A Reinforcement Learning Approach for Motion Planning of Hopping Rovers

Benjamin Hockman
Department of Mechanical Engineering
Stanford University
Email: bhockman@stanford.edu

***Abstract*—This paper considers the problem of navigating a hopping rover on the surface of small Solar System bodies, such as asteroids and comets. Specifically, by controlling the torques applied to three internal flywheels, the rover is able to perform controllable long-range hops and short tumbling maneuvers. The highly stochastic dynamics of bouncing in microgravity and the sequential and discrete nature of control inputs makes it difficult to apply traditional motion planning algorithms (e.g., graph search and sampling-based methods). Building on a previous RL approach to motion planning in a simplified 2D world, this project casts the full 3D motion planning problem as a tractable MDP. Model-free RL techniques (i.e. Q-iteration) with linear regression value function approximation is used to derive approximately optimal control policies which outperform previous control heuristics in simulation.**

I. Introduction

Hopping rovers have been a subject of increasing interest in the space community for exploring small Solar System bodies, such as asteroids and comets, where the microgravity environment prohibits conventional wheeled or legged locomotion [1], [2]. In fact, four small hoppers are currently en route to asteroid Ryugu aboard JAXA's Hayabusa 2 spacecraft—a MASCOT lander developed by DLR [3] and three MINERVA landers [4], which are both designed to perform small hops, albeit with minimal control.

Over the past few years, a team from Stanford and JPL have been developing a new type of hopping rover that uses *internal actuation* via three orthogonal flywheels to hop and tumble—a paradigm that enables *controllable* trajectories (see Fig. 1) [2], [5], [6].

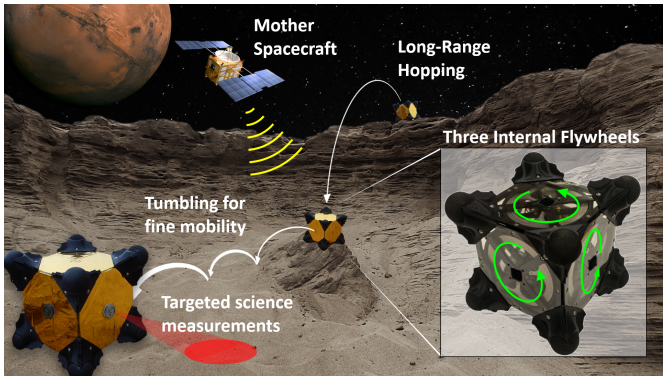


Fig. 1. Hedgehog rovers spin three internal flywheels to controllably hop and tumble in microgravity environments.

The rover, called "Hedgehog," is a cube-shaped structure with three internal flywheels that can be used to exchange momentum with the structure, causing it to spin and push off from the surface. Hedgehog has eight spikes (one on each corner) that act as feet for gripping the terrain and protect it from impacts with the surface. Depending on how the flywheels are actuated, Hedgehog can perform a variety of controlled maneuvers or *motion primitives*, ranging from long range hops (over 100 m) to short, precise tumbling. (see Fig. 2). The dynamics and control of these motion primitives has been studied in detail via dynamic models [6], simulations, and reduced gravity experiments [7], which characterizes properties such as takeoff angle and velocity, and their uncertainty[1].

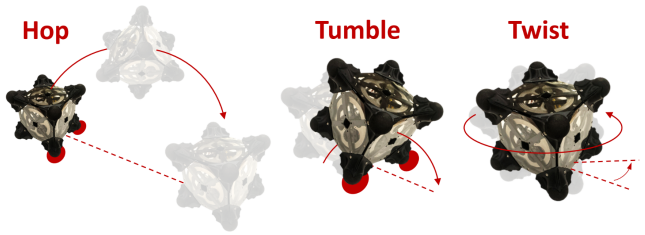


Fig. 2. Hedgehog's three fundamental motion primitives.

While the dynamics control of individual maneuvers has been studied extensively, this is just one step towards achieving the ultimate objective of *targeted, point-to-point mobility* on small bodies–a task that may require *many* hops. Concatenating multiple hopping maneuvers to achieve some mission objective (i.e. navigate to a goal region) is an open problem.

