Planetary Micro-Rover Operations on Mars Using a Bayesian Framework for Inference and Control

Mark A. Post[c,*], Junquan Li[d], Brendan M. Quine[d]

[a]*Space Mechatronic Systems Technology Laboratory, Department of Design, Manufacture and Engineering Management, University of Strathclyde, 75 Montrose St. Glasgow, United Kingdom. G1 1XJ +44-(0)141-574-5274*

[b]*Department of Earth and Space Science and Engineering, York University, 4700 Keele Street, Toronto, ON, Canada. M3J 1P3 +1(416)736-2100 x22095*

Abstract

With the recent progress toward the application of commercially-available hardware to small-scale space missions, it is now becoming feasible for groups of small, efficient robots based on low-power embedded hardware to perform simple tasks on other planets in the place of large-scale, heavy and expensive robots. In this paper, we describe design and programming of the Beaver micro-rover developed for Northern Light, a Canadian initiative to send a small lander and rover to Mars to study the Martian surface and subsurface. For a small, hardware-limited rover to handle an uncertain and mostly unknown environment without constant management by human operators, we use a Bayesian network of discrete random variables as an abstraction of expert knowledge about the rover and its environment, and inference operations for control. A framework for efficient construction and inference into a Bayesian network using only the C language and fixed-point mathematics on embedded hardware has been developed for the Beaver to make intelligent decisions with minimal sensor data. We study the performance of the Beaver as it probabilistically maps a simple outdoor environment with sensor models that include uncertainty. Results indicate that the Beaver and other small and simple robotic platforms can make use of a Bayesian network to make intelligent decisions in uncertain planetary environments.

*Corresponding author

Email addresses: `mark.post@strath.ac.uk` (Mark A. Post), `junquanl@yorku.ca` (Junquan Li), `bquine@yorku.ca` (Brendan M. Quine)

Keywords:
Rover, Decision-making, Probabilistic, Bayesian, Inference, Mars

Planetary Micro-Rover Operations on Mars Using a Bayesian Framework for Inference and Control

Mark A. Post[c,*], Junquan Li[d], Brendan M. Quine[d]

[c]*Space Mechatronic Systems Technology Laboratory, Department of Design, Manufacture and Engineering Management, University of Strathclyde, 75 Montrose St. Glasgow, United Kingdom. G1 1XJ +44-(0)141-574-5274*
[d]*Department of Earth and Space Science and Engineering, York University, 4700 Keele Street, Toronto, ON, Canada. M3J 1P3 +1(416)736-2100 x22095*

1. Introduction

Intelligent autonomous operation is easily the most difficult problem in mobile robotics. While known and finite sets of conditions can be planned for and responses pre-programmed using a variety of methods, giving a robot the ability to appropriately handle unexpected and uncertain circumstances remains an open and very challenging problem. Planetary rovers would easily benefit the most from full autonomy, given that they must operate in uncertain conditions while isolated from any direct human assistance. Certainly real-time control of a rover on another planet is infeasible, as the delay in communicating a signal to Mars at the speed of light ranges from 3 to 21 minutes, not even considering temporary communications blackouts and time for retransmitting due to packet errors. However, due to the complexities involved, autonomy on space hardware has been very slow in adoption because of the inherent risks of autonomy failures with the extremely high costs of putting space hardware on other planets. Rovers work in a partially unknown environment, with narrow energy/time/movement constraints and, typically, limited computational resources that limit the complexity of online planning and scheduling. This is particularly true of micro-rovers and other small, inexpensive robots that are desirable to reduce potential losses if

*Corresponding author
Email addresses: `mark.post@strath.ac.uk` (Mark A. Post), `junquanl@yorku.ca` (Junquan Li), `bquine@yorku.ca` (Brendan M. Quine)

problems occur with individual units, but require low-power embedded systems for control that over long periods will need to adapt to unknowns in less reliable hardware like mechanical wear, system failures, and changes in environmental conditions. Considerable research work has been done on efficient planning algorithms [1] and algorithms to autonomously modify planning to handle unexpected problems or opportunities [2]. In particular, probabilistic methods have been used to great effect [3]. Adaptive learning and statistical control is already available for planetary rover prototypes, and can be significantly improved to decrease the amount of planning needed from humans [4], but is still often underutilized.

1.1. Bayesian Reasoning

Bayesian Networks (BN) are well-suited for handling uncertainty in cause-effect relations, and handle dependence/independence relationships well provided that the network is constructed using valid relational assumptions. Gallant et al. [5] show how a simple Bayesian network can operate to determine the most likely type of rock being sensed given a basic set of sensor data and some probabilistic knowledge of geology. To make it possible for the small, efficient planetary rovers of the future be decisionally self-sufficient, focus is needed on the design and implementation of efficient but robust embedded decision-making systems for tasks such as navigation and identification. Some drawbacks of this method are that the variables, events, and values available must be well-defined from the beginning, and the causal relationships and conditional probabilities must be available initially [6]. This makes construction of the Bayesian network a considerable challenge. Bayesian networks have been used extensively for image recognition, diagnostic systems, and machine behaviours. However, the potential of these concepts for distributed machine learning and problem-solving is considerable, and warrants further real-world research. In the reference [7], a self-organized control method for a planetary rover has been studied, but only extends to the navigation problem.

In this work, we apply a probabilistic framework of this type to the problem of using sensors with a probabilistic model to map obstacles sensed by a micro-rover prototype. A directional sensor fusion model for simple infrared range sensors is used to improve the accuracy of object determination, and Bayesian methods are used to infer the likelihood of object presence at a given location on the map, which functions as an occupancy grid that is calculated as a two-dimensional probability distribution. In addition, the uncertainty in the measurement made is estimated, mapped in the same way, and used to drive the search pattern. A set of behaviours is then applied to the likelihood map to implement obstacle avoidance. To evaluate performance in a real-world scenario, this system is implemented on a small micro-rover prototype and tested in an outdoor area with obstacles present. The use of a common Bayesian framework simplifies operational design and calculation methods throughout the whole system. Following this Section 1, Section 2 provides background information on Bayesian networks and inference methods in the context of discrete reasoning, Section 3 explains our implementation of a Bayesian network and compares our methodology to the original concept of Bayesian Robot Programming as described by Lebeltel et al., Section 4 details the design and methodology of a simple obstacle mapping system using the Bayesian framework, and Section 5 shows the results of actual testing using this system. Section 6 then concludes the paper.

1.2. The Beaver μrover

The original application for development of our probabilistic system is the Northern Light mission, a Canadian initiative to send a lander such as the one shown in Figure 1 to the surface of Mars for a short-range mission to observe the surface and position a ground-penetrating radar at a distance from the lander module for subsurface imaging [8]. For this purpose, it is planned to include a micro-robot, known as the Beaver rover, which will leave the lander and perform geological surveying and imaging of the Martian surface. The primary science payloads for this mission are an Argus infrared spectrometer for spectral analysis of surface rocks, the same type as is currently used on the CANX-2 Nanosatellite for atmospheric imaging, and a ground-penetrating radar system, which is currently under parallel development. For this mission, the Beaver will have to traverse a distance of under a kilometre while avoiding obstacles and taking sensor measurements. Naturally, extended and reliable operation on the surface of Mars would be preferred.

The Beaver micro-rover (μrover) prototype currently in use was developed for this mission as a stand-alone, self-powered, autonomous ground roving vehicle of 6kg mass designed to gather data and perform simple tasks in distant or hostile environments such as Mars, the Moon, or here on Earth [9]. The μrover is powered from solar panels and can recharge outdoors while operating, or by powering down onboard systems for extended periods. A low-cost, modular design using commercial off-the-shelf (COTS) components

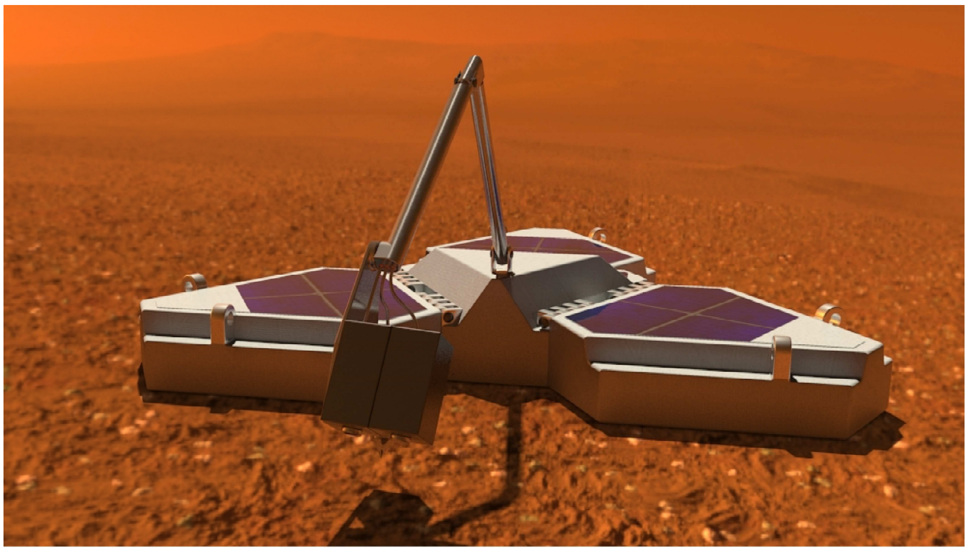


Figure 1: Northern Light Lander Module (Air Whistle Media/Thoth Technology Inc.)

makes it very flexible and useable for both planetary and terrestrial research purposes [10]. It has a power-efficient ARM-based onboard computer, a color CMOS camera, magnetometer, accelerometer, six navigational infrared sensors, and communicates via long-range 900MHz mesh networking. A variety of communications interfaces including RS-485 serial, SPI, I2C, Ethernet and USB are used for payload interfacing [11]. A diagram of the onboard systems available is shown in Figure 2, and the electronics are implemented in a PC/104+ form factor stack with external payload connectors. Programming is done in the C language using fixed-point numerical processing for efficiency. The autonomy algorithms are probabilistic in nature, using adaptive Kalman filtering [12] and Bayesian networks to handle uncertainty in the environment with minimal computation requirements. The Beaver uses probabilistic methods for mission planning, operations, and problem-solving, where each state variable in the system is considered to be a random variable.

