

A Probabilistic Programming Idiom for Active Knowledge Search

Malte R. Damgaard

Rasmus Pedersen

Thomas Bak

Abstract—In this paper, we derive and implement a probabilistic programming idiom for the problem of acquiring new knowledge about an environment. The idiom is implemented utilizing a modern probabilistic programming language. We demonstrate the utility of this idiom by implementing an algorithm for the specific problem of active mapping and robot exploration. Finally, we evaluate the functionality of the implementation through an extensive simulation study utilizing the HouseExpo dataset.

I. INTRODUCTION

Making decisions under uncertainty to obtain new knowledge about an environment is a recurring problem within robotics. To efficiently solve this problem, the robot needs to continuously learn about its environment while keeping track of the uncertainty about current knowledge. The decision-making is further complicated if an extrinsic reward signal cannot guide the robot and if predefined constraints should be satisfied.

The most well-known and studied problem of this type within robotics is probably active mapping and robot exploration [1]. Most solutions to active mapping and robot exploration heavily exploit the structure of the stored knowledge, i.e., the map, to derive efficient algorithms. E.g. for grid map representations, it is common to apply frontier exploration [2]–[4]. These methods exploit the property, that it is possible to identify frontiers between the knowledge represented by grid cells in a grid map, that the robot is currently certain about, and the knowledge for which it is uncertain. Actions are chosen to guide the robots towards these frontiers, by which the map is explored. While such approaches exploiting problem-specific properties can result in efficient solutions, they do not easily generalize to other types of knowledge. E.g. because such exploration frontiers cannot easily be defined for other types of knowledge.

Other solutions to active mapping and robot exploration take a deep-learning approach, to learn an efficient policy for acquiring new knowledge. E.g., in [5] they feed the current knowledge, again in the form of a grid map, into an artificial neural network and let the output of the network control the actions of the robot. They then train the network with a reward equal to the newly discovered area at each time-step, by which they obtain a policy for exploration. While such an approach can be very efficient at specific tasks, end-to-end learning often limits the generalizability of the solution due to a lack of structural transferability. In many cases, the

All authors are with Department of Electronic Systems, Automation and Control, Aalborg University, Denmark, (e-mail: {mrd, rpe, tba}@es.aau.dk).

artificial neural network would have to be re-trained to work for other problems requiring other inputs and outputs.

Opposite to the problem-specific approaches already mentioned, the goal of cognitive architectures is to create computational entities with general problem-solving capabilities, that should function across a multitude of tasks. In recent years a community consensus about the overall structure and components of cognitive architectures has begun to emerge, called the Standard Model of the Mind [6]. Especially, the realization of the need for an efficient combination of symbolic and statistical processing is a massive change compared to early research in cognitive architectures. In [7] we presented a generalized framework for developing such cognitive architectures for robotics applications. This was done in an effort to standardize work and promote better cooperation. One of the main ideas of the framework is to develop and identify general and reusable fragments of probabilistic programs, i.e., probabilistic programming idioms, for which inference could be done efficiently utilizing variational inference methods.

Inspired by some of the main concepts of the Standard Model of the Mind, the goal of the presented efforts is to develop such a general and reusable probabilistic programming idiom for the problem of making decisions under uncertainty to obtain new knowledge about an environment. The main contributions of this paper are:

- 1) Derivation and implementation of the said probabilistic programming idiom,
- 2) and validation of the said idiom used in an active mapping and robot exploration context through simulations on a large dataset.

We choose to validate the idiom based on the active mapping and robot exploration problem because it is a well-studied problem with a relatively simple problem formulation for which results are easily interpretable via visual inspection of the robot’s trajectory. Still, the problem is sufficiently hard due to the non-convex constraints implied by objects in the environment.

Section II presents preliminaries necessary to understand the content of the following sections. In Section III the derivation of the probabilistic programming idiom is presented. In Section IV the application of the idiom for the active mapping and robot exploration problem is presented, together with the results of an extensive simulation study. Finally, in Section V we conclude upon the presented work, and hint to future lines of research.

