2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Choosing the Right Tool for the Job: Online Decision Making over SLAM Algorithms

Samer B. Nashed[1], Roderic A. Grupen[1] and Shlomo Zilberstein[1]

***Abstract*— Nearly all state-of-the-art SLAM algorithms are designed to exploit patterns in data from specific sensing modalities, such as time-of-flight and structured light depth sensors, or RGB cameras. This specialization increases localization accuracy in domains where the given modality detects many high-quality features, but comes at the cost of decreasing performance in other, less favorable environments. For robotic systems that may experience a wide variety of sensing conditions, this difficulty in generalization presents a significant challenge. In this paper, we propose running several computationally cheap SLAM front ends in parallel and choosing the most promising feature set online. This problem is similar to the Algorithm Selection Problem (ASP), but has several complicating factors that preclude application of existing methods. We first provide an extension of the ASP formalism that captures the unique challenges in the SLAM setting, and then, based on this formalism, we propose modeling the SLAM ASP as a partially observable Markov decision process (POMDP). Our experiments show that dynamically selecting SLAM front ends, even myopically, improves localization robustness compared to selecting a static front end, and that using a POMDP policy provides even greater improvement.**

I. Introduction

Deployed robotic systems are complex, often comprised of dozens or even hundreds of sub-systems and algorithms, each designed for a specific purpose and specific operating conditions. For example, different methods for simultaneous localization and mapping (SLAM) may be designed for different sensors (camera [22], LiDAR [23]), extrinsic sensor calibrations (front facing [7], top facing [9]), or environmental structures or affordances [25]. While such specialization often allows greater performance by leveraging structure in data to perform more efficient or accurate computation, it comes at the cost of restricting the set of operating conditions in which the system can perform reliably. Moreover, outputs of these sub-systems are often inputs to other sub-systems and thus affect the quality of future computation in ways that are frequently too uncertain or too complex to model.

One strategy to combat these complexities is modularity. Roboticists have identified common processes (e.g. SLAM) where similar data (e.g. RGB or depth images) are processed to produce similar outputs (e.g. pose estimates). Here many algorithms may be interchanged without affecting the module's interface with other parts of the system. However, as robot deployments encounter greater variety in operational conditions, it becomes increasingly difficult to design singular, one-size-fits-all modules (algorithms) that can perform reliably under all possible conditions. Moreover, as sensors become cheaper, and sensor suites larger, it may be difficult for robots deployed in demanding or dangerous environments to incorporate all possible strategies for contingencies with different operational sensor suites within a single module. We term the development of single modules the 'one-size-fits-all' approach to system design, and our hypothesis is that this is not always the best approach to developing robust, broadly capable robotic systems.

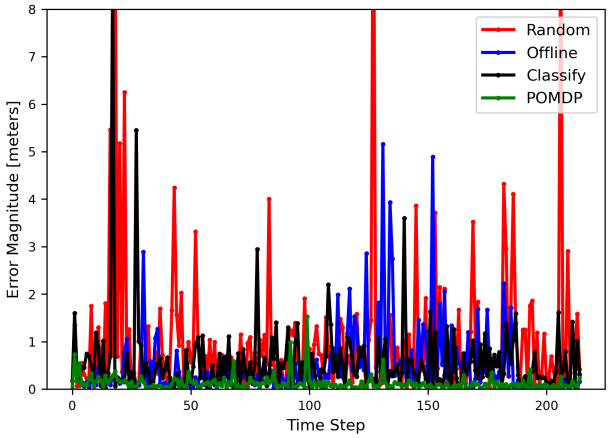

Fig. 1. Time series of localization errors for several approaches to the SLAM ASP. The POMDP-based approach is the only one capable of reasoning about both the immediate quality of particular sensors and the long-term effects of constructing optimization problems using these sensors.

This paper presents an alternative to the 'one-size-fits-all' architecture. Instead, we propose storing several redundant algorithms in memory, each of which may be substituted within a particular module in the stack, and then selecting the most appropriate instance of that module online. This approach essentially replaces the task of designing singular modules that operate reliably under all possible conditions with the task of efficiently identifying which existing algorithm is most reliable in the current situation. This architecture relieves the tension between specialization and generalization inherent in many robotics choices, since if a reliable algorithm exists for the current conditions and we can identify it, the system can 'generalize' to that situation without compromising performance in other situations.

Our primary contributions are (1) a formal definition of the ASP extension this architecture presents that highlights the fundamentally sequential nature of this task in robotics, (2) a solution concept using a combination of classification and belief-space planning, and (3) a detailed simulation and set of experiments showing the potential benefit (see Figure 1) of online decision making over SLAM algorithms and in particular the effectiveness of belief-space planning.

[1]University of Massachusetts, Amherst, MA, USA. Emails: {snashed, grupen, shlomo}@cs.umass.edu. Partial support for this work was provided by NSF grants 1954782 and 2205153.