Localization and motion tracking is performed on two scales. For short distances, high-precision tracking is performed using an HMC5883L magnetometer as a compass to obtain heading information, and a downward-pointing optical sensor to monitor horizontal motion [13]. The optical sensor is calibrated using reference distances obtained from encoders on the drive motors [14]. Over longer distances of $> 1m$, the rover's position is periodi-

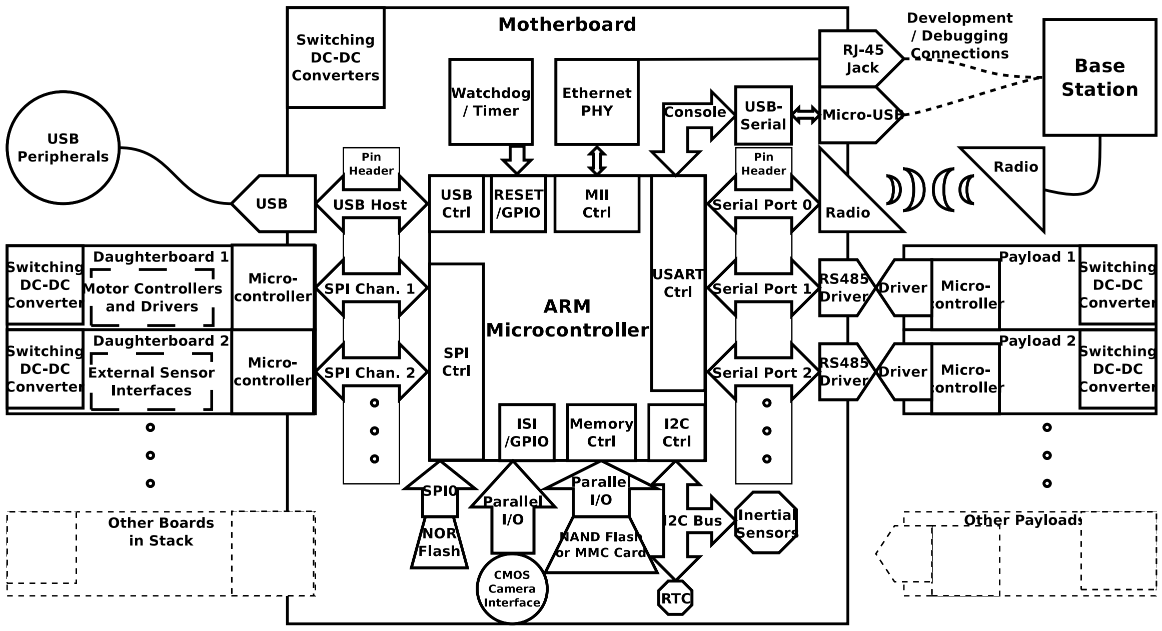


Figure 2: Beaver Micro-rover Modular Electronic Systems

cally updated with the position obtained from a Trimble Lassen IQ GPS unit to identify and correct accumulated positional error. A resolution of $0.5m$ is used for the occupancy grid as this follows the current level of accuracy of the GPS position estimation system. Rollover risk and terrain variations are handled with a nonlinear controller that adjusts the suspension angle by using differential torques from the four drive motors to raise, lower, and tilt the suspension as needed. A monocular vision system has also been developed to augment the mapping of terrain by building feature maps in three dimensions by triangulation of features detected by the ORB algorithm using a dedicated DSP board for vision. [15] [16]. As localization methods are improved, eventually the GPS will not be needed for navigation of the micro-rover. A picture of the Beaver μrover under testing in a sandy outdoor environment is shown in Figure 3.

2. Methods

A wide variety of approaches to dealing with probability exist. In general, the concept of uncertainty is encapsulated in a "random variable", which is defined as having a value that is to some degree determined by randomness, in

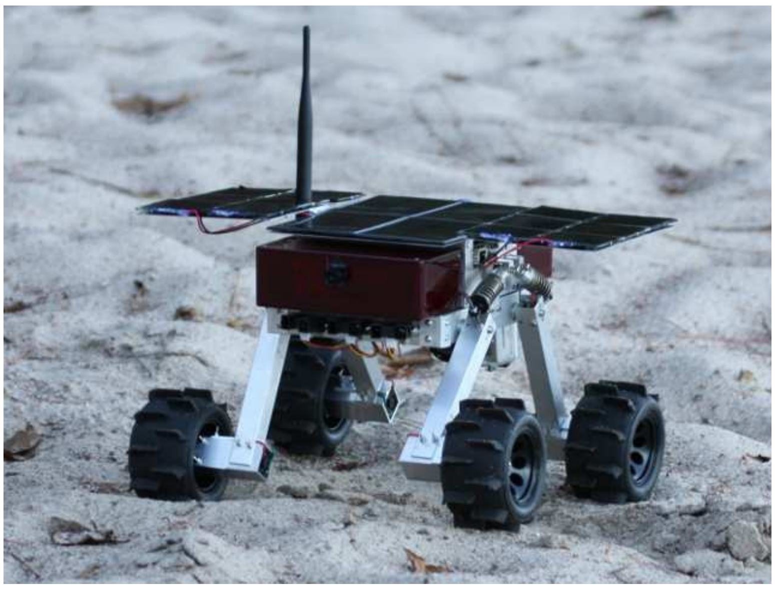


Figure 3: Beaver μrover Prototype Testing in Sand

contrast to "definite" variables that have a certain known or unknown value. In a stochastic system, states are determined by the probability distributions in random variables, where the values that a random variable takes on have precise probabilities associated with them, and the sum (or integral) of all probabilities in a random variable is defined to be 1 to reflect that there must be some value associated with the random variable that is present. Probability theory provides a framework for logically handling random variables and the relationships between them so that useful results can be determined. Logically connecting many random variables together based on probabilistic dependencies requires the concept of "evidence", on which a change in a given set of probabilities can be based. The interpretation of Bayesian probability makes use of propositional logic to enable "hypotheses" to be tested and updated based on probabilistic data. This process of "Bayesian inference" is central to our treatment of probabilistic systems. The term "Bayesian" refers to the 18th century statistical theologian and Presbyterian minister Thomas Bayes, who formulated the basic theorem of statistical inference based on conditional probability.

Bayesian networks are similar in concept to fuzzy systems, but have one major theoretical difference: *Fuzzy memberships are constant and imprecise,* *whereas probabilities are precisely defined but may change when an event occurs.* Rather than characterize a variable as having partial memberships, probabilistic or “random” variables can only have one value, but with a certain probability that varies depending on other factors. These other factors can be probabilities themselves, leading to the concept of a Bayesian network of conditional probabilities. Probability in this case implies directed causality $a \rightarrow b \rightarrow c$, and Bayesian networks allow associations to be defined to determine the likelihood of both a and b if c is known. Using associated probabilities allows a machine learning system to calculate not only what specific actions should have a certain effect, but what actions are unlikely to have that effect, and also what actions may have already had that effect or not. This ability to characterize multiple causes and “explain away” unlikely causes makes a Bayesian network a very powerful predictive tool.

2.1. Naive Bayesian Modelling

The most useful contribution of Bayesian networks to real probability calculations is reducing the storage and computation required. A joint probability distribution $\mathrm{P}(X_1, \ldots, X_L)$ for random variables with N values requires $N^L - 1$ separate probability values to store all N^L combinations of the values in all variables with related probability data for each combination. By exploiting independence between random variables, we can parameterize the distribution into a set of specific outcomes. For example, if we let X_m be the result of a coin toss out of M coin tosses with two possible values for x_m being $X = x_{heads}$ and $X = x_{tails}$, we can define the parameterized probability p_m as the probability that the m'th coin toss results in *heads*. Using the probabilities $\mathrm{P}(X = x_{heads}) = 0.5 = p_m$ and $\mathrm{P}(X = x_{tails}) = 1 - 0.5 = 0.5$, we can now represent the distribution using only the M values of p_m rather than all 2^M possible combinations of random variable values. The critical assumption is that *each random variable is at least marginally independent* of the other variables, which allows us to write the distribution over M coin tosses as

$$\begin{aligned} &\mathrm{P}(X = x_1, X = x_2, \ldots, X = x_M) \\ &= \mathrm{P}(X = x_1)\mathrm{P}(X = x_2)\ldots\mathrm{P}(X = x_M) = \prod_{m=1}^{M} p_m. \end{aligned} \tag{1}$$

Parameterization of joint distributions provides an effective way to make storage and calculation more tractable. However, obtaining useful probability estimations for use in a joint distribution is not trivial. Most commonly, probability estimates for values of random variables such as X and Y are obtained through averaging of repeated trial measurements of the stochastic processes they represent. The probability values in a joint distribution $\mathrm{P}(X,Y)$ can be completely different from those in the separate marginal distributions $\mathrm{P}(X)$ and $\mathrm{P}(Y)$. Given a large enough database of experience, it may be possible to estimate the joint distribution completely, but there is little information that can be re-used and every joint distribution would require a separate dataset. It is much more practical to use the chain rule for conditional random variables to factor the distribution into a conditional probability

$$\mathrm{P}(X,Y) = \mathrm{P}(X)\mathrm{P}(Y|X). \tag{2}$$

Of course, this example only considers a single relationship, and probabilistic models of much more complicated systems are needed, with multiple probability dependencies. For example, a vision system operating in concert with the obstacle sensor may be able to detect and triangulate features on nearby objects, and we can use it to increase the accuracy of our estimation of object presence. Let W be the random variable of features being detected within the range of the obstacle sensor, and let $\mathrm{P}(W = w_f)$ be the parameterized probability that enough features are detected to constitute an obstacle. For the moment, we need not consider the complexities of determining whether the features in question constitute an obstacle, as this complexity can be effectively "hidden" by the use of the distribution over W, as we will clarify later. It is natural to think of the information from these two sensors as being related (and hence the variables W and X), since they are effectively observing the same set of obstacles. We can write the conditional joint distribution over these two variables similarly to Equation 2 as

$$\begin{aligned} \mathrm{P}(W,X,Y) &= \mathrm{P}(W,X|Y)\mathrm{P}(Y) \\ &= \mathrm{P}(W|Y)\mathrm{P}(X|Y)\mathrm{P}(Y). \end{aligned} \tag{3}$$

In this way, we can generalize the factorization of a joint distribution of M variables $X_1 \ldots X_M$ that are marginally independent but dependent on a condition Z to

$$\mathrm{P}(X_1, \ldots, X_M, Z) = \mathrm{P}(Z) \prod_{m=1}^{M} \mathrm{P}(X_m|Z). \tag{4}$$

This principle of factorizing parameterized joint distributions is extremely useful. Besides being able to "split up" a joint distribution into more easily-obtainable conditional factors, if we assume N values for each random variable we can again reduce the number of actual probabilities required across M random variables from the order of $N^M + 1$ to the order of $N \times M + 1$ by means of parameterization, as well as correspondingly reducing the amount of calculation required to a set of multiplications. If $\mathrm{P}(W|Y)$ and $\mathrm{P}(X|Y)$ are considered to be "likelihoods" and $\mathrm{P}(Y)$ is a "prior", then only "evidence" is missing from this expression, indicating that for our example we are only describing a probability of general object detection and the actual reading of the sensor is missing. If we divide both sides of Equation 4 by $\mathrm{P}(X)$ and apply the chain rule so that $\mathrm{P}(W, X, Y)/\mathrm{P}(X) = \mathrm{P}(W, Y|X)$, we return to a conditional expression that has a change of dependency on the left hand side

$$\mathrm{P}(W, Y|X) = \frac{\mathrm{P}(W|Y)\mathrm{P}(X|Y)\mathrm{P}(Y)}{\mathrm{P}(X)}. \tag{5}$$

The sensory model we describe here is consequently known as a naive Bayes model, which is often used for classification of observed features in sensory systems. For these systems, it is assumed that the conditional variable Y contains a set of "classes" that are responsible for the "features" represented by variables $X_1 \ldots X_M$. If we want to compute a measure of confidence in whether a class y_1 or y_2 better fit the observed features, we can compare two distributions directly by taking the ratio of their posterior probability distributions

$$\begin{aligned} &\frac{\mathrm{P}(X_1 = x_1, \ldots, X_M = x_m, Y = y_1)}{\mathrm{P}(X_1 = x_1, \ldots, X_M = x_m, Y = y_2)} \\ &= \frac{\mathrm{P}(Y = y_1)}{\mathrm{P}(Y = y_2)} \prod_{m=1}^{M} \mathrm{P}\frac{(X_m|Y = y_1)}{(X_m|Y = y_2)}. \end{aligned} \tag{6}$$

As we now have methods for building probabilistic relationships and making probabilistic inferences, we can make use of this framework to allow a μrover to make intelligent decisions and respond appropriately to uncertain situations. The basis for our approach is inference using a Bayesian network as an abstraction of expert knowledge regarding the rover itself and assumptions about its environment, such as obstacles and hazards. We approach the problem of probabilistic robotics using the Bayesian Robot Programming (BRP) methodology developed by Lebeltel, Bessiere et al [17] [18] [19], which provides a quite comprehensive framework for robotic decision-making using inference and learning from experience. Despite having considerable promise and providing a novel solution for reasoning under uncertainty, BRP has not been developed significantly since the initial publications during 2000-2004. We add to this method by formally using Bayesian networks as a knowledge representation structure for programming, and by constructing these networks dynamically with implicit information obtained from the μrover bus and from a store of mission information. There are several advantages to this approach, which include clarity of representation, a practical structure for constructing joint distributions dynamically, and reliance on a proven probabilistic methodology. Also, the use of recursive inference in a Bayesian network avoids the need to manually partition and decompose large joint distributions, which greatly simplifies the programming process.