In this paper, I present one of the first, if not the first study of stochastic motion planning for microgravity hoppers, modeled as an MDP. Section II present the problem structure and formally defines the MDP. Sect. III discusses the simulation environment used as a generative model for data collection. Sect. IV proposes a model-free RL algorithm called "Least-squares fitted Q-iteration with parametric approximation" for approximating the optimal value function, and discusses feature selection. The results, including algorithm convergence and policy evaluation, are discussed in Sect. V, and the paper concludes with Sect. VI.

[1]For more information on this project and for a list of publications, see the project website at: http://asl.stanford.edu/projects/surface-mobility-on-small-bodies/

II. Problem Formulation

Because of the sequential nature of control inputs and the highly stochastic dynamics, an MDP is a natural way to approach this problem. Specifically, the state of the rover can be thought of as it's resting location and orientation on the surface of the body, or more generally, its "belief state" (although, partial observability is left for future work). The actions the rover can take are the space of motion primitives (see Fig. 2). The reward model is a design choice that can be structured to encode mission objectives (e.g., big bonus in goal states and penalties in hazardous states). A previous study considered a simplified 2D version of this problem, whereby the discretized (1D) state space was a series of piecewise-linear surface patches and the action space was also a 1D discretized span of velocities. The low-dimensional discrete state and action spaces allowed the transition model to be estimated directly from simulation data as a MLE of the transition matrix [8].

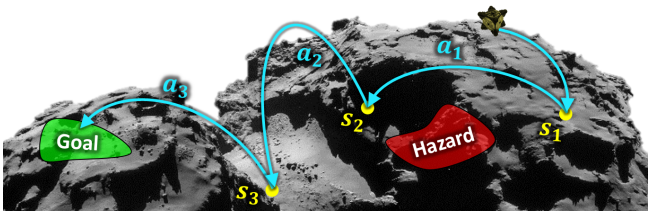


Fig. 3. Motion planning of a hopping rover formulated as an MDP.

This approach does not scale well to higher dimensions. In the full 3D formulation, the state (position and orientation) lives in $\mathbb{R}^6$ and actions (velocity vector) live in $\mathbb{R}^3$ (see Fig. 3). This dimensionality can be approximately reduced by making two simplifying assumptions: (1) ignore the rover's orientation ($S : \mathbb{R}^6 \rightarrow \mathbb{R}^3$) and (2) fix the elevation angle of the hop velocity relative to the local surface normal vector ($A : \mathbb{R}^3 \rightarrow \mathbb{R}^2$). In practice, reducing the state to only the location on the surface is reasonable assuming a lower level controller to adjust orientation[2]. The hop angle assumption is, in fact, a physical constraint of the hopping control (generally, the rover hops at a 45^o angle) [6]. However, even with this dimensionality reduction, discretizing the state and action spaces requires very course binning to make the transition function storable. The need for large, "global" state transitions as well as controlled *local* movement prohibits state discretization. Thus, for this study we consider the *continuous* state space:

$$\boxed{\textbf{States: } S = [x, y, z] \subset \mathbb{R}^3.} \tag{1}$$

and the action space in $\mathbb{R}^2$ is discretized as:

$$\boxed{\begin{aligned}\textbf{Actions: } & A = [A_1, A_2], \\ & A_1 = \{\psi_1, ..., \psi_{n_\psi}\}, A_2 = \{v_1, ..., v_{n_v}\},\end{aligned}} \tag{2}$$

[2]On a convex body, $\mathbb{R}^2$ is sufficient to parametrize the surface (i.e. latitude and longitude), but such parametrization is not always possible on small bodies with irregular, non-convex shapes

where the direction (ψ_i) and speed (v_i) are uniformly distributed bins from 0 to 2π and v_{min} to v_{max} respectively. For this study, we choose $n_\psi = 8$ and $n_v = 10$, which is large enough to sufficiently cover the action space, but small enough to remain tractable.