II. Related Work

The "Algorithm Selection Problem" (ASP) was first outlined by Rice in 1976 [29], and has since been studied more closely in several sub-fields of computer science. In particular, combinatorial search [14], [11] and optimization [19], [13], [21] have been among the most prolific adopters, with the SAT solver SATzilla [36] likely one of the most successful applications of ASP methods to-date. Systems that solve ASPs vary substantially in how they operate. For example, they may select a single algorithm at the beginning of the problem [36], select a schedule of multiple algorithms to run sequentially [28], select a subset of algorithms to run in parallel [8], or monitor progress and revisit these decisions during the solve [1]. Because our goal is to choose between SLAM front ends in order to maintain highly accurate location estimates *indefinitely*, we focus on the online, or dynamic, version of this problem.

Many ASP methods employ machine learning techniques to either evaluate potential algorithm performance or select algorithms directly. This application has strong ties with meta-learning [33], [12]. There have also been proposals to use sequential decision making in the form of reinforcement learning (RL) for algorithm selection when the algorithms have a recursive nature [15], [27], [26]. In some cases meta-RL systems, such as RL^3 [3], may eventually allow a form of automatic adaptation to new ASPs. However, in this paper we focus on understanding the unique challenges of robotics ASPs, particularly for SLAM, and thus we are concerned more with formalizing the decision-making problem and developing a performant decision-making model.

Despite applications of ASP methods to other hard problems there have been few serious efforts to bring insights from ASP research into practice in robotics. The idea of using portfolios of models has been explored for localization [24], and the closely related problems of hyperparameter optimization and online hyperparameter tuning have been studied in the context of motion planning [20], [4]. However, engagement with formal ASP constructs has been, to the best of our knowledge, limited to the study of decentralized heuristic selection for coordination [30], and some computer vision tasks with relevance to robotic perception [17], [18]. In summary, this work offers the first formalisms, models, and solution methods that address the unique and fundamental challenges of applying the spirit of algorithm selection to SLAM and similar robotic perception problems. While not theoretically limited to perception systems, we see SLAM as a natural and important application of this formalism.

III. Algorithm Selection Problems for Robotics

Most applications of algorithm selection have been able to use essentially the original formalism from Rice with little modification. In this section we will briefly introduce this formalism, explain how robotics applications such as SLAM require more complex considerations to solve optimally due to their recursive nature and potential indefinite and reactive operation requirements, and then present an extended formalism that captures these aspects of the ASP for robotics.

A. The Algorithm Selection Problem

Formally, the ASP considers a set of algorithms $\mathcal{A}$, sometimes called a *portfolio*, and a problem instance x drawn from a some space of possible problems $\mathcal{P}$. Solving problem instance $x \in \mathcal{P}$ with algorithm $A \in \mathcal{A}$ results in a performance (often time or cost), which in our SLAM application we will call solution error $\varepsilon(A, x)$. The original ASP is thus to design or learn some selector function $S : \mathcal{P} \rightarrow \mathcal{A}$ such that $\forall x \in \mathcal{P}$, $S(x) = A^*$, where $A^* \in \text{argmin}_{A \in \mathcal{A}} \varepsilon(A, x)$.

In general, there may be other notions of maximizing performance, such as minimizing processor time or minimizing the cost of a plan for solving x generated by A. Additionally, in many cases it is not possible to guarantee $S(x) = A^* \forall x \in \mathcal{P}$, and in practice this depends on the quality of the features that can be derived from x in order to inform $S(x)$. Common extensions include selecting a set of algorithms $A \subset \mathcal{A}$ to run on x in parallel, i.e. $S(x) = \{A_1, A_4, A_{11}\}$, selecting a sequence or schedule of algorithms to run, i.e. $S(x) = \{A_1 : (t_0, t_4), A_4 : (t_4, t_{11}), A_{11} : (t_{11}, t_f)\}$, or dynamically adjusting the algorithm selected online, i.e. $S(x_t) = S(S(x_{t-1})(x_{t-1}))$.

B. Additional Robotics Considerations

There are four properties of SLAM systems that preclude some of the most common ASP approaches, as well as make the base objective a less accurate descriptor of success. First, most of the time, even after the algorithm has run, a SLAM system will not know the true value of $\varepsilon(A, x)$. This is in contrast to most other applications where, for example, the total processing time or the cost of the resultant plan is available. This makes dynamic selector functions, typically implemented as classifiers or regressors that rely specifically on fully observable features or feedback, less applicable. We could of course still apply such methods, but as we will see there are alternative frameworks that more naturally handle this partial observability constraint and do so while maintaining decision-theoretic optimality.

Second, all SLAM systems, whether Kalman filters, particle filters, or pose-graph optimizers, are *recursive*, where the state estimates produced at time t are used as inputs at time $t + 1$. Thus, achieving particular intermediate data values or representations is to some degree a function of the choice of solution strategy, which is true for many complex problems but not frequently modeled explicitly in ASP applications. As shown in Figure 2, instances of x encountered at different points in the SLAM problem are generally not independent. Moreover, most state-of-the-art SLAM systems use sliding window pose-graph optimization [32], meaning that optimization problems become generally more stable as time progresses [16] and that switching modalities essentially reduces the active optimization window size back to just the current time step. Thus, there is a cost to the stability, and therefore accuracy, associated with switching between sensing modalities online in the context of a SLAM system.