2.2. Bayesian Networks

For us to be able to properly organize and represent a large set of joint distributions using factorization in this way, we need a method of clearly associating random variables that are dependent on each other. Using the naive Bayes model for random variable dependence, a directed graph can be constructed, with nodes that represent random variables connected by edges that show the direction of dependence of one random variable on another. A graph or "network" of probability distributions over random variables, with one independent random variable representing a node and directed edges showing its dependencies, is called a Bayesian Network (BN), and forms the representation of our probabilistic relationships.

A Bayesian Network (BN) can be visualized as a directed acyclic graph (DAG) which defines a factorization (links) of a joint probability distribution over several variables (nodes). The probability distribution over discrete variables is the product of conditional probabilities ("rules"). For a causal statement $X \rightarrow Y$ often it is needed to find $\mathrm{P}(X|Y = y)$ using the distribution $\mathrm{P}(X)$. Bayes' rule states that

$$\mathrm{P}(X|Y=y) = \frac{\mathrm{P}(Y=y|X)\mathrm{P}(X)}{\mathrm{P}(Y=y)}. \tag{7}$$

and a Bayesian network, when viewed from the point of view of a single joint probability distribution, can be represented by:

$$\mathrm{P}(X_1 \cdots X_n) = \prod_{i=1}^{n} \mathrm{P}(X_i|\mathrm{Pa}(X_i)) \tag{8}$$

where the parents or depended-on variables) of each i'th node X_i are denoted by $\mathrm{Pa}(X_i)$. To construct a BN, you identify the variables and their causal relationships and construct a DAG that specifies the dependence and independence assumptions about the joint probability distributions. Bayesian networks can be constructed using semi-automated means if there is information regarding the probability and independence of network variables. We construct the Bayesian network with information about the components of the rover itself and the mission plan, where causality is implicit as the structure is broken down into components and sub-components. Applying the chain rule for conditional probabilities [20], to all nodes in a Bayesian network, the probability distribution over a given network or subnetwork of nodes $\wp = \{X_1 \dots X_M\}$ can be said to factorize over $\wp$ according to the dependencies in the network if the distribution can be expressed as a product

$$\mathrm{P}(\{X_1 \dots X_M\}) = \prod_{m=1}^{M} \mathrm{P}(X_m|Pa(X_m)). \tag{9}$$

The chain rule is particularly useful for calculating the Conditional Probability Distribution (CPD) of a given node in the network. Due to the dependency structure of the network, the conditional probability of a given node X depends on all its ancestors, so that $\mathrm{P}(X|\mathrm{Pa}(X))$ must be calculated recursively. A query for the posterior probability distribution of X involves first querying all of its parent nodes $Y \in \mathrm{Pa}(X)$ to determine their probability distributions, then multiplying them by the probability distributions of each parent node Y such that by a simplification of the chain rule

$$\mathrm{P}(X=x) = \sum_{Y \in \mathrm{Pa}(X)} \mathrm{P}(X=x|Y=y)\mathrm{P}(Y=y). \tag{10}$$

For discrete random variables, which we will be primarily using, it is important to note that with only one parent of a node, this is effectively just a

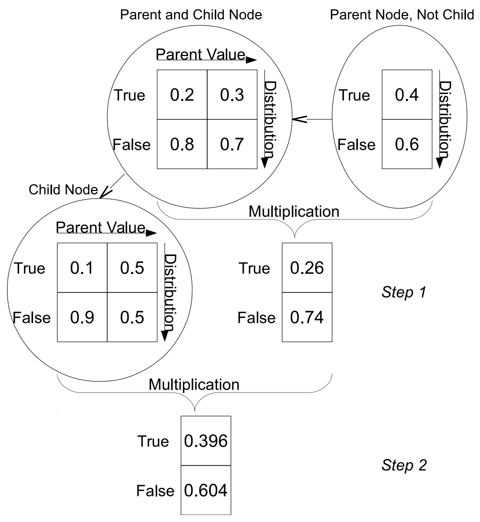


Figure 4: Matrix Calculation for Querying a Random Variable with One Parent

matrix multiplication. This process is graphically illustrated in Figure 4. The CPD $\mathrm{P}(X)$ represents a local probabilistic model within the network, which for each node represents a local summary of probability for its own random variables. For a continuous distribution, it can be a function evaluation. Representing probability distributions by tables, the distribution $\mathrm{P}(X|Y)$ will be an $L+1$-dimensional matrix for L parents, with the major dimension of size N for N random variable values in $\mathrm{V}(X)$. Each parent Y is assumed to have a distribution of size M (although each could be different with no change in the concept), so that the distribution $\mathrm{P}(Y)$ is a $N \times 1$ matrix. For nodes with two or more parents. the process is more complex, as each possible value of the inferred distribution must be calculated as a sum of the probabilities that lead to that value, as shown in Figure 5. To calculate the conditional distribution for X, the size $M \times N$ plane of the matrix representing the values from the parent versus the values from the child node, which is the distribution $\mathrm{P}(X|Y)$ is multiplied with the $N \times 1$ distribution $\mathrm{P}(Y)$. This results in an $N \times 1$ matrix that is effectively a temporary posterior distribution estimate for $\mathrm{P}(X)$ which avoids frequent recalculation for determining the conditional distributions of children $\mathrm{Ch}(X)$ while traversing the network.

In this way, any node in a Bayesian network can be queried to obtain a probability distribution over its values. While exact inference as described

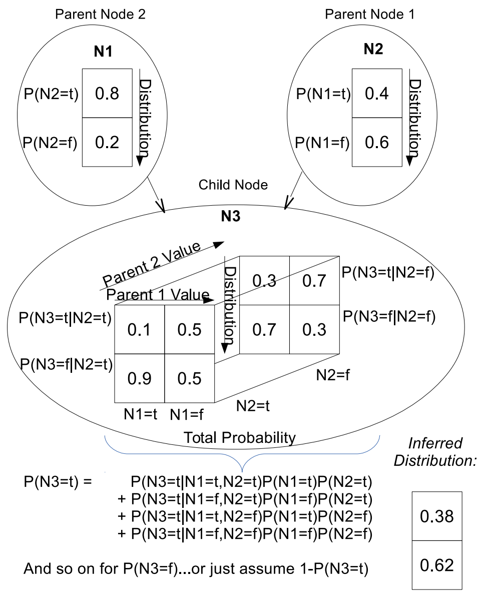


Figure 5: Matrix Calculation for Querying a Random Variable with more than One Parent

into a Bayesian network is widely understood to be an NP-hard problem [21] [22], it is still much more efficient than the raw computation of a joint distribution, and provides an intuitive, graphical method of representing dependencies and independences. It is easy to see how any system that can be described as a set of independent but conditional random variables can be abstracted into a Bayesian network, and that a wealth of information regarding the probability distributions therein can be extracted relatively efficiently. This forms our basic methodology for probabilistic modelling.

$$\begin{aligned} &\mathrm{P}(X_1 \cap X_2 \cap \ldots \cap X_N) \\ &= \mathrm{P}(X_1)\mathrm{P}(X_2|X_1)\ldots\mathrm{P}(X_N|X_1 \cap \ldots \cap X_{n-1}). \end{aligned} \tag{11}$$

This allows any number of events to be related in terms of other events, and is a very important result for inference systems. The order or numbering of the events is not important because the conjunction operator $\cap$ is commutative, meaning that the chain can be re-formulated to better suit the set of known priors. For most robotic systems, the desired result is an expression for a conditional probability that provides a degree of confidence in a certain event with respect to other known events.

$$\begin{aligned} \mathrm{P}(Y|X) &= \frac{\mathrm{P}(X|Y)\mathrm{P}(Y)}{\mathrm{P}(X)} \\ &= \frac{(\textit{likelihood}) \cdot (\textit{prior})}{(\textit{evidence})}. \end{aligned} \tag{12}$$

2.3. Probability Queries

Once a probability distribution is defined, it is necessary to have a consistent method of extracting information from it. In most cases, rather than knowing the entire probability distribution, we only are interested in a specific probability of a single value or set of values, usually the highest probability. This is referred to as a Maximum A Posteriori (MAP) query, or alternately, “Most Probable Explanation” (MPE). This refers to the most likely values of all variables that are not used as evidence (because evidence is already known and has certain probability), and is described using $\arg\max_x \mathrm{P}(x)$ (the value of x for which $\mathrm{P}(x)$ is maximal) as

$$\mathrm{MAP}(X|Y=y) = \arg\max_x \mathrm{P}(x \cap y). \tag{13}$$

A very important point to note is that while the MAP query for a single variable $\mathrm{MAP}(X|Y=y)$ is equivalent to just finding the highest probability of the single-variable distribution $\mathrm{P}(X)$, it is not the same as finding all the maximum values in a joint conditional distribution $\mathrm{P}(X \cap Z)$, because the underlying probabilities in this joint distribution depend on both values of X and Z. Rather, a MAP query over a joint distribution finds the most likely complete set of values (x, z) as each combination of values has a different probability. This leads to an important generalization of the MAP query, in which we use a smaller subset of X to find a less likely set of events by independently searching a joint, conditional distribution of the subset. This is called a marginal MAP query, and is used frequently in Bayesian inference engines.

$$\mathrm{MAP}(X|Y=y) = \arg\max_x \sum_Z \mathrm{P}(X \cap Z|Y) \tag{14}$$

2.4. Logical Propositions

Inference in Bayesian programming generally relies on the concept of logical propositions. A logical proposition is essentially a formalization of the state of a probabilistic system, which by our probabilistic framework is represented by a value assignment for a discrete random variable $X = x$ or several discrete random variables. Conjunctions $(X = x \cap Y = y)$ and disjunctions $(X = x \cup Y = y)$ are then also, by extension, propositions. To distinguish the use of random variable sets from the use of propositions (although logically, sets and propositions are isomorphic to each other) we will use the propositional logic operators $\wedge$ for conjunction and $\vee$ for disjunction, as well as the shorthand notation for random variables. Hence, propositions will take the form (x), $(x \wedge y)$, and $(x \vee y)$. We add to this the concept of the negation of a proposition $\neg x$, which represents the complement of the value x. For a probability assignment $\mathrm{P}(X = x)$, this represents the complementary probability $1 - \mathrm{P}(X = x)$, or informally, the probability of a given event not happening.