For traditional robot motion planning, it is often easy to construct cost functions that capture physical objectives (e.g., minimizing distance, energy, time, etc.). This is less straightforward when casting a planning problem as an infinite horizon MDP with discounted rewards. For hopping rovers, we would like to incentivize actions that *minimize the expected time to real the goal*. However, since actions take various amounts of time, discounting a terminal reward alone is not sufficient. Accordingly, an additional penalty is added to the reward function:

$$\boxed{\begin{aligned}\textbf{Rewards: } & R(s_g, \cdot) = 1, \quad R(s_h, \cdot) = -1, \quad \text{and} \\ & R(s, a) = \frac{-t(s, a)}{t_{\text{max}}}, \quad \gamma = 0.99\end{aligned}} \tag{3}$$

where $s_g \in S_{\text{goal}} \subset S$ is the goal region(s), $s_h \in S_{\text{hazard}} \subset S$ defines hazardous regions, and $t(s, a)$ is the total travel time for taking action a from state s to state s'. With this construction, and for low discounting ($\gamma = 0.99$), the optimal policy approximates a minimum-time policy. In other words, the optimal value function V^* can be interpreted as a time-to-goal metric, where $\mathbb{E}(\text{time to goal}) \approx t_{\text{max}}(1 - V^*)$.

III. Data Collection

So far, Eqs. (1)–(3) define four of the five elements required in an MDP (S, A, T, R, γ). The *transition model* requires special attention, as it cannot be modeled explicitly due to the highly nonlinear and stochastic dynamics of ballistic flight in irregular gravity fields and bouncing on uneven surfaces. Instead, individual trajectories can be *sampled* from a high-fidelity simulation environment, which forward propagates the nonlinear dynamics and captures uncertainty in the (1) the initial hop vector (i.e. control errors), (2) rebound velocities, and (3) gravity field. Figure 4 shows an example of 20 trajectories randomly sampled from a single state/action pair.

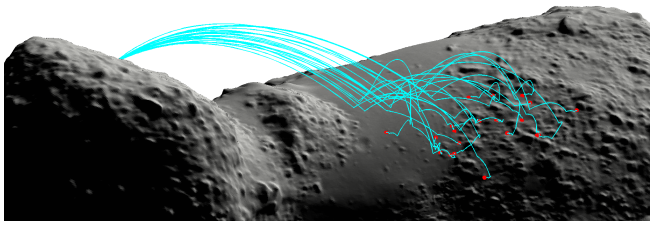


Fig. 4. Monte Carlo simulations of 20 hopping trajectories on the surface of asteroid Itokawa using a 3-million facet shape model.

Specifically, the simulation models the rover as a simple particle (i.e. its CG) that is subject to gravitational and contact forces. A polyhedral gravity model is used to propagate the ballistic trajectories, which integrates over the volume of a shape model and assumes constant density (cite Scheeres). The initial launch vector and rebound velocities are sampled [BH:

from predetermined distributions characterizing control uncertainties and surface properties, respectively. A data set of $500,000$ trajectories was generated (in roughly 7 hrs) to train the RL algorithm in Sec. IV, with inputs $\{s_i, a_i\}$ and outputs $\{t_i, s_{i+1}\}$. States and actions were sampled mostly at random, with some bias towards "interesting" regions. For example, more weight was given to "flat surface" states that the rover is more likely fall into. At the same time, the samples are spatially distributed so that the data set can be used to train policies for various objectives (i.e. visiting different locations).

IV. Reinforcement Learning Approach

Although model-based RL demonstrated good performance in the simplified 2D model considered in [8], the dynamics in 3D are non-linear and much more chaotic, making it difficult to approximate the transition model. For example, small perturbations in the hop velocity can yield drastically different settling distributions. Instead, I chose to investigate a model-free approach that directly approximates the value function. Specifically, I implemented an "offline least-squares fitted Q-iteration with parametric approximation" algorithm, as outlined in [9]. For this application, it is important to leverage *offline* computation as much as possible so that the implementation on the CPU-constrained rover hardware is minimized (ideally, just a lookup table encoding the policy). For simplicity and convergence guarantees, I chose to estimate the Q-functions for each discrete set of actions as a linear combination of state features:

$$Q(s,a) = \phi^T(s)\theta_a, \quad \phi(s), \theta_a \in \mathbb{R}^{n_f} \tag{4}$$

Thus, the goal is to fit the $n_a = n_\psi n_v$ weight vectors (θ_a) for each unique set of actions. A batch variant of fitted Q-iteration is outlined in Algorithm 1.