Third, SLAM systems can only react to changes in data quality or data quantity rather than preemptively act to affect sensing conditions. At each time step, at least some

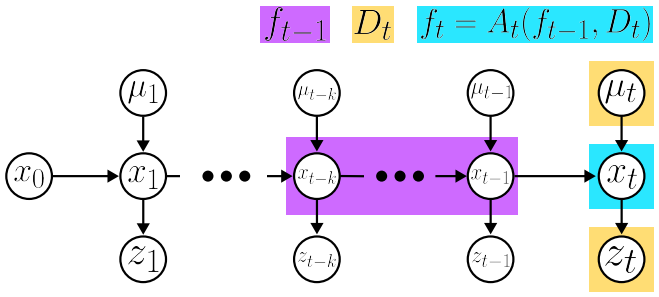

Fig. 2. A dynamic Bayesian network (DBN) representation of SLAM inference. As new data (D_t, yellow) from proprioception (μ_t) and exteroception (z_t) is observed in an uncontrolled process, it is used alongside one or more previous location estimates (f_{t-1}, purple), corresponding to the DBN nodes $\{x_{t-k}, \ldots, x_{t-1}\}$, to estimate the current pose x_t (more generally, f_t, blue). Here, μ_t and z_t represent geometric constraints on the transformation of the robot's position over time. In practice, these geometric constraints must be derived from raw data, and it is precisely this process which may produce large errors when sensing conditions diverge from expectations. While many SLAM systems use different optimization procedures, including loop-closures and other explicit references to a persistent map or other model of the world, we note that the recursive, reactive, and indefinite characteristics of the problem remain the case in all SLAM systems.

component of the data required for localization is provided in a manner beyond the control of the agent. For this reason, selecting a single algorithm at the outset is only robust if the deployment is very constrained. Moreover, robotic systems are often highly compute bound, and SLAM optimization is a notoriously computationally expensive process. While it is possible to run multiple SLAM front end feature extraction routines for various sensing modalities in parallel, it is not possible to run multiple full SLAM systems in parallel.

Last, indefinite simply indicates that total the size of the problem or number of steps is not known. As SLAM algorithms operate indefinitely while the robot is deployed, selecting a schedule of different algorithms to run during the deployment is not possible due to the unknown duration. We now formalize the robotics ASP, designed to capture the *recursive*, *reactive*, and *indefinite* properties specifically for SLAM and similar robotics problems. Selecting an algorithm to sort a list [15], for example, is recursive but neither reactive nor indefinite. Together, these properties create a more challenging problem requiring sequential reasoning.

We retain the notation for the portfolio of algorithms $\mathcal{A}$. Instead of a problem $x \in \mathcal{P}$, we must represent data processed in a sequence, part of which is recursive and influenced by the algorithm selected in the previous time step and part of which is generated independently by the environment and only available to the robot incrementally. We represent the latter data at time t as D_t, and the former as $f_t = A_t(f_{t-1}, D_t)$; see Figure 2 for more details. Both f_{t-1} and D_t affect the performance of A_t. We will denote the entire, unbounded sequence of data as $\tau = D_0, \ldots, D_\infty$, and let $\tau \in \mathcal{P}$. Of course, τ is not known at run time, but in some cases we may have domain knowledge that indicates some τ are more likely than others.

Because we care about the output quality at every intermediate time step in addition to the final time step, our objective is instead to minimize the sum $\sum_{t=t_0}^{t_\infty} \varepsilon(A_t, f_{t-1}, D_t)$. However, as the actual error is not fully observable, $\varepsilon(A_t, f_{t-1}, D_t)$ is instead a probability distribution and our objective is to minimize this sum in expectation: $\sum_{t=t_0}^{t_\infty} \mathbb{E}[\varepsilon(A_t, f_{t-1}, D_t)]$. Finally, we see that since $f_t = S(f_{t-1}, D_t)(f_{t-1}, D_t)$, S must represent a sequence of dependent decisions with stochastic outcomes, which are naturally expressed by the notion of a policy from sequential decision making. We now give a formal definition of this problem using the notation developed above.

Definition 1. *Given an unbounded data stream τ from the set of streams $\mathcal{P}$, the* Perception Algorithm Selection Problem *is to, at each time step, select using selector S an algorithm A_t from a portfolio of algorithms $\mathcal{A}$ such that the cumulative expected error $\sum_{t=t_0}^{t_\infty} \mathbb{E}[\varepsilon(A_t, f_{t-1}, D_t)]$ is minimized.*

This formulation generalizes several variants. For example, if we relax the partial obervability condition on the data quality or error, we no longer need to maintain belief over the error and thus minimize the object $\sum_{t=t_0}^{t_\infty} \varepsilon(A_t, f_{t-1}, D_t)$, which we can see is solvable using an MDP. If we further relax the recursive condition, we get an objective minimizing $\sum_{t=t_0}^{t_\infty} \varepsilon(A_t, D_t)$ which is solvable via repeated classification. Last, if we relax the reactive (streaming) data condition, we minimize $\sum_{t=t_0}^{t_f} \varepsilon(A_t, D_t)$, which could then be solved using existing schedule building techniques from the ASP literature since τ is known completely in advance.

