Motion Planning and Stochastic Control with Experimental Validation on a Planetary Rover

Rowan McAllister, Thierry Peynot, Robert Fitch and Salah Sukkarieh

***Abstract*— Motion planning for planetary rovers must consider control uncertainty in order to maintain the safety of the platform during navigation. Modelling such control uncertainty is difficult due to the complex interaction between the platform and its environment. In this paper, we propose a motion planning approach whereby the outcome of control actions is learned from experience and represented statistically using a Gaussian process regression model. This model is used to construct a control policy for navigation to a goal region in a terrain map built using an on-board RGB-D camera. The terrain includes flat ground, small rocks, and non-traversable rocks. We report the results of 200 simulated and 35 experimental trials that validate the approach and demonstrate the value of considering control uncertainty in maintaining platform safety.**

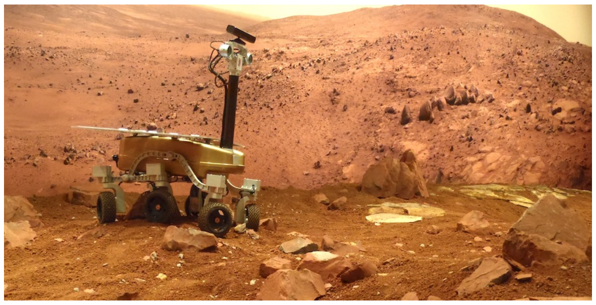


Fig. 1. Planetary rover "Mawson" used for experimental validation, shown in the Mars yard at the Powerhouse Museum in Sydney, Australia.

I. INTRODUCTION

Motion planning for mobile robots in unstructured environments must consider various forms of uncertainty. One significant source of uncertainty in outdoor terrain is *control uncertainty*. Robots such as planetary rovers are designed for mobility in challenging environments, but understanding the associated control uncertainty for the purpose of motion planning is difficult due to the complexity of this type of environment. It is critical to consider control uncertainty in motion planning, particularly in environments that expose the robot to the risk of serious mechanical damage. We are interested in this problem in the context of planetary rovers [1]. Our goal is to navigate while maintaining the safety of the platform in potentially dangerous terrain.

The goal of classical geometric motion planning is to minimise time or distance while avoiding obstacles [2]. The conceptual distinction between free space and obstacles for planetary rovers, however, is less clear. It is important to avoid obstacles, but it is also desirable to avoid free space where, due to control uncertainty, the robot has high likelihood of encountering an obstacle during execution. This situation cannot be modelled by simple distance thresholds surrounding obstacles because risk varies across free space, and is probabilistic.

Accurately predicting executed behaviour in response to a given control input is difficult for planetary rovers due to complex terramechanics [3]. For previously unobserved terrain, prior models of terrain properties may not be available. It is thus important to model control uncertainty with a method that can be feasibly executed online during operation of the robot, and to validate such a model experimentally.

Our approach is to build a statistical model directly from observed behaviour, represented as a Gaussian process (GP). We consider uncertainty in the heading of the platform and in distance travelled. We use this GP model to build a stochastic transition function for use in motion planning. The planning goal is to compute a policy that allows the robot to reach a given goal location while maintaining the safety of the platform. We assume that a map of the environment is available, represented as a digital elevation map. Platform safety is represented by a cost function over this terrain map, which is constructed *a priori* using on-board sensors. We compute the policy using dynamic programming (DP), where the resolution of discretised geometric states is equal to that provided in the elevation map.

In this paper, we present the details of our approach and its implementation for the planetary rover shown in Fig. 1. The environment consists of flat terrain, traversable rocks, and non-traversable rocks. We learn GP models for rock traversal that map environment features to a distribution of resulting rover configurations (in state space) for two types of control actions. The cost map is constructed from data collected by an on-board RGB-D camera. We report results from 200 simulated and 35 experimental trials that evaluate the rover's ability to traverse flat terrain and small rocks while avoiding non-traversable rocks. We compare rover performance in executing policies constructed with and without control uncertainty. Our results show empirically that planning with control uncertainty improves the rover's ability to navigate while avoiding non-traversable areas, and demonstrate the value of planning under uncertainty for planetary rovers using a real platform in a realistic environment.

The work was supported by the Australian Research Council, the Australian Space Research Program *Pathways to Space: Empowering the Internet Generation*, the Australian Centre for Field Robotics, and the Air Force Research Laboratory under agreement FA2386-10-1-4153.

R. McAllister, T. Peynot, R. Fitch and S. Sukkarieh are with the Australian Centre for Field Robotics, The University of Sydney, Australia {r.mcallister,tpeynot,rfitch}@acfr.usyd.edu.au

II. RELATED WORK

A common approach for considering control uncertainty in motion planning is to express the uncertainty as a cost and then to plan a path that minimises this cost assuming deterministic control [4], [5]. Another family of approaches plans a path using a sampling-based algorithm, and then evaluates the control uncertainty along the path selected [6], [7]. In classical motion planning, the desired path is provided to a feedback controller for execution. Various forms of control strategies (such as LQG) can be used to model potential deviations from the path and hence to select a path with least risk in terms of platform safety [8], [9].

For non-determinisitic systems Markov decision processes (MDPs) are commonly used to formulate problems in motion planning with uncertainty [2], [10]. Control uncertainty is represented as a stochastic transition function, and a policy can be computed using dynamic programming [11]. The partially-observable Markov decision process (POMDP) is another common formulation [11]. However, these techniques are most often evaluated in simulation only and there is a critical need for further validation using real robots.

