A Reinforcement Learning Approach for Motion Planning of Hopping Rovers

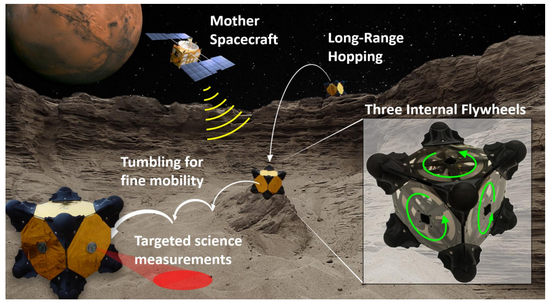
Benjamin Hockman, bhockman@Stanford.edu

Stanford University

Autonomous Systems Lab

Objective

Hopping rovers are a promising form of mobility for exploring small Solar system bodies, such as asteroids and comets, where gravity is too low for traditional wheeled rovers (<1mg). Stanford and JPL have been investigating a new internally-actuated rover concept, called "Hedgehog," that can perform long range hops (>100m) and small tumbling maneuvers simply by applying torques to internal flywheels [1], [2], [3].



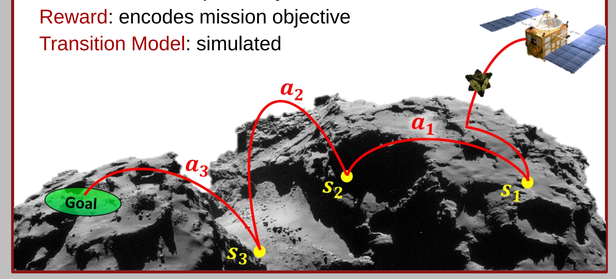
While the controllability of single maneuvers (i.e. hopping trajectories) has been studied extensively via dynamic models, simulations [1], and reduced gravity experiments [2], the ultimate objective is to achieve targeted point-to-point mobility. Akin to a game of golf, the sequential hopping maneuvers with highly stochastic bouncing is well modeled as an MDP [3].

Motion Planning as an MDP

After deployment from the mothership, the rover bounces and comes to rest at location $\boldsymbol{s_1}$ on the surface. Then, the rover hops with velocity $\boldsymbol{a_1}$, bounces, and eventually settles at location $\boldsymbol{s_2}$ on the surface, collecting reward $\boldsymbol{r_1}$. This process repeats until the rover reaches one or many goal regions.

Summary:

State: rover location on surface
Action: nominal hop velocity
Reward: encodes mission objective
Transition Model: simulated



MDP Formulation

State: Unlike spherical bodies, the surface locations on highly irregular bodies cannot always be parametrized in $\mathbb{R}^2$ (i.e. latitude / longitude). Thus, the state space is consider as a 2D manifold within $\mathbb{R}^3$ implicitly defined by a surface mesh model:

$$S \subset \mathbb{R}^3.$$

Actions: While the Hedgehog rover can control its hop speed (v) and azimuthal direction (ψ), the inclination angle relative to the surface is constrained to 45º. Thus, the action space is $A = \mathbb{R}^2$, which is discretized as follows: ($n_\psi = 8$, $n_v = 10$)

$$A = [A_1, A_2] \quad A_1 = \{\psi_1, \dots, \psi_{n_\psi}\} \quad A_2 = \{v_1, \dots, v_{n_v}\},$$

Where ψ_i and v_i are uniformly distributed bins from 0 to 2π and v_{min} to v_{max}, respectively.

Rewards: We want 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:

$$R(s_g, \cdot) = 1, \qquad R(s_h, \cdot) = -1, \qquad R(s, a) = \frac{-T}{T_{max}},$$

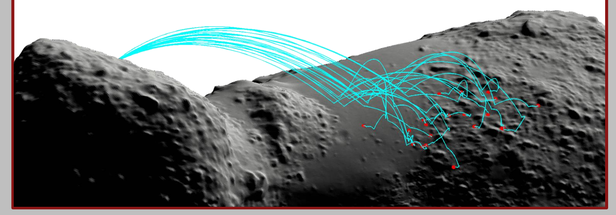
Where $s_g \in S_{goal} \subset S$ is the goal region(s), $s_h \in S_{hazard} \subset S$ defines hazardous regions, and T is the total travel time. Thus, the optimal value function will roughly correlate to the optimal time-to-goal relative to the maximum allowable time, T_{max}.

Data Collection

Due to the highly irregular gravity fields and chaotic bouncing dynamics, an explicit state-transition model is not available. Instead, individual simulated trajectories are sampled from a high-fidelity *generative* model, which captures uncertainty in:

- the initial hop vector (i.e. control errors),
- rebound velocities (due to unknown surface properties),
- gravity field, which assumes a constant-density model

500,000 trajectories were simulated in ~7hrs. States and actions were sampled mostly at random, with some bias towards "interesting" regions.



Reinforcement Learning Model

Due to the chaotic transition probabilities and continuous, high dimensional state space, model learning is intractable. Instead, a model-free approach is taken to learn the value functions directly. Specifically, I implemented a least-squares fitted Q-iteration algorithm with linear function approximation.

