Formal Performance Guarantees for Behavior-based Localization Missions*

D. M. Lyons *Senior Member, IEEE*, R. C. Arkin *Fellow, IEEE*, S. Jiang *Student Member, IEEE*, M. O'Brien, F. Tang, P. Tang

(Walter, Taubig, & Luth, 2010)

***Abstract*— Localization and mapping algorithms can allow a robot to navigate well in an unknown environment. However, whether such algorithms enhance any specific robot mission is currently a matter for empirical validation. In this paper we apply our *MissionLab*/VIPARS mission design and verification approach to an autonomous robot mission that uses probabilistic localization software.**

Two approaches to modeling probabilistic localization for verification are presented: a high-level approach, and a sample-based approach which allows run-time code to be embedded in verification. Verification and experimental validation results are presented for two different waypoint missions using each method, demonstrating the accuracy of verification, and both are compared with verification of an odometry-only mission, to show the mission-specific benefit of localization.

I. Introduction

One of the most impactful recent developments in robotics has been efficient mapping and localization algorithms [1]: Techniques whereby a robot can use information from its sensors to construct a map of its environment and, at the same time, determine its location with respect to this map. However, whether such algorithms enhance, or are even necessary for any specific robot mission is currently just a matter for empirical validation.

Formal verification can be used as a design tool to determine whether a piece of robot software will function as desired without having to execute the software. The field has made significant strides in recent years with the development of model-checking [2] and SMT engines [3]. However, it can at best produce an approximation of robot performance, due to the undecidability of the underlying verification problem. A crucial issue therefore in selecting a verification approach is to understand what aspects of the robot software problem to focus on. Behavior-based robot programming is an important tool in autonomous robotics because it can yield programs that are robust to uncertainty about exactly what environment the robots will face during execution. For this reason, verification of behavior-based robot programs has become a topic of interest [4] [5] [6], and we focus on that approach here.

In recent work for the *Defense Threat Reduction Agency* [7] we have developed a unique combination of static analysis techniques and probabilistic reasoning to provide performance guarantees for behavior-based robot programs operating in environments with uncertain obstacles. Rather than addressing purely computational verification problems such as absence of deadlock or absence of run-time errors [8] [9], or verifying software generated control signals without consideration of the physical platform [10], our work focuses on establishing performance guarantees for the mission software with a complex and uncertain environment model.

In this paper we apply our technique to a behavior-based robot program that includes probabilistic localization (ROS AMCL) [11]. This the first time to our knowledge that a formal V&V method has been applied to such a system. Verification of this application is challenging because it absolutely requires an environment model, separate from, and interacting with, the behavior-based software. The model has to include the physical location of the robot, the geometry of the map, and the relationship between these and the sensor measurements. Uncertainty in physical location (at the least) needs to be modeled. However, verifying over all possible environment models, or even a sizeable subset of this deemed to have sufficient information for effective localization introduces overwhelming combinatorics.

Our approach is somewhat different: We argue that the *ultimate purpose of localization* is to improve mission performance and not to generate an accurate map without any consideration of how the map is needed or used. So our approach is to verify performance results for a behavior-based mission [11] with and without localization, thereby verifying whether including localization has been of value to the mission performance criteria or is even necessary. While we are verifying any potential execution of the mission software, we verify all those potential executions for a single map. We will conduct the mission verifications using maps previously generated by probabilistic mapping.

The platform that we use for verification is the *MissionLab* mission design toolkit [12] with *VIPARS* verification module [13], briefly reviewed in Section II. Section III presents some background on our formal verification technique and environment modelling, while Section IV presents the two approaches to modeling localization. One approach represents the functionality of localization at a high level. A second approach uses the actual ROS AMCL code during the verification process. Section V presents our results. We close the loop by comparing verification results with experimental validation for the mission. Results for each of these approaches is presented and compared with experimental validation results. The results demonstrate the accuracy of our verification and show that the benefit of localization is indeed mission-specific.

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

D.M. Lyons, F. Tang and P. Tang are with the Dept. of Computer & Information Science, Fordham University, NY 10458, USA (Ph: 718-817-4485, Fx: 718-817-4488, Em: dlyons@fordham.edu). R.C. Arkin, S. Jiang and M. O'Brien are with the Mobile Robotics Laboratory, Georgia Institute of Technology, GA 30332, USA (Em: arkin@cc.gatech.edu).

II. System Architecture

As robots grow in complexity along with their task demands, so do the opportunities for their failures and the difficulty to foresee those failures. Poor judgments of robot capabilities have led to failures of many robotic systems [14]. Furthermore, critical emergency response missions are typically characterized by a stringent window of opportunity for successful action. Therefore, it is imperative that the performance of robotic systems be guaranteed before mission execution. The goal of our research is to develop tools to provide such performance guarantees which mission operators can use to make the appropriate decision regarding robot deployments. The result of our research effort is a verification framework VIPARS, which provides the performance guarantee for a given mission based on how well the specified performance criteria are satisfied by the given control program, robot, and the environment models.

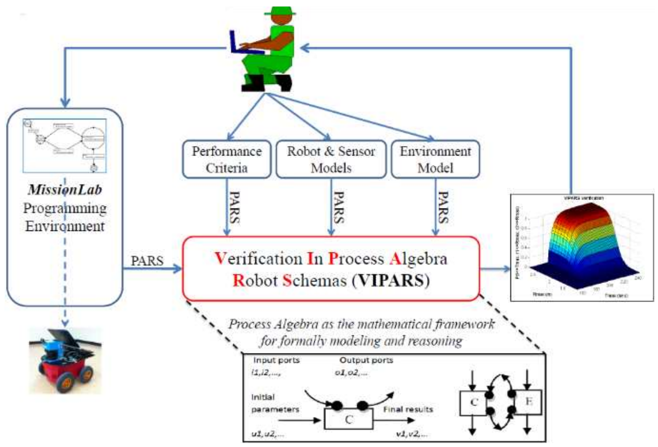


Figure 1: System Architecture (reproduced from [7])

The verification framework, VIPARS [13], is built upon *MissionLab* [12], a behavior-based robot mission specification environment (Fig. 1). *MissionLab* provides a usability-tested graphical programming interface, where the robot's program is specified in the form of a finite state automaton (FSA), assembled from a library of primitive behaviors. The output of VIPARS is the performance guarantee, currently quantified as probability distributions, that describes the likelihood of mission success. This output effectively forms a feedback loop that allows the mission operator to take preemptive measures against potential mission failures.