Algorithm 1 Least-squares fitted Q-iteration

Input: discount factor γ,
samples $\{(s_i, a_i, s'_i, r_i) | i = 1, ..., n_s\}$
feature mapping $\phi(s_i)$

1: initialize parameter matrix $\Theta_0 = [\theta_{1,0}, ..., \theta_{n_a,0}]$
2: **repeat** at every iteration $l = 0, 1, 2, ...$
3: **for** $i = 1, ..., n_s$ **do**
4: $Q_{i,l+1} \leftarrow r_i + \gamma \text{max}_{a'}[\phi(s'_i)^T \Theta_l]$
5: **end for**
6: **for** $a = 1, ..., n_a$ **do**
7: $\theta_{a,l+1} \leftarrow \text{argmin}_\theta \sum_{i|a_i=a} (Q_{i,l+1} - \phi(s_i)^T \theta_{a,l})^2$
8: **end for**
9: **until** Θ_{l+1} satisfactory

Output: $\hat{\Theta}^* = [\theta_{1,l+1}, ..., \theta_{n_a,l+1}]$

Line 1 initializes the parameter vectors, which can speed convergence if chosen wisely. Leveraging the intuition that the value function should decrease monotonically as the distance from the goal increases, θ_0 was set to $[0, ..., 0, 1, 0, ..., 0]^T$, where the non-zero element j corresponds to a radial basis feature around x_g (the center of S_g): $\phi_j(s_i) = e^{-||s_i - x_g||_2}$ (see the first heatmap in Fig. 5).

Lines 3–5 calculate Q_i for each data sample based on the observed reward (see Eq. (3)) and the optimal value at the next state, discounted by γ. This can be implemented as a 1-line matrix multiplication:

$$Q_{l+1} = \begin{bmatrix} r_1 \\ \vdots \\ r_{n_s} \end{bmatrix} + \text{colmax} \begin{bmatrix} \phi^T(s'_1) \\ \vdots \\ \phi^T(s'_{n_s}) \end{bmatrix} \Theta_l. \tag{5}$$

Lines 6–8 involve partitioning the data into action sets and performing n_a least squares problems:

$$\theta_{a,l+1} = (\Phi_a \Phi_a^T)^{-1} \Phi_a Q_{a,l+1}, \quad \Phi_a = \begin{bmatrix} \vdots \\ \phi^T(s'_k) \\ \vdots \end{bmatrix}_{a_k = a} \tag{6}$$

The stopping condition for convergence is met when $||\theta_{a,l+1} - \theta_{a,l}||_2 < \theta_{\text{tol}}, \forall a = \{1, ..., n_a\}$. Because of the efficient use of data, this algorithm typically converges in a few tens of iterations. The complexity of Eq. (5) is $O(n_s n_a n_f)$ and the complexity of Eq. (6) is $O(n_s n_f^2)$, indicating that large feature vectors are particularly expensive.

A. Feature Selection

As with any linear function approximator, the "optimal" value function is only as good as the span of features used to represent it. Recall that by discretizing the action space, ϕ is only a function of the state, s_i. It is also important to note that, although the raw state is represented with $(x, y, z) \in \mathbb{R}^3$, we are only concerned with the value of the states *on the surface*—a subspace within $\mathbb{R}^3$. In other words, Eq. (4) has value for all $s \in \mathbb{R}^3$, but only points on the surface are physically realizable.

Some features can be intelligently designed based on "expert knowledge" of the dynamics and the mission objectives. For example, since the locations of the goal and hazard regions are predefined, radial basis features centered on these regions are likely to be strong indicators of the value of nearby states. These features are:

$$\phi_{g_1}(s) = \frac{d_{g,\text{max}} - d_g(s)}{d_{g,\text{max}}}, \qquad \phi_{g_2}(s) = e^{-d_g(s)},$$

$$\phi_{g_3}(s) = \mathbf{1}\{d_g(s) < D_g\}, \qquad \phi_{g_4}(s) = \frac{d_{g,\text{max}}}{d_{g,\text{max}} + d_g(s)},$$

where $d_g(s_i) = ||x_{s_i} - x_g||_2$ is the distance of state s_i from the center of goal region g, and D_g is its radius. These features represent (in order) linear, exponential, binary, and inverse functions of the distance from the center of each goal (and hazard). Thus, for a mission scenario with n_g goal regions and n_h hazard regions, there are a total of $4(n_g + n_h)$ "hand-crafted" features.