Algorithm 1 Least-squares fitted Q-iteration

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

1: initialize parameter matrix $\theta_0 = [\theta_{(\psi_1, v_1)}, \dots, \theta_{(\psi_{n_\psi}, v_{n_v})}]$
2: **repeat** at every iteration $l = 0, 1, 2, \dots$
3: **for** $i = 1, \dots, n_s$ **do**
4: $Q_{l+1,i} \leftarrow r_i + max_{a'}[\phi(x_i')^T \theta_l]$
5: **end for**
6: $\theta_{l+1} \leftarrow \text{argmin}_\theta \sum_{i=1}^{n_s} (Q_{l+1,i} - \phi(x_i)^T \theta_l)^2$
7: **until** θ_{l+1} satisfactory

Output: $\hat{\theta}^* = \theta_{l+1}$

- Each set of actions has it's own parameter vector, $\theta_{(\psi_i, v_j)}$
- Lines 3-5 can be implemented as a matrix multiplication:

$$Q_{l+1} = \begin{bmatrix} r_1 \\ \vdots \\ r_{n_s} \end{bmatrix} + \max_{\text{row}} \begin{bmatrix} \phi^T(x_1') \\ \vdots \\ \phi^T(x_{n_s}') \end{bmatrix} \begin{bmatrix} \theta_{(\psi_1, v_1)} & \cdots & \theta_{(\psi_{n_\psi}, v_{n_v})} \end{bmatrix}$$

- Line 6 involves partitioning the data and solving $n_a = n_\psi n_v$ least squares problems:

$$\theta_{(\psi_i, v_j),\, l+1} = \left(\Phi^{\mathrm{T}}\Phi\right)^{-1} \Phi^T Q_{(\psi_i, v_j),\, l+1}, \quad \Phi = \begin{bmatrix} \vdots \\ \phi^T(x_k') \\ \vdots \end{bmatrix}_{a_k = (\psi_i, v_j)}$$

$$i = 1, \dots, n_\psi, \qquad j = 1, \dots, n_v$$

This batch algorithm makes efficient use of data and typically converges within about 20 iterations.

Feature Selection

We need a set of features that map the raw data ($\boldsymbol{x} \in \mathbb{R}^3$) to value functions. First, a set of "radial" exponential and binary features are "expertly" designed for each goal and hazard:

$$\phi_g = e^{-\|x - x_g\|}, \quad \phi_{g+n_g} = \mathbf{1}\{\|\boldsymbol{x} - \boldsymbol{x}_g\| < d_g\}, \quad g = \{g_1, \dots, g_{n_g}\}$$

For additional spatial representation, a set of k^{th} order monomials are also included:

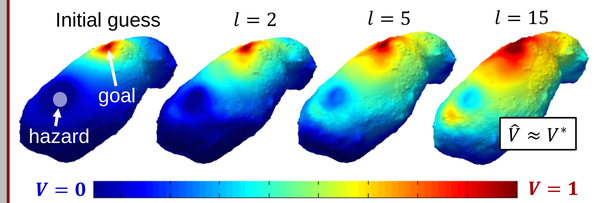
$$\phi_j = x^{k_1} y^{k_2} z^{k_3}, \qquad \forall \{k_1, k_2, k_3 \in \mathbb{N}_0 | \; k_1 + k_2 + k_3 \leq k\}$$

This produces $\binom{k+3}{3}$ features. Thus, choosing k presents the tradeoff between bias and variance and depends on the size of the data set. Through cross-validation, $k = 5$ provided the best fit on the test data set, for a total of 61 features.

Results and Discussion

Value function convergence

With educated initial guess, least-squares fitted Q-iteration converges within roughly 15 iterations:



The specially constructed reward function gives the optimal value function a beautiful interpretation:

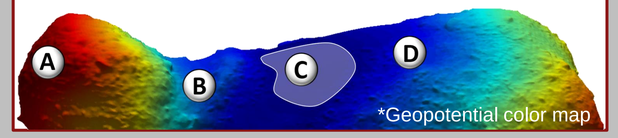
Optimal time-to-goal $\approx T_{max}(1 - V^*)$

In this example on Asteroid Itokawa, the goal can be reached from anywhere on the surface within 7 hours (in expectation).

Policy Evaluation

The extracted policy ($\pi(s) = \max_a \phi^T(s)\hat{\theta}^*$) is executed in simulation and compared to a "hop-towards-the-goal" heuristic. The learned policy universally outperformed the heuristic, especially with hazards and large potential gains.

Start	Goal	Hazard	Policy	% Success	Mean Time (hrs)
D	B	none	Heuristic	95	3.8
			Learned	99	3.2
D	B	C	Heuristic	41	3.8
			Learned	99	3.7
D	A	none	Heuristic	3	14.3
			Learned	96	5.3



Conclusion

This study presents the *first ever demonstration of autonomous mobility for hopping rovers on small bodies*. Future work will consider other constraints such as battery life, and partial state observability from localization uncertainty.

References

[1] B. Hockman, et al., Design, control, and experimentation of internally-actuated rovers for the exploration of low-gravity planetary bodies. In Conf. on Field and Service Robotics, June 2015. *Best Student Paper Award.*

[2] B. Hockman, R. Reid, I. A. D. Nesnas, and M. Pavone. Experimental Methods for Mobility and Surface Operation of Microgravity Robots. International Symposium on Experimental Robotics, October 2016

[3] B. Hockman and M. Pavone. Autonomous Mobility Concepts for the Exploration of Small Solar System Bodies. Stardust Final Conference on Asteroids and Space Debris, ESTEC, Netherlands. November 2016

Learn more about our project at: http://asl.stanford.edu/projects/surface-mobility-on-small-bodies/