IV. Choosing SLAM Algorithms Online

Given the key differences in robotics versions of the ASP compared to traditional combinatorial search applications, dynamic ASP methods that treat repeated algorithm selection independently and thus employ classification or regression techniques alone will not maximize the updated objective. In this section we will first describe a POMDP decision-making model we developed to solve this problem and cover some key design choices. We will then discuss a method for generating simulations of SLAM systems across a range of sensing conditions which we use to conduct our experiments.

A. Sequential Algorithm Selection as a POMDP

Below, we denote members of the POMDP model with an overbar to distinguish them from other variables. A *partially observable Markov decision process* (POMDP) [10] is a tuple $\langle \bar{S}, \bar{A}, \bar{T}, \bar{R}, \bar{\Omega}, \bar{O} \rangle$. $\bar{S}$ is a set of states of the world. $\bar{A}$ is a set of actions. The transition function $\bar{T} : \bar{S} \times \bar{A} \times \bar{S} \to [0, 1]$ gives the probability of ending up in state s' after performing action a in state s. The reward function $\bar{R} : \bar{S} \times \bar{A} \to \mathbb{R}$ maps each state s to an immediate reward. $\bar{\Omega}$ is a set of observations. The observation function $\bar{O} : \bar{S} \times \bar{A} \times \bar{\Omega} \to [0, 1]$ maps each state s' and action a to the probability of emitting observation $\omega \in \bar{\Omega}$. In a POMDP, the agent does not know the true state of the world and instead maintains a belief b, informed by noisy observations, over all states. Upon taking action a and observing ω, the agent updates its belief as $b'(s'|b, a, \omega) = \alpha \bar{O}(a, s', \omega) \sum_{s \in S} \bar{T}(s, a, s') b(s)$, where α is the normalization constant $\alpha = Pr(\omega|b, s)^{-1}$. We propose the following POMDP as a step towards decision-theoretic solutions to the ASP for robotics sub-systems.

$\bar{S}$: $D_l \times D_c \times D_k \times f \times A$, where A is the previous algorithm $\{\text{LASER}, \text{CAMERA}, \text{KINECT}\}$, f is the quality (conditioning, number of correct data associations) of the previous solve $\{1, \ldots 5\}$, and D_l, D_c, and D_k, all drawn from $\{1, \ldots, 3\}$ represent the partially observable data quality for each modality. Thus there are a total of 405 states.

$\bar{A}$: Our set of actions is the set of algorithms, $\mathcal{A}$.

$\bar{T}$: There is no uncertainty in algorithm execution. If algorithm A is selected, algorithm A is executed. However, there is uncertainty in the quality of the next data D_{t+1} with respect to each modality and uncertainty with respect to the quality of the output f_t with respect to subsequent solve attempts. We implement a small bias towards sensing conditions remaining the same or similar since on balance this is most likely, and we also encode a small chance of increasing the quality of f_{t+1}, provided the same algorithm is used and the current optimization problem maintained.

$\bar{R}$: Reward is the negative expected error of Equation (1), given α, β, and δ, which are functions of either D_l, D_c, or D_k, depending on the which algorithm is currently selected. We also include a mitigating factor of quality$(f_t)^{-\frac{1}{2}}$.

$\bar{\Omega}$: $D_l \times D_c \times D_k$, where D_l, D_c, and D_k represent the quality estimated by a classifier (e.g. a neural network).

$\bar{O}$: The observation function encodes the noise characteristics (roughly 80% accurate) of the classifiers evaluating the quality of each SLAM front end on the current data D_t.

This POMDP is too large to solve exactly, so we use an approximate technique based on value iteration, implemented in the pomdp_py library [38], which we represent compactly as a finite state controller [5]. There are many possible extensions to this model, including using more basic information, such as the residuals from the optimizer after each step, analyzing the density of the information matrix, or employing other, more complicated forms of error estimation or correction prediction [2]. Moreover, using value approximators, such as neural networks, in the context of reinforcement learning could help make this formulation tractable for higher dimensional variants, such as adding the ability to control external entities in some capacity [37].

Overall, the key idea is to exploit the fact that most SLAM back ends are modality agnostic, and thus provide substantial opportunity to intercede between raw data acquisition and factor graph construction. Intermediate data quality metrics may be flexible, so long as they can be estimated and correlate with solution quality in a manner learnable by, for example, supervised learning. Thus, we can avoid singular, monolithic front ends that in spite of their complexity are often still susceptible to individual sensor failures.