To capture finer spatial resolution in the value function across the surface, an additional set of features is required. There are many features that have been successfully used for parametric function approximation such as distributed radial basis functions (RBFs), indicator functions for aggregated state

clusters, and monomials. For this problem, I chose to add a set of k^{th} order monomials of the raw state:

$$\phi_j(s) = x^{k_1} y^{k_2} z^{k_3}, \quad \forall \{k_1, k_2, k_3, \in \mathbb{N}_0 | k_1 + k_2 + k_3 \leq k\}.$$

This produces a set of $\binom{k+3}{3}$ linearly independent monomial features. Thus, choosing k presents the classic tradeoff between bias and variance and depends on the size of the data set. For the $500,000$ trajectory samples, $50,000$ were randomly selected to be held out as part of the test set. Through cross-validation on this test set, $k = 5$ was chosen as the best representation, which produces 56 monomial features. All features were normalized to lie in the range of $[0, 1]$.

V. Results

With the MDP definition from Sect. II, the generative sampling model from Sect. III, and the RL algorithm in Sect. IV, we can now evaluate the performance of this method. First, we consider the convergence of Algorithm 1 and then we discuss the quality of the policies that it generates.

A. Value Function Convergence

One of the nice things about using linear value function approximation in a Q-iteration algorithm, is that there are provable convergence guarantees, as discussed in [9]. In other words, the error of the least squares fit in line 7 of Algorithm 1 will decrease monotonically. However, this is of course only true for states represented in the data set. In other words, for a given action, the value function may represent a good fit in regions where there is sufficient data, but may swing wildly to unrealistic values in other regions with sparse or no coverage. This is why I chose to sample states mostly at random and keep the number of actions low, resulting in distributed state coverage and a large number of data samples per action (6000).

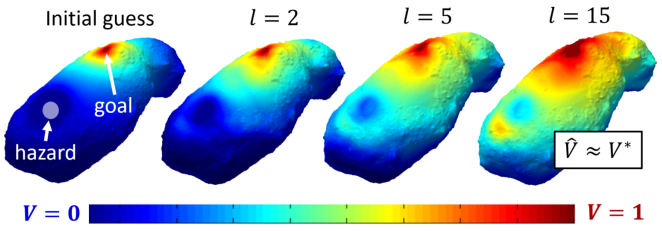


Fig. 5. Heat maps of the value function projected onto the surface of asteroid Itokawa, showing convergence of Algorithm 1 within 15 iterations. Smart initializations can help speed convergence.

Thankfully, for this problem, there is a nice way to plot the value function over the entire state space: as a surface heat map. Figure 5 shows a value function heat map overlayed onto the surface of asteroid Itokawa at various iterations in Alg. 1, from initialization to convergence. In this example, a single goal and hazard region are defined according to the reward model in Eq. 3, and $t_{\max}$ is set to be 8 hours[3]. The optimal value function had a mean absolute error of 0.04 (which alone is not that useful).

[3]Motion is typically *very* slow in microgravity, with the duration of a single hop taking anywhere from minutes to hours.

So do these heatmap plots of the value function make sense? Well, recall from Sect. II that the reward model is constructed in such a way that gives the optimal value function a beautiful physical interpretation: it represents the expected time-to-goal according to $\mathbb{E}(\text{time to goal}) \approx t_{\max}(1 - V^*)$, which is only true for very low discounting ($\gamma = 0.99$). So we would expect the value function of states "close" to the goal region to be higher than more distant states, which is indeed the case in Fig. 5. Also, we can see how the presence of hazards also affects the nearby state values, which can be interpreted as the additional time it takes to carefully go around the hazard. For $t_{\max} = 8$ hrs, the optimal value function for states farthest from the goal is very close to zero, indicating that the expected time to reach the goal from those location is nearly $t_{\max}$. Note that an additional "do nothing" state was added to prevent negative values (i.e. $R(s, 0) = 0$).

