6-26-2013

Verifying Performance for Autonomous Robot Missions with Uncertainty

Damian M. Lyons
Fordham University, dlyons@fordham.edu

Ronald Arkin
Georgia Institute of Technology, arkin@gatech.edu

Tsung-Ming Liu
Fordham University

Shu Jiang
Georgia Institute of Technology

Paramesh Nirmal
Fordham University

Verifying Performance for Autonomous Robot Missions with Uncertainty

Damian Lyons*, Ronald Arkin,**
Tsung-Ming Liu*, Shu Jiang **, Paramesh Nirmal*

** Fordham University, Bronx NY 10458 USA*
(Tel: 718-817-4480; e-mail: dlyons@ fordham.edu)
*** Georgia Institute of technology, Atlanta GA, USA*
(e-mail: arkin@gatech.edu)

Abstract:

Establishing performance guarantees for robot missions is especially important for C-WMD applications. Software verification techniques, such as model checking (Clark 1999, Jhala & Majumdar 2009), can be applied to robotic applications but characteristics of this application area, including addition of a robot environment model and handling continuous spatial location well, exacerbate state explosion, a key weakness of these methods.

We have proposed an approach to verifying robot missions that shifts the focus from state-based analysis onto the solution of a set of flow equations (Lyons et al. 2012). The key novelty introduced in this paper is a probabilistic spatial representation for flow equations. We show how this representation models the spatial situation for robot motion with environments or controllers that include discrete choice (constraints).

A model such as we propose here is useful only if it can accurately predict robot motion. We conclude by presenting three validation results that show this approach has strong predictive power; that is, that the verifications it produces can be trusted.

Keywords: Mobile Robots, Performance Guarantees, Formal Methods, Design Tools

1. INTRODUCTION

A robot or team of robots acting autonomously to search an area for a high impact target, a biological weapon for example, must perform predictably despite the uncertainty associated with the mission environment. We are developing a mission design toolkit that allows designers to include automatic verification of performance properties as part of the mission design cycle. When a robot operates in a dynamic and uncertain environment, its state at any point can only be characterized uncertainly. In this paper, we build upon our previous work in mission verification (Lyons et al. 2012, 2012b, Arkin et al 2012) and present here a probabilistic framework for verifying the performance of autonomous robot missions with uncertainty.

In Arkin et al. 2012 we introduced our approach, building on the *MissionLab* (MacKenzie et al. 1997) robot mission design, simulation and testing toolkit. Formally, we represent the robot program and the environment with which it interacts as concurrent communicating processes, and we use the techniques of *Process Algebra* (Baeton 2005) to analyze this interaction. In Lyons et al. (2012) we proposed an approach to analyzing robot behavior in uncertain and dynamic environments based on the identification of a behavioral periodicity, the *system period*. In particular we showed how verification of the combined robot and environment system could be reduced to the solution of a set of recurrent equations that we called the system flow equations. The variables in the flow equations, message communications in the underlying process networks, represent characteristics of the robot such as its location and velocity as well as the locations or other properties of parts of the environment.

In this paper, we address the issue of the probabilistic representation for variables that represent robot and environment characteristics. We begin by introducing a number of robot missions that show our application focus. We use these to motivate our selection of probabilistic representation. We consider how the process equations we develop for robot programs and environment models, in cases with and without discrete choice, give rise to probabilistic flow equations. The paper presents two novel results: First, we define a mixture of Gaussian representation for random variables in our flow functions, and we show how these are treated in the case of discrete choice/constraint (which can be in the environment, e.g., a wall, as well as a conditional program statement). Second, the method we propose here has value only if it accurately verifies the behavior of real robots. We report therefore validation results of our verification predictions for several robot examples to show that this approach produces results that can be trusted.

The remainder of the paper is laid out as follows. Section 2 places this work in the context of the literature in robot uncertainty and software verification. Section 3 presents examples of the kinds of robot mission on which our work is predicated. Section 4 briefly reviews our approach to mission verification and then presents our first main result, the

probabilistic representation in flow functions. In Section 5, we present the validation experiments that show the strong predictive power of the approach we have proposed here. Section 6 presents our conclusions and discusses future work.

2. LITERATURE REVIEW

Emergency response incidents such as counter weapons of mass destruction (C-WMD) and urban search and rescue (USAR) provide unique challenges for autonomous robotic systems. The operating environments in these domains could be highly unstructured (caused by an earthquake) or unknown (lack prior knowledge). While verification of robot missions under these kinds of naturalistic environments poses a greater challenge than traditional software verification, it is at the same time a necessary process to ensure robot mission success and safety under these conditions.

C-WMD and USAR missions are verified against specified performance criteria which vary drastically based on the emergency situation. Humphrey (2009) presented eight CBRNE (Chemical, Biological, Radiological, Nuclear, and Explosive) incident response tasks for robots: survey, identification, scene observation/object tracking, medical initial assessment, medical victim transportation, decontamination, hazard disposal, and resource hauling. Each mission category has different uncertainties associated with it. Thrun (2000) observed that the five primary sources of uncertainty in robotics are the environment, robots, sensors, models, and computation. The real world environment is dynamic and unpredictable; robots have imperfect actuation; sensor measurements are usually corrupted by noise; internal models are approximations of the real world; and uncertainty in computation involves algorithmic approximations needed real-time execution.