B. Modeling SLAM Systems

Empirical measures of SLAM system performance, characterized by the probability density function (PDF) of the magnitude of their location estimate errors, rarely coincide exactly with any known parametric distribution. Most commonly, such error distributions are modeled as half-normal distributions [35], or mixed distributions that include the half-normal distribution [6]. This approach is adequate

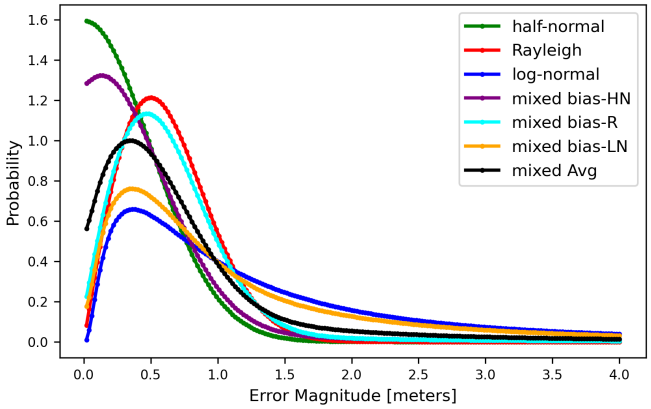

Fig. 3. Error distributions used in the simulator. Half-normal, Rayleigh, and log-normal distributions are shown in green, red, and blue respectively, while several mixed distributions (bias-HM, bias-R, bias-LN) correspond to $\alpha = 0.8$, $\beta = 0.8$, and $\delta = 0.8$ (others set to 0.1), respectively. Mixed Avg has $\alpha = \beta = \delta = 0.\bar{3}$.

although not perfect if the algorithms operate in the sensing regimes for which they were designed. However, as sensing conditions (localization affordances) change due to the passage of time or the motion of the robot in the world, these error distributions also shift in accordance with how well the given SLAM algorithm can accurately estimate state given the current quality of the sensing data. Since there are virtually no models for SLAM algorithm performance in *unintended* deployment conditions, we propose a mixed distribution composed of a variable linear combination of half-normal, log-normal, and Rayleigh distributions in order to model empirically observed error distributions more realistically. This also allows more control over the shape of the error PDF depending on the simulated sensing conditions. More formally, we simulate the error distribution of SLAM algorithm A_t on data D_t as

$$\begin{aligned} \mathrm{P}(\varepsilon | A_t, D_t) = & \alpha \left(\frac{\sqrt{2}}{\sigma_{\mathcal{N}} \sqrt{\pi}} e^{\frac{-\varepsilon^2}{2\sigma_{\mathcal{N}}^2}} \right) + \beta \left(\frac{\varepsilon}{\sigma_{\mathcal{R}}^2} e^{\frac{-\varepsilon^2}{2\sigma_{\mathcal{R}}^2}} \right) \\ & + \delta \left(\frac{1}{\varepsilon \sigma_{\mathcal{L}} \sqrt{2\pi}} e^{\frac{-\ln(\varepsilon)^2}{2\sigma_{\mathcal{L}}^2}} \right), \end{aligned} \tag{1}$$

where α, β, and δ are functions of A_t and D_t, and $\alpha + \beta + \delta = 1$. $\sigma_{\mathcal{N}}$, $\sigma_{\mathcal{R}}$, and $\sigma_{\mathcal{L}}$ are the standard deviation values for the half-normal, log-normal, and Rayleigh distributions, respectively. They may be tuned to provide even finer control although in our simulator and experiments we use constant values of $\sigma_{\mathcal{N}} = 0.5$, $\sigma_{\mathcal{R}} = 0.5$, and $\sigma_{\mathcal{L}} = 1$. Figure 3 shows several example error distributions.

In addition to sensing conditions we also consider the amount of data currently represented in the sliding window of the optimizer. Changing modalities means that recent features cannot be loop-closed against, for example because ORB [31] features from RGB images and FLIRT [34] features from lasers have no meaningful correspondence. Thus, there exists a trade off between switching front end algorithms immediately upon receiving bad data to avoid a bad location estimate, and maintaining a fully populated local map in order to be more robust to future bad inputs.

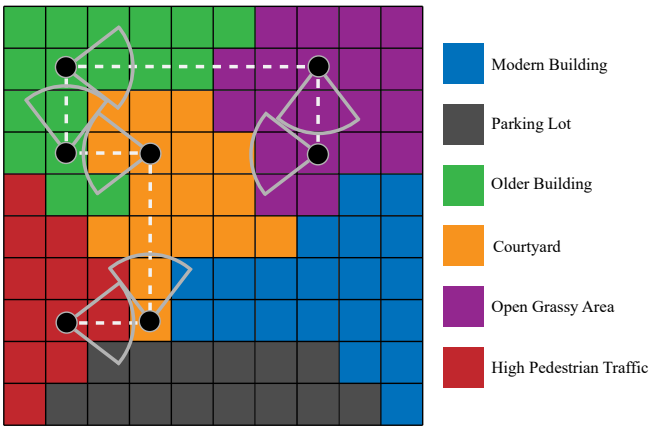

Fig. 4. Example path consisting of 7 total waypoints an agent may take in an environment with 6 sub-environments, each of which have potentially unique localization affordances. These affordances are affected by 6 parameters: Level of ambient light, amount of clutter, level of dynamics, amount of perceptual aliasing, amount of empty space, and natural versus artificial light. Each parameter can take two possible values for a total of 64 unique possible environments. In this example, the high pedestrian traffic area may have a high level of dynamics, the courtyard may have a large amount of open space and natural light, and the older building may have low ambient light and no natural light.