B. Policy Evaluation

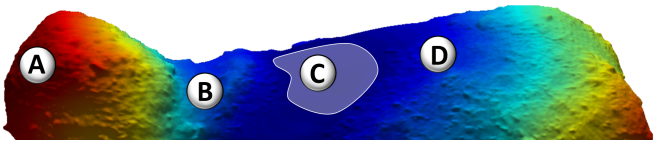
The convergence of the value function produces low error residuals and tends to agree with intuition, but this says nothing about the quality of the control policies. As a baseline for comparison, the learned control policy was compared to a heuristic policy, which is a best attempt at controlling the rover without any simulation data to learn from. The "hop-towards-the-goal" heuristic is a 3D extension of the control heuristic proposed in [5] and is computed as follows:

1) Calculate the plane that contains the current state position (s_i), the goal position (s_g), and the local gravity vector at this vector $g(s_i)$.
2) Define the hop velocity unit vector ($\hat{v}$) that lies in this plane and also obeys the elevation angle constraint relative to the local surface normal (this should have a unique solution).
3) Given the approximate plane of motion, the initial velocity unit vector, and assuming constant gravity, use 2D projectile motion equations to calculate the required speed to hit the goal state exactly.
4) Limit the hop speed to $v_{\max}$ or the estimated escape velocity at s_i, whichever is lower.

In summary, this heuristic directs the rover to hop directly at the goal region and assumes planar, parabolic trajectories and ignores bouncing dynamics. Planar motion is a reasonable assumption for small-scale hops, but becomes much worse for more distant hops, which often produce curved 3D trajectories. Thus, this heuristic policy has been shown to work well in simulation and actual experiments, but only for relatively flat and obstacle-free terrains. The goal here is to show that the policy generated from the RL approach presented in Sect. IV is more effective in complex terrains.

To compare the performance of this heuristic policy to the learned policy, three mission scenarios were constructed, cast as the MDP reward model from Eq. 3, and executed in simulation 1000 times (refer to Fig. 6). Two metrics were used to compare the performance of each policy: (1) the "% success" which simply reflects the percentage of simulations that reach the goal within $t_{\max}$ without entering the hazardous

region, and (2) the "mean time" (of the successful samples) to reach the goal.



Start	Goal	Hazard	Policy	% Success	Mean Time (hrs)
D	B	none	Heuristic	95	3.8
			Learned	99	2.9
D	B	C	Heuristic	41	3.8
			Learned	98	3.6
D	A	none	Heuristic	3	14.3
			Learned	96	5.7

Fig. 6. Three navigation scenarios across the surface of asteroid Itokawa (plotted with a geopotential color map), comparing the learned and heuristic policies. "% Success" indicates the percentage of the 1000 simulations that reached the goal safely (i.e. not escaping or entering the hazardous region) within $t_{\max}$. The "mean time" reflects the mean total time to reach the goal *of the successful samples*

The first scenario represents motion planning task in which the rover must traverse relatively flat, obstacle-free terrain, so it is no surprise that the heuristic policy has comparable performance to the learned policy. The second scenario introduces a large hazardous region between the starting and goal states[4]. Ignorant of this hazard, the heuristic policy often hops or bounces into it, whereas the learned policy exhibits "intelligent avoidance" actions such as hopping around it, or aggressively over it. Finally, one very difficult situation for a hopping rover is to reach locations with high gravitational potential (i.e. at the top of a hill), as illustrated by the third scenario in Fig. 6 (note the extreme geopotential color difference between A and D). The randomness usually drives the residual bounces towards regions of lower gravitational potential, making higher states "unstable" in a sense. Indeed, the heuristic policy drives the rover straight up the steepest part of the ascent, which typically causes it to bounce back down and thus, has a low success rate. On the other hand, the learned policy is able to strategically position the rover into states from which the goal region is more easily reachable. To observe what these policies look like executed in simulation, follow the link to my video online at: https://youtu.be/j-bgiX4QjkQ, which shows the execution of the learned policy for the first scenario.

VI. Conclusion

This paper presented the motion planning problem for the navigation of hopping rovers on small Solar System bodies. By casting this problem as an MDP and using model-free reinforcement learning methods to approximate the optimal value function, the extracted policies have demonstrated a previously unobtained level of performance. The batch least-squares fitted Q-iteration with linear value function approximation algorithm (Alg. 1) made efficient use of simulation data to converge on the optimal policy with very few iterations. In simulations, these learned policies outperformed a "best-guess" heuristic policy in every scenario. To the best of the author's knowledge, these results constitute one of the first ever demonstrations of MDP motion planning for hopping rovers on small bodies.