As with Bayesian networks, the process of inference is used to extract information from propositions, based on conditional probability distributions over a set of random variable dependencies. It is assumed that most propositions are conditional on the same basic set of prior knowledge (or in graph parlance, the random variables most queried are child nodes of the same set of parent nodes). The conjunction of the set of random variables that are considered “prior knowledge” for a given proposition are often shortened to the proposition π for brevity, and can be thought of in a Bayesian network as parents of the node being examined. Based on the rules used for probabilistic inference, only two basic rules are needed for reasoning [23]: the Conjunction Rule and the Normalization Rule. We assume that the random variables used such as X are discrete with a countable number of values, and that a logical proposition of random variables $[X = x_i]$ is mutually exclusive such that $\forall i \neq j, \neg(X = x_i \wedge X = x_j)$ and exhaustive such that $\exists X, (X = x_i)$. The probability distribution over a conjunction of two such variables using shorthand notation is then defined as the set $\mathrm{P}(X, Y) \equiv \mathrm{P}(X = x_i \wedge Y = y_i)$. Using this concept of propositions, the Conjunction and Normalization Rules are stated as [17]

$$\begin{aligned} \forall x_i \in X, \forall y_j \in Y : \mathrm{P}(x_i \wedge y_j | \pi) &= \mathrm{P}(x_i, y_j | \pi) \\ &= \mathrm{P}(x_i | \pi)\mathrm{P}(y_j | x_i, \pi) = \mathrm{P}(y_j | \pi)\mathrm{P}(x_i | y_j, \pi) \end{aligned} \tag{15}$$

and an equivalent disjunction rule can be stated as

$$\begin{aligned}\forall x_i \in X, \forall y_j \in Y : \mathrm{P}(x_i \vee y_j|\pi) \qquad\qquad (16)\\ = \mathrm{P}(x_i|\pi) + \mathrm{P}(y_j|\pi) - \mathrm{P}(y_j|\pi)\mathrm{P}(x_i, y_j|\pi)\end{aligned}$$

while the Normalization Rule becomes

$$\forall y_j \in Y : \sum_{\forall x_i \in X} \mathrm{P}(x_i|\pi) = 1. \qquad (17)$$

The marginalization rule may then be derived for propositions as

$$\forall y_j \in Y : \sum_{\forall x_i \in X} \mathrm{P}(x_i, y_j|\pi) = \mathrm{P}(y_j|\pi). \qquad (18)$$

For clarity, when applying these rules to distributions over random variables, we do not necessarily have to state the individual propositions (or values). As with common random variable notation, the conjunction rule can be stated without loss of generality as

$$\mathrm{P}(X, Y|\pi) = \mathrm{P}(X|\pi)\mathrm{P}(Y|X, \pi), \qquad (19)$$

the normalization rule as

$$\sum_X \mathrm{P}(X|\pi) = 1, \qquad (20)$$

and the marginalization rule as

$$\sum_X \mathrm{P}(X, Y|\pi) = \mathrm{P}(Y|\pi). \qquad (21)$$

It is assumed that all propositions represented by the random variable follow these rules.

3. Theory and Implementation

There are many existing software packages for building and using Bayesian networks, but for μrovers we require a a software framework that is portable, highly resource-efficient, and usable on embedded systems, as well as being accessible and open for development. The μrover does not run Java or Python virtual machines as a rule (although future implementations may support Python), and the need for simple integration at a low level with other robotic components makes the use of a MATLAB embedded target binary impractical. Due to the constraints involved in running Bayesian inference processes on low-power ARM hardware and the need to adopt other architectures in the future, the C language was chosen as the lowest common denominator for compatibility between systems despite the extra complexity required for implementation in such a low level language. C++ bindings are also planned for development in order to make interfacing with other programs easier. The original framework for Bayesian Robot Programming, known as Open Probabilistic Language (OPL), was originally made available freely [18] but was later renamed and redistributed as a proprietary package called ProBT [19]. Several other currently-maintained and open-source packages were considered for use on the μrover, such as Bayes++, Mocapy++, and LibDAI, which is a recent but promising system for inference [24]. However, there is no known general framework that is implemented in bare C for efficiency and portability as is preferred for μrover software, and is focused specifically on robotic use in embedded systems, particularly with consideration to fixed-point math. It was therefore determined that a new Bayesian inference engine for the μrover had to be created.

3.1. Bayesian Robot Programming

A Bayesian program has been defined by Lebeltel et al. as a group of probability distributions selected so as to allow control of a robot to perform tasks related to those distributions [17]. A "Program" is constructed from a "Question" that is posed to a "Description". The "Description" in turn includes both "Data" represented by δ, and "Preliminary Knowledge represented by π. This "Preliminary Knowledge π consists of the pertinent random variables, their joint decomposition by the chain rule, and "Forms" representing the actual form of the distribution over a specific random variable, which can either be parametric forms such as Gaussian distributions with a given mean and standard deviation, or programs for obtaining the distribution based on inputs [18]. Rather than unstructured groups of variables, we apply these concepts to a Bayesian network of M random variables $\wp = X_1, X_2, \ldots, X_N \in \pi, \delta$, from which an arbitrary joint distribution can be computed using conjunctions. It is assumed that any conditional independence of random variables in π and δ (which must exist, though it was not explicitly mentioned by Lebeltel et al.) is represented appropriately by the Bayesian network, thus significantly simplifying the process of factorization for joint distributions. The general process we use for Bayesian programming, including changes from the original BRP, is as follows:

1. **Define the set of relevant variables.** We use the edges between nodes to represent dependencies. For example, for a collision-avoidance program, the relevant variables include those from the nodes associated with the obstacle sensors and vision system, and are generally easy to identify due to the structure of the network. Usually, a single child node is queried to include information from all related nodes.

2. **Decompose the joint distribution.** Rather than partitioning variables $\mathrm{P}(X_1, \ldots, X_M|\delta, \pi)$ into subsets [19], we make use of the properties of the Bayesian network for implicitly including information in parent nodes when queried. A question such as a MAP query of any given node involves knowing the distributions for the parents of that node, and so on recursively until a node with no parents is reached, by $\mathrm{P}(X) \prod_{m=1}^{M} \mathrm{P}(X_m|\delta, \pi)$.

3. **Define the forms.** For actual computations, the joint and dependent distributions must be numerically defined. The most common function to be used, and the function used for distributions in this work, is the Gaussian distribution with parameters mean $\bar{x}$ and standard deviation σ that define the shape of the distribution, commonly formulated as $\mathrm{P}(X = x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}$.

4. **Formulate the question.** While queries into a BRP system traditionally involve partitioning random variables into three sets: "searched" (Se), "known" (Kn), and "unknown" (Un) variables. The use of a Bayesian network formalizes the relationships of these sets, so that "searched" nodes can be queried, and implicitly will include all relevant known and unknown information in the network. It is important to note that a "question" is functionally just another conditional distribution, and therefore operates in the same way as an additional node in the Bayesian network.

5. **Perform Bayesian inference.** To perform inference into the joint distribution, the "Question" that has been formulated as a conjunction of the three sets Se, Kn, and Un is posed to the system and solved as

a Bayesian inference. The "Answer" is obtained as a probability distribution, or in the case of a MAP query, a value from the "searched" variable. For our Bayesian network implementation, nodes associated with actions to be taken typically have conditional distributions that act as "questions" regarding their operational state.

3.2. Bayesian Inference

The last step in Bayesian programming is the actual inference operation used to determine the probability distribution for the variable or set of variables in question. Obtaining the joint distribution $\mathrm{P}(Se|Kn, \pi)$ is the goal, and requires information from all related random variables in $\{Kn, Un, \pi\}$, which in the Bayesian network are visualized as parents of Se. This distribution can always be obtained using the following inference method [25]. The marginalization rule from Equation 21 first allows the inclusion of Un, as

$$\mathrm{P}(Se|Kn, \delta, \pi) = \sum_{Un} \mathrm{P}(Se, Un|Kn, \delta, \pi). \tag{22}$$

By the conjunction rule from Equation 19, this can be stated as

$$\mathrm{P}(Se|Kn, \delta, \pi) = \frac{\sum_{Un} \mathrm{P}(Se, Un, Kn|\delta, \pi)}{\mathrm{P}(Kn|\delta, \pi)}. \tag{23}$$

Applying the marginalization rule again to sum the denominator over both Se and Un, we have

$$\mathrm{P}(Se|Kn, \delta, \pi) = \frac{\sum_{Un} \mathrm{P}(Se, Un, Kn|\delta, \pi)}{\sum_{\{Se,Un\}} \mathrm{P}(Se, Un, Kn|\delta, \pi)}. \tag{24}$$

The denominator of Equation 24 acts as a normalization term, and for simplicity will be replaced with the constant $\Sigma = \sum_{\{Se,Un\}} \mathrm{P}(Se, Un, Kn|\delta, \pi)$, giving

$$\mathrm{P}(Se|Kn, \delta, \pi) = \frac{1}{\Sigma} \sum_{Un} \mathrm{P}(Se, Un, Kn|\delta, \pi). \tag{25}$$

To complete the inference calculation, we only need to reduce the distribution $\sum_{Un} \mathrm{P}(Se, Un, Kn|\delta, \pi)$ into factors that can be determined. To do this, we must assume that these factors are at least marginally independent. While BRP originally reduced these factors into marginally independent subsets, we can assume that independence is denoted by the structure of the Bayesian network, so we only need be concerned with the ancestors of Se. Using only the ancestors of a given node removes the need to scale by Σ. Given that inference into a Bayesian network typically involves querying a single node, we will assume that Se is the singleton $Se = \{X\}$. This can also be accomplished if Se is larger by making X a parent of all nodes in Se. Applying the chain rule again to Bayesian networks, the probability distribution over Se directly depends on the distributions of its parents, which for the moment we will assume are known to be unconditional random variables for clarity. We can factorize the immediate vicinity of $Se = \{X\}$ as

$$\mathrm{P}(Se|Kn, \delta, \pi) = \sum_{Un} \prod_{Y \in \{X, \mathrm{Pa}(X)\}} \mathrm{P}(X|Y)\mathrm{P}(Y). \tag{26}$$

This gives us a factorization for a single node. Of course, we cannot assume that the parents of X have no dependencies, and in general should be assumed to have some other dependencies Z so that we have $\mathrm{P}(Y|Z)$. In this case we must consider the parent nodes of the node being queried $\mathrm{Pa}(X)$, the parents of the parent nodes $\mathrm{Pa}(\mathrm{Pa}(X))$, and so on recursively until we have spanned the complete set of ancestors Y with $Y \in \mathrm{An}(X)$. From a purely algorithmic perspective, we can walk the Bayesian network backwards through the directed edges from X, determining the conditional distribution of each node from its parents as we go, and therefore breaking down the determination of the joint distribution into smaller, separate calculations. Considering Z to be the parents of each ancestor node Y and following the method of Equations 9, Equation 10, and Equation 26, a general expression for the factorization of $\mathrm{P}(Se|Kn, \delta, \pi)$ through the Bayesian network is

$$\mathrm{P}(Se|Kn, \delta, \pi) = \sum_{Y \in \{X, \mathrm{An}(X)} \left[\prod_{Z \in \mathrm{Pa}(Y)} \mathrm{P}(Y|Z)\mathrm{P}(Z) \right]. \tag{27}$$

This is a recursive calculation, as we must first obtain the conditional distributions $P(Z|Y)$ for the ancestors Z furthest from the node X before the closer ancestors and parents of X (as in depth-first traversal of the branches of a dependency tree). To save calculations, the temporary estimate $\mathrm{P}(Y) = \mathrm{P}(Y|Z)\mathrm{P}(Z)$ is saved for each node Y for use when calculating $\mathrm{P}(\mathrm{Ch}(Y))$ for the children of Y.

To construct an appropriate machine representation of a Bayesian network, it is necessary to consider both the numerical properties of a Bayesian node (or random variable), and the underlying requirements of the hardware and software that support the information contained in the network. As Bayesian nodes are essentially random variables associated with other random variables by way of a joint distribution, an object-oriented approach is taken to describing them using C structures. Unlike most Bayesian network implementations, our implementation is unique in that it uses fixed-point math for storage and calculation and is programmed in C for better portability and calculation efficiency on small-scale embedded systems.

3.3. Programming the Bayesian Network

The fundamental properties of a Bayesian Node are the probability distribution of the random variable, and the way that distribution is affected by the knowledge of other nodes nearby. For numerical compactness and code efficiency, the values of a node are represented as $M \times N$ distribution matrices, where each row represents the probability distribution of the random variable, and each column represents the effects of linked nodes. Total probability requires that all values in each row m sum to 1. The joint distribution P of probability values associated with a given random variable is the content actually stored, as well as a vector of labels for each of the local values in the random variable itself.