Though there has been work on optimizing the size of sliding windows in SLAM solvers [16], there has been less on characterizing the effect of window size on error distributions. The established wisdom is that window size offers diminishing returns, reducing errors by a factor of roughly $1/n$ after n repeated observations [32]. Thus, after drawing an error from the distribution in (1), we apply a slightly more conservative reduction of $1/\sqrt{n}$, up to $1/\sqrt{n_{max}}$, where $n_{max} = 5$ is the maximum sliding window size.

The agent moves continuously through the world between waypoints, shown in Figure 4, according to some maximum linear and angular velocities. It has a bounded field of view and range for each sensor, roughly in accordance with real-world parameters. The agent may view more than one type of environment if accessible to its sensors (it cannot see through walls), and multiple environments may simultaneously affect the classifier predictions regarding the localization affordances of the current location. For example, camera-based SLAM systems operate well in cluttered, well-lit environments, but struggle in highly aliased, high contrast, or very low light settings. If the agent is viewing two areas which each have these characteristics, the incoming data D_t may be classified differently than if it was viewing an obviously poor or obviously well-suited scene. We do not include the effects of error accumulation over time or the loop-closure detection process.

V. Results

The primary measure of efficacy for SLAM systems is localization error. In the following experiments, we measure the effect of different modality selection strategies on the average error magnitude, the distribution of errors, and the robustness of the system to sensor failures. We also show that certain strategies become relatively more effective as the space of operating environments becomes less homogeneous. In particular, we compare the following four strategies, each of which select from the same set of 3 sensors at each time-step: Random, Offline, Classify, and POMDP.

Random selects a sensor randomly with uniform probability. Offline first analyzes the entire map and estimates the quality of data from each sensor for each part of the map. The sensor with the overall highest average quality is then selected and used exclusively for the entire deployment. This method does not use information about the planned trajectory of the robot, which may visit some regions of the environment more often than others. Although this may seem like a relatively weak baseline, this is in fact essentially the current state-of-the-art approach, except that humans are the ones typically doing the pre-deployment evaluation of the environment and matching it with a SLAM front-end.

Classify runs an imperfect classifier at each time step, and selects the highest scoring modality to use that frame with probability 0.8 and a random other modality is chosen otherwise. This is equivalent to acting in a greedy manner exclusively on the POMDP observations. POMDP selects an algorithm based on a policy representing an approximate solution of a POMDP modeling the problem. The primary qualitative difference between Classify and POMDP is that the POMDP represents and reasons about the effects of its current algorithm selection on the *quality of future inputs*, whereas Classify does not.

For all experiments, data was collected by randomly generating an environment, establishing several waypoints for the robot to navigate to, and then simulating and recording the localization errors. For a given number of sub-environments (2-10), all methods were run a total of 10 times over the same 10 randomly generated maps.

A. Minimizing Cumulative Error

Unsurprisingly, the POMDP method accumulates the least localization error in simulation,[1] with an average per-pose error across all 90 trails of 0.27m and a standard deviation of 0.12m. Classify, Offline, and Random obtained averages of 0.41 ± 0.14, 0.44 ± 0.15, and 0.93 ± 0.10 meters, respectively. The differentiating factor seems to be approaches to transitions in sensing conditions. While the Classify method always selects the highest scoring method regardless of history, and is thus susceptible to noise, the belief dynamics within the POMDP act as a sort of low-pass filter on the noisy sensor quality observations, enabling the POMDP to continue using high-quality optimization problem initializations (f_{t-1}) when the lapse in reported signal quality is transient. In 81% of sequences where the robot moves from one modality to another, it took more than two consecutive observations where the current choice was ranked lower than the sensor it eventually selected.

[1]When interpreting the simulation results, we focus on the relative performance of different methods rather than their absolute performance, since the latter has little meaning in the isolation of simulation.

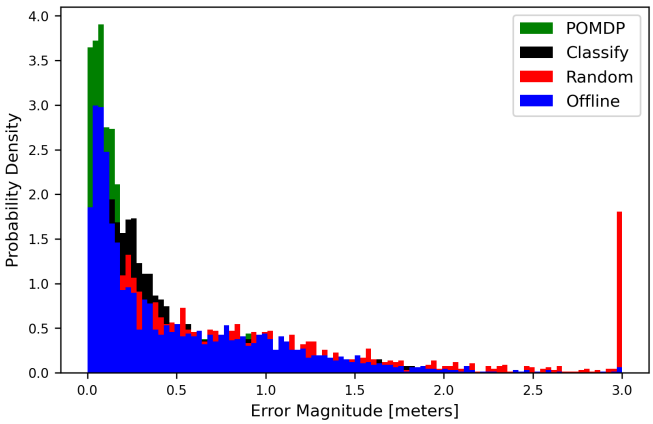

Fig. 5. Distribution of localization errors for all trails with a total of 4 sub-environment types. Other numbers of sub-environments show a similar trend, although their means shift slightly. Note that errors exceeding 3m were capped at 3m for the purposes of visualization.

B. Avoiding Catastrophic Failures

Perhaps the most important characteristic for deployed SLAM systems is to avoid large errors. Large errors can not only cause immediate problems in terms of trajectory following or route planning, but they can also cause difficulty when trying to re-localize or detect loop closures. Therefore, techniques that can reduce such catastrophic state estimation errors are employed frequently, and include a wide array of methods from simple thresholds to complex graph optimization. Here, we show the distribution of errors produced by each approach (Figure 5).