II. PRELIMINARIES

Within this paper Z is used to denote latent variables, X is used to denote observed variables, and C is used to denote a collection of both types of variables. We use a superscript in curly brackets to indicate the index of a variable. Specially, for time indexes, we indicate the set of indexes of future variables as $\{t\}^+ = \{t + 1, \dots, t + \bar{T}\}$. Similarly, we indicate the set of indexes of past variables as $\{t\}^- = \{t - \underline{T}, \dots, t\}$. We develop our model primarily for approximate inference with stochastic variational inference. In general, variational inference refer to methods that approximates one conditional distribution, $p(z|x = \bar{x})$ with another unconditional distribution, $q(z)$, through an optimization problem of the form

$$\min_{q(z) \in Q} D[p(z|x = \bar{x})||q(z)]$$

where Q is the family of distributions from which q can be picked, and D is a divergence measure quantifying the difference between p and q . In stochastic variational inference, q is assumed to be parameterised by a set of parameters ϕ , and stochastic gradient ascent is used to solve a tractable dual-problem [8]. To solve this dual-problem, we do not need to know the conditional distribution, $p(z|x = \bar{x})$, but only need to specify the unconditional model, $p(z, x = \bar{x})$, making it a lot easier to work with. However, to use stochastic variational inference we need to ensure that our unconditional model, $p(z, x = \bar{x})$, preserves the differentiability of the dual-problem. Within this paper, we will make use of divergence measures from the family of f-divergences, defined by

$$\begin{aligned} D_f[p(z)||q(z)] &= \int_z q(z) f\left(\frac{p(z)}{q(z)}\right) \\ &= E_{q(z)}\left[f\left(\frac{p(z)}{q(z)}\right)\right] \end{aligned}$$

where f is an arbitrary convex function [9]. Based on f-divergence we can define the f-information measure as

$$\begin{aligned} I_f[z, y] &= D_f[p(z)p(y)||p(z, y)] \\ &= E_{p(y)}[D_f[p(z)||p(z|y)]]. \end{aligned}$$

The commonly used KL-divergence, D_{KL} , and mutual information is defined by $f(u) = -\log(u)$ such that

$$D_{KL}[p(z)||q(z)] = E_{q(z)}[\log(q(z)) - \log(p(z))]$$

Similarly, the inverse-KL-divergence, $D_{\overline{KL}}$, and Lautum information, I_L , is defined by $f(u) = u \cdot \log(u)$, from which we can obtain the conditional Lautum information measure

$$\begin{aligned} I_L[z, y|x] &= E_{p(y|x)}[D_{\overline{KL}}[p(z|x)||p(z|y, x)]] \quad (1) \\ &= E_{p(y|x)}\left[\log(E_{p(z|x)}[p(y|z, x)]) - E_{p(z|x)}[\log(p(y|z, x))]\right] \quad (2) \end{aligned}$$

For more information about these measures and their properties, we refer the reader to [9], [10]. Within this paper, we

will also be using the following approximate "probabilistic logic"

$$\begin{aligned} p(z \in \bar{z} \vee y \in \bar{y}) &\stackrel{\text{def}}{=} \\ p(z \in \bar{z}) + p(y \in \bar{y}) - p(z \in \bar{z}) \cdot p(y \in \bar{y}) \\ p(z \in \bar{z} \wedge y \in \bar{y}) &\stackrel{\text{def}}{=} p(z \in \bar{z}) \cdot p(y \in \bar{y}) \\ p\left(\bigwedge_{i=1}^I z^{\{i\}} \in \bar{z}^{\{i\}}\right) &\stackrel{\text{def}}{=} \prod_{i=1}^I p(z^{\{i\}} \in \bar{z}^{\{i\}}) \end{aligned}$$

where we have used \vee and \wedge to denote the approximate *or* and the *and* operation, respectively. These approximate "probabilistic logic" rules simply constitute a probabilistic union and intersection with an implied independence assumption, respectively.

III. DECISION MODEL

According to [6] it is commonly agreed that the memory structure of mind like architectures at a top-level conceptually can be divided into *working memory* and *long-term memory* each of which constitutes relations over symbols supplemented by quantitative metadata to provide a hybrid symbolic-subsymbolic representation. Besides the two main types of memory, it is also agreed that there exists an architectural component denoted *perception* for converting external signals into appropriate memory representations. Similarly, there exists an architectural component denoted *motor* for translating internal memory representations into external signals. The relations between each of the aforementioned are illustrated in Fig. 1a. *Long-term memory* is responsible for the storage of information over extended periods. The *working memory* includes temporary information necessary for behavior production and problem-solving,