Recent work by Brooks and Iagnemma [12] models control uncertainty as a function of terrain in a self-supervised learning framework. This approach uses visual features to classify terrain types and learn associated proprioceptive mechanical properties.

Finally, physics-based approaches that study terramechanics provide detailed mobility models by considering features such as soil cohesion and density [13]. Statistical mobility prediction using terramechanics has been proposed that generates a Gaussian distribution over predicted future states on homogeneous terrain [3]. However, it is difficult to precisely model non-homogeneous terrain that includes rocks of different sizes and shapes that may move in reaction to the force exerted by a rover wheel.

In our work we directly search for a path with low risk of entering a non-traversable area, but our model of stochastic actions is tied to observed environmental features that vary across the terrain and learned through experience. We furthermore consider risk at the level of primitive actions and construct a policy that is executed directly. Our approach uses statistical regression techniques in performing the inference and direct learning is applied showing meaningful improvements to motion planning are possible without a complex terramechanics model.

III. MOTION PLANNING AND LEARNED CONTROL UNCERTAINTY

With the aim of reaching a given goal region *safely* while optimising the total cost of traversal over the *executed* trajectory, our approach is to take into account the stochasticity of the control of the robot at the planning stage. This requires modelling the control uncertainty, which we achieve by experience, using machine learning. This section first describes the planning algorithm used in our approach (Sec. III-A), followed by the presentation of the learning technique used to model the control uncertainty (Sec. III-B).

A. Planning Algorithm

We compute a control policy for the robot using dynamic programming (DP). DP computes an optimal policy with respect to a discrete set of (stochastic) primitive motions and given resolution of the state space [2]. DP is a feasible method in low dimensional state spaces; in our problem the state s can be defined using two lateral dimensions x and y and one dimension for orientation ψ, i.e., $s = \{x, y, \psi\}$.

This formulation treats the motion planning problem as a Markov Decision Process, assuming accurate localisation and stochastic control. The optimal motion policy is computed using the Bellman optimality equation iteratively:

$$V^*(s) = \max_a \Big\{ \sum_{s'} P(s'|s,a)(R(s'|s,a) + \gamma V(s')) \Big\}, \quad (1)$$

where s is the robot state (discretised into uniform cells), a is an action (or motion primitive) from the action set A, and $\gamma = 1$ is the discount factor. The transition function $P(s'|s,a)$ describes the probability that a state-action pair (s,a) transitions to state s'. The optimal policy is given by:

$$\pi^*(s) = arg \max_a \Big\{ \sum_{s'} P(s'|s,a)(R(s'|s,a) + \gamma V(s')) \Big\}, \quad (2)$$

where the reward $R(s'|s,a)$ is computed from a cost map that represents the difficulty of the terrain. $P(s'|s,a)$ is not known *a priori*. Therefore, these state transitions are learned from experience, as described below.

B. Learning-based Mobility Prediction

$P(s'|s,a)$ can be expressed using a PDF of the relative transition between states, $p(\Delta s|s,a)$, where $\Delta s \equiv s' - s$:

$$P(s'|s,a) = \int p(\Delta s|s,a) f(s + \Delta s, s') \, \mathrm{d}\Delta s, \quad (3)$$

where $f(s_1, s_2) = 1$ if the discretisation of s_1 corresponds to s_2 and $f(s_1, s_2) = 0$ otherwise. In unstructured terrains, Δs may strongly depend on factors such as the terrain profile along the executed path and also the action executed. Terrain profiles are represented by a vector of features (or characteristics): $\boldsymbol{\lambda}(s,a)$. Therefore, $p(\Delta s|s,a)$ is learned as a function of $\boldsymbol{\lambda}(s,a)$ and a:

$$p(\Delta s|s,a) = p(\Delta s|\boldsymbol{\lambda}(s,a), a). \quad (4)$$

The estimation of relative change in state is achieved using Gaussian Process (GP) regression. GP is a standard framework to learn a model of spatially correlated data and provides estimations with uncertainty. The GP framework is especially effective in cases where the input data are sparsely populated.

We use the GP formulation from [14]. The input vector $\boldsymbol{x}$ is a function of the terrain features, shifted such that the input has zero mean, i.e., $\boldsymbol{x} = \boldsymbol{\lambda} - \bar{\boldsymbol{\lambda}}_{train}$, where $\bar{\boldsymbol{\lambda}}_{train}$ is the mean of each terrain feature in the training data. We define $\Delta s_{i,a}$ as the i^{th} single-valued component of the change of state Δs resulting from executing action a. We define a GP for each $\Delta s_{i,a}, i \in [\![1, N]\!], a \in A$, where $N = Dim(s)$. The output value $y_{i,a}$ of one of these

GPs is defined as $y_{i,a} = \Delta s_{i,a} - \bar{\Delta s}_{i,a}$, where $\bar{\Delta s}_{i,a}$ is the mean value of Δs_i across all executions of action a in the training data. $y_{i,a} = f_{i,a} + w_{i,a}$ has a zero mean and variance σ_n^2 equal to the variance of the additive noise $w_{i,a}$, i.e., $p(y_{i,a}|f_{i,a}) = \mathcal{N}(0, \sigma_n^2)$.