This paper extends our prior to address the verification of probabilistic behavior. Behaviors such as motion towards a goal or avoidance of a sensed obstacle are enhanced in [11] so that they use information from a map, previously generated by a probabilistic mapping algorithm. Furthermore, the robot location input for these behaviors is enhanced to leverage probabilistic localization. Because our verification approach is designed specifically for behavior-based programming, it is well-positioned to verify missions based on the approach in [11].

III. Environment Model for Localization

In prior work [7] [13] [15], Lyons et al. designed a framework for verifying the performance of autonomous behavior-based robot missions in uncertain environments. *MissionLab* [12] missions are autotranslated [15] to a *process-algebra* notation *PARS* (Process Algebra for Robot Schemas) for analysis. Environment models are also processes in this notation and we have proposed that a standardized set of environment models could be used to capture different classes of environment (e.g., motion uncertainty [13]; obstacle uncertainty [7]). The first phase of VIPARS verification is a static analysis of the concurrent, communicating mission software and environment model to generate a set of recurrent flow-functions capturing how the two interact. The second phase of VIPARS uses a Bayesian network to predict the performance of the mission from these functions. To provide context for this paper, below we review briefly the static analysis technique to extract flow functions and how a Bayesian network is then used to predict performance. The full details are in [13].

A. Automatic Verification with VIPARS

A behavior-based program and its environment are modeled in PARS as a set of interconnected, recurrent processes, where a process $\boldsymbol{P}$ is written as:

$$\boldsymbol{P}\langle \boldsymbol{u_1}, \ldots, \boldsymbol{u_n}\rangle(\boldsymbol{i_1}, \ldots, \boldsymbol{i_j})(\boldsymbol{o_1}, \ldots, \boldsymbol{o_k})\langle \boldsymbol{v_1}, \ldots, \boldsymbol{v_m}\rangle \quad (1)$$

where $u_1,\ldots,u_n$ are the *initial* values for the *process variables*, $i_1,\ldots,i_j$ and $o_1,\ldots,o_k$ are *input and output port connections*, and $v_1,\ldots,v_m$ are *final result values* of the process variables. Processes compute result values from initial values, and this computation may be influenced by any communications that occur over port connections.

Processes can be defined as combinations of other processes using composition operators: parallel ('|'), disabling ('#') and sequential (';'). Bounded recursion is captured using tail-recursive process definitions, e.g.,:

$$\mathbf{P}\langle x\rangle = \mathbf{Q}\langle x\rangle\langle y\rangle \,;\, \mathbf{P}\langle y\rangle \quad (2)$$

Here process **P** activates process **Q** with input value x. **Q** delivers output value y, which is then used to recur **P**.

A variable *flow function* f_P relates the values of variables at the start of each recursive step of **P** to those at the end. The flow-function for atomic processes are *specified a-priori*; those for composite process, those defined as compositions of other processes, e.g., (2), are *composed* from the flow functions of the component processes. This can be automated to generate flow functions given a set of processes [13] with complexity linear in the number of processes.

The system to be verified is expressed as the parallel, communicating composition **Sys** of a robot controller, e.g., **Ctr** with variable $r1$, and an environment model, e.g., **Env** with variable $r2$, written:

$$\begin{aligned} \mathbf{Sys}\langle r_1,r_2\rangle &= \mathbf{Ctr}\langle r_1\rangle(a)(b) \mid \mathbf{Env}\langle r_2\rangle(b)(a) && (3)\\ &= \mathbf{Sys'}\langle r_1,r_2\rangle \,;\, \mathbf{Sys}\langle f_{Sys}(r_1,r_2)\rangle && (4)\\ f_{Sys}(r_1,r_2) &= (f_{Sys,r1}(r_1,r_2),\ f_{Sys,r2}(r_1,r_2)) && (5) \end{aligned}$$

In eq. (3), the input of **Ctr** is connected to the output of **Env**, (a), and the output of **Env** is connected to the input of **Ctr**, (b). If (3) were a sequential composition like (2) then we could extract flow functions for the combined interaction of controller and environment. Therefore, in [13] we developed *an interleaving theorem*[1] for behavior-based systems to convert (3) to sequential form. The intuition is that a

[1] In process algebra, an interleaving theorem relates the sequential and parallel composition operations.

behavior-based system has behavioral 'states' each with an associated set of sensory triggered responses. A static analysis algorithm *Sysgen* was developed to identify the set of processes for these states and rewrite parallel compositions of the form (3) into a sequential composition (4) where **Sys'** is the automatically identified behavioral state processes, referred to as the *system period*. Once *Sysgen* analysis is complete, a *system flow function* can be extracted from **Sys'**. In the small example of eqs. (3), (4) above, the function extracted is shown in eq. (5). This is a recurrent function that evaluate the values for *r1* and *r2* as computed by the interactions between **Ctr** and **Env** in each execution of the system period **Sys'**.

Process variables, e.g., r_1, r_2, can be random or deterministic. Random variables are represented as multivariate mixtures of Gaussians, and operations on random variables are automatically translated by VIPARS into operations on distributions [16]. Flow functions relate variable values at recursion step t of **Sys'** to those at $t+1$, which can be written as conditional probabilities, e.g.,:

$$f_{Sys,r1}(r_{1,t}, r_{2,t}) = P(r_{1,t+1} \mid r_{1,t}, r_{2,t}) \qquad (6)$$

In the final phase of VIPARS processing, extracted flow functions are converted to conditional probabilities e.g., (6). These are then the basis of a Dynamic Bayesian Network [17] used to carry out filtering, forward propagation of probability distributions, to determine whether the combination of controller and environment will meet a performance specification.