This work leaves a number of extensions open for future work. First, defining the state in $\mathbb{R}^3$ (and therefore defining the value function in $\mathbb{R}^3$), is a burdensome representation, requiring many more features than necessary. In the future, I will use conformal mapping techniques to parametrize the irregular surface model in $\mathbb{R}^2$. Combined with PCA feature-reduction, this would have much more representation power and produce more accurate value function approximations.

One issue that became apparent in observing policy execution is the effect of local variation in surface slope. Recall that the inclination angle of the hop vector is constrained *relative to the local surface normal* to 45^o. As a result, the same action executed in nearby states that have different slopes (i.e. due to surface roughness) may produce drastically different trajectories. The "smooth" k^{th} order monomial features used here do not have sufficient spatial resolution to capture sharp value function changes like this. In the future, I will augment the state variable with some notion of surface slope to account for this.

Some parallel work being done by our colleagues at JPL is the problem of localization, primarily through onboard vision-based techniques, whereby state estimation will likely have some uncertainty. Future work will consider tractable ways of capturing this partial observability in a POMDP formulation.

Finally, NASA and other space agencies are primarily concerned with *minimizing risk* of missions—a subtle but important difference from maximizing expected rewards. In the future, I will also explore the use of time-consistent risk metrics such as "conditional value at risk" (CVaR) presented in [10], which allows for imposing probabilistic risk constraints (e.g. $< 1\%$ probability of hitting an obstacle).

[4]Scientists believe that this area may indeed be covered in very soft, loose granular soil, or "regolith," in which the hopper would likely sink.

Acknowledgment

I would like to thank Prof. Marco Pavone for advising me along this project, the rest of the project team (Robert Reid, Issa Nesnas, Andy Frick, and Julie Castillo at JPL, and Jeff Hoffman at MIT) for their insightful contributions, NASA Innovative Advanced Concepts (NIAC) for funding the project, and François Germain for the instructive feedback.

References

[1] "Decadal Survey Vision and Voyages for Planetary Science in the Decade 2013–2022," National Research Council, Tech. Rep., 2011, available at http://solarsystem.nasa.gov/2013decadal/.

[2] J. C. Castillo Rogez, M. Pavone, I. A. D. Nesnas, and J. A. Hoffman, "Expected Science Return of Spatially-Extended In-situ Exploration at Small Solar System Bodies," in *IEEE Aerospace Conference*, Big Sky, MT, Mar. 2012, pp. 1–15. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6187034

[3] C. Dietze, S. Herrmann, F. Kuß, C. Lange, M. Scharringhausen, L. Witte, T. van Zoest, and H. Yano, "Landing and mobility concept for the small asteroid lander MASCOT on asteroid 1999 JU3." in *61st International Astronautical Congress*, 2010.

[4] "JAXA Hayabusa mission," JAXA, Tech. Rep., 2011, available at http://hayabusa.jaxa.jp/e/index.html.

[5] R. Allen, M. Pavone, C. McQuin, I. A. D. Nesnas, J. C. Castillo Rogez, T.-N. Nguyen, and J. A. Hoffman, "Internally-Actuated Rovers for All-Access Surface Mobility: Theory and Experimentation," in *Proc. IEEE Conf. on Robotics and Automation*, Karlsruhe, Germany, May 2013, pp. 5481–5488. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6631363

[6] B. Hockman, A. Frick, I. A. D. Nesnas, and M. Pavone, "Design, Control, and Experimentation of Internally-Actuated Rovers for the Exploration of Low-Gravity Planetary Bodies," in *Conf. on Field and Service Robotics*, Toronto, Canada, Jun. 2015, in Press.

[7] B. Hockman, R. G. Reid, I. A. D. Nesnas, and M. Pavone, "Experimental Methods for Mobility and Surface Operations of Microgravity Robots," Tokyo, Japan, Oct. 2016, in Press.

[8] B. Hockman and M. Ahumada, "An mdp approach to motion planning for hopping rovers on small solar system bodies," AA228 Final Project, December 2015.

[9] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2010, vol. 39.

[10] Y.-L. Chow and M. Pavone, "Stochastic optimal control with dynamic, time-consistent risk constraints," in *2013 American Control Conference*. IEEE, 2013, pp. 390–395.