To provide performance guarantees for robots operating in real C-WMD and USAR domains, uncertainty needs to be properly represented and incorporated into the verification process. This research tackles this challenge by addressing the problem of verification of robot missions with uncertainties in robots, sensors, and the operating environment. Clark et al. (1999) describe how formal verification typically handles verification of digital hardware designs, network protocols, and verification of software. Verification of robot mission software shares many concerns with these but we argue it also has some very unique aspects.

In software verification, the performance criteria are expressed as liveness and safety conditions on program variable values. The ultimate effect of a robot program is however, motion of the robot and, possibly, an effect on the robot's environment; performance guarantees for the verification of robot software should therefore be liveness and safety conditions on the robot motion and on parts of the environment. The first implication of this is that any analysis of the performance of robot mission software must include a model of the robot's environment, since the mission software will behave quite differently in different environments (Lyons & Arkin 2004). The second implication is that the focus of robot mission software verification be on conditions on the robot motion and environment rather than on the values of arbitrary variables in the robot program.

In Lyons et al. (2012b) we present examples of these kinds of performance guarantees as probabilistic constraints on robot motion. There is an established literature on probabilistic representations for robot motion. Smith (1990) proposed to represent the uncertain spatial relationship in robot navigation by estimates of the mean and covariance of the system state vector. POMDPs have been a popular technique for planning under uncertainty in which the robot's state uncertainty is explicitly modelled and the robot chooses action based on the probabilistic distribution over state space (Vlassis et al. 2005). Filter-based methods (e.g., Kalman filters and particle filter) are also popular methods for robot location and mapping (Thrun 2005).

Software verification has focused on automata models (Jhala & Majumdar 2009) because of the need to verify conditions on arbitrary variables, and recent work in formal methods for robotics (Kress-Gravit & Wongpiromsam 2011) has followed that trend. We proposed an alternate model that focuses on processes rather than on states (Lyons & Arkin 2012, Lyons et al. 2012). Rather than requiring state-enumeration to verify a performance criterion, our approach generates a set of recurrent flow equations whose solution tests the performance guarantee. This differs from other work in software verification (e.g., SPIN, BLAST, etc. (Jhala & Majumdar 2009)) in moving away from a state-based approach. Probabilistic verifiers, such as PRISM (Kwiatkowska et al. 2011), are closer to our approach. Our intended focus on continuous spatial distributions to represent the robot and environment however, distinguish us from that work.

Our concern in this paper is how to represent probabilistic robot and environment motion in flow equations. We propose to represent robot motion, and other environment variables, by random variables with Gaussian mixture distributions (Bishop 2006). These allow us to capture the continuous, multimodal spatial distributions that result from probabilistic algorithms and interactions with motion, sensor and environment uncertainty as seen for example in mapping algorithms (Thrun 2005).

3. ROBOT MISSIONS

Emergency response (e.g., C-WMD, USAR) incidents present critical missions that are characterized by various stress factors: time pressure, high-stake risks, dynamic conditions and uncertainty. The objective of this research is to support a human robot operator's mission specification process in these naturalistic settings by providing feedback on the predicted performance of the robotic system. This section presents three robot missions as examples of the type of missions that our verification framework will analyze and provide performance guarantee for.

The Back and Forth mission was introduced in Lyons et al. (2012b), Figure 1, where the robot goes back and forth between points A and B. While this is a seemingly simple mission, uncertainties in the robot motion and environment can cause difficulty in its accomplishment. If the robot is conducting this mission in an open space indoors, where GPS cannot be used, the robot would have difficulty accurately localizing itself due to slippage between wheels and the floor.

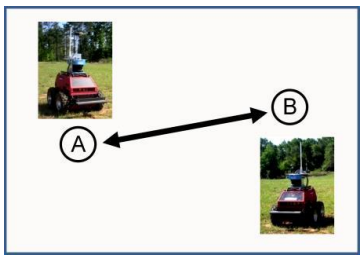


Figure 1: Back and Forth

A waypoint-based biohazard search mission is shown in Figure 2, where the robot is tasked to enter a building to look for a biohazard (Arkin et al. 2012). This is an example of the CBRNE survey task presented in Humphrey (2009) and it also presented a robot operating in a realistic environment (i.e., the basement of an office building).

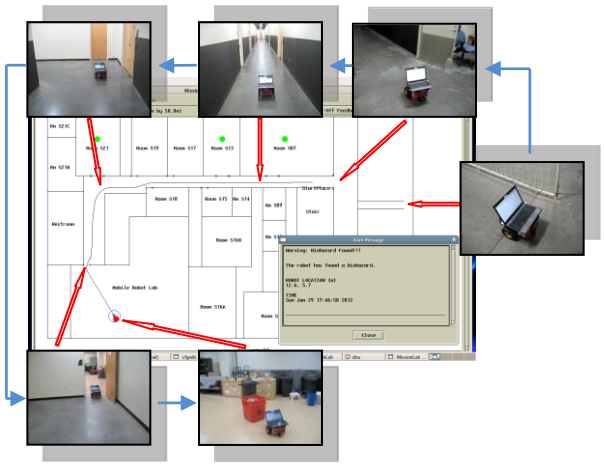


Figure 2: Biohazard Search

Figure 3 presents a multi-robot mission, where the robots alternate in advancing forward and taking *overwatch* position (i.e., covering for the advancing robot). This scenario is inspired by the military tactic called bounding overwatch, which is used by infantry to move forward under enemy fire. The ellipses in Figure 3 indicated where each robot stops and takes the overwatch position while the other robot advances.