The covariance function used to describe the spatial correlation between two input vectors is the squared exponential:

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x'})^\top \Lambda^{-2}(\boldsymbol{x}-\boldsymbol{x'})\right) + \sigma_n^2 I, \quad (5)$$

where σ_f^2 is the input variance and Λ is a length scale matrix of diagonal elements that describes the smoothness of the input data. The predictive distribution is given by a Gaussian,

$$p(\Delta s_{i,a}|\boldsymbol{\lambda}(s,a)) - \bar{\Delta s}_{i,a} = p(f_*|X, Y, \boldsymbol{x}_*) \sim \mathcal{N}(\mu_*, \Sigma_*), \quad (6)$$

with predictive mean

$$\mu_* = K(\boldsymbol{x}_*, X)[K(X,X) + \sigma_n^2 I]^{-1} Y,$$

and variance

$$\begin{aligned}\Sigma_* = {} & K(\boldsymbol{x}_*, \boldsymbol{x}_*) \\ & - K(\boldsymbol{x}_*, X)[K(X,X) + \sigma_n^2 I]^{-1} K(X, \boldsymbol{x}_*),\end{aligned}$$

where X is the $n \times m$ matrix of all n training input vectors, Y is the $n \times 1$ vector of all training output values, and $\boldsymbol{x}_*$ is the test input vector. $K(X, \boldsymbol{x}_*)$ is a covariance matrix which stores the covariance of each training input value against the test input values.

Thus, $p(\Delta s_{i,a})$ can be computed for untraversed terrain profiles using Eq. (6), where the test input vector $\boldsymbol{x}_*$ has not been observed directly but rather estimated by employing a kinematics model of the robot over the series of states that would be traversed by executing action a from state s.

Finally, our planner considers the uncertainty in each component Δs_i separately by using the full distribution learnt from Δs_i and the expectation of the other components. Representing $\boldsymbol{\lambda}(s,a)$ as $\boldsymbol{\lambda}$, Eq. (4) is calculated as:

$$\begin{aligned}& p(\Delta s|\boldsymbol{\lambda}, a) = p(\{\Delta s_1, ..., \Delta s_i, ..., \Delta s_N\}|\boldsymbol{\lambda}, a) \approx \\ & \{\mathbb{E}(\Delta s_1|\boldsymbol{\lambda}, a), ..., p(\Delta s_i|\boldsymbol{\lambda}, a), ..., \mathbb{E}(\Delta s_N|\boldsymbol{\lambda}, a)\}. \quad (7)\end{aligned}$$

C. System Outline

Fig. 2 shows an outline of the system. Once an elevation map of the robot's local environment has been computed, a kinematics model is used to estimate both the terrain profile characteristics and cost of traversal ($R(s'|s,a)$) at each state-action pair. Characteristics of the terrain profile, as observed by the Inertial Measurement Unit (IMU) and the localisation system, are used as input data to train the GP offline. The GP can then be used to estimate the stochastic transition function $P(s'|s,a)$ on terrain similar to that encountered during training. Using both $P(s'|s,a)$ and $R(s'|s,a)$, the motion planner computes the value function (Eq. (1)) over the observed state space to follow greedily as per the policy given in Eq. (2).

IV. IMPLEMENTATION

This section describes the implementation of the proposed approach on our experimental Mars rover shown in Fig. 3(a).

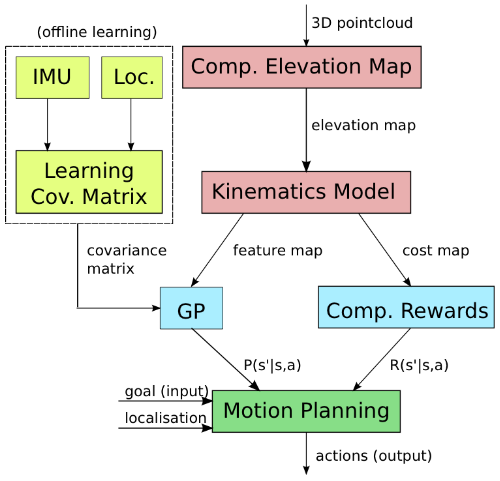
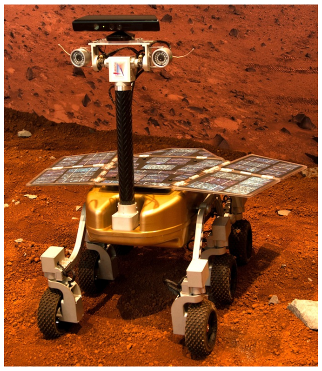
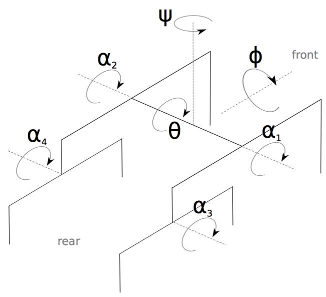


Fig. 2. System Outline. Colours indicate perception (red), offline learning (yellow), estimation (blue) and planning (green).



(a) Mawson



(b) Orientation and Internal angles

Fig. 3. The Mawson Rover (a) and its chassis configuration (b).

A. The Robot

Mawson is a six-wheeled rover with individual steering servo motors on each wheel and a Rocker-bogie chassis. The platform is equipped with:

- an RGB-D camera (Microsoft Kinect) mounted on a mast, tilted down 14°, used for terrain modelling and localisation,
- a 6-DOF IMU used to measure the roll (ϕ) and pitch (θ) of the robot,
- three potentiometers to observe the configuration of the chassis by measuring both bogie angles and the rocker differential (α_i in Fig. 3(b)).

For localisation and terrain modelling we use the RGB-D SLAM algorithm [15], implemented in the Robot Operating System (ROS) [16], which uses data from the RGB-D camera to perform simultaneous localisation and mapping (SLAM) online. An elevation map is generated from the point clouds supplied by the RGB-D camera by distributing elevation points in a regular Cartesian grid. The grid resolution is $0.025m \times 0.025m$.

