajectory. The MEXEC planner is allowed to do whatever it wants in terms of spacecraft control (it can balance the orbit request from AIM with other GNC inputs from the spacecraft and additional spacecraft constraints), which allows for the AIM module to be implemented as a reactive component, responding to the current projected trajectory, and generating imaging events based on the expected trajectory. Such a split observation provides the ability for validation and verification of orbit maneuvers which are necessary for real-world applications of autonomy.

*Challenges of using Deep RL in Real World Autonomy*

While we have shown that deep reinforcement learning can provide promising results in terms of spacecraft control, our results provide only a small part of the results necessary before we can implement such systems in real-world autonomous spacecraft. The primary challenge which faces the implementation of Deep RL on spacecraft is the lack of validation and verification which can be performed on the RL policy networks. Because we have little ability to understand the complex decision surfaces of the networks involved, we cannot reliably validate that networks will perform how we expect. There has been promising research in this area in terms of flight systems such as in [39,40], however, additional research is necessary into explainable systems and fail-safe deep learning.

In addition to the validation and verification of spacecraft performance, we also need to be sure that we are optimizing the correct objectives. In both this work, as well as [19], we perform our optimization based on a hand-tuned reward function. During the first construction of the environment, we, as researchers, forgot to include the penalization of the reward for approaching the body and we found that the policy consistently attempted to crash the spacecraft into the body to exploit the fact that the body is much more visible from the inside. Such an example of a poorly designed reward function allows for "reward hacking behavior", and there have been many cases of policies exploiting poorly designed reward functions to generate unwanted behavior [41]. These challenges should clearly be addressed before using such techniques in a real-world environment.

One final issue that should be mentioned is that of the hardware required for running RL systems onboard spacecraft systems. Because of the necessary requirements for space-faring computational hardware, it is unlikely that the computational power available on-board modern spacecraft could handle the significant computational requirements of the RL systems. While inference in neural systems can be relatively fast (In this model, we require only the evaluation of a four layer fully-connected network consisting of solely matrix multiplications), training on-board the spacecraft to adapt to incoming data would likely be an impossibility as there is a large amount of GPU computation necessary for training some RL models (See section 4 for a discussion of our training hardware). Thus, until the computational power onboard spacecraft improves, it may be impossible to run deep RL based models with the necessary efficiency for any kind of decision making.

The above issues are only a few potential challenges of using deep reinforcement learning in real-world scenarios. While this paper provides some interesting and promising results, we believe that additional work is required and that with further research we can move towards more autonomous space-faring systems.

## 7. Conclusion

In conclusion, we have presented a novel deep reinforcement learning approach to solving the autonomous small body mapping problem. We have presented a complex formulation of the mapping problem in an end-to-end format as a Partially Observable Markov Decision Process which allows us to accurately define the interactions that an autonomous spacecraft has with the world. We then used the definition of the POMDP we have constructed to create OSIM, the orbital simulation environment, to facilitate reinforcement learning. By training a simple network on this environment with REINFORCE, we showed that not only can a policy trained with REINFORCE outperform simple benchmark models, but the policy can generalize to new scenarios with no additional training, and continue to outperform those simple benchmarks. We then discussed the splitting of the module into two distinct components to facilitate real-world usage in the context of the MEXEC flight system software, and some of the challenges that deep RL faces in real-world autonomy problems. While the area of using deep reinforcement learning for space exploration is a relatively new field, we believe that it is a promising application for learning systems and can provide a new way to think about the autonomous spacecraft of the future.

## Acknowledgments

The research was partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Partial Government sponsorship acknowledged. Thanks additionally to NVIDIA for providing the GTX Titan GPU used in this research, and to the Berkeley AI Research Laboratory for their support.

## References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[2] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[3] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[4] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," *arXiv preprint arXiv:1709.10489*, 2017.

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

[6] R. Gaskell, O. BARNOUIN-JHA, D. J. Scheeres, A. Konopliv, T. Mukai, S. Abe, J. Saito, M. Ishiguro, T. Kubota, T. Hashimoto *et al.*, "Characterizing and

navigating small bodies with imaging data," *Meteoritics & Planetary Science*, vol. 43, no. 6, pp. 1049–1061, 2008.

[7] F. Preusker, F. Scholten, K.-D. Matz, T. Roatsch, K. Willner, S. Hviid, J. Knollenberg, L. Jorda, P. J. Gutiérrez, E. Kührt *et al.*, "Shape model, reference system definition, and cartographic mapping standards for comet 67p/churyumov-gerasimenko–stereo-photogrammetric analysis of rosetta/osiris image data," *Astronomy & Astrophysics*, vol. 583, p. A33, 2015.