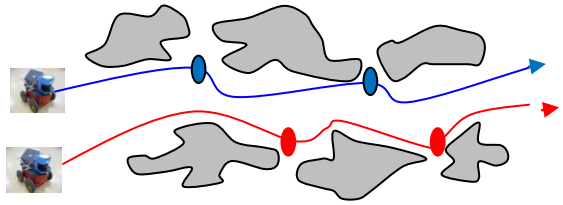


Figure 3: Bounding Overwatch

4. PROBABILISTIC REPRESENTATION

In Lyons et al. (2012) we introduced a process algebra, PARS[1], for representing robot missions: both the robot control program and also the environment in which the program will be carried out. From a system perspective: The robot controller is built in *MissionLab* and is translated to a PARS representation for the verification step (Arkin et al. 2012). The robot, sensor, and environment models are available in *MissionLab* as user-selectable libraries. The designer can ask the VIPARS[2] verification module whether a combination of controller and robot, sensor, environment models meets a performance specification. We do not address the system architecture issues further in this paper; Arkin et al. (2012) presents them in more detail. We begin by very briefly reviewing PARS.

4.1 PARS

Programs and environment models are specified in PARS as networks of communicating, concurrent processes. The *process* is the basic unit of program and environment model structure. An extended *port automaton* model (Lyons et al. 2012) provides the semantics for a process. We formalize processes as automata, and communication connections between processes as ports. We formalize the ways in which the automata can be composed to a port connection automaton as process algebra composition operations.

In PARS, we write a process **P** with initial parameter values *u1,u2,…* and which produces final result values *v1,v2,…* as: $\mathbf{P}_{u1,u2,\ldots}\ \langle v1,v2,\ldots\rangle$. Processes are defined as compositions of other processes using operators such as sequence (;), parallel-max (|) or parallel-min (#), ultimately in terms of set of predefined basic processes that implement port communications (**In** $_{p}\langle r\rangle$, **Out** $_{p,r}$), random sampling (**Ran**), timing (**Delay**$_t$) and so forth. In Lyons et al. (2012) we investigated the properties of a number of controllers and environments, including a robot controller **MoveTo** (1) and non-deterministic environment model **NEnv** (2) combined into a system **Sys** (3)**:**

$$\mathbf{MoveTo}_g = \mathbf{In}_p\langle r\rangle\ ;\ \mathbf{Neq}_{r,g}\ ;\ \mathbf{Out}_{v,\,s(g-r)}\ ;\ \mathbf{MoveTo}_g \quad (1)$$

$$\mathbf{NEnv}_{r,q} = (\mathbf{Delay}_t\ \#\ \mathbf{NOdo}_q\ \#\mathbf{At}_r)\ ;\ (\mathbf{Ran}_{N(0,s1)}\langle e1\rangle \mid \mathbf{In}_v\langle u\rangle)\ ;\ \mathbf{NEnv}_{r+(u+e1)\,t,\ q+ut} \quad (2)$$

$$\mathbf{NOdo}_q = \mathbf{Ran}_{N(0,s2)}\langle e2\rangle\ ;\ \mathbf{Out}_{p,q+e2}\ ;\ \mathbf{NOdo}_q$$

$$\mathbf{Sys}_{pi,g} = \mathbf{MoveTo}_g \mid \mathbf{NEnv}_{pi,pi} \quad (3)$$

The **MoveTo** robot controller process (eq. (1)) reads an input, a robot position from port *p* into a result variable *r* which is then tested to see if it is equal to the goal location *g*. If it isn't equal (**Neq**$_{r,g}$) then a value *s(g-r)*, a velocity proportional to difference between current and desired locations, is written to the robot velocity output port *v*, and this sequence then repeats. The sequence stops in the case *r* = *g*.

The **NEnv** environment process starts with three parallel processes; a timer process that stops after *t* time units (**Delay**$_t$), a process that repeatedly transmits the current robot location with some associated sensor noise *e2~N(0,s2)* (zero-mean with variance *s2)*, and a process that represents the position of the robot (**At**$_r$). After *t*, this network terminates and the velocity information *u (*from port *v)*, along with associated motor noise *e1~N(0,s1)*, is used to calculate a new location *r+(u+e1) t* and repeat the sequence.

4.2 System Flow Function

We have proposed an approach to efficiently analyze

[1] Process Algebra for Robot Schemas

[2] Verification In PARS.

properties of process networks such as the concurrent composition of (1) and (2) based on a novel process algebra expansion theorem (Baeten 2005). If the concurrent composition of controller and environment, **Sys,** consists of the processes **P1,** … , **Pn**, where, each process can be written recurrently as:

$$\begin{aligned} \mathbf{P1} &= \mathbf{P1'}\ ;\ \mathbf{P1}, \\ \mathbf{P2} &= \mathbf{P2'}\ ;\ \mathbf{P2}, \\ &\cdots, \\ \mathbf{Pm} &= \mathbf{Pm'}\ ;\ \mathbf{Pm} \end{aligned} \qquad (4)$$

$$\mathbf{Sys}_r = \mathbf{P1} \mid \mathbf{P2} \mid \ldots \mid \mathbf{Pm}$$

We developed an expansion theorem:

$$\mathbf{Sys}_r = \mathrm{F}(\mathbf{P1'}, \mathbf{P2'}, \ldots, \mathbf{Pm'})\ ;\ \mathbf{Sys}_{f(r)} \qquad (5)$$

The parameter r is the system flow variable, a vector of values that characterize the state of the program and the environment. We call F**(P1', P2', …, Pm')** the system period and $f(r)$ the system flow function. Both F and f can be generated by looking only at a single period **Pi** for each process i (though of course the result may be highly conditional). The result of all possible executions of **Sys** can now be characterized in terms of $f^n(r)$. In Lyons et al. (2012) we show how this result is leveraged for verification of performance guarantees for several examples. Our focus here is on the situation where r contains probabilistic information.