B. Kinematics Model

To predict the attitude angles $\{\phi, \theta\}$ and chassis configuration $\{\alpha_2 - \alpha_1, \alpha_3, \alpha_4\}$ of the rover at given positions on the elevation map, we use a method similar to [17]. Although it does not take into account the dynamics of the platform, this simplified model is a sufficient approximation since the rover operates at low speeds. This kinematics model is used to: 1) estimate terrain profile characteristics for (s, a) pairs the planner queries, and 2) compute a cost map of the observed terrain (see Fig. 2).

1) Feature Map: From the estimates of the configurations of the rover on the map, a feature map is built for each (s, a) pair using the GP model to predict $P(\Delta s)$ for each (s, a) pair the planner considers.

2) Cost Map: The cost function chosen to generate the cost map from the elevation map penalises large absolute values of roll, pitch, and configuration angles of the chassis at a given position $s = \{x, y, \psi\}$:

$$cost_{terrain}(s) = (cost_{\phi\theta}(s) + cost_{\alpha}(s))^2, \quad (8)$$

where

$$cost_{\phi\theta}(s) = (\phi^2 + \theta^2), \quad (9)$$

$$cost_{\alpha}(s) = (\alpha_2 - \alpha_1)^2 + \alpha_3^2 + \alpha_4^2. \quad (10)$$

Since the configuration of the robot at a given position on the elevation map depends on its orientation, a 2D (x, y) cost map needs to be generated for each discretised orientation. The result is a 3-dimensional (x, y, ψ) cost map.

C. Planning

1) State Space: As mentioned in Sec. III-A, the rover's state s is defined as its position and orientation:

$$s \triangleq \{x, y, \psi\} \in \mathbb{R}^3. \quad (11)$$

This definition specifies all other orientations and internal angles (ϕ, θ, α_i) at each state using the kinematics model. State resolution was required to be smaller than the uncertainty bounds of resultant positions of actions in order for uncertainty to be considered by the DP. A discretisation of $0.025m \times 0.025m \times \frac{\pi}{32} rad$ is sufficient.

2) Action Set: We define two action types for the rover: *crabbing* and *rotation*. Crabbing corresponds to executing a straight line translation in the xy-plane by a given distance and heading, with no change in ψ, and constant linear velocity ($0.11m/s$). The rover is able to crab in any direction. Rotation is a spin-on-the-spot motion primitive at constant angular velocity ($0.24rad/s$). It changes ψ by a given magnitude. In total, the action set A is composed of 2 rotation and 8 crabbing motion primitives:

$$A \triangleq \{\text{rotate}(\pi/4), \text{rotate}(-\pi/4), \text{crab}(0.3m, n\pi/4)\ \forall n \in [\![-3, 4]\!]\}. \quad (12)$$

Restricting the actions to this set was the result of a trade-off between rover dexterity and algorithm complexity.

3) Reward Function: The reward $R(s'|s, a)$ of an action is defined as the negative of the average cost of states that lie on a linear interpolation between a start state s and a resultant state s':

$$R(s'|s,a) = -\xi - \frac{1}{K}\sum_{i=0}^{K} Cost\Big(s_x + \frac{i}{K}(s'_x - s_x), s_y + \frac{i}{K}(s'_y - s_y), s_\psi + \frac{i}{K}(s'_\psi - s_\psi)\Big), \quad (13)$$

where $\xi = 0.003$ is a small penalty used to deter excessive motions on flat terrain and K is the sampling resolution.

D. Learning and GP

To achieve mobility prediction from training data, we used the GP implementation from [14]. These training data were obtained through experimental runs of the rover executing each action a from A multiple times, while logging discretised sets of $\{\phi, \theta, time, s, a\}$ continuously. The training data collected comprise the rover's attitude angles, provided by the on-board IMU, and the deviation from the expected motion when executing a given action, measured using the localisation system. Due to the left-right symmetry of the platform, training was only required on 6 of the 10 actions from A (Fig. 4 shows which set of training data can be combined between symmetric actions).

Due to slow localisation updates, Δs could only be measured at the end of each action primitive. However, multiple values of $\{\phi, \theta\}$ were recorded during the execution of an action primitive, resulting in a vector of these values $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$, representing the terrain profile. Terrain profiles were encoded more compactly by extracting features of the evolution of $\{\boldsymbol{\phi}, \boldsymbol{\theta}\}$ to avoid overfitting. A combination of terrain profile features were tested with GP regression using cross validation. The features (vector of functions $\boldsymbol{\lambda}$) which produced the lowest Root Mean Square error between the GP mean estimations and test datum output were chosen:

$$\boldsymbol{\lambda} \triangleq \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}, \quad (14)$$

where

$$\lambda_1(\boldsymbol{\phi}, \boldsymbol{\theta}) = max(\phi_j - \phi_i),\ \forall i, j : i < j \quad (15)$$

$$\lambda_2(\boldsymbol{\phi}, \boldsymbol{\theta}) = min(\phi_j - \phi_i),\ \forall i, j : i < j \quad (16)$$

$$\lambda_3(\boldsymbol{\phi}, \boldsymbol{\theta}) = max(\theta_j - \theta_i),\ \forall i, j : i < j \quad (17)$$

$$\lambda_4(\boldsymbol{\phi}, \boldsymbol{\theta}) = min(\theta_j - \theta_i),\ \forall i, j : i < j, \quad (18)$$

i.e., largest increase of ϕ, largest decrease of ϕ, largest increase of θ and largest decrease of θ during an action primitive.