Although [13] discusses more complicated performance guarantees, we basically restrict our attention to the guarantee that a mission will achieve some criterion on environment variables (usually a spatial accuracy for a waypoint goal and/or a temporal requirement for achieving the mission) with probability greater than a threshold before a time-limit has expired. We demonstrated that this approach is fast and accurate when validated against physical executions (most recently [7]).

B. Localization Mission System Process

The system process **Sys** for the localization mission is shown in eq. (7).

$$\begin{aligned}\textbf{Sys} = (\ & \textbf{Mission}\,(clp, clh, cl)(cv) \mid \\ & \textbf{Map}\langle sysmap\rangle()(cm) \mid \\ & \textbf{Localization}\langle D0\rangle(cp,co,ch,cl,cm)(clp,clh) \mid \\ & \textbf{MB\_Laser}\langle ms, mo, lo\rangle(cm,cp,ch)(cl)\,) \mid \\ & \textbf{Robot}\langle P0,H0\rangle(cv)(cp, ch, co)\,. \end{aligned} \qquad (7)$$

The **Mission** process is the translation of a waypoint mission in Section V (Fig. 5), and is fundamentally similar to all prior waypoint missions we have verified and validated. It has inputs <u>*clp*</u> (position), <u>*clh*</u> (heading) and <u>*cl*</u> (laser readings); and output *cv* (velocity). **Robot** is the environment model, capturing the motion and odometry error and robot interactions with obstacles, also fundamentally similar to our prior work. *PO, HO* are initial position and heading, inputs *cv* (velocity) and outputs *cp, ch* (odometry position and heading) and *co* (real position distribution, i.e., without sensing noise – used for performance estimation and high-level localization model).

However, there are three new processes: In the behavior-based localization approach [11], the obstacle avoidance sensor gets its information from the map, rather than directly from measuring sensory input. **Map** makes mapping information (from the a-priori generated *sysmap*) available on its output *cm*; **MB_Laser** uses the map to generate map-based laser data on its output *cl*. **Localization** implements a localization method using the map *cm*, laser *cl*, and robot *cp, co, ch* inputs. *D0* is the initial position uncertainty. The output of **Localization,** *clp*, is the localized position (and heading *clh*) used by the **Mission** process.

C. Map Representation

A key difference between this localization mission and prior missions to which we have applied our verification approach [13] [7] [15] is the map and the role it plays in the obstacle avoidance behavior and in localization. The **Map** process in (7) contains a map data structure *sysmap*. Random variables are represented in VIPARS as Mixtures of Gaussians distributions (MG). If $a \sim MG(CM)$, for $CM=\{(\mu_i, \Sigma_i, w_i) \mid i \in 1\ldots m\}$ the set of the mixture parameters (means, variances, weights), then a_i refers to mixture member $N(\mu_i, \Sigma_i)$, and $w(a_i)=w_i$ are the mixture weights, where $\sum_{i=1}^{m} w_i = 1$, and $CMG(x;\ CM) = \sum_{i=1}^{m} w_i N(x; \mu_i, \Sigma_i)$. The mixture size is written $|\,a\,| = m$.

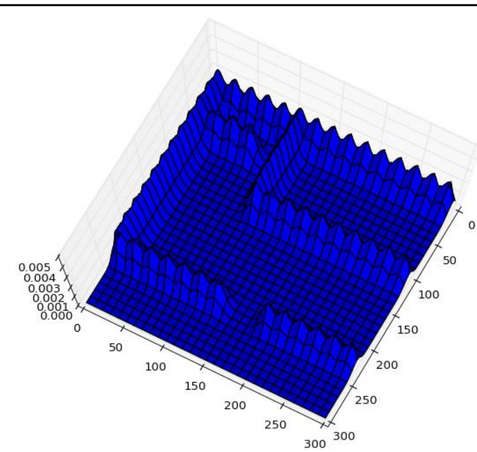


Figure 2: Example VIPARS Map Representation

Map information – the locations and geometry of obstacles, walls and other physical aspects of the mission environment – can be directly represented using this model. The interactions of the map with the robot and map-based sensor is analyzed in VIPARS by measuring the overlap between random variable distributions [7]. The advantage of this approach to representing physical geometry is that there is no restriction on the spatial location or extent of obstacles, and finer precision of modeling can be obtained at the cost of adding more mixture members (Fig. 2).

Definition. An *indexed mixture of Gaussians* is a mixture of Gaussians distribution $a \sim MG(CM)$ together with an index set *I*. The mixture is restricted as follows:

- $a[x] \equiv a_i$ where $\mu(a_i) = x \in I,\ i \in 1\ldots m$.
- $\mu(a_i) \in I$, *for all* $i \in 1\ldots m$; *a* only contains members indexed by <u>*I*</u>.
- For any $x \in I$, $|\{a[x]\}| \leq 1$; <u>*a*</u> has at most one member for each index.

We define $w[x]$ and $\Sigma[x]$ similarly to $a[x]$ to label member weights and covariances. A map is defined as a indexed bivariate mixture of Gaussians where $I=[0\ldots X]\times[0..Y]$ and where each member is a Gaussian kernel with covariance $\Sigma[x,y]=\sigma_m^2 I$, and where σ_m represents the map resolution. This corresponds somewhat intuitively with an occupancy grid representation, where $w[x,y]$ is related to probability of occupancy for the location (x,y).

During verification, the location random variable (the connection *cp* in eq. (7)) represents the location of the robot for *all* possible executions. It's relevant to compare this with the representation of robot location in a localization algorithm: the representation there may be also be a random variable, but the interpretation is different. In any single execution, the robot can really only be at a *single* physical location; the localization distribution is an estimate of this. In verification, the objective is not to find the single most likely location, but to propagate the effects of being at all locations. Rather than using a ray trace algorithm to determine how each location is supported by sensor readings and refining the position estimate based on that, the ray trace algorithm is used by the **MB_Laser** process to gather *all* possible sensor readings that can arise due to the robot location distribution.