4.2 Probabilistic Flow Variables

Returning to the controller and environment in (3): The system flow function for (3) from Lyons et al (2012) is

$$f(r) = r + [s(\ g - (\ r + e2\ )\ ) + e1]t \qquad (6)$$

An evaluation of $f(r)$ requires a sampling of the random variables $e1$ and $e2$ from the two noise distributions in eq. (2). But this means that $f^n(r)$ no longer captures all possible executions of **Sys** – it's just one sample of an execution; different choices for $e1$ and $e2$ would have produced a different sample.

Let us consider the initial position p to be a random variable from a multivariate normal distribution $N(\mu_{p0},\Sigma_{p0})$. In that case, the ports, parameters and result variables that contain results calculated from the value of p must also be represented as random variables. Whenever variables are added (or subtracted) in the flow we need to replace this by convolution operations. If r and q are independent random flow variables with distributions $P_r(x)$ and $P_q(x)$ then $p+q$ is a random variable with distribution:

$$P_{p+q}(x) = \int_{-\infty}^{\infty} P_p(x-z)\,P_q(z)dz = P_p * P_q(x) \qquad (7)$$

And if $P_r(x)$ and $P_q(x)$ are normal distributions, $N(\mu_p,\Sigma_p)$ and $N(\mu_q,\Sigma_q)$, then so is $P_{r+q}(x)$, $N(\mu_p+\mu_q,\Sigma_p+\Sigma_q)$ (Bishop(2006)).

The random variable modified flow function, $f_{rv}(r)$ is therefore:

$$f_{rv}(r) = r * [s(\ g - (\ r * N(0,s2)\ )\ ) * N(0,s1)\ ]t \qquad (8)$$

Now when we look at $f^n_{rv}(r)$ we will see all possible executions of **Sys** again. Figure 4 below shows $r = N(\mu_{p0},\sigma_{p0})$ and the result of evaluating $f^n_{rv}(r)$ for several different values of n.

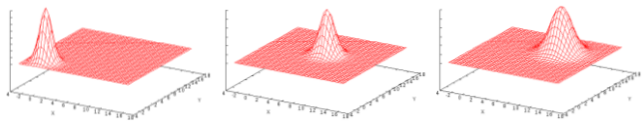


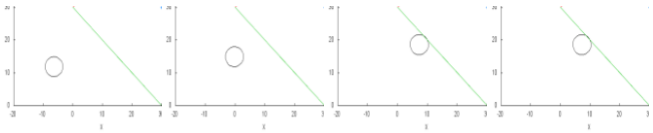
Figure 4: Example of $f^n_{rv}(r)$ solved for 3 values of n.

4.3 Conditional Flow Functions

The random variable flow function in (8) expressed an unconditional transformation of its input r to its output $f_{rv}(r)$. While there are many examples of this kind of system in robotics, there are also many examples where the transformation is conditional. One obvious way this conditionality can arise is from conditionals ("if statements") in robot program/controllers. Perhaps less obviously, it can also arise due to environment constraints. Consider the environment model below:

$$\begin{aligned} &\mathbf{WEnv}_{r,q} = (\mathbf{Delay}_t \,\#\, \mathbf{NOdo}_q \,\#\mathbf{At}_r)\ ; \\ &(\ \mathbf{Ran}_{N(0,s1)}\langle e1\rangle \mid \mathbf{In}_v\langle u\rangle\ )\ ; \\ &\quad (\ \mathbf{GTR}_{\,r+(u+e1)\,t\,,\,L}\ ;\ \mathbf{PASS}_{r,q}\langle nr,nq\rangle \mid \\ &\qquad \mathbf{LTE}_{\,r+(u+e1)\,t\,,\,L}\ ;\mathbf{PASS}_{\,r+(u+e1)\,t\,,\,q+ut}\ \langle nr,nq\rangle\ )\ ;\ \mathbf{WEnv}_{nr,nq} \end{aligned} \qquad (9)$$

The **WEnv** model above is almost the same as **NEnv** in (2) and uses the same variable naming. However, now each new position of the robot, $r+(u+e1)t$, is 'filtered' so that the robot is can only proceed if it is on one side of the wall L (see Figure 5) that bisects its world. **GTR** only allows its parallel branch of the program to be carried out if the projected position is greater than the line L: $r+(u+e1)t > \mathrm{L}$. Similarly, **LTE** only allows its branch to be executed if the projected position is less than or equal to the line L: $r+(u+e1)t \le \mathrm{L}$. The value of the flow variables nr and nq (which are analogous to r and q variables in (2)) depend on which **PASS** process passes the values along, which in turn depend on whether which condition process (**GTR** or **LTE**) succeeded.



(a) (b) (c) (d)

Figure 5: Example of conditional environment constraint: a wall L preventing the robot access to a portion of the world; (a) shows robot location distribution initially, a ellipse of 1SD centered on the starting location; and (b,c,d) show the distribution after some time has passed. The view is top down, and the (infinite) normal distribution is represented with 1SD ellipse.

When the position of the robot is represented as a probability distribution, then some part of the distribution will need to be evaluated with one branch of eq. (9) and the other part with the other branch of eq. (9).