When planning over previously untraversed states, $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are estimated using the kinematics model from a state-action pair: $\hat{\boldsymbol{\phi}}(s, a)$ and $\hat{\boldsymbol{\theta}}(s, a)$. The $\boldsymbol{\lambda}$ functions are then applied to these estimates, which are input to the GP. Thus, the expression

$$p(\Delta s_i | \boldsymbol{\lambda}(s, a), a) = p(\Delta s_i | \boldsymbol{\lambda}(\hat{\boldsymbol{\phi}}(s, a), \hat{\boldsymbol{\theta}}(s, a), a))$$

is substituted in Eq. (7).

The components Δs_i of Δs (see Sec. III-B) were defined according to radial coordinates, as per the control space of crabbing and rotations, opposed to the Cartesian representation of the state space $\{x, y, \phi\}$ itself. The Δs_i components we consider are heading and distance travelled for crabbing actions

$$\Delta s_1 \quad = \Delta s_{head} \quad = atan2(\Delta y, \Delta x) \tag{19}$$
$$\Delta s_2 \quad = \Delta s_{dist} \quad = \sqrt{(\Delta x)^2 + (\Delta y)^2}, \tag{20}$$

and yaw for rotation actions

$$\Delta s_3 = \Delta s_{yaw} = \Delta \psi. \tag{21}$$

Therefore, Δs is represented by the tuple

$$\Delta s \triangleq \{\Delta s_{head}, \Delta s_{dist}, \Delta s_{yaw}\}. \tag{22}$$

In the rest of the paper, we define control errors (δs) as the difference between observed change in state (Δs) and "ideal" change in state ($\tilde{\Delta s}$) (i.e., if the controller followed the action-command perfectly); i.e., $\delta s = \Delta s - \tilde{\Delta s}$, a tuple analagous to Eq. (22).

V. EXPERIMENTAL RESULTS

We evaluated our approach both in simulation and through experiments with a planetary rover robot. Our experimental methodology consisted of two phases. First, we learned statistics of control error through empirical trials, described in Sec. V-A. Then, we performed navigation experiments using these learned data to build the motion planner's stochastic transition function. Sec. V-B describes experiments in simulation, and Secs. V-C and V-D describe experiments using the robot.

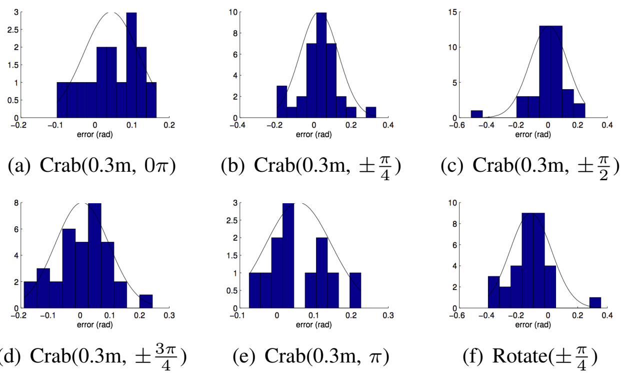
Experiments were conducted in the Mars Yard environment shown earlier in Fig. 1. This environment is $117m^2$ in area and was designed to reproduce typical Martian terrain. It contains rocks of various sizes, small craters, and various grades of sand, dirt and gravel.

A. Training on Flat and Rough Terrain

Training was conducted for two cases: flat-terrain traversal and rough-terrain traversal. For the flat terrain case, control errors were learned by executing multiple runs for each action. Rough-terrain training additionally involved traversal of various rocks (one at a time).

In flat-terrain traversal, variations of values of $\{\phi, \theta\}$ were negligible, therefore, motion errors were learnt with respect to action only. Fig. 4 shows example heading errors for each action, and Table I lists learned statistical values. Although the terrain was mostly flat, the variance in motion primitive error is significant, which validates the need to take uncertainty into account in the planning. We found that the distributions could reasonably be approximated by a Gaussian.

During rough-terrain traversal, various rocks were traversed. Note that in some cases some rocks shifted under the weight of the rover, slightly sinking into the sand or rolling over. These types of situations, which are extremely difficult to predict by modelling, were therefore captured in our learning data. As expected, the control errors were more significant in rough terrain data. Table II shows the GP hyperparameters obtained.



(a) Crab(0.3m, 0π) (b) Crab(0.3m, $\pm\frac{\pi}{4}$) (c) Crab(0.3m, $\pm\frac{\pi}{2}$)
(d) Crab(0.3m, $\pm\frac{3\pi}{4}$) (e) Crab(0.3m, π) (f) Rotate($\pm\frac{\pi}{4}$)

Fig. 4. Mobility prediction by action on flat terrain.

TABLE I
MOBILITY PREDICTION BY ACTION, GP FEATURES NOT INCLUDED

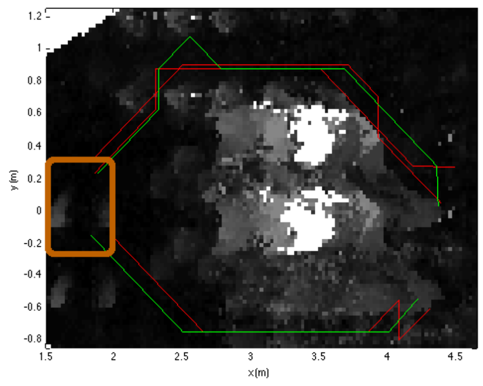
Action:	crab 0π	crab $\pm\pi/4$	crab $\pm\pi/2$	crab $\pm 3\pi/4$	crab π	rotate $\pm\pi/4$
Error:	δs_{head}	δs_{head}	δs_{head}	δs_{head}	δs_{head}	δs_{yaw}
Flat Terrain						
mean (rad)	0.043	0.028	0.004	0.006	0.058	-0.117
std. (rad)	0.074	0.103	0.127	0.091	0.088	0.140
# samples	15	34	39	34	12	32
Rough Terrain - *marginalised by Action*						
mean (rad)	0.044	0.010	0.060	0.037	0.037	-0.063
std. (rad)	0.081	0.115	0.158	0.117	0.089	0.119
# samples	43	58	58	48	35	54