[8] K.-H. Glassmeier, H. Boehnhardt, D. Koschny, E. Kührt, and I. Richter, "The rosetta mission: flying towards the origin of the solar system," *Space Science Reviews*, vol. 128, no. 1-4, pp. 1–21, 2007.

[9] C. Capanna, G. Gesquière, L. Jorda, P. Lamy, and D. Vibert, "Three-dimensional reconstruction using multiresolution photoclinometry by deformation," *The Visual Computer*, vol. 29, no. 6-8, pp. 825–835, 2013.

[10] S. Bhaskaran, J. E. Riedel, and S. P. Synnott, "Autonomous target tracking of small bodies during flybys," 2004.

[11] C. Olson, R. P. Russell, and S. Bhaskaran, "Spin state estimation of tumbling small bodies," *The Journal of the Astronautical Sciences*, vol. 63, no. 2, pp. 124–157, 2016.

[12] C. Olson, "Sequential estimation methods for small body optical navigation," Ph.D. dissertation, 2016.

[13] C. Cocaud and T. Kubota, "Autonomous navigation near asteroids based on visual slam," in *Proceedings of the 23rd International Symposium on Space Flight Dynamics, Pasadena, California*, 2012.

[14] H. Xiangyu, C. Hutao, and C. Pingyuan, "An autonomous optical navigation and guidance for soft landing on asteroids," *Acta Astronautica*, vol. 54, no. 10, pp. 763–771, 2004.

[15] K. J. Åström, "Optimal control of markov processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, pp. 174–205, 1965.

[16] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[17] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable markov processes over a finite horizon," *Operations research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[18] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," *Exploring artificial intelligence in the new millennium*, vol. 8, pp. 236–239, 2003.

[19] V. Pesce, A.-a. Agha-mohammadi, and M. Lavagna, "Autonomous navigation and mapping of small bodies," *IEEE Aerospace Conference*, 2017.

[20] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of machine learning research*, vol. 4, no. Dec, pp. 1107–1149, 2003.

[21] D. Braziunas, "Pomdp solution methods," *University of Toronto*, 2003.

[22] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[24] K. Lee, S.-A. Kim, J. Choi, and S.-W. Lee, "Deep reinforcement learning in continuous action spaces: a case study in the game of simulated curling," in *International Conference on Machine Learning*, 2018, pp. 2943–2952.

[25] K. Doya, "Reinforcement learning in continuous time and space," *Neural computation*, vol. 12, no. 1, pp. 219–245, 2000.

[26] W. C. Lasunncty. (2010) Digram illustrating and explaining various terms in relation to orbits of celestial bodies. [Online]. Available: https://commons.wikimedia.org/wiki/File:Orbit1.svg

[27] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[28] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[29] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[30] S. B. Broschart, M. Abrahamson, S. Bhaskaran, E. G. Fahnestock, R. Karimi, G. Lantoine, T. A. Pavlak, and L. Chappaz, "The small-body dynamics toolkit and associated close-proximity navigation analysis tools at jpl," in *AAS Guidance and Control Conference*, 2015, pp. 1–12.

[31] B. Sicardy, J. Ortiz, M. Assafin, E. Jehin, A. Maury, E. Lellouch, R. Gil-Hutton, F. Braga-Ribas, F. Colas, J. Lecacheux *et al.*, "Size, density, albedo and atmosphere limit of dwarf planet eris from a stellar occultation," in *EPSC-DPS Joint Meeting 2011, held 2-7 October 2011 in Nantes, France. http://meetings. copernicus. org/epsc-dps2011, p. 137*, 2011, p. 137.

[32] G. Matthews, "Celestial body irradiance determination from an underfilled satellite radiometer: application to albedo and thermal emission measurements of the moon using ceres," *Applied optics*, vol. 47, no. 27, pp. 4981–4993, 2008.

[33] NASA. (2017) A collection of 3d models, textures, and images from inside nasa. [Online]. Available: https://github.com/nasa/NASA-3D-Resources/tree/master/3D%20Models

[34] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[36] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[37] V. Verma, D. Gaines, G. Rabideau, S. Schaffer, and R. Joshi, "Autonomous science restart for the planned europa mission with lightweight planning and execution."

[38] NASA, "Fprime software architecture," https://github.com/nasa/fprime, 2017.

[39] J. Ding, X. Kang, and X.-H. Hu, "Validating a deep learning framework by metamorphic testing," in *Metamorphic Testing (MET), 2017 IEEE/ACM 2nd International Workshop on*. IEEE, 2017, pp. 28–34.

[40] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*. IEEE, 2016, pp. 1–10.

[41] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.