One approach is to consider the mass of the position distribution that 'meets' each of the two branch conditions, and to use this to produce an output that is a weighted sum of both. Let $M(c(r))$ be the fraction of the mass of the distribution of the random variable r that meets the condition $c(r)$. We will refer to $M(c(r))$ as the *mass function* for the condition $c(r)$ on r:

$$M(c(r)) = \int_{z \in c(r)} P_r(z)\, dz \qquad (10)$$

For example, *M(r>L)* would be the fraction of the distribution *r* greater than the line *L*. In that case, *M(r≤L)* is the mass less than or equal to *L*, and *M(r≤L) = 1 - M(r>L)* since the probability mass is normalized to 1. In that case, we can write a flow function for eq. (9) as

$$f_{rv}(r) = M(r>L) * r + M(r \le L) * r * [s(g - (r * N(0,s2))) * N(0,s1)]t \qquad (11)$$

Now $f^{n}_{rv}(r)$ is the distribution that generalizes to the results of both branches. Consider an example of this approach where a robot is moving with some velocity uncertainty (and no use of sensors to avoid collision) as it passes a sharp corner. Let us assume that the flow function for the robot position $f^{n}_{rv}(r)$ is solved for a fixed value of *n* and the distribution is plotted as an ellipse. As the robot moves by the corner some portion of the distribution will get 'snagged' by the corner and be unable to progress, while the remainder will pass the corner unhindered and reach the goal. Using eq. (11), the resulting distribution has to capture this spread of results with a single distribution of large variance. However, this wide spread caused by the weighted sum approach used in (11) hides the fact that there are really just two results: a distribution close to the goal, and a distribution by the corner.

4.4 Conditionals using Mixture of Gaussians (MoG).

If we want to shows the two actual results in the previous example, we need to use a multimodal representation for our probability distribution. Using a normal distribution as a representation for random variables had the advantage of a large established literature (e.g., (Bishop 2006)); we would like to continue this advantage, so we adopt a mixture of normal distributions (MoG) as our model:

$$\mathrm{MG}(x; \{(\mu_i, \Sigma_i, w_i) | i \in 1,..,m\}) = \sum_{i=1}^{m} w_i\, N(x; \mu_i, \Sigma_i) \qquad (12)$$

$$\text{and} \sum_{i=1}^{m} w_i = 1$$

The effect of a conditional such as that in eq. (9) will be to generate additional modes (members) in the mixtures, avoiding the issue of overgeneralization we saw with the weighted-sum unimodal approach. The mass function *M(c(r))* needs to be redefined for this MoG case. There are two parts to the definition. If the flow variable *r* is represented by a MoG model, then the mass function is applied to each member of the model according to the member weights:

$$M(c(r)) = \sum_{i=1}^{m} w_i M(c(r_i)) \qquad (13)$$

The effect applying the mass function to a single Gaussian, *M(c(r_i))* is to generate a new Gaussian $N(\mu, \Sigma)$ representing the mass of the original distribution on one side of the condition $c(r_i)$ and its corresponding weight as a fraction of the original distribution:

$$M(c(r_i)) = (\mu, \Sigma, w) \qquad (14)$$

The mean and variance is arrived at by computing the expected value of the truncated normal distribution (Robert 1995).

$$\mu = \int_{z \in c(r_i)} z P_{r_i}(z)\, dz \qquad (15)$$

The addition operator in (6) now needs to take two MoGs M1 and M2 generated as indicated in eqs. (13) and (14) above and combine them into a single MoG assuming independence as follows:

$$\mathrm{M1} + \mathrm{M2} = \{(\mu_i, \Sigma_i, w_i) \mid i \in 1,..,m1,\ldots,m1+m2\} \qquad (16)$$

$$where\ (\mu_i, \Sigma_i, w) \in M1\ iff\ i \le m1$$

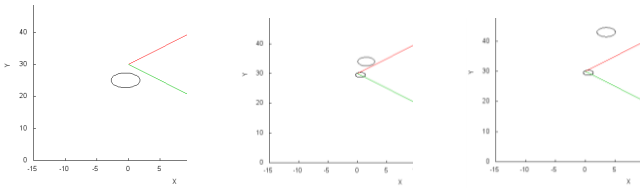
$$(\mu_{i-m1}, \Sigma_{i-m1}, w) \in M2\ else$$

$$w_i = {}^{w_j}/_{W} \text{ where } (\mu_i, \Sigma_i, w_{j=i}) \in M1\ iff\ i \le m1$$

$$and\ (\mu_i, \Sigma_i, w_{j=i-m1}) \in M2\ else$$

$$W = \sum_{(\mu_i, \Sigma_i, w_i) \in M1 \cup M2} w_i$$

Figure 6 below shows this example, the motion of a robot under uncertainty as it passes a corner. Note that the two spatial modes show up clearly in Figure 6(b) after the intersection with the wall.

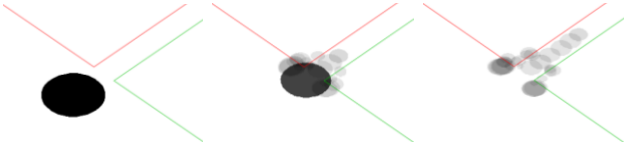


(a) (b) (c)

Figure 6: MoG distribution of location after moving (a) to (c) by a sharp corner with motion uncertainty.

4.5 Motion through a doorway

A common situation for a robot traversing an indoor site is moving through a doorway or passageway. This kind of scenario offers a lot of conditional interactions with the environment. Figure 7 below shows a robot moving through a passageway, again with motion uncertainty and no use of sensors to avoid collisions.