B. Simulation of Flat Terrain Traversal

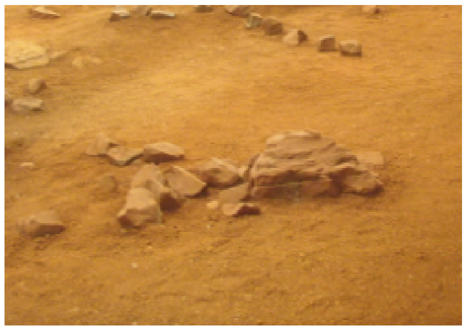
We simulated the robot traversing flat terrain. Control uncertainty was simulated using the learned data described in Sec. V-A. The robot's environment was simulated using a point cloud acquired by the robot's RGB-D camera. This environment is a roughly flat area with a cluster of rocks, shown in Fig. 5(b). Trials consisted of placing the robot randomly around the cluster of rocks and directed to a unique goal region opposite the rock cluster. We used a cost function where any rock on the terrain appears as an obstacle.

TABLE II
GP HYPERPARAMETERS TRAINED FROM TRAVERSALS IN ROUGH TERRAIN

Action	Error	Λ_{11}^{-2}	Λ_{22}^{-2}	Λ_{33}^{-2}	Λ_{44}^{-2}	σ_f	σ_n
crab 0π	δs_{head}	0.056	0.052	10.59	10.56	0.251	0.029
crab $\pm\frac{\pi}{4}$	δs_{head}	0.068	0.010	0.063	0.087	0.183	0.032
crab $\pm\frac{\pi}{2}$	δs_{head}	0.644	0.448	0.015	0.024	0.000	0.111
crab $\pm\frac{3\pi}{4}$	δs_{head}	1.89	0.058	0.068	1.78	0.284	0.040
crab π	δs_{head}	0.055	0.042	0.151	2.47	0.163	0.032
rotate $\pm\frac{\pi}{4}$	δs_{yaw}	0.007	0.246	0.036	0.014	0.135	0.042



(a) Simulated trajectories



(b) The real terrain

Fig. 5. (a) shows a few samples of simulated trajectories to navigate around a cluster of rocks (shown in (b)). The cost on the map is shown as levels of grey, with white indicating the highest cost, for a single orientation value: the rover facing left. The brown rectangle on the left indicates the common goal region. Red trajectories were computed without considering uncertainty, while green trajectories considered uncertainty.

Planning without uncertainty (whereby the expectation of the transition function $\mathbb{E}(P(\Delta s|\boldsymbol{\lambda}(s,a),a))$ learned is used instead of the full distribution) and planning considering uncertainty were each tested 100 times. When planning considering uncertainty, the flat-terrain learning data were used, whereby Δs_{head} was considered during crab actions, and Δs_{yaw} was considered during rotate actions. Resultant trajectories were assessed in light of the known Δs_{head} and Δs_{yaw} distributions to determine the probability of colliding with a rock as well as the expectation of accumulated cost for each trajectory. Results obtained for both methods can be compared using the statistics on all *executed* trajectories in Table III. These statistics represent: the averages of total cost accumulated form start position to goal, the probability of hitting an obstacle summed over the entire trajectory and the minimum distance to an obstacle over the trajectory. Fig. 5(a) shows a few examples of trajectories executed.

Results in Table III highlight two of the major consequences of planning without uncertainty: a platform will be more likely to collide with an obstacle and will, on average, accumulate more cost in traversing to the goal region. In fact, when planning without uncertainty, the projected accumulated cost will always be underestimated when the robot deviates at least once from the lowest cost path it computes,

TABLE III

SIMULATED PLANNING AROUND A CLUSTER OF ROCKS

No Uncertainty	Cost	Prob.	Min. Dist. (m)
mean	1.132	0.028	0.154
std.	0.605	0.101	0.131
max.	6.879	0.68	0.530
min.	0.727	0	0.015
Head. Uncertainty	Cost	Prob.	Min. Dist. (m)
mean	1.080	0.005	0.161
std.	0.138	0.035	0.113
max.	1.449	0.34	0.505
min.	0.733	0.0	0.025

which is frequently the case with imperfect control. Thus planning without uncertainty cannot provide any guarantee of total cost accumulated to reach a goal region, which is important when a decision needs to be made regarding if a goal region is worth visiting up to a certain cost of traversing there. The safety issue of rock collisions occurs when the planner follows paths very close to an obstacle without considering consequences of slight deviations from its path.

C. Experiments of Flat Terrain Traversal

We conducted a series of experiments using the physical robot traversing flat terrain. The robot navigated from a starting position to a goal region whilst avoiding large rocks. The terrain was flat sand and gravel, however due to the imperfect control on the loose terrain, the control uncertainties were significant, as shown earlier in the training data in Sec. V-A.

Trajectories were planned and executed on the robot 10 times for planning with uncertainty (in heading during crabbing actions and yaw during rotation actions). For comparison, 10 trajectories were also planned and executed without accounting for uncertainty. In both cases, two different starting points were used, while the goal region was the same.