IV. Modeling Localization

A common approach to verification is to manually implement the algorithm to be verified in a formal framework. Of course, this implementation may not represent the actual code. Published descriptions, even for widely known algorithms, have been shown to contain errors [18]. It also means that verification requires a huge investment of expertise and manpower [10]. Our prior work takes a different approach: Mission designers work directly in the *MissionLab* design toolkit, and their software can be automatically translated to PARS [15]. The approach is predicated on being able to provide a library of atomic behaviors that have been expressed in PARS already. So, to include a localization behavior in verification, it is necessary to build a model of the *MissionLab* implementation in PARS. But this is basically following the same flawed verification approach just discussed. We use two new approaches to this problem and we will evaluate both in our validation trials.

The first approach involves modeling localization at a high level: modeling not the actual collection of sensory data that produces improved position estimates, but just position estimates that improve with time according to some parameterization. This has the advantage that different localization algorithms can be included in verification by just changing the parameterization, not requiring as many hours of expert effort as implementing a new localization algorithm directly in the formal framework. It has the disadvantage that it decouples the localization from predicted sensor measurements, and may miss the effect of measurements that greatly improve or degrade the localization estimate.

The second approach involves the incorporation of existing localization code *directly* into the VIPARS verification algorithm. The main difficulty here is that localization code is designed to execute a single instance of a robot mission, whereas VIPARS is probabilistically reasoning about all executions that are possible given the a-priori environment model information. Our approach considers the embedded code to be capable of transforming a sample from a PARS random variable, and we define a framework for sampling and reconstructing variable distributions. This approach has the advantage of using the actual code that will get executed by the robot at run-time for the mission. It has the disadvantage of potentially lengthening verification times, since multiple samples need to be evaluated for a representative result.

A. High-level Model Approach

Localization starts with the odometry estimate of position at time step t, $q(t) \sim MG$. Through comparisons of sensory returns and the map, it refines the odometry estimate, bringing it closer to the actual position of the robot at time t, $p(t) \sim MG$. At any time, therefore the localization position is some combination of the odometry and the actual position:

$$\ell(t) = (1-k(t))\, p(t) + k(t)\, q(t) \tag{8}$$

where $k(t)\in [0,1]$ is a time varying gain with $k(t_0)=1.0$, forcing localization to start with just the odometry estimate. The improvement of localization with time is modeled by a monotonic decreasing dynamics for k:

$$k(t+\Delta t)= t_c\, k(t) \tag{9}$$

For time constant $t_c\in [0,1]$ determined from calibration measurements of the localization algorithm to be verified.

B. Sampling Approach

Consider that the C++ program we want to add to a mission is **P**. A PARS process wrapper for **P** is built, so the code behaves like a 'black box' process $\mathbf{P}\langle x\rangle\langle y\rangle$. Then, like every PARS process, it has an associated flow function $f_P(x)=(y)$ which is calculated by VIPARS. However, when **P** is called, it will map one input value x to an output, y; only one possible execution of **P**, whereas verification has to check *all* possible executions. So this approach to embedding **P** doesn't work, but, embedded code can *only* be called in this way.

Our approach is to define an extension to the flow function f_P from the process/program **P**: the mixture extended flow function F_P takes a random variable x as input and produces a random variable **y** as output. It samples the input distribution x and calls f_P on the samples, and reconstructs the output distribution mixture $p(\, y \mid x\,)= F_P(\boldsymbol{x})$ from the result..

Definition. Let $f_P(x)=y$ be the flow-function for the code to be embedded in verification, defined only by executing that code. Let $\boldsymbol{x}$, $\boldsymbol{y} \sim MG(CM)$ be random variables over the type of the variables x, y which we denote T. The *mixture extended flow function* (MEF) F_P is defined as follows.

- f_P: $T\rightarrow T$, where $y=f_P(x)$, for $x, y\in T$,
- F_P: MG$\rightarrow$MG, where $\mathbf{y} = F_P(\boldsymbol{x})$, for $\boldsymbol{x}, \boldsymbol{y}\in$MG (where MG is the set of all MG), and
- where we define **y=x**
- except $\mu(y_i) = f_P(\mu(x_i))$ for all x_i in $\boldsymbol{x}$, and
- where $\sigma(y_i)$ is calculated as follows:
  - $\mu'_j = f_P(s_i)$ for s_i a sample of the input x_i
  - $\sigma(y_i)= \sum_{j=1}^{k} N(s_j; \mu(x_i), \sigma(x_i)) \left(\left(\mu'_j - \mu(y_i)\right)^2\right)$

The MEF preserves number of members (|**y**|=|**x**|). Each mean is transformed directly $\mu(y_i) = f_P(\mu(x_i))$, requiring multiple executions of the embedded code. Finally, each variance is calculated by carrying out further sample executions for each member $\mu'_j = f_P(s_i)$.