There are two key takeaways. First, the POMDP-based solution clearly enables the most robust data conditions for localization. Second, we see that the OFFLINE method has 2 modes. The first ($\mu \approx 0.1$) results from localization when it is operating in the environment for which its chosen sensor is well-suited, and the second ($\mu \approx 0.8$) occurs when it is operating in unfavorable sensing conditions. This set of events represents types of deployments roboticists may currently elect to avoid due to lack of generalizability.

C. Fault Tolerance

One key aspect of this system is that it can also deal with unexpected sensor failures. In fact, there is no need to model this event differently than, for example, entering a very dark area while previously using a camera to localize. As long as the meta-data or classifier for a given sensor can reliably detect an abnormality in sensor function, it can output an assessment of its quality as it would for any other frame. This quality is then used as an observation as it would be if the sensor was functioning nominally. Most powerfully, this allows systems to employ multi-modal SLAM systems and still have the opportunity to fall back on algorithms designed for a subset of modalities.

Table I shows the results of a set of experiments designed to test this capability. In these experiments, we artificially restrict the set of sensors available to the RANDOM, OFFLINE, and CLASSIFY methods, while leaving the POMDP policy unchanged. We then run simulations (10 trials with 5 different sub-environments) as before, but generate *noisy* observations that reflect the ground truth that one sensor is malfunctioning. Here, we can see that even without this prior information the policy is able to reason about the available options online and apply them at least as effectively as the other baseline methods, which were designed specifically leveraging this information.

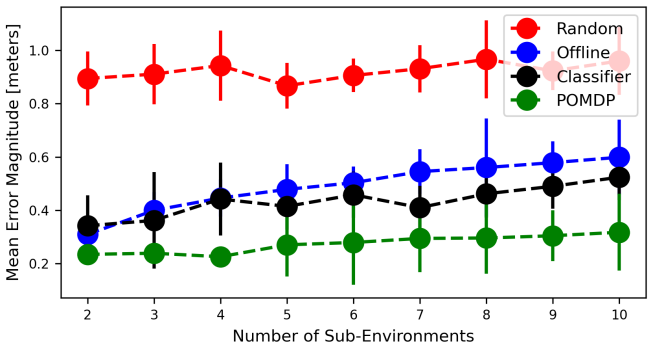

Fig. 6. Average localization error for all trials as a function the number of sub-environments in the map. Vertical bars represent one standard deviation.

TABLE I
ROBUSTNESS TO SENSOR FAILURE

Method	Working Sensors	Mean Error	Standard Deviation
RANDOM	2/3	0.87	0.13
OFFLINE	2/3	0.43	0.09
CLASSIFY	2/3	0.39	0.14
POMDP	2/3	0.32	0.11

D. Effect of Environment Heterogeneity

While the OFFLINE method works well in cases where we can predict pre-deployment how likely different sensing conditions will be during deployment, we can see that the more heterogeneous or unpredictable the operating environment becomes, the more challenging it is for non-reactive systems to perform well. Figure 6 shows a graph of average localization error across all trials as a function of the number of sub-environments present in the map. Large values on the x-axis indicate a less homogeneous operational environment.

Because OFFLINE selects one sensor, a uniform traversal of the map provides a lower bound on the rate of optimal sensor selection of just $|\mathcal{A}|^{-1}$. Moreover, if we consider arbitrary multimodal systems, this becomes roughly $2^{-|\mathcal{A}|}$. Therefore, we expect the performance of OFFLINE to decrease initially before converging as environment diversity increases.

VI. CONCLUSION

In this paper we outline a new type of algorithm selection problem for robotic perception, devise a solution to this problem using partially observable Markov decision processes, and detail several experiments showing the effectiveness of online decision making and sequential reasoning for such problems. We presented results from a detailed simulation of SLAM algorithms based on multiple modalities which show that SLAM systems the ability to modulate reliance on front-end data resources experience less localization error on average as well as significantly fewer catastrophic localization errors. Future work will include improving classifiers used to establish the reliability of incoming data and establishing these multi-SLAM systems on deployed robots.

REFERENCES

[1] A. Alejandro, H. Youssef, and S. Michele. Online heuristic selection in constraint programming, 2009. In *International Symposium on Combinatorial Search-2009*, 2009.

[2] Z. Alsayed, G. Bresson, A. Verroust-Blondet, and F. Nashashibi. 2d slam correction prediction in large scale urban environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5167–5174. IEEE, 2018.

[3] A. Bhatia, S. B. Nashed, and S. Zilberstein. RL ^{3}: Boosting meta reinforcement learning via RL inside RL ^{2}. *arXiv preprint arXiv:2306.15909*, 2023.

[4] A. Bhatia, J. Svegliato, S. B. Nashed, and S. Zilberstein. Tuning the hyperparameters of anytime planning: A metareasoning approach with deep reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pages 556–564, 2022.

[5] D. Braziunas. Pomdp solution methods. *University of Toronto*, 2003.

[6] J. Guo, P. V. Borges, C. Park, and A. Gawel. Local descriptor for robust place recognition using lidar intensity. *IEEE Robotics and Automation Letters*, 4(2):1470–1477, 2019.