Fig. 6 shows a few examples of the actual trajectories executed by the rover. Occlusions shown in the cost map are due to "shadows" of rocks when they were observed by the robot.

The results of planning with uncertainty of heading during crabbing actions and yaw during rotation actions are compared statistically against results of planning without uncertainty in Table IV. Results indicate that the robot would, on average, plan wider berths around rocks with uncertainty considered and execute safer paths in practice.

D. Experiments on Unstructured Terrain: Traversing Rocks

We also conducted a series of experiments where the robot traverses rough terrain. Trained GP models were used to predict control uncertainty as described in Sec. V-A. As in the training phase, the experiments were conducted in unstructured and rough terrain because of the presence of rocks that the robot sometimes has to travel across, but they were limited to areas with negligible terrain slopes.

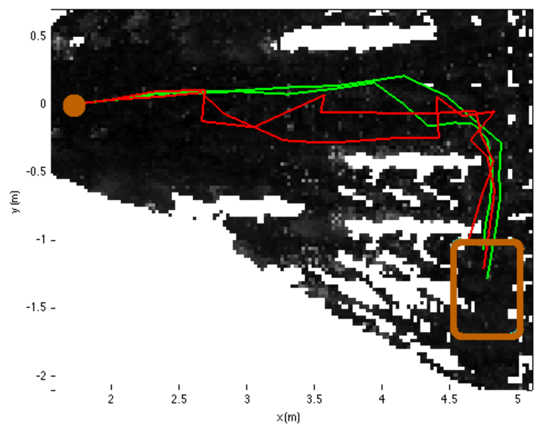


Fig. 6. Example of trajectories taken to avoid several rocks on otherwise flat terrain. Red: without uncertainty considered. Green: with uncertainty in heading. The starting position is indicated by the brown circle on the left and the goal region is shown as the brown rectangle at the bottom right. Planning with uncertainty generates paths that are less sensitive to control error.

The learned rough-terrain models were limited to traversal of one rock at a time, and so a rock field (Fig. 7) was set up accordingly. However, the layout of rocks was dense enough to cause the robot to traverse multiple rocks while navigating to the goal region.

In addition to Δs_{yaw} during rotation actions, two types of uncertainty were considered (separately) with crab actions: Δs_{head} and Δs_{dist}. Fig. 8 shows an example policy computed by our planning algorithm.

Resultant trajectories in Fig. 9 show that whilst each planning method attempted to navigate between rocks, the method of planning without uncertainty (red) would tend to "zig zag" more severely in attempts to attain the least possible cost path, finely navigating every rock. Methods considering uncertainty (green, cyan), by contrast, appear smoother. The "zig zag" action sequences can actually result in more rock traversals in total, due to a greater distance travelled in the rock field.

Results shown in Table V indicate that policies chosen by the planning with uncertainty case were favourable, with less accumulated cost. In these tests, considering heading uncertainty was most significant to overall cost. In this table, "stuck states" refers to situations where the rover dug one of its wheels in next to a rock during the test and could not initially mount the rock as the result. This occurred 20% of the time when no uncertainty was considered. When considering uncertainty in distance travelled, a stuck state still occured. However, planning with heading uncertainty again made more impact, with all stuck states avoided. Although the sample size is limited, this trend in the data supports the claim that our approach to consider the control uncertainty can significantly improve the safety of the platform at the execution of planned trajectories.

Note that in these experiments we considered two sources of uncertainty independently. Considering them in combination should then have an even stronger impact on the platform safety. This will be tackled in future work.

TABLE IV

FLAT TRAVERSAL: PROBABILITY ASSESSMENT

No Uncertainty			Cost	Prob. Hit	Min. Dist. (m)
# trials:	10	Mean:	1.295	0.380	0.095
# collisions:	1	Std.:	0.385	0.492	0.084
Uncertainty			Cost	Prob. Hit	Min. Dist. (m)
# Trials:	10	Mean:	1.177	0.016	0.165
# Collisions:	2	Std.:	0.262	0.005	0.088

TABLE V

PLANNING WITH TRAVERSAL OVER ROCKS

Uncertainty considered	# Trials	# Stuck States	Mean Cost	Std. Cost
none	5	1	1.46	0.050
Δs_{dist},Δs_{yaw}	5	1	1.31	0.083
Δs_{head},Δs_{yaw}	5	0	1.19	0.061

VI. CONCLUSION

Since the motion of any real mobile robot is stochastic to some degree, considering control uncertainty at the planning stage enables us to significantly enhance the safety of the platform (e.g. mitigating chances of collisions) and lower the cost accumulated on average during the execution of planned trajectories in real environments. These claims were demonstrated in this paper using a holonomic planetary rover. A model of uncertainty was built using learning data and Gaussian processes to predict motions over flat and unstructured terrain in a Mars yard setup. The trained model was then exploited to plan policies using dynamic programming, and to execute paths following the planned policies. Experimental validation was achieved both in simulation and in real experiments, in a variety of situations, taking into account uncertainties in heading and distance travelled. The experimental validation demonstrates the results obtained on the actual executed trajectories compared to trajectories executed when planning without considering uncertainty (de-

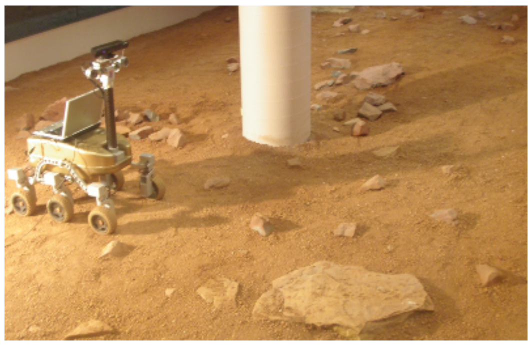


Fig. 7. Rock traversal experiment setup: rover is shown at its starting position. It must traverse over a "rock field" to reach a goal region to the right (out of the picture).