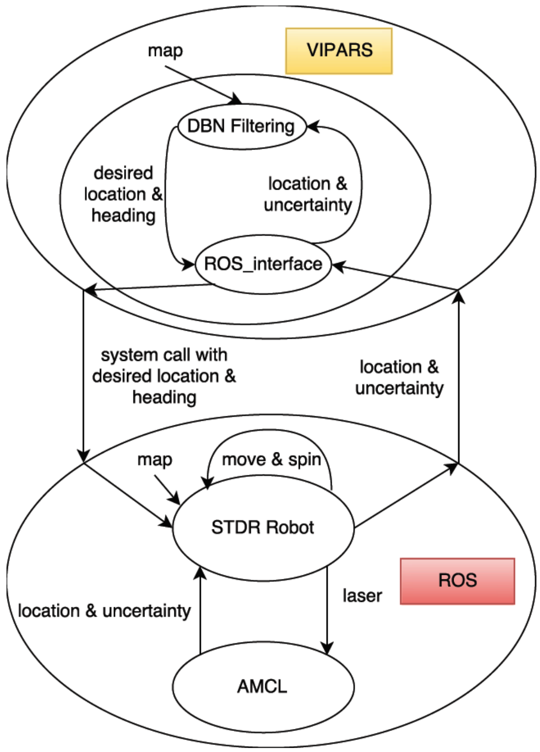


Figure 3: VIPARS-ROS Architecture

C. *Embedding ROS AMCL Localization*

The localization algorithm used in this paper was Adaptive Monte Carlo Sampling (AMCL) [19] as implemented in ROS. In the sampling approach, the DBN filtering engine of VIPARS issued requests to a ROS-based AMCL server to evaluate the MEF function for the **Localization** process. The interaction is shown in Fig. 3: Whenever the flow function for the **Localization** process needed to be evaluated on a position random variable, the position variable was sent from the DBN filtering engine (Top, Fig. 3) via a pipe to a concurrently running ROS system (Bottom, Fig. 3). The STDR simulator node was instructed to move the robot to the appropriate position, and localization data collected from the AMCL node. For simplicity, the MEF function was restricted to single member variables, and rather than calculating the variance by evaluating multiple samples, only the mean value was transformed and the variance calculated by convolving the mean with a zero-mean distribution $N(0, \sigma_s)$. This simplified the hysteresis issue with calling AMCL. The hysteresis challenge in fully implementing the MEF Definition for AMCL is discussed in the Conclusion.

V. VERIFICATION AND VALIDATION

To assess the effectiveness of the verification in providing performance guarantees for probabilistic robot behaviors, we present two waypoint missions, where the robot is tasked to navigate through a series of waypoints toward a goal with behaviors that are based on probabilistic algorithms. The general assessment process consists of three steps: 1) verification – use VIPARS to generate a performance guarantee for the mission with respect to some specified performance criteria, 2) validation – conduct experimental trials of the mission with a real robot, 3) evaluation – compare the predicted performance generated by VIPARS with the actual performance of the robot.

The waypoint missions are illustrated in Figure 4. The mission proceeds with robot starting at (2, 2) and navigates by following a series of waypoint to the goal locations at (11.7, 12.5) and (1.0, 7.3) respectively for each mission. The behavior of the robot for *Mission-B* (Fig. 4a) is shown in Fig. 5, which was created in *MissionLab* in the form of an FSA. The robot FSA consists of a series of *GoToGuarded* and *Spin* behaviors, whose transitions are prompted by *AtGoal* and *HasTurned* triggers. The behavioral FSA for *Mission-A* is similar to the one shown in Fig. 5, and is omitted for brevity.

In contrast to the behaviors we had examined in our prior work, the behaviors here have leveraged the probabilistic robotic algorithms to improve mission performance. However, these probabilistic behaviors present new verification challenges we have not addressed previously. Specifically, the perceptual schemas of *MoveToGuarded* and *AvoidObstacles*, two of the constituent primitive behaviors of the high-level *GoToGuarded* behavior, are augmented with a SLAM-based spatial map [11]. The *MoveToGuarded* primitive behavior drives the robot to a specified location with a radius of velocity dropoff around the goal. Instead of using odometry for localization, the perceptual schema of *MoveToGuarded* is replaced with the adaptive Monte Carlo localization (AMCL) algorithm [20]. This probabilistic localization algorithm takes the robot odometry and an a-priori acquired map as inputs, and outputs an estimated pose of the robot along with a covariance matrix representing the uncertainty of the estimated pose. Furthermore, the *AvoidObstacles* behavior uses the spatial map to generate repulsion vectors instead of using direct sensory reading from the laser scanner. The perceptual schema of the *AvoidObstacles* is modified to turn the spatial map into a pseudo laser scans of the environment through beam tracing within the occupancy map. As a result, the *GoToGuarded* behavior utilizes perceptual information (i.e., robot pose and obstacles) generated by probabilistic algorithms to generate motor response while navigating through the waypoints.

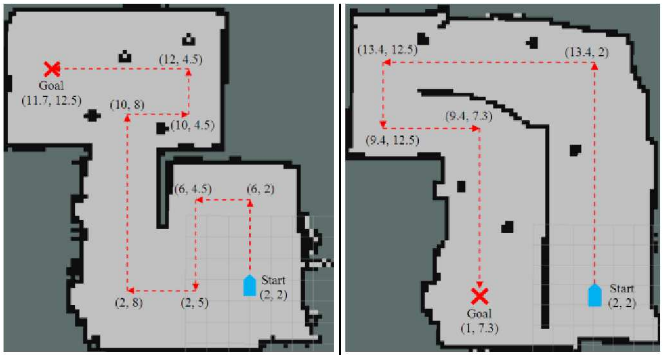
Performance criteria are mission specifications that a robotic system needs to meet. For missions A and B, this is:

- R_{max} – maximum radius of spatial deviation allowed from the goal.
- T_{max} – maximum allowable mission completion time.

Moreover, each waypoint mission is considered successful only when both performance criteria are met. Thus, the overall mission success is defined as:

$$Success = (r \leq R_{max}) \text{ and } (t \leq T_{max}) \tag{10}$$

where r is the robot's relative distance to its goal location and t is the time the robot to finish a mission. The objective of VIPARS is then to verify how well these performance criteria are satisfied by the combination of the robot, its behavioral FSA, and the operating environment.



a. Waypoints Mission A

b. Waypoints Mission B

Figure 4: Waypoint Missions for Verification and Validation

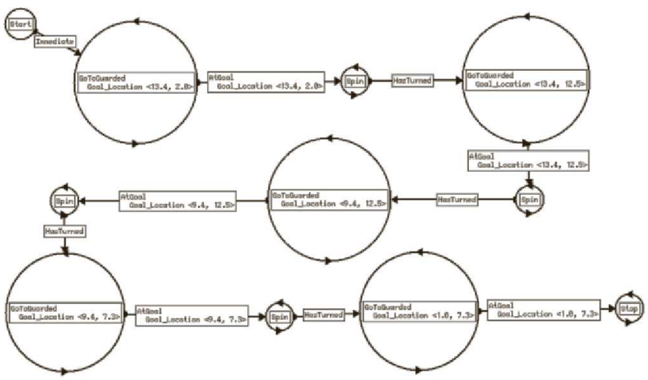


Figure 5: Behavioral FSA for *Mission-B*

A. *Verification*

Both verification approaches were applied to both waypoint missions. For the high-level approach, **Localization** in eq. (7) implemented (8), (9) with the gain parameter from (9), $t_c = 0.99$. This value was empirically determined from experimentation with ROS AMCL running on a Pioneer 3-AT robot carrying out short waypoint missions.