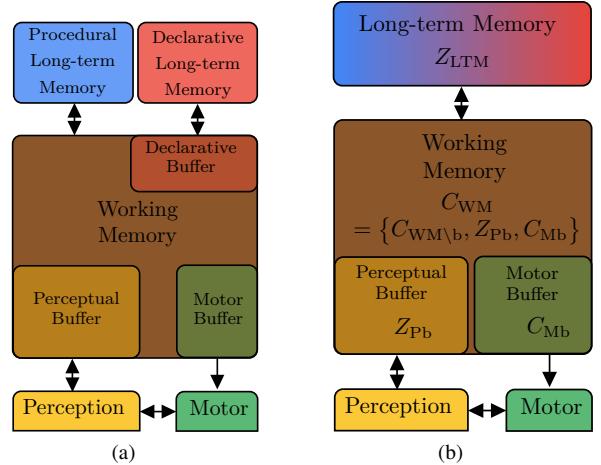


Fig. 1. (a) The conceptual memory structure of the Standard Model [6]. The red, blue, brown, green, and yellow colour are related to declarative long-term memory, procedural long-term memory, working memory, perception component, and motor component, respectively. Here we have used rectangles with rounded corners to symbolise pure memory, and sharp corners to indicate a relation to external signals. (b) The conceptual memory structure used within this paper is without the distinction between procedural and declarative long-term memory, and without the *declarative buffer*. The figure also indicates the symbols used for each type of memory.

such as information about goals, but also contains different buffers for temporarily storing information from the *perception* component, the *motor* component, and some types of long-term memory. As such *working memory* acts as a linkage between the other components. It is customary to sub-divide long-term memory further into specialized types of memories. However, since we in this paper are focusing on decision making to acquire new knowledge, that is updating all types of long-term memory, we will not make such distinctions, as illustrated in Fig. 1b. Neither will we make a distinction between the declarative buffer and general long-term memory, and jointly refer to them as long-term memory. Furthermore, to keep our presentation relatively concise, we will not consider the relation between working memory, and the *perception* and *motor* components. Instead we will assume that appropriate *perception* and *motor* components are present.

To accommodate the need for a hybrid symbolic-subsymbolic representation as suggested by the standard model, we will derive a probabilistic model of decision making. Based on the division of memory and the symbol definitions indicated in Fig. 1b we make the following definition of the joint probability distribution

$$\begin{aligned} p(C_{WM\backslash b}, C_{Mb}, Z_{Pb}, Z_{LTM}) \\ = p(Z_{WM\backslash b}^{\{t\}-}, C_{WM\backslash b}^{\{t\}+}, X_{Mb}^{\{t-1\}-}, Z_{Mb}^{\{t-1\}+}, Z_{Pb}^{\{t\}-}, Z_{Pb}^{\{t\}+}, Z_{LTM}) \\ \stackrel{\text{def}}{=} \underbrace{p(C_{WM\backslash b}^{\{t\}+}, Z_{Mb}^{\{t-1\}+}, Z_{Pb}^{\{t\}+} | \check{Z}_{WM\backslash b}^{\{t\}-}, \check{Z}_{LTM})}_{\text{Planning/Decision Making}} \\ \cdot \underbrace{p(Z_{WM\backslash b}^{\{t\}-}, X_{Mb}^{\{t-1\}-}, Z_{Pb}^{\{t\}-}, Z_{LTM})}_{\text{Learning/Reasoning}} \end{aligned} \quad (3)$$

where we have used sub-script "WM\backslash b" to denote the set of variables representing the working memory except of the set of variables representing the two buffers, i.e., C_{Mb} and Z_{Pb} . In Eq. (3) we have assumed that the distribution of future variables, $C_{WM\backslash b}^{\{t\}+}$, $Z_{Mb}^{\{t-1\}+}$, and $Z_{Pb}^{\{t\}+}$ are conditional independent of previous information in the perceptual buffer, $Z_{Pb}^{\{t\}-}$, and motor buffer, $X_{Mb}^{\{t-1\}-}$, given the previous variables in the rest of the working memory, $Z_{WM\backslash b}^{\{t\}-}$, and the long-term memory, Z_{LTM} . The last fraction of Eq. (3) deals with inference over variables internal to an agent based on past experience in the form of the variables of the perceptual buffer, $Z_{Pb}^{\{t\}-}$, and the motor buffer, $Z_{Mb}^{\{t-1\}-}$, related to the past. As such this fraction corresponds to reasoning and *learning*. Similarly, the first factor of Eq. (3) only deals with future variables based on what have already been learned from past experiences. Since it is assumed that the working memory includes information necessary for behavior production this fraction is responsible for decision making and *planning* guided by preferences contained in the working memory. By the nature of the problem, the probabilistic causation between *learning* and *planning* should only be one way, from *learning* to *planning*. In other words, we can consider inference over the variables in the *learning*

part in isolation, and when performing inference in the *planning* part we should keep the *learning* distribution fixed. To emphasize this, we have used breves over the variables $Z_{WM\backslash b}^{\{t\}-}$ and Z_{LTM} in the first fraction of Eq. (3). The proposed model effectively divides the cognitive tasks of an agent into *learning* and *planning*. Assuming that we have access to the *learning* distribution, this allows us to focus the rest of the paper on the *planning* part.