At minimum, a random variable with N possible values will have a $1 \times N$ distribution matrix. A parent node will increase the number of distributions that must be accounted for in the child node, causing at most M possible probability distributions for each one of its M possible variable values. If two or more parent nodes are present, the total number of combinations of affecting values must be considered in the child node. Hence, a parent with 2 values and a parent with 3 values will together contribute $2 \times 3 = 6$ possible probability distributions, and if the node itself has 4 possible values, a total of $4 \times 2 \times 3 = 24$ probability values must be stored in total. In general, if each parent has a distribution N_l values in size, and there are L parents, then the number of distributions M possible in the child node are

$$M = \prod_{l=1}^{L} N_l. \tag{28}$$

As each child node must have an $N \times M$ matrix, assuming that parent nodes have similar numbers of values, the storage size of the node scales roughly as N^L. This can be mitigated by designing a deeper graph with more nodes and less parents per node, as the simplifying properties of the Bayesian network will decrease the total storage required. A parent node with an $M_l \times N_l$ distribution matrix, regardless of the number of parents and the size of M_l, will still only contribute N_l values to its child nodes, making the speed of storage size increase dependent on the size of the probability distributions in question. A given node X will then have to store a table of size $|\mathrm{V}(X \cup \mathrm{Pa}(X))|$.

The actual method of storing the distributions is not trivial. Because the dimensionality of the distribution matrix effectively increases with each parent (adding a new set of combinations of variables), fixed-dimension matrices are not practical for nodes where new parents may have to be added dynamically. Many languages use nested template classes and other object-oriented methods for implementing N-dimensional storage. However, for speed and compactness of storage, we use a single C array for storage of the distribution, and index it with a linear index that is a function of the parent numbers of the node. To create the index, when addressing an array as an $L+1$-dimensional matrix for L parents we use an extension of the conventional mapping to a linear array index i for a row-major-order matrix, which for row (m) and column (n) indices is formulated as $n + m * columns$. By recognizing that each additional dimension must be indexed by multiplying past the sizes of all preceding dimensions, we can consistently index into a linear array at location i using matrix indices m_1 for dimension 1 of size M_1 (we choose columns here for consistency), m_2 for dimension 2 of size M_1(we choose rows here for consistency), and $m_3, m_4, \ldots$ and above for additional matrix dimensions of size $M_3, M_4, \ldots$ respectively, obtaining

$$\begin{aligned} m_1 + m_2 M_1 + m_3 M_2 M_1 + \ldots + m_{L+1} \prod_{l=1}^{L} M_l \\ = \sum_{n=1}^{L+1} \left(m_n \prod_{l=1}^{n-1} M_l \right) = i. \end{aligned} \tag{29}$$

This $O(L)$ complexity operation must be done for every index into the array, although shortcuts can be taken when traversing a dimensional axis, such as incrementing by m_1 for traversing rows, $m_2 M_1$ for columns, etc.

A set of functions for creating Bayesian networks have been implemented in C utilizing the fixed-point math system. The functions were specifically targeted at making the system reliable, efficient and small for use on embedded processors. Nodes of the network are stored as structures indexable in a static array to ensure that all nodes can be searched easily and no memory leakage occurs. The linked nodes and probability distribution in each node are also dynamically allocated. Each time a dynamic element is accessed, the pointer to it is tested for validity. This lowers the chance of segmentation faults and corrupted data. Currently, the network is built from both hardware data and XMLBIF files that contain the network structure and probability distributions.

In a Bayesian network representation, everything the rover knows is represented as linked random variables. The knowledge (priors, etc.) is initially provided by the rover's self-aware devices. Abstractions that need to be inferred are provided by the mission plan. Using the communications system detailed above, known values are obtained directly from hardware via the system bus, with models, capabilities, and links between the nodes provided by devices themselves. Abstractions such as the definitions of obstacles and mapped areas of interest are expert knowledge that is typically included in the mission planning data. Figure 6 shows a simple example of a Bayesian network constructed in this manner.

3.4. Node Types

While all nodes in the Bayesian network we construct represent random variables, the variables represent a variety of different real abstractions, and are consequently constructed using different data sources and roles within the Bayesian network. All nodes are kept as similar as possible in terms of data representation and programming interface, and all effectively function as probability distributions over random variables, although data nodes in particular usually call external sources to calculate appropriate responses to queries in real time.

- *Ability* or (A) nodes provide the abstraction of functions, and include the sensor models and movement models used to convert physically measurable quantities into concepts that are suitable for inference and reasoning such as "high", "medium", and "low". Ability nodes operate similar to fuzzifier/defuzzifier rules in a fuzzy logic system, but can

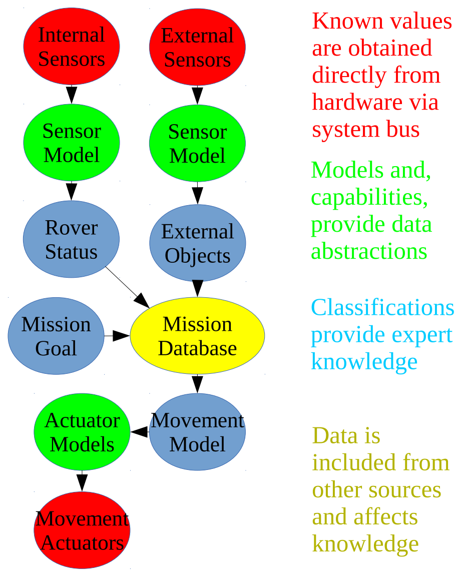

Internal Sensors
External Sensors
Sensor Model
Sensor Model
Rover Status
External Objects
Mission Goal
Mission Database
Actuator Models
Movement Model
Movement Actuators
Known values are obtained directly from hardware via system bus
Models and, capabilities, provide data abstractions
Classifications provide expert knowledge
Data is included from other sources and affects knowledge

Figure 6: A General Bayesian Network Illustrating Dependencies and Node Types

also be implemented as probabilistic algebraic functions, such as the sensor model for the infrared range sensors. They are usually a parent or child of a bus node. Ability nodes for sensors are typically included in the set of knowns Kn and ability nodes for actuators are typically searched in the set Se to make decisions about actuator movement and operational functions.

- *Bus* or (B) nodes include hardware devices connected to the system bus such as the obstacle sensors, inertial sensors, environmental sensors, and actuator drivers. Sensor nodes usually are parent nodes only that provide actual measurement values to ability nodes, while actuator nodes are usually child nodes only that are dependent on ability nodes representing their actual function, and have joint distributions representing a "question" regarding the actuator state. A bus node acts as a "terminal" or "endpoint" in the network that allows interaction with the rover itself and the outside world, and generally is associated with nodes in Se or Kn.

- *Classification* or (C) nodes are used to interpret and translate information within the network. For example, an inertial sensor can only give information about probability of body orientation and an obstacle sensor about the probability of the presence of an obstacle, but determining the likelihood of the rover tipping over or the likelihood of a collision are conditional judgments based on inference. Classification nodes act as drivers in determining behaviours to respond to external or internal events, and are most similar to the BRP concept of "Parametric Forms". Classification nodes are also generally included in the set of unknowns Un.

- *Data* or (D) nodes act as an interface to additional information outside the network. These include probability maps built as two or three-dimensional distributions, and mission information databases used to build probability distributions dynamically based on external instructions. Data nodes typically use function pointers to refer queries to functions that provide the appropriate information based on the system state, and are similar to the BRP concept of "Program Forms". As they provide data, they are usually included in the set of knowns Kn.

In Figures 6 and 8, ability nodes are coloured green, bus nodes are coloured red, classification nodes are colored blue, and data nodes are colored yellow. Learned probability data is stored and shared in XML format. There have been several different proposed Bayesian Network Interchange Formats (BNIF) developed, such as BIF, XMLBIF, and XBN [26]. Despite a lack of current information and limited scope of adoption, the XBN standard is the most current, although the earlier XMLBIF format appears more efficient to parse. Both formats can currently be used for storage, with additional XML tags implemented to support complete storage of the node structure such as node type $\{A, B, C, D\}$.

4. Experiments

To make use of the probabilistic systems we have described for the basic task of environmental mapping and navigation, we apply Bayesian proba-

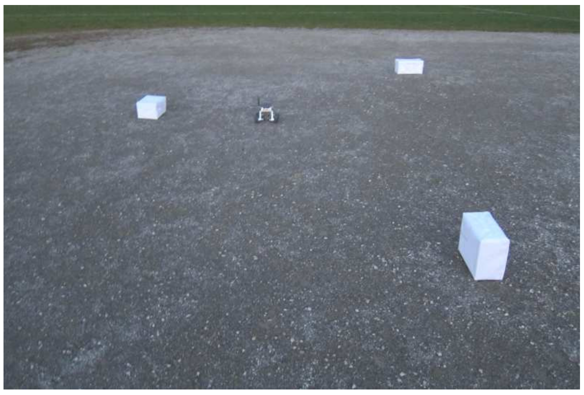


Figure 7: Beaver μrover Prototype in Outdoor Test Area with Obstacles

bilistic models to the sensor and mapping hardware on the μrover and allow it to map a small area of outdoor space with known obstacles using only the infrared range sensors for obstacle detection. The area mapped by the μrover is shown in Figure 7. It is open except for three boxes, which due to the low sensitivity of the infrared sensors are covered with white paper for high reflectivity. In this test, time-averaged GPS is used for coarse position sensing.

We use simple single-output sensors with a statistical model to monitor the environment in front of the micro-rover. To avoid necessitating the involvement of human operators, the system for data collection and decision making has to be implemented on the embedded micro-rover hardware itself. We make use of a small Bayesian network to perform the obstacle detection and navigation task. Due to the limitations present on the micro-rover platform used in this study, all numerical algorithms are being implemented in fixed-point arithmetic, with the necessary scaling and normalizing applied. This kind of Bayesian system has not been applied and tested in an actual planetary micro-rover of this type. As a goal for a simple Bayesian sensing and control system, we construct an uncertainty map of the rover's surroundings that can be used to identify prominent features reliably and avoid collisions during forward motion. We do not attempt to localize the rover itself using this information due to the limited information provided by the range sensors, which would require too much movement during measurements to perform simultaneous localization and mapping. A sensor model for simple infrared range sensors is used with a directional sensor fusion model to improve the accuracy of the sensors, and a Bayesian method is used to infer the likelihood of object presence at a given location on the map, structured as a Bayesian network to simplify design and calculation. In addition, the uncertainty in the measurement made is estimated, mapped, and used to drive the search methodology. A set of behaviours is then applied to the likelihood map to implement obstacle avoidance. To evaluate performance in a real-world scenario, this system is implemented on a small micro-rover prototype and tested in an outdoor area with obstacles present. The Bayesian network structure used to relate the various aspects of μrover operation for the mapping task is shown in Figure 8. An early version of this mapping methodology was tested using a simpler monolithic decision-making algorithm and without using a the Bayesian network [27] and this was used as a reference for development of the current network of variables, which offers better probabilistic performance, and more importantly, vastly more flexible programmability.