(a) (b) (c)

Figure 7: Traversal (a) through (c) of a passageway with uncertainty; Area of ellipse indicates variance; shading of ellipse indicates its weight in the mixture.

The flow function for the robot position $f^{n}_{rv}(r)$ is solved for a fixed value of *n* and all the distribution members plotted as ellipses. The result is a probability landscape for where the robot will be at three successive times.

The robot starts just outside the entrance to the passageway, and its distribution is shown in Fig. 7(a) as a single Gaussian with a weight of 1.0. As the robot moves towards the passageway, some probable locations will result in it colliding with the wall. The small, light-colored ellipses to each side of the doorway in Fig. 7(b) represent situations where the uncertainty has caused the robot to collide with one or the other side of the doorway.

The 'safe' positions of the robot (i.e., that did not collide) are smeared out along the passageway in Fig. 7(c), the result of the left and right wall constraints cropping the original position distribution to its central portion. The darker, overlap regions along the centre just indicate the summing of the individual member distributions. This is an artefact of drawing the distributions as bounded ellipses when in fact they are infinite.

The 'trail' of ellipses along each side of the passageway represents the (relatively small) number of situations where the uncertainty results in the robot jamming against the side of the passageway. After a short distance into the hallway these situations have too low a probability to see.

5. VALIDATION

The approach we propose here has value only if it accurately verifies the behavior of real robots. In this section, we report on our results in validating predictions made with our approach. We present three different validation results:

The first is a validation of the precision of motion, comparing our prediction of robot position after carrying out a single waypoint mission to the measured location of the robot after carrying out the mission.

The second is a similar validation of the verification prediction for a two waypoint mission.

The final validation is of the corner motion shown in Figure 6, comparing our prediction of the *proportion* of successes to the measured number of successes.

We calibrate the robot uncertainty model using empirically collected motion data that characterizes a *Pioneer-3AT* mobile robot operating in an indoor, laboratory environment. To collect this data, the robot was commanded to traverse straight-line motions of varying distances, and the error along the *x* and *y* plane is recorded. Rotational uncertainty was measured in a similar fashion. Based on the collected data, we characterized the resulting uncertainties using a linear model parameterized by the distance (angle) moved. The results of these uncertainty measurements are normal distributions for the translation and rotation error that are used during the verification of a controller.

5.1 Validation of Single Waypoint

In the single waypoint mission, the robot is given a goal location *10m* from its start location. The PARS model for the single waypoint mission in an uncertain environment, as presented in (3), was used incorporating the uncertainty calibration data.

The VIPARS verification module, (see Arkin et al. (2012) for a system diagram) is then used to predict the position of the robot. The module identifies the system period, extracts the flow equations, and solves them for specific goal conditions given by the performance criterion. In all the following cases, the performance criterion is that the robot have a cumulative probability of 80% of having reached the goal before a maximum time T_{max}. The output of VIPARS includes whether the performance criterion was met or not, and a spatial distribution for the robot location in either case.

In the single waypoint validation example, the robot start position *p0* was the distribution

$$N(\mu = (0,0),\quad \Sigma = \begin{bmatrix} 3.5\text{e}-4 & 0 \\ 0 & 3.5\text{e}-4 \end{bmatrix}) \qquad (17)$$

For the *10m* traverse (waypoint location=*(10,0)*) with velocity=*0.8m/s*, VIPARS confirmed that the performance criterion are met and returned the final position *p1* distribution:

$$N(\mu = (9.761,0.161),\quad \Sigma = \begin{bmatrix} 3.64\text{e}-4 & 0 \\ 0 & 3.5\text{e}-4 \end{bmatrix}) \qquad (18)$$

This, and the following, waypoint examples represent random variables as normal distributions rather than mixtures to allow for statistical significance testing.

This prediction is then validated against a set of empirically collected data points for the robot carrying out the *10m* traverse. Because of the inherent motion uncertainty, every time the traverse is carried out, the final robot position may be different. Our null hypothesis is as follows:

$$H_0:\quad \mu_{PARS} = \mu_{Ob} \qquad (19)$$

where μ_{PARS} is the mean final position predicted by the PARS verification module, μ_{Ob} is the observed mean position of the robot (in meters).

We validate PARS predictions using the χ^2 test as follows:

$$\chi^2 = [x, y] \cdot S^{-1} \cdot [x, y]^T \qquad (20)$$

Where [*x, y*] is the difference between the predicted position and the observed position, and *S* is the covariance of the PARS population. The critical value is:

$$\chi^2_{crit} = Q^{-1}\left(1 - \left(1 - Q(k)\right) + 0.95 \cdot Q(k)\right) \qquad (21)$$

Where $k = [0.005, 0.005] \cdot S^{-1} \cdot [0.005,0.005]^T$ is the χ^2-score for a *5mm* position error (which we consider as the minimum error), and *Q*(*k*) is the P-value of *k* obtained from the *two-dof* χ-table. If $\chi^2 < \chi^2_{crit}$ holds, then the results support the null hypothesis that $\mu_{PARS} = \mu_{Ob}$ within the 95% confidence interval given a minimum error of *5mm*.