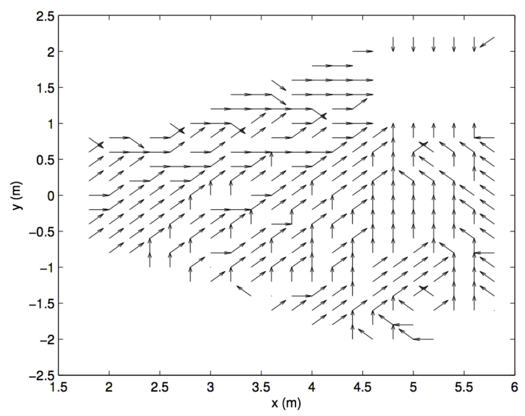


Fig. 8. Example policy obtained for rough terrain experiment with Δs_{head} considered, projected onto the $x - y$ plane. The goal is the empty square in the upper right corner of the figure. Arrows indicate the preferred crab actions at each state. Dots indicate rotations.

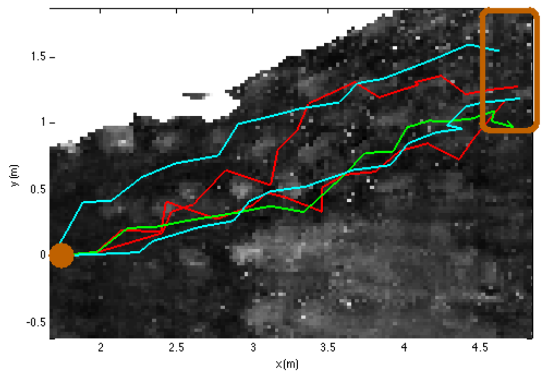


Fig. 9. Examples of trajectories during the rock traversal experiments, comparing planning when considering different uncertainty sources. Red: without uncertainty considered. Green: with uncertainty in heading. Cyan: with uncertainty in distance. Cost map shown in grayscale, from dark to light as cost increases. The starting position is indicated by the brown disk on the left and the goal region is shown as the brown rectangle at the top right corner. The planner generally attempts to navigate between most of the rocks which are high cost.

terministic control). These results show the improvement in safety and accumulated cost when accounting for uncertainty.

In future work, we will consider more complex terrain with larger slopes, denser collections of small rocks. We are particularly interested in studying the ability of our approach to learn and predict the deviations of control actions due to loose rocks that shift during traversal. We will also investigate an extension of the GP learning to include observations collected during navigation.

REFERENCES

[1] P. S. Schenker, T. L. Huntsberger, P. Pirjanian, E. T. Baumgartner, and E. Tunstel, "Planetary rover developments supporting mars exploration, sample return and future human-robotic colonization," *Autonomous Robots*, vol. 14, pp. 103–126, 2003.

[2] S. LaValle, *Planning Algorithms*. Cambridge Univ. Press, 2006.

[3] G. Ishigami, G. Kewlani, and K. Iagnemma, "Statistical Mobility Prediction for Planetary Surface Exploration Rovers in Uncertain Terrain," in *IEEE International Conference on Robotics and Automation*, 2010.

[4] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila, "Autonomous Rover Navigation on Unknown Terrains: Functions and Integration," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 917–942, 2002.

[5] D. Helmick, A. Angelova, and L. Matthies, "Terrain Adaptive Navigation for planetary rovers," *Journal of Field Robotics*, vol. 26, no. 4, pp. 391–410, 2009.

[6] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *IEEE International Conference on Robotics and Automation*, 2011.

[7] G. Ishigami, K. Nagatani, and K. Yoshida, "Path planning for planetary exploration rovers and its evaluation based on wheel slip dynamics," in *IEEE International Conference on Robotics and Automation*, 2007.

[8] S. Patil, J. Berg, and R. Alterovitz, "Motion planning under uncertainty in highly deformable environments," in *Robotics: Science and Systems VII*, 2011.

[9] J. Berb, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," in *Robotics: Science and Systems VI*, 2010.

[10] S. M. LaValle and S. A. Hutchinson, "An Objective-Based Framework for Motion Planning under Sensing and Control Uncertainties," *The International Journal of Robotics Research*, vol. 17, no. 1, pp. 19–42, 1998.

[11] H. Kurniawati, T. Bandyopadhyay, and N. Patrikalakis, "Global motion planning under uncertain motion, sensing, and environment map," in *Robotics: Science and Systems VII*, 2011.

[12] C. A. Brooks and K. Iagnemma, "Self-Supervised Terrain Classification for Planetary Surface Exploration Rovers," *Journal of Field Robotics, Special Issue on Space Robotics, Part I*, vol. 29, no. 3, pp. 445–468, 2012.

[13] K. Iagnemma, H. Shibly, A. Rzepniewski, and S. Dubowsky, "Planning and control algorithms for enhanced rough-terrain rover mobility," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2001.

[14] C. E. Rasmusen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[15] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the rgb-d slam system," in *IEEE International Conference on Robotics and Automation*, 2012.

[16] M. Quigley et al., "Ros: an open-source robot operating system," in *Open-Source Software Workshop, IEEE International Conference on Robotics and Automation*, 2009.

[17] D. Bonnafous, S. Lacroix, and T. Simeon, "Motion generation for a rover on rough terrains," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.