4.1. Range Sensor Model

For the purposes of this study, the rover is tasked to observe all objects encountered and statistically identify obstacles. The μrover's infrared range sensors are used to detect objects based on infrared light reflected from their surfaces, and each provide a range observation measurement r. Each sensor model is considered to be of a Bernoulli type with a probability of correct object detection β_r, a fourth-order polynomial function of the range sensor state, modelled as a random variable R, where [28]

$$R = ||\mathbf{x} - \bar{\mathbf{x}}|| \in \mathbb{R}^+ \tag{30}$$

with $\bar{\mathbf{x}}$ being the rover's estimated current location and $\mathbf{x}$ being the location of a sensed object. We can quantify the the probability β_r that the range sensor state R is correct by defining it as

$$\beta_r = \begin{cases} \beta_b + \frac{1-\beta_b}{{r_{max}}^4}({r_{max}}^2 - R^2)^2, & \text{if} \quad R \leq r_{max} \\ \beta_b, & \text{if} \quad R > r_{max} \end{cases} \tag{31}$$

where β_b is the base likelihood of correct object detection, assumed to be $\beta_b = 0.5$ so that an even likelihood of correctness is assumed if the object is outside the sensor's range since no actual information of object presence will be provided to the sensor. r_{max} is the sensor's maximum range of approximately $2m$, beyond which correct and incorrect object detection are equally likely. The peak value of β_r if the object being observed is located

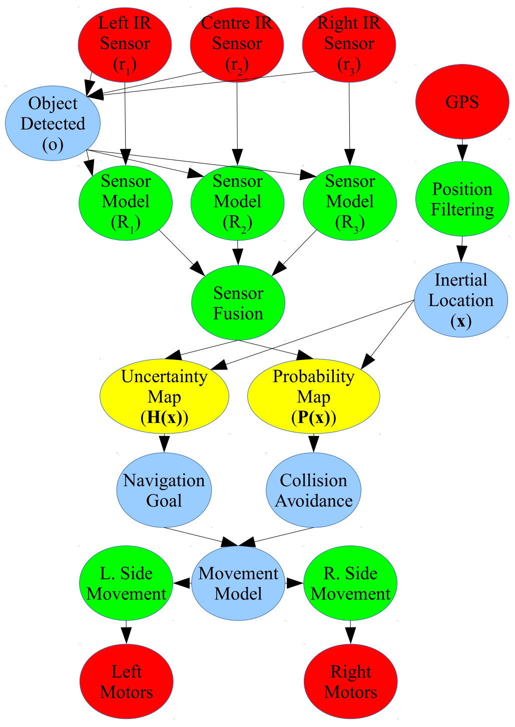


Figure 8: The Bayesian Network Used in the μrover for Mapping with IR Sensors

at $\bar{\mathbf{x}}$ is 1 (certainty) and occurs closest to the sensor, which generally has higher accuracy when closer to an object. This provides a model by which the assumption of object presence at R can be made based on the actual measurement of r.

A probabilistic method is used to update the probability of object presence given a range observation r. We define the likelihood of an object being present at range R from the current location and time $t+1$ given the range observations r as $\mathrm{P}(R|r,t+1)$ and taking into account the reliability of the sensor. Although we have no other reference for object detection besides the range sensors, we can increase accuracy by including any knowledge we have already regarding object presence at a given map location $\mathbf{x}$ with the variable $o=\{0,1\}$ with 1 defining an object being present and 0 defining an object not being present. This can be written in the form $\mathrm{P}(R|r,o,t+1)$, which can be solved for by applying Bayes' rule as [29]

$$\mathrm{P}(R|r,o,t+1) = \frac{\mathrm{P}(r|R,t)\mathrm{P}(R,t)}{\mathrm{P}(r,t)} \tag{32}$$
$$= \begin{cases} \frac{\beta_r \mathrm{P}(R,t)}{2\beta_r \mathrm{P}(R,t)-\beta_r-\mathrm{P}(R,t)+1}, & \text{if} \quad o=1 \\ \frac{(1-\beta_r)\mathrm{P}(R,t)}{-2\beta_r \mathrm{P}(R,t)+\beta_r+\mathrm{P}(R,t)}, & \text{if} \quad o=0 \end{cases}$$

where we use the law of total probability and the fact that $\mathrm{P}(r|R,t)$ is given by β_r. Equation 32 represents the "Sensor Model" nodes in the Bayesian network of Figure 8. This probability distribution is shown for the case where the probability of object detection by itself is constant as $\mathrm{P}(R,t) = \mathrm{P}(R) = 0.2$ in Figure 9. Of note is the fact that regardless of object detection o, the distribution converges to $\mathrm{P}(R,t)$ at $r = r_{max}$. To determine whether an object has been contacted, we need to make sure the output of the range sensor is above the noise level of the device, for which we make sure at least two samples in a row indicate that contact well above the RMS error σ_r is made.

$$o = \begin{cases} 1, & \text{if } r_{raw}(t) > 2\sigma_r \wedge \ r_{raw}(t-1) > 2\sigma_r \\ 0, & \text{otherwise} \end{cases} \tag{33}$$

4.2. Range Sensor Fusion

Three directional infrared sensors are placed on the front of the rover for use in obstacle detection, with the side sensors angled 30° out from the central

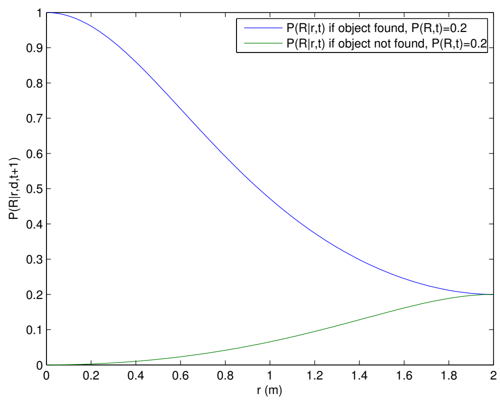


Figure 9: Probability Distributions $\mathrm{P}(R|r, d, t+1)$ for Infrared Range Sensor Model

sensor. To improve the detection reliability of objects in the probability map, the sensor data is combined using a linear opinion pool. Each sensor set at an angle θ_r is assumed to have a sensor angle of view w_r and a Gaussian horizontal detection likelihood, so the detection probability incorporating angular deviation for each sensor is estimated as

$$\alpha_s = \frac{1}{w_r^2\sqrt{2\pi}} * e^{-(\frac{\theta_r}{w_r})^2}. \tag{34}$$

To combine probability information in the Bayesian network, we can create a node that depends on other nodes above it in the structure. To combine the information from the three sensors together, we use the "Sensor Fusion" node, which provides an opinion pool for N sensors at angles of $\theta_n, n = 1 \ldots N$ with respect to a primary direction by

$$\begin{aligned} &\mathrm{P}(R|r, o, t, \theta_1, \theta_2, \cdots \theta_N) \\ &= \sum_{n=1}^{N} \mathrm{P}(R|r, o, t) * \frac{1}{w_r^2\sqrt{2\pi}} * e^{-(\frac{\theta_n}{w_r})^2}. \end{aligned} \tag{35}$$

4.3. Map Updates

Mapping of object probabilities is done at each point $\mathbf{x}$ within the sensor range r_{max} of $\bar{\mathbf{x}}$ at the angle θ_r. The rover's map is structured as an occupancy grid spanning the area in question, and functioning as a probability distribution over obstacle presence O. Initially, every grid element in the map is initialized to $\mathrm{P}(R)$, the estimated probability of encountering an obstacle in this environment, and the map is updated at the locations pointed to by the sensors. We consider map updates to be a Bayesian parameter estimation problem using a joint probabilistic model to obtain the probability $\mathrm{P}(O|R,\mathbf{x},t)$ of an obstacle being present at $\mathbf{x}$. We use the estimate of the probability of a range measurement given an obstacle at $\mathbf{x}$, the probability of an obstacle $\mathrm{P}(O,\mathbf{x},t)$, and the prior for any obstacle detection with Bayes' rule to form

$$\mathrm{P}(O|R,\mathbf{x},t+1) = \frac{\mathrm{P}(R|O,\mathbf{x},t)\mathrm{P}(O,\mathbf{x},t)}{\mathrm{P}(R,\mathbf{x},t)}. \tag{36}$$

A two-dimensional Gaussian function can be used to estimate $\mathrm{P}(O|R,\mathbf{x},t+1)$ on the map to capture the uncertainty in positional measurement. While the main goal is statistical identification of obstacles, it is also important to know how much certainty is present at each point in the map. The uncertainty in a measurement can be modelled as the informational entropy present [30]. The information entropy at $\mathbf{x}$ and time t for a Bernoulli distribution P_s=$\{P_s, 1-P_s\}$ are calculated as

$$\begin{aligned} &\mathrm{H}(O|R,\mathbf{x},t+1) \\ &= \min_t(-\mathrm{P}(R,\mathbf{x},t)\ln(\mathrm{P}(R,\mathbf{x},t)) \\ &-(1-\mathrm{P}(R,\mathbf{x},t))\ln(1-\mathrm{P}(R,\mathbf{x},t))). \end{aligned} \tag{37}$$

The minimum of all measurements over t taken at a mapped point $\mathbf{x}$ is used to reinforce that uncertainty decreases over time as more data is gathered. The use of a probabilistic sensor model allows the rover's view of the world to be fuzzy, so that rather than assuming a precise location for obstacles and landmarks, the rover can choose the path that is least likely to contain obstacles, and also consider the locations on the map that contain the least information to be the least reliable. This makes the system more robust to position errors and sensor inaccuracies, as it will attempt to choose the best solution while considering the presence of statistical errors such as Gaussian noise.

The rover maintains two maps of its operating area, one for detected object probability $\mathrm{P}(O|R,r,\mathbf{x})$ and one for accumulated entropy $\mathrm{H}(O|R,r,\mathbf{x})$, which are constant over t unless sensory data is added at any point $\mathbf{x}$. As the rover is currently only intended to travel several meters to carry out a mission goal, this is sufficient for short-range travel. The "Probability Map" node provides an interface from the occupancy grid to the Bayesian network by representing the map as a two-dimensional discrete probability distribution. As a probability distribution, the map can be part of a query to obtain the probability of obstacles at a specific location, or updated as part of a learning process with the original map serving as the prior distribution. The main difference in considering the map a probability distribution is that each row or column must be normalized to the size of the map for queries to be accurately carried out.

4.4. Mapping Methodology

Assuming the rover is present at the centroid of a grid element $\bar{\mathbf{x}}$ then a sensor reading at range r will affect positions $\mathrm{P}(\mathbf{x}_x + r\cos(\theta_r), \mathbf{x}_y + r\sin(\theta_r))$ and any adjacent locations within the distribution spread of the sensor. Entropy is mapped in much the same way, as $\mathrm{H}(\mathbf{x}_x + r\cos(\theta_r), \mathbf{x}_y + r\sin(\theta_r))$ for orthogonal Cartesian components $\mathbf{x}_x$ and $\mathbf{x}_y$ for each θ_r. Before any data is gathered, the probability map is initialized to $\mathrm{P}(R)$, while the entropy map is normally initialized to 1 at every point. As the search process within the map is not generally randomized, this leads to the same pattern being repeated initially. To evaluate the impact of varying initial uncertainty on the search pattern, the entropy map was also initialized using a pseudo-random value $\delta_h < 1$ as $1 - \delta_h < \mathrm{H}(\mathbf{x}) < 1$ in a separate set of tests. For efficiency, the rover is assumed to only evaluate a local area of radius d_{max} in its stored map at any given time t. Mapping continues until there are no remaining points with uncertainty exceeding a given threshold, $(\forall \mathbf{x}, \mathrm{H}(\mathbf{x}) < H_{desired})$.

Considering the set Δ_s as all points within this radius, where $\{\forall \mathbf{x} \in \Delta_s, ||\mathbf{x} - \bar{\mathbf{x}}|| < d_{max}\}$, the target location $\hat{\mathbf{x}}$ is generally chosen to be the point with maximum uncertainty:

$$\hat{\mathbf{x}'} = \arg\max_{\Delta_s}(\mathrm{H}(O|R, r, \mathbf{x})). \tag{38}$$

However, as this typically results in mapping behaviour that follows the numerical map search algorithm, it is desirable to provide a more optimal search metric. We choose the map location with maximum uncertainty within the margin δ_h and minimum distance from the rover and use a logical OR with Equation 38 in the algorithm as

$$\hat{\mathbf{x}} = \hat{\mathbf{x}'} \vee [\arg\max_{\Delta_s}(\mathrm{H}(O|R, r, \mathbf{x}) - \delta_h) \wedge \arg\min_{\Delta_s}||\mathbf{x} - \bar{\mathbf{x}}||]. \tag{39}$$

To make the target destination available as a probability distribution, we use a maximum likelihood estimation process to create a Gaussian distribution over a target location random variable T with the mean at the horizontal angle θ_t aiming toward $\hat{\mathbf{x}}$ and with a standard deviation of π radians so that at least a 180° arc has probability of reaching the target.

$$\mathrm{P}(T|R, \mathbf{d}, \theta_t) = \frac{1}{\frac{\pi}{2}\sqrt{2\pi}} e^{-\frac{(x-\theta)^2}{\pi^2/4}} \tag{40}$$

The node "Navigation Goal" in Figure 8 encapsulates this so that queries can be performed. The spread of the Gaussian distribution allows a wide variety of choices for steering angle even if obstacles are present between the rover and the target.