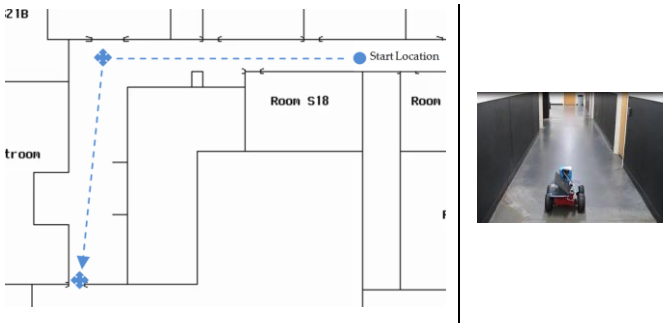
For the *10m* traverse, χ^2_{crit}=6.06 and the χ^2 test statistic computed from eq. (20) was 0.2578. Clearly $\chi^2 << \chi^2_{crit}$; the result emphasizes that our approach has the potential to produce accurate predictions of the behavior of the actual robot – that is, that the verification can be trusted.

5.2 Validation of Two Waypoint Mission

A more challenging validation case is the two waypoint mission shown in Figure 8. The robot carries out several motions in order to reach the final waypoint. We can validate the spatial accuracy of the final location as we did in the previous section.

Using the same translation and rotational uncertainty calibration as we did before, the VIPARS module confirms the performance criterion and its prediction for the robot position distribution after the second waypoint is:

$$N\left(\mu = (58.56, 33.10),\ \Sigma = \begin{bmatrix} 1.51 & 0.01 \\ 0.01 & 3.09 \end{bmatrix}\right) \quad (22)$$



(a) (b)

Figure 8: Two waypoint mission. (a) The mission is shown in dotted blue. (b) Robot near the start location

After collecting *n=10* samples of the Pioneer 3-AT robot carrying out the two waypoint mission, the observed mean location is *(57.83, 32.17)*. Carrying out a χ^2-test as in (11) and (12), we find that the critical value χ^2_{crit}=5.9916 and the χ^2 test statistic is χ^2=0.6274. Once again, $\chi^2 << \chi^2_{crit}$ showing strong evidence of the predictive power of our method and emphasizing that the verification can be trusted.

5.3 Validation of Missions with Environment Interactions

In both of the previous cases, the PARS verification only involved solving flow functions of the form (4) without discrete conditions. In our final example, we will validate a prediction from solving flow functions with discrete conditions, as in (6). We will use the example from Figure 6 of a robot moving in a straight line past a corner obstruction (without any sensing). Due to uncertainty, the robot sometimes collides with the wall. Our prediction in this case will produce a probability distribution member along the wall, and also a probability distribution member at the goal. In the two waypoint validation example, we did not explicitly model the wall collisions; we do so in this example.

Our statistic validation framework is different from the last two examples. It is time-consuming to repeat the empirical measurements of Section 5.1 and 5.2 for multiple waypoint missions and for missions where the robot end location varies widely. Also, we needed to restrict our random variables to be normal distributions rather than mixtures, to support the significance testing we were doing. Of course, for the corner example in Figure 6, we have already argued that a normal distribution is not sufficient and we extended our approach to normal mixtures. We need to be able to validate verification results that include mixtures.

We will use the results from the VIPARS prediction to determine what *proportion* of the distribution of predicted robot locations successfully passed the obstruction. The remaining proportion would have hit the wall at some point during the transit. From this we predict our success proportion p_{PARS}. The proportion of collisions predicted from verification will be given by the sum of the MoG member weights for distribution members at the goal location (as opposed to at the wall).

To validate the prediction, we collect empirical data for the robot carrying out this mission *n* times. We only count how many times the robot successfully reaches the end goal (versus how many times its motion uncertainty causes it to collide with the wall) which we record as the observed success proportion p_{Ob}. Using a *1-proportion z-test* we can test our hypothesis:

$$H_0: \quad p_{PARS} = p_{Ob} \quad (23)$$

The z-test statistic is calculated as

$$z = \frac{p_{PARS} - p_{Ob}}{\sqrt{\frac{p_{PARS}(1-p_{PARS})}{n}}} \quad (24)$$

Empirical measurements were taken in this case, by modeling the corner in Figure 6 in the lab with a box, Figure 9. Each time the robot hit the box was counted as a failure and each time it successfully reached the goal was counted as a success. After *n* trials the success proportion p_{Ob} was calculated as the ratio of successes to *n*.

Figure 9: Experimental setup for robot moving by a corner.

The VIPARS module reported that the performance criterion in this case would not be met (i.e., that by the time T_{max} there was not an 80% cumulative probability that the robot reached its goal location). The spatial distribution returned was inspected and the success proportion calculated as indicated above. The prediction was $p_{PARS} = 0.278\%$.

The empirical testing was carried out for *n=40* trials, and recorded a success proportion $p_{Ob} = 0.30$ The z-test statistic for this case was calculated from eq. (24), and is *z=0.31*. Such a low z-statistic is strong evidence in favor of the VIPARS prediction. Once again it shows that the prediction, in this case that the controller will not operate according to the performance criterion in this environment, can be trusted.

6. CONCLUSIONS

Being able to establish performance guarantees for robot missions is especially important for C-WMD applications. Unfortunately applying software verification techniques, such as model checking (Jhala & Majumdar 2009), to robotic

applications is difficult because of the many special robotic characteristics that exacerbate state explosion, including the necessity to verify a robot controller in conjunction with a model of its environment, and the importance of the continuous spatial state in performance guarantees (Lyons et al. 2012b). We have proposed an approach to verifying robot missions that shifts the focus from state-based analysis onto the solution of a set of flow equations (Lyons et al. 2012). In this paper we have introduced a novel probabilistic spatial representation for flow equations. We show how this representation models the spatial situation for robot motion with models (environment or controller) that include discrete choice (constraints). All of the examples focused on the conditional effects of the environment on the robot, since this is the less obvious, though no less important case to consider. The effect on the robot position distribution of conditionals in the program can be handled in exactly the same way. In Lyons et al. (2012) for example, we show an example controller and flow function for obstacle avoidance.