[7] M. Holder, S. Hellwig, and H. Winner. Real-time pose graph slam based on radar. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1145–1151. IEEE, 2019.

[8] B. A. Huberman, R. M. Lukose, and T. Hogg. An economics approach to hard computational problems. *Science*, 275(5296):51–54, 1997.

[9] W. Jeong and K. M. Lee. Cv-slam: A new ceiling vision-based slam technique. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3195–3200. IEEE, 2005.

[10] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Journal of AI Research*, 1998.

[11] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45, 2019.

[12] I. Khan, X. Zhang, M. Rehman, and R. Ali. A literature survey and empirical study of meta-learning for classifier selection. *IEEE Access*, 8:10262–10281, 2020.

[13] A. Kostovska, A. Jankovic, D. Vermetten, J. de Nobel, H. Wang, T. Eftimov, and C. Doerr. Per-run algorithm selection with warm-starting using trajectory-based features. In *International Conference on Parallel Problem Solving from Nature*, pages 46–60. Springer, 2022.

[14] L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *Data mining and constraint programming: Foundations of a cross-disciplinary approach*, pages 149–190, 2016.

[15] M. G. Lagoudakis, M. L. Littman, et al. Algorithm selection using reinforcement learning. In *ICML*, pages 511–518, 2000.

[16] S. Lim, T.-k. Lee, S. Lee, S. An, and S.-y. Oh. Adaptive sliding window for hierarchical pose-graph-based slam. In *2012 12th International Conference on Control, Automation and Systems*, pages 2153–2158. IEEE, 2012.

[17] M. Lukac and M. Kameyama. Adaptive functional module selection using machine learning: Framework for intelligent robotics. In *SICE Annual Conference 2011*, pages 2480–2483. IEEE, 2011.

[18] M. Lukac, A. Zhurtanov, and A. Ospanova. High-level verification of multi-object segmentation. In *2016 International Conference on Information and Digital Technologies (IDT)*, pages 173–179. IEEE, 2016.

[19] L. Meunier, H. Rakotoarison, P. K. Wong, B. Roziere, J. Rapin, O. Teytaud, A. Moreau, and C. Doerr. Black-box optimization revisited: Improving algorithm selection wizards through massive benchmarking. *IEEE Transactions on Evolutionary Computation*, 26(3):490–500, 2021.

[20] M. Moll, C. Chamzas, Z. Kingston, and L. E. Kavraki. Hyperplan: A framework for motion planning algorithm selection and parameter optimization. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2511–2518. IEEE, 2021.

[21] D. Müller, M. G. Müller, D. Kress, and E. Pesch. An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning. *European Journal of Operational Research*, 302(3):874–891, 2022.

[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[23] S. Nashed and J. Biswas. Human-in-the-loop slam. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[24] S. B. Nashed, D. M. Ilstrup, and J. Biswas. Localization under topological uncertainty for lane identification of autonomous vehicles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6000–6005. IEEE, 2018.

[25] S. B. Nashed, J. J. Park, R. Webster, and J. W. Durham. Robust rank deficient slam. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6603–6608. IEEE, 2021.

[26] S. B. Nashed, J. Svegliato, A. Bhatia, S. Russell, and S. Zilberstein. Selecting the partial state abstractions of mdps: A metareasoning approach with deep reinforcement learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11665–11670. IEEE, 2022.

[27] S. B. Nashed, J. Svegliato, M. Brucato, C. Basich, R. Grupen, and S. Zilberstein. Solving markov decision processes with partial state abstractions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 813–819. IEEE, 2021.

[28] L. Pulina and A. Tacchella. A self-adaptive multi-engine solver for quantified boolean formulas. *Constraints*, 14:80–116, 2009.

[29] J. R. Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier, 1976.

[30] A. Rosenfeld, G. A. Kaminka, S. Kraus, and O. Shehory. A study of mechanisms for improving robotic group performance. *Artificial Intelligence*, 172(6-7):633–655, 2008.

[31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[32] G. Sibley, L. Matthies, and G. Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, 2010.

[33] K. A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)*, 41(1):1–25, 2009.

[34] G. D. Tipaldi, M. Braun, and K. O. Arras. Flirt: Interest regions for 2d range data with applications to robot navigation. In *Experimental Robotics: The 12th International Symposium on Experimental Robotics*, pages 695–710. Springer, 2014.

[35] P. Trybała. Lidar-based simultaneous localization and mapping in an underground mine in złoty stok, poland. In *IOP Conference Series. Earth and Environmental Science*, volume 942. IOP Publishing, 2021.

[36] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008.

[37] Z. Yang, H. Zhang, H. Zhang, B. Di, M. Dong, L. Yang, and L. Song. Metaslam: Wireless simultaneous localization and mapping using reconfigurable intelligent surfaces. *IEEE Transactions on Wireless Communications*, 22(4):2606–2620, 2022.

[38] K. Zheng and S. Tellex. pomdp_py: A framework to build and solve pomdp problems. In *ICAPS 2020 Workshop on Planning and Robotics (PlanRob)*, 2020.