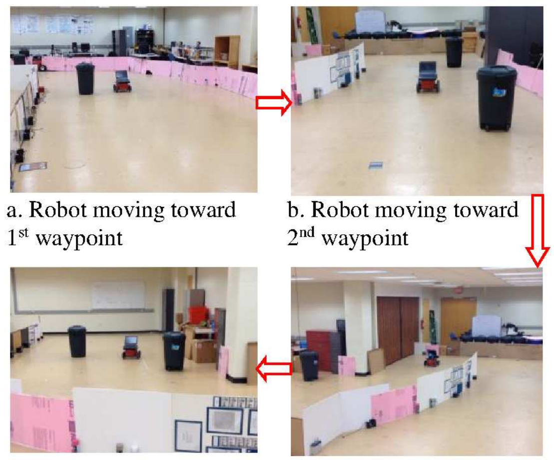
The sample-based approach implemented the architecture of Figure 3 using ROS version Indigo. A third, odometry only version of the mission was also run through verification for the purpose of comparing with both localization methods, and determining whether localization helped the mission or not. No additional validation was done on the odometry only version since that replicates our prior work.

The results of carrying out verification using both approaches with both waypoint missions was a set of performance graphs (as described in [7]) showing the predicted performance of the missions with respect to the performance criteria (10).

B. *Validation*

Validation experiments of the waypoint missions were conducted to illustrate that VIPARS' predicted performance of the mission is consistent with the robot's actual performance. The robot used for the experimental trials is the Pioneer 3-AT, a four-wheeled skid-steered mobile robot. The robot is also equipped with a forward-facing SICK laser scanner. The complete validation experiment consists of 50 trial runs for each waypoint mission respectively, which resulted in a total of 100 trial runs. Snapshots of the waypoint mission B are shown in Figure 6. Mission success is defined by how well the performance criteria (10) are met. For each trial, the following performance variables were measured:

- t – Mission completion time
- r – Robot's relative distance to its goal location



a. Robot moving toward 1st waypoint

b. Robot moving toward 2nd waypoint

c. Robot after turning a corner

d. Robot moving toward goal location

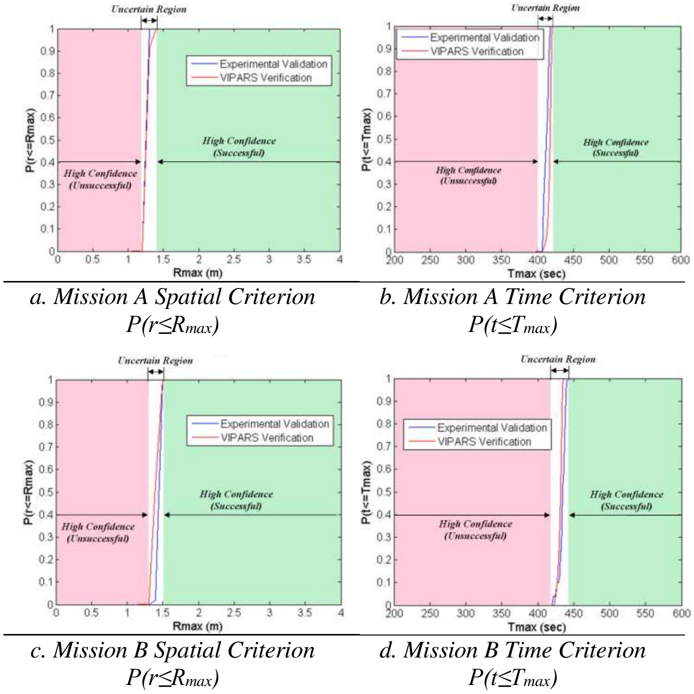
Figure 6: Snapshots of Validation for *Mission-B*

C. *Verification vs. Validation (V&V)*

Verification and validation are conducted independently by our two research groups, and the results are not shared until the final comparison stage. Figure 7 shows the results of verification and validation of the waypoint missions. The performance guarantee is quantified as a probability distribution that represents the robot mission's likelihood for success. These results also serve as the basis for performance feedback; and how this information should ultimately be presented to the mission operator was investigated in our recent human-subjects study [21].

Figure 7 shows the validation results of the performance guarantees for the two waypoint missions. These results are obtained with the sampling-based model of the probabilistic localization as described in Section IV. Figs. 7a and 7c show the V&V results for the spatial criteria $P(r \leq R_{max})$, the probability that the robot arrives within R_{max} radius of its goal location. Figs. 7b and 7d show the comparisons for the time criteria $P(t \leq T_{max})$, the probability that the waypoint mission is completed under the time limit, T_{max}. The results illustrate that the VIPARS verification of performance guarantees are consistent with the outcomes from experimental validation. The V&V results can be divided into three regions for further interpretation: *High Confidence (Unsuccessful)*, *Uncertain*, and *High Confidence (Successful)* regions. The *High Confidence (Unsuccessful)* is the region of near zero verification error and the mission has a zero probability of success. The *Uncertain* region is the region where verification error is significantly greater than zero and the probability of mission success is between 0 and 1.0. As a result, the robot is not guaranteed to succeed with the mission. The *High Confidence (Successful)* is region of near zero verification error and the mission is guaranteed to succeed with probability of 1.0. Consequently, the mission operator's decision for robot deployment can be based on which region of the mission criteria fall into. For instance, if the specified performance criterion falls within the

Unsuccessful region (e.g., R_{max}=0.5m), the operator can either abort the mission or modify mission parameters or design.



a. Mission A Spatial Criterion $P(r \le R_{max})$

b. Mission A Time Criterion $P(t \le T_{max})$

c. Mission B Spatial Criterion $P(r \le R_{max})$

d. Mission B Time Criterion $P(t \le T_{max})$

Figure 7: Results of VIPARS Verification and Experimental Validation of Spatial and Time Performance Criteria for Waypoint Missions A and B. Figures 6a & 6b show the V&V results of spatial and time performance respectively for Mission-A, where the results are divided into three regions based the performance guarantees: *High Confidence (Unsuccessful), Uncertain*, and *High Confidence (Successful)*. Figures 6c and 6d show the V&V results of Mission-B.