A model such as we propose is useful only if it can accurately predict robot motion. We concluded by presenting three validation results that show our approach has strong predictive power, that is, that its verifications can be trusted.

Comparing our work to other well known probabilistic verifiers such as PRISM (Kwiatkowska et al. 2011), the first important point of difference is our focus on spatial distributions representing the robot's physical location, and the mixture-based representation of random variables. Note that while probabilistic spatial filtering methods are common in mapping and localization (Thrun 2005), our flow functions are not restricted to just spatial variables and can represent other relevant mission variables as desired. A second point of difference is the system period (5) as mechanism to automatically construct the probabilistic flow functions (e.g. (6, 11)) used for filtering.

A practical aspect that we have not discussed in depth here is management of the number of members in the MoG for a random variable. In our implementation we have set a fixed maximum number of members for each variable. During the calculation of (10) the number of members will increase. At the end of the calculation, the number of members is again reduced to the maximum allowed by removing members with low weights and renormalizing. Other management policies are possible here including merging some low weight members that are spatially close. We note the similarity here to the issue of hypothesis pruning in techniques such as MHT and expect that similar concerns apply.

This paper has not addressed the software/architecture aspect of this work. That is addressed by Arkin et al. (2012) and includes the overall system diagram, the verification algorithms and their integration with the *MissionLab* mission design toolkit.

The examples shown in this paper have focused on fairly simply robot missions, with little or no sensor use. This is because this level of verification must function demonstrably well before the results of more complex missions can be evaluated. We are now working on versions of the waypoint mission that include laser ranging and visual sensing and its verification, and will also consider multi-robot missions.

ACKNOWLEDGEMENT

This research is supported by the Defense Threat Reduction Agency, Basic Research Award #HDTRA1-11-1-0038.

REFERENCES

Arkin R.C., Lyons, D.M., Nirmal, P., Shu, J. & Zafar, M., (2012) Getting it Right the First Time: Predicted Performance Guarantees from the Analysis of Emergent Behavior in Autonomous & Semi-autonomous Systems, *Unmanned Systems Tech. XIV*, Baltimore MD.

Baeten, J., (2005) A Brief History of Process Algebra. *Elsevier J. Theo. Comp. Sci. – Process Algebra*, 335(2-3).

Bishop, C.M., (2006) Pattern Recognition and Machine Learning, Springer.

Clark, E., Grumberg, O., Peled, D., (1999) *Model Checking*. MIT Press.

Humphrey, C.M., Adams. J.A. (2009), Robotic tasks for CBRNE incident response. *Adv. Robotics*, 23:1217-1232.

Jhala, R., Majumdar, R., (2009), Software Model Checking. *ACM Computing Surveys*, V41 N4.

Kress-Gazit, H., Wongpiromsarn, T., (2011), Corrective, Reactive, Highlevel Control. *Rob. & Aut. Magazine* 18(3).

Kwiatkowska, M., Norman, G., and Parker, D., (2011), PRISM 4.0: Verification of Probabilistic Real-time Systems. *Proc. 23rd Int. Conf. Computer Aided Verification*, v6806 LNCS, Springer.

Lyons, D.M., Arkin, R.A., (2004), Towards Performance Guarantees for Emergent Behavior, *IEEE Int. Conf. on Rob. & Aut.*, New Orleans LA.

Lyons, D., Arkin, R., Nirmal, P., and Jiang, S., (2012) Designing Autonomous Robot Missions with Performance Guarantees', *IEEE/RSJ Int. Conf. on Int. Rob. and Sys. (IROS 2012)*, Algarve, PT.

Lyons, D., Arkin, R.C., Fox, S., Jiang, S., Nirmal, P., and Zafar, M., (2012b) Characterizing Performance Guarantees for Real-Time Multiagent Systems Operating in Noisy and Uncertain Environments, *Proc. Performance Metrics for Intelligent Systems Workshop*, Baltimore MD.

MacKenzie, D., Arkin, R.C., Cameron, R., (1997) Multiagent Mission Specification and Execution. *Aut. Robots*, 4(1), 29-52.

Robert, Christian P. (1995). Simulation of truncated normal variables. *Statistics and Computing* **5** (2): 121–125

Smith, R., Self, M., Cheeseman, P. (1990), Estimating uncertain spatial relationships in robotics, *Autonomous Robot Vehicles*, Vol. 4, pp. 167-193.

Thrun, S., Beetz, M., Bennewitz, M., Burgard, W. Cremers, A.B. Dellaert, F. Fox, D. Hähnel, D. Rosenberg, C. Roy, N. Schulte, J. and Schulz, D. (2000), Probabilistic algorithms and the interactive museum tour-guide robot Minerva. *Int. J. of Robotics Research*, 19(11):972-999.

Thrun, S. Burgard, W. and Fox, D. (2005), Probabilistic Robotics. MIT Press, Cambridge, MA.

Vlassis, N., Gordon, G., & Pineau J. (2005), Reasoning with Uncertainty in Robotics (RUR-05), *IJCAI Workshop Notes*, Edinburgh, Scotland.