4.5. Navigational Decisions

For forward navigation, we would like to travel to the target location by the shortest route possible while minimizing the risk of a collision. From a naive Bayes standpoint, what we need is a probability distribution that includes both the probability of collision across the possible steering angles of the rover, and the distance to potential obstacles so that closer obstacles count as more dangerous than distant ones. This can accomplished by first obtaining the vector $\mathbf{d} = \hat{\mathbf{x}'} - \bar{\mathbf{x}}$ from the rover to the target point, and considering the area of the map occupying the solid angle θ_d from this vector centred on the rover, such that the angles $\theta \in [-\theta_d \ldots \theta_d]$ with respect to $\mathbf{d}$ are considered. A set of M discrete angles $\theta_m, m = 1 \ldots M$ can then be evaluated by summing the normalized total probability of encountering an obstacle over all locations along the length d_{max} vector $\mathbf{x}_\theta$ to form the probability distribution

$$\mathrm{P}(O|R, \mathbf{d}, \theta_m) = \frac{\sum_{\mathbf{x}_\theta} \mathrm{P}(O|R, \mathbf{x}_\theta)}{d_{max}}. \tag{41}$$

This provides a metric for the likelihood of encountering an obstacle in the direction θ and allows existing map data to help plan routes. To incorporate the concept of distance, the sum of the distribution is weighted by the distance $|\mathbf{x}_\theta - \bar{\mathbf{x}}|$, effectively increasing the likelihood of encountering closer obstacles.

$$\mathrm{P}(O|R,\mathbf{d},\theta_m) = \frac{\sum_{\mathbf{x}_\theta}(d_{max} - |\mathbf{x}_\theta - \bar{\mathbf{x}}|)\mathrm{P}(O|R,r,\mathbf{x}_\theta)}{d_{max}} \tag{42}$$

The probability distribution $\mathrm{P}(O|R,r,d,\theta_m)$ is implemented in the "Collision Avoidance" node in Figure 8, and is used together with the target location to drive the "Movement Model" node, which calculates a distribution $\mathrm{P}(M|O,R,r,\mathbf{d})$ over a random variable of movement direction M to prioritize the target point, but avoid areas with high obstacle likelihood.

$$\mathrm{P}(M|O,R,\mathbf{d}) = \mathrm{P}(T|R,\mathbf{d},\theta_t) + (1 - \mathrm{P}(O|R,\mathbf{d},\theta_m)) \tag{43}$$

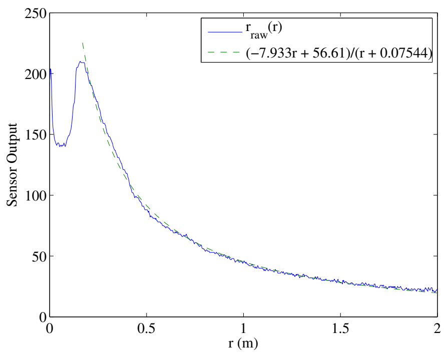
The query $\arg\max_M \mathrm{P}(M|O,R,r,\mathbf{d})$ is then used to determine the best choice of direction for forward movement.

5. Results and Discussion

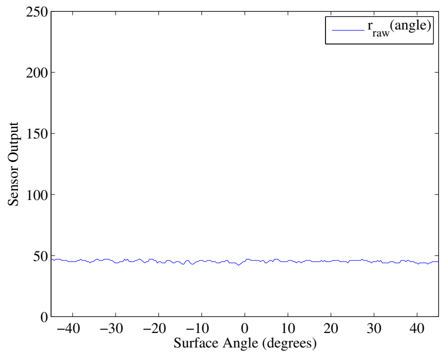
The Beaver was given a 20m by 20m area for motion, and mapping was constrained to this area in software, although the rover could physically leave the map due to turning radius constraints. Using the Bayesian mapping strategy with the goal of exploring all the given map area thoroughly, the rover was allowed to move freely in an area with no obstacles. For this test, $\beta_b = 0.2$, $d_{max} = 8m$, $\mathrm{P}(R) = 0.2$, and a grid with $0.5m$ resolution were used. The grid resolution reflects not only the noise and uncertainty in GPS measurements, but also the safety margin around obstacles that is desired to avoid collisions.

5.1. Range Sensor Characterization

To evaluate the performance of the μrover infrared range sensors used for navigation, the μrover was placed at $2m$ from one of the mapping obstacles and driven forward while taking positional measurements. By driving the infrared emitter with its maximum design voltage of $7V$, a maximum range of $r_{max} = 2m$ is possible, which is assumed by the sensor model. Additionally, to ensure that the range sensor would function at oblique angles to a target, a mapping obstacle was placed at $1m$ distance and the normal of the facing surface rotated through $\vartheta_o = (45° \ldots - 45°)$ with respect to the line of sight of the sensor. Profiles of digital range r_{raw} versus actual range and digital range with respect to the facing surface angle of the target ϑ_o are plotted in Figure 10.



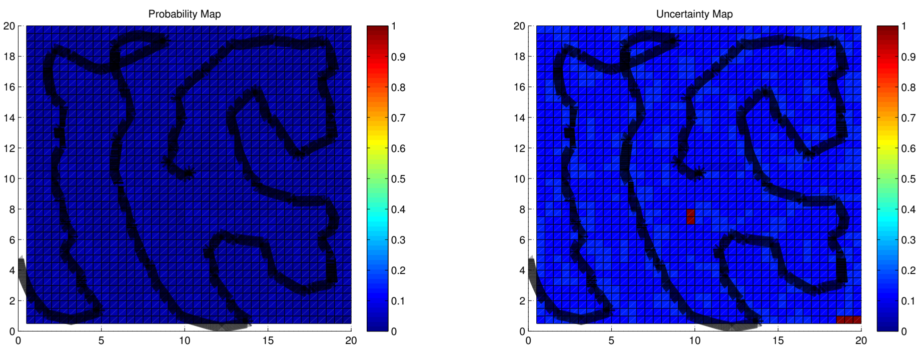
(a) Output Variation with Distance



(b) Output Variation with Angle to Plane Normal

Figure 10: Infrared Range Sensor Profiles for Distance (a) and Angle (b)

An 8-bit ADC is used for measurement of the range sensor, and the sensor output noise level remains within ±1 8-bit unit throughout most of the test. Fitting the profile of r_{raw} to a first-order rational function yields the polynomial fit $r_{raw} \approx (-7.933r + 56.61)/(r + 0.07544)$, which is solved for r to obtain the transfer function $r = -(0.08(943r_{raw} - 707625))/(1000r_{raw} + 7933)$ used to calculate the actual range from the given measurements. The combination of sensor, ADC, and polynomial fit noise results in an RMS error of $\sigma_r = 2.777$ 8-bit units (1.33%) in r_{raw}, which is more than acceptable for the main driver of uncertainty in the map, our rover positioning error. Variation with target surface angle of $r_{raw}(\vartheta_o)$ is also very low, showing no appreciable dependence on ϑ_o and only slightly higher noise than with no rotation. As there is no fitting estimation, an RMS error of 1.015 is observed for this case. In practice, infrared range sensors such as these are limited in capability by ambient sunlight. Strong direct sunlight can overwhelm the intensity of the infrared beam generated from the sensor and make it almost impossible for the sensor to distinguish the location of the reflected beam on nearby objects. For this reason, the current models of infrared sensor are considered to be low-cost development and testing alternatives to more robust range sensors such as laser rangefinders or LIDAR systems that would be used on actual flight hardware.



(a) $\mathrm{P}(O|R, r, \mathbf{x})$ (b) $\mathrm{H}(O|R, r, \mathbf{x})$

Figure 11: Probability and Uncertainty Maps Found by μrover, No Obstacles

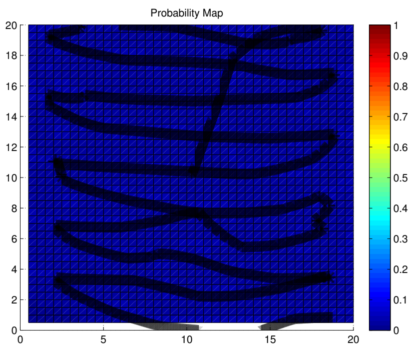
5.2. Autonomous Mapping Without Obstacles

First, each location in the probability map was initialized to 0 and each in the entropy map was initialized to 1. Figure 11 shows the probability and uncertainty maps. The path that the rover took during the test is shown as an overlaid black line. The initial entropy map was then modified with a pseudo-random offset as suggested above with $\delta_h = 0.1$. Figure 12 shows the probability and uncertainty maps. The new path that the rover chose is shown.

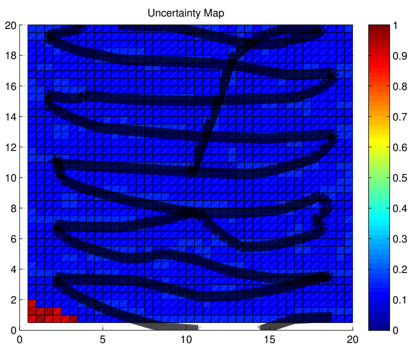
5.3. Autonomous Mapping With Obstacles

Three obstacles were then placed in the 20m by 20m area to test the obstacle avoidance methodology. Two $1mx1m$ obstacles were placed at $(14m, 16m)$ and $(7m, 10m)$, and a $0.5mx1m$ obstacle was placed at $(12.5m, 3m)$ in grid coordinates. The layout of the testing area used is shown in Figure 7. The rover was run with the same parameters as the tests above with the uncertainty map initialized to 0 and the entropy map initialized to 1. The resulting path and maps are shown in Figure 13. The test was repeated with the pseudo-random offset as suggested above with $\delta_h = 0.1$, and the results are shown in Figure 14.