The overall mission success (Eq. 10) is defined in terms of both spatial and time criteria. Thus, we examined further in Figs. 8 and 9 the effects of various combinations of spatial and time criteria (R_{max} and T_{max}) on the mission success and verification error. The results can also be used to answer queries regarding the performance guarantee for a specific combination of T_{max} and R_{max}. Fig. 8 shows the effects of the time criterion T_{max} on the V&V results of the spatial criterion $P(r \le R_{max})$ for *Mission A*. While the T_{max}'s in both of its high confidence regions (Fig. 7b) have no effect on the verification error for $P(r \le R_{max})$, T_{max}'s that are in the *Uncertain* region (e.g., T_{max}= 415 sec) incur significant verification errors. For instance, for T_{max}=415sec, VIPARS predicted a success probability of 0.18, while the robot was actually successful 76% of the time in experimental trials.

Fig. 9 shows the effects of the spatial criterion R_{max} on the V&V results of the time criterion $P(t \le T_{max})$. While similar observations can be made here as in Fig. 8, in this case, R_{max}'s have much less impact on the verification error of $P(t \le T_{max})$ due to VIPARS's accuracy in predicting the spatial performance of mission even in the uncertain region (as shown in Fig. 7a). Nonetheless, missions with performance criteria in the *Uncertain* regions should generally be avoided.

Lastly, we have also examined the different verification results of VIPARS based on how the probabilistic localization mechanism is modeled: sampling-based and high-level model-based (Section IV). These results are also compared in Mission B to the verification result for the case when *only odometry information* is used for localization. These verification results are shown in Figs. 10-11 along with the validation result for *Mission-A*. While the verification results for different localization modeling approaches are comparable for the time criterion (Fig. 10), the performance based on the sampling-based model is more closely aligned with the validation result for both spatial and time criteria.

The odometry-only Mission B was 100% *unsuccessful* during verification due to collisions. However, with the final waypoints moved just 15 cm, the odometry-only mission finishes, and with almost identical accuracy to the nominal (original waypoint) mission shown in Figs. 10, 11. Because a small modification enables the odometry-only mission to be potentially successful, it is also clear that localization is not always required for mission success.

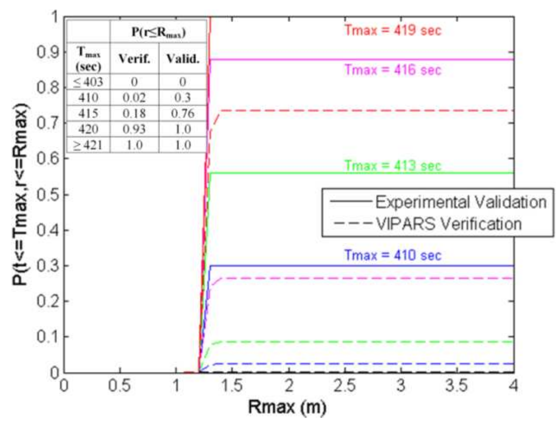


Figure 8: V&V of Spatial Criterion at various T_{max} for *Mission A*

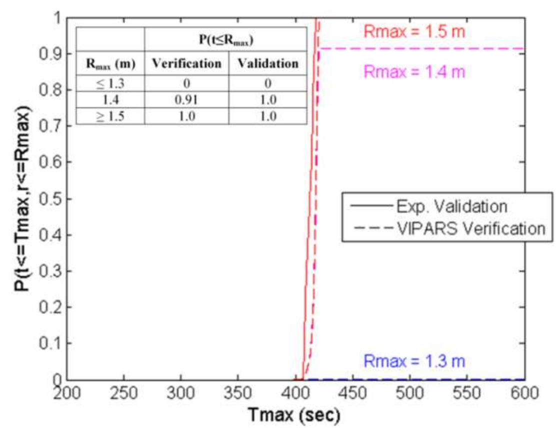


Figure. 9: V&V of Time Criterion at various R_{max} for *Mission A*

VI. Conclusion

Localization and mapping techniques intuitively offer advantages for robots navigating in unknown environments. This paper has applied our work in verification of autonomous behavior based robot missions to the problem of determining whether there is a mission-specific benefit to using localization. The *MissionLab*/VIPARS mission design and verification approach was extended to handle two approaches to modeling localization: a high-level approach in which only position estimate improvement is modeled, and a sample-based approach, in which the run-time localization code is embedded in verification. Extensive experimental validation is reported for two different waypoint missions using localization. The discussion of Figures 10 and 11 indicates that the sample-based approach yields the more accurate estimate, even for the sampling simplification made in this paper.

While there is support (Fig. 11) for the intuition that localization is an asset to mission performance (100% failure of the non-odometry mission); a minor modification of 15cm will allow the mission to be verified successful, indicating that the need for localization is mission-specific. Thus we argue that benefit to the mission is a better evaluation of localization than (the more common) general spatial accuracy.