The obstacles are not overly obvious given the statistical nature of the mapping, but they are visible as points of high probability and low uncertainty, while the remainder of the mapped area retains an uncertainty of close to 0.1 on average. The peaks in obstacle probability vary between runs due

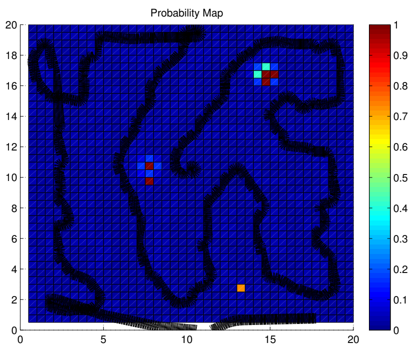


(a) $\mathrm{P}(O|R, r, \mathbf{x})$

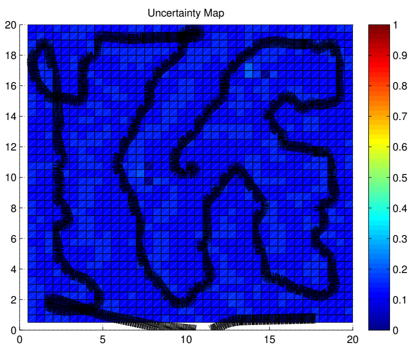


(b) $\mathrm{H}(O|R, r, \mathbf{x})$

Figure 12: μrover Probability and Initially-Randomized Uncertainty Maps Found by μrover, No Obstacles

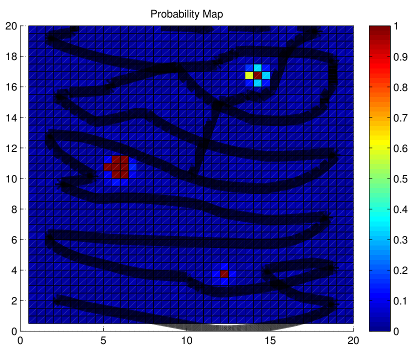


(a) $\mathrm{P}(O|R, r, \mathbf{x})$

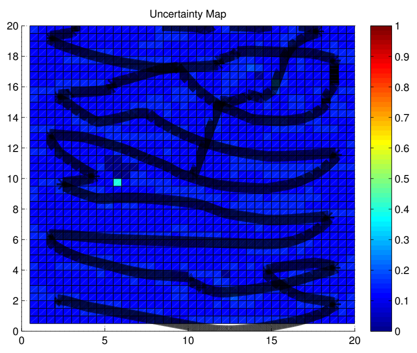


(b) $\mathrm{H}(O|R, r, \mathbf{x})$

Figure 13: Probability and Uncertainty Maps Found by μrover, With Obstacles



(a) $\mathrm{P}(O|R, r, \mathbf{x})$



(b) $\mathrm{H}(O|R, r, \mathbf{x})$

Figure 14: Probability and Initially-Randomized Uncertainty Maps Found by μrover, With Obstacles

to small differences in location $\mathbf{x}$ or sensor reading r that occur due to real-world uncertainties. However, even though the probability map varies with each run, the results obtained regarding object presence are very consistent, as statistical methods are generally robust to uncertainties. Because mapped probability depends on previous map measurements as well as current ones, if the micro-rover moves too fast, the reliability of the measurements will decrease as well. It can be noted that obstacles that lie directly in the path of the rover have better characterization in terms of high probability, because if the uncertainty driver does not force the rover to get close to obstacles, the sensor model will not place as high a reliability on the resulting probability.

The path was fairly consistent between test runs, with the exception of occasional sensor errors that momentarily cause the rover to begin obstacle avoidance behaviours, and show up as small course deviations or loops in the path. The pseudo-random offset caused the rover to prefer a Cartesian side-to-side movement indicating that the greatest uncertainty search in Eq. 36 was dominant, while a flat uncertainty initialization caused a shorter and less regular pattern indicating that the closest point search in Eq. 37 was dominant. The same random seed was used on all tests, so observing the same motion pattern in both cases is expected. This also causes a more thorough traversal of the map, resulting in better obstacle characterization. In both cases, it is evident that the obstacles are detected as peaks in the probability map and successfully avoided, although due to the high granularity of the mapping, the obstacle location accuracy is quite coarse. No comparison was done in this study between different test areas and obstacle layouts, and it is possible that a less predictable path could be desired. In this case, greater randomization in the algorithm would likely suffice.

6. Conclusions

Improving the reasoning capability on modern robots is an important part of making small planetary rovers more independent of humans. One of the greatest challenges for small planetary rovers is being able to make intelligent, appropriate decisions in environments that are inherently unknown and uncertain. Probabilistic methods are a good solution to this problem, providing a way to interpret large interrelated numbers of variables logically and choose more likely options while explaining away others. For this reason, we base the autonomy and machine intelligence of the μrover on Bayesian networks that are constructed from both prior knowledge and known relationships between hardware components and abstract concepts. A comprehensive description of how statistical and probabilistic methods are applied to the μrover has been given, with particular emphasis on the practicality of probability distributions with random variables as a way of representing abstracted data and making queries in a robotic system. Making decisions using logical propositions is described within the context of the Bayesian Robot Programming paradigm. Applying this paradigm specifically to the Bayesian network structure has lead to an efficient and intuitive way of representing robotic knowledge and drawing conclusions based on relationship information encapsulated in the network.

The novel, efficient, structured framework we have implemented for practical use of Bayesian networks and probabilistic queries can be applied to a wide range of robotic applications. By replacing definite variables with random variables and building networks of the four different kinds of nodes with either discrete or continuous probability distributions, any set of information can be obtained with relevance dependent on the accuracy of the distributions represented. The programming is efficient enough that queries can be made at high speed even on the embedded μrover hardware. Using this framework, we have shown the feasibility of using Bayesian methods for intelligent control on embedded micro-rover hardware for processing and mapping statistical data from a basic set of sensors. Results indicate that statistical methods can be used effectively with a simple sensor set and embedded hardware to provide systematic mapping and obstacle avoidance algorithms for outdoor navigation. This method can be extended to much more complex systems simply by adding sensors and their appropriate networked random variables.

It is expected that the Bayesian programming system will contribute to the success of the Northern Light mission by increasing the number of variables with associated uncertainty that can be considered when making decisions, and correspondingly decreasing the amount of manual planning and interaction between the rover and ground station. To justify the use of autonomy methods such as this on other planets, extensive development and testing of robust methods is required, and the cost savings in required personnel for operation will have to become significant with respect to the mission cost. It is expected that the importance of including statistical characterizations and uncertainty in planetary robotic systems such as the μrover will only increase as technology develops and more missions are fielded. Development of intelligent and robust autonomous systems should remain a high priority for future missions.

7. References

[1] G. D. Penna, B. Intrigila, D. Magazzeni, F. Mercorio, Resource-optimal planning for an autonomous planetary vehicle, International Journal of Artificial Intelligence and Applications 1 (3) (2010) 15 – 29.

[2] T. Estlin, D. Gaines, C. Chouinard, F. Fisher, R. Castano, M. Judd, R. C. Anderson, , I. Nesnas, Enabling autonomous rover science through dynamic planning and scheduling, in: IEEE Aerospace Conference (IAC 05). Big Sky, Montana., 2005.

[3] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), The MIT Press, 2005.

[4] T. Huntsberger, Onboard learning of adaptive behavior: Biologically inspired and formal methods, in: Advanced Technologies for Enhanced Quality of Life Conference, 2009, pp. 152 – 157.

[5] M. Gallant, A. Ellery, J. Marshall, Science-influenced mobile robot guidance using bayesian networks, in: Electrical and Computer Engineering (CCECE), 2011 24th Canadian Conference, 2011, pp. 001135 – 001139.

[6] U. B. Kjrulff, Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis, Springer Science, 2008.

[7] R. Kozma, T. Huntsberger, H. Aghazarian, R. Ilin, E. Tunstel, W. J. Freeman, Intentional control for planetary rover srr, Advanced Robotics 22 (12) (2008) 1309 – 1327.

[8] B. Quine, R. Lee, C. e. a. Roberts, Northern light - a canadian mars lander development plan, in: Conference Proceedings of CASI ASTRO 2008, Montreal, Canada, 2008.

[9] M. A. Post, Planetary micro-rovers with bayesian autonomy, Ph.D. thesis, York University (2014).

[10] M. A. Post, B. M. Quine, R. Lee, Beaver micro-rover development for the northern light mars lander, in: CASI ASTRO 2012, Quebec City, Quebec, Canada, 2012.

[11] M. A. Post, R. Lee, , B. Quine, Modular design for space engineering research platforms, in: 8th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Noordwijkerhout, Netherlands, 2011.

[12] J. Li, M. A. Post, R. Lee, A novel adaptive unscented kalman filter attitude estimation and control system for a 3u nanosatellite, in: 12th biannual European Control Conference, Zurich, Switzerland, 2013.

[13] D. Sekimori, F. Miyazaki, Precise dead-reckoning for mobile robots using multiple optical mouse sensors, in: J. Filipe, J.-L. Ferrier, J. A. Cetto, M. Carvalho (Eds.), Informatics in Control, Automation and Robotics II, Springer Netherlands, 2007, pp. 145–151, 10.1007/978-1-4020-5626-0_18.
URL http://dx.doi.org/10.1007/978-1-4020-5626-0_18

[14] M. Dille, B. P. Grocholsky, S. Singh, Outdoor downward-facing optical flow odometry with commodity sensors, in: Proceedings Field & Service Robotics (FSR '09), 2009.

[15] R. I. Hartley, P. Sturm, Triangulation, Computer Vision and Image Understanding 68 (2) (1997) 146 – 157. doi:http://dx.doi.org/10.1006/cviu.1997.0547.

URL http://www.sciencedirect.com/science/article/pii/S1077314297905476

[16] E. Rublee, V. Rabaud, K. Konolige, G. R. Bradski, Orb: An efficient alternative to sift or surf, in: ICCV 2011, 2011, pp. 2564–2571.

[17] P. Bessiere, O. Lebeltel, O. Lebeltel, J. Diard, J. Diard, E. Mazer, E. Mazer, Bayesian robots programming, in: Research Report 1, Les Cahiers du Laboratoire Leibniz, Grenoble (FR, 2000, pp. 49–79.

[18] O. Lebeltel, J. Diard, P. Bessiere, E. Mazer, A bayesian framework for robotic programming, in: Twentieth International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering (MaxEnt 2000), Paris, France, 2000.
URL http://hal.archives-ouvertes.fr/hal-00089153

[19] O. Lebeltel, P. Bessière, J. Diard, E. Mazer, Bayesian robot programming, Autonomous Robots 16 (1) (2004) 49–79.

[20] D. Koller, N. Friedman, Probabilistic Graphical Models, Principles and Techniques, MIT Press, Cambridge, Massachusetts, 2009.

[21] P. Dagum, M. Luby, Approximating probabilistic inference in bayesian belief networks is np-hard, Artificial Intelligence 60 (1) (1993) 141 – 153. doi:http://dx.doi.org/10.1016/0004-3702(93)90036-B.
URL http://www.sciencedirect.com/science/article/pii/000437029390036B

[22] D. Wu, C. Butz, On the complexity of probabilistic inference in singly connected bayesian networks, in: D. Slezak, G. Wang, M. Szczuka, I. Dntsch, Y. Yao (Eds.), Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, Vol. 3641 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 581–590.
URL http://dx.doi.org/10.1007/11548669_60

[23] J. A. Robinson, A machine-oriented logic based on the resolution principle, J. ACM 12 (1) (1965) 23–41. doi:10.1145/321250.321253.
URL http://doi.acm.org/10.1145/321250.321253

[24] J. M. Mooij, libDAI: A free and open source C++ library for discrete approximate inference in graphical models, Journal of Machine Learning

Research 11 (2010) 2169–2173.
URL http://www.jmlr.org/papers/volume11/mooij10a/mooij10a.pdf

[25] O. Lebeltel, P. Bessière, Basic Concepts of Bayesian Programming, in: Probabilistic Reasoning and Decision Making in Sensory-Motor Systems, Springer, 2008, pp. 19–48.
URL http://hal.archives-ouvertes.fr/hal-00338715

[26] R. Cover, Xml belief network file format (April 14 1999).
URL http://xml.coverpages.org/xbn.html

[27] M. A. Post, B. M. Quine, R. Lee, Bayesian decision making for planetary micro-rovers, in: AIAA Infotech@Aerospace 2012, Garden Grove, California, 2012.

[28] I. I. Hussein, D. M. Stipanovic, Effective coverage control for mobile sensor networks with guaranteed collision avoidance, IEEE Transactions on Control System Technology 15 (4) (2007) 642–657.

[29] Y. Wang, I. I. Hussein, Bayesian-based decision making for object search and classification, IEEE Transactions on Control Systems Technology 19 (6) (2011) 1639–1647.

[30] Y. Wang, I. I. Hussein, Multiple vehicle bayesian-based domain search with intermittent information sharing, in: American Control Conference (ACC), 2011, 2011, pp. 1280 – 1285.