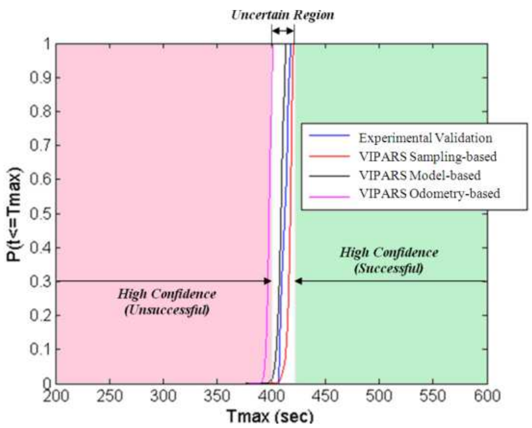


Figure 10: V&V of Time Criterion and Models of Localization

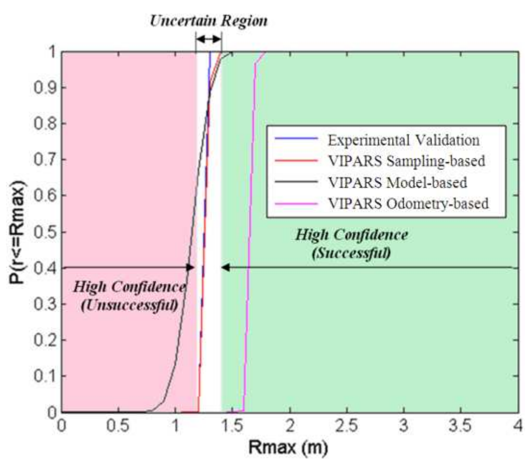


Figure 11: V&V of Spatial Criterion and Models of Localization

To completely implement the mixture extended function for the sampling-based approach in this paper, the full motion history for each sample request would need to be sent to the STDR node and AMCL reset between samples. The ability to cache these multiple sensory histories would improve computation time, but at the cost of directly instrumenting AMCL – a step we were avoiding for reasons discussed.

VII. References

[1] T. Bailey and H. Durrant-Whyte, "Simultaneous Localization and Mapping (DLAM): Parts I & II," *IEEE Robotics and Automation Magazine,* June, September 2006.

[2] R. Jhala and R. Majumdar, "Software Model Checking," *ACM Computing Surveys,* vol. 41, no. 4, 2009.

[3] L. DeMoura and N. Bjorner, "Satisfiability Modulo Theories: Introduction and Applications," *CACM,* 54(9), 54-67, 2012.

[4] A. Cowley and C. Taylor, "Towards Language-Based Verification of Robot Behaviors," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ Int. Conf. on*, 2011.

[5] M. Proetzsch, K. Berns, T. Schuele and K. Schneider, "FORMAL VERIFICATION OF SAFETY BEHAVIOURS OF THE OUTDOOR ROBOT RAVON," *4th Int. Conf. on Inf., Aut. and Control*, Dortmund, Germany, 2007.

[6] Ropertz, T. and R. Berns., "Verification of behavior-based networks using satisfiability modulo theories," in *ISR/Robotik 2014; 41st International Symposium on Robotics*, 2014.

[7] D. Lyons, R. Arkin, S. Jiang, D. Harrington, F. Tang and P. Tang, "Probabilistic Verification of Multi-Robot Missions in Uncertain Environments," in *IEEE Int. Conf. on Tools with AI*, Vietro sul Mare, Italy, 2015.

[8] P. Trojanek and K. Eder, "Verification and testing of mobile robot navigation algorithms," in *IEEE/RSJ Int. Conf on Intelligent Robots and Systems (IROS*) Chicago, 2014.

[9] D. Walter, H. Taubig and C. Luth, "Experiences in Applying Formal Verification in Robotics," in *29th Int. Conf. on Comp. Safety, Reliability and Security*, Vienna Austria, 2010.

[10] M. Kim, K.-C. Kang and H. Lee, "Formal Verification of Robot Movements - a Case Study on Home Service Robot SHR100," in *IEEE Int. Conf. Robotics and Automation*, 2005.

[11] S. Jiang and R. Arkin, "SLAM-Based Spatial Memory for Behavior-Based Robots," in *11th IFAC Symposium on Robot Control (SYROCO)*, Salvador, Brazil, 2015.

[12] D. MacKenzie, R. Arkin and R. Cameron, "Multiagent Mission Specification and Execution," *Autonomous Robots,* vol. 4, no. 1, pp. 29-52, 1997.

[13] D. Lyons, R. Arkin, S. Jiang, T.-L. Liu and P. Nirmal, "Performance Verification for Behavior-based Robot Missions," *IEEE Trans. on Robotics,* vol. 31, no. 3, 2015.

[14] J. Brahman, "Verification and Analysis of Goal-Based Hybrid Control Systems," Ph.D. Thesis, California Institute of Technology, Pasadena CA, 2009.

[15] M. O'Brien, R. Arkin, D. Harrington, D. Lyons and S. Jiang, "Automatic Verification of Autonomous Robot Missions," in *4th Int. Conf. on Simulation, Modelling and Prog. for Aut. Robots*, Bergamo, Italy, 2014.

[16] D. Lyons, R. Arkin, T.-L. Liu, S. Jiang and P. Nirmal, "Verifying Performance for Autonomous Robot Missions with Uncertainty," in *IFAC Intelligent Vehicle Symposium*, Gold Coast Australia, 2013.

[17] S. Russel, P. Norvig, Artificial Intelligence, Prentice-Hall, 2010.

[18] A. Zaks and R. Joshi, "Verifying Multi-threaded C programs with SPIN," *15th Int. SPIN Workshop* Los Angeles CA, 2008.

[19] D. Fox, "KLD–Sampling: Adaptive Particle Filters," in *Neural Information Processing Systems 14 (NIPS)*, Vancouver Canada, 2001.

[20] F. Dellaert, D. Fox, W. Burgard and S. Thrun, " Monte Carlo localization for mobile robots," in *IEEE Int. Conf. on Rob. & Aut.*, Detroit, 1999.

[21] M. O'Brien and R. Arkin, "An Analysis of Displays for Probabilistic Robotic Mission Verification Results," *7th Int. Conf. on App. Human Factors & Ergonomics.* Las Vegas NV, 2016.