For the purpose of decision making, and to make our model resemble the classical Markov decision process, we introduce the following variables as a part of the working memory. State variables, Z_s , representing the state of the agent itself and the environment. Decision variables, C_D , explicitly represent preferences such as goals and constraints. That is, $Z_s^{\{t\}-} \in C_{WM\backslash b}^{\{t\}-}$ and $\{Z_s^{\{t\}+}, C_D^{\{t\}+}\} \in C_{WM\backslash b}^{\{t\}+}$. Furthermore, adopting the Markov property between state variables also used in the Markov decision process we define the planning distribution from Eq. (3) as

$$\begin{aligned} p(C_{WM\backslash b}^{\{t\}+}, Z_{Mb}^{\{t-1\}+}, Z_{Pb}^{\{t\}+} | \check{Z}_{WM\backslash b}^{\{t\}-}, \check{Z}_{LTM}) \\ \stackrel{\text{def}}{=} \prod_{\tau=t+2}^{t+\bar{T}} \left[\begin{array}{l} p(C_D^{\{\tau\}} | Z_s^{\{\tau\}}, \check{Z}_{WM\backslash b}^{\{\tau\}-}, \check{Z}_{LTM}) \\ \cdot p(Z_s^{\{\tau\}} | Z_s^{\{\tau-1\}}, Z_{Mb}^{\{\tau-1\}}) p(Z_{Mb}^{\{\tau-1\}}) \\ \cdot p(C_D^{\{t+1\}} | Z_s^{\{t+1\}}, \check{Z}_{WM\backslash b}^{\{t\}-}, \check{Z}_{LTM}) \\ \cdot p(Z_s^{\{t+1\}} | \check{Z}_s^{\{t\}}, Z_{Mb}^{\{t\}}) p(Z_{Mb}^{\{t\}}) \end{array} \right] \end{aligned} \quad (4)$$

The causality structure of Eq. (4) goes as follows. The current possible content of the motor buffer, $Z_{Mb}^{\{\tau\}}$, together with the belief over the state at that time instance, $Z_s^{\{\tau\}}$, determines the belief over the next possible states, $Z_s^{\{\tau+1\}}$. The next possible states, $Z_s^{\{\tau+1\}}$, together with the variables in the long-term memory, Z_{LTM} , and all variables related to the past in the working memory except the buffers, $Z_{WM\backslash b}^{\{t\}-}$, potentially contributes to the current belief over the decision variables, $Z_D^{\{\tau\}}$. Except for the decision variables and the explicit inclusion of the long-term memory variables, most parts of Eq. (4) resembles elements known from other decision models such as the Partially observable Markov decision process. As stated earlier, the decision variables are meant to guide the decision process, and as such might be problem-dependent.

For the purpose of making decisions in order to obtain new knowledge, and inspired by [11] we chose to include and combine the following general purpose decision variables: progress, z_p , information gain, z_i , constraint, z_c , and attention, x_A . From these we define $C_D^{\{\tau\}} = \{x_A^{\{\tau\}}, z_p^{\{\tau\}}, z_i^{\{\tau\}}, z_c^{\{\tau\}}\}$. The meaning of these variables are described in the following sections. For reference the structure of the combined model is indicated in Fig. 2.

A. Progress

The progress variable is meant to quantify how different a given state, $Z_s^{\{\tau\}}$, is from the past states, $Z_s^{\{\tau\}-}$. To quantify the progress while taking uncertainty into account we can make use of the divergence measures described in Section II.

However, calculating such divergence measures inside a probabilistic program amounts to a form of nested inference which potentially can cause problems. E.g. when we want to use stochastic variational inference as the main inference algorithm we have to make sure that we can calculate the gradient of the nested inference performed. Here we choose to use the following one-point estimate of the KL-divergence as a measure of progress

$$\begin{aligned}
D_{\text{KL}} & \left[p(Z_s^{\{t-l\}}) \| p(Z_s^{\{\tau\}} | Z_s^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}}) \right] \\
& = E_{\hat{Z}_s^{\{\tau\}}} \left[\log \left(\frac{p(Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}} | Z_s^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}})}{p(Z_s^{\{t-l\}} = \hat{Z}_s^{\{\tau\}})} \right) \right] \\
& \approx \frac{1}{I} \sum_{i=1}^I \left(\frac{\log(p(Z_s^{\{t-l\}} = \hat{Z}_s^{\{\tau\}, \{i\}}))}{-\log(p(Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}, \{i\}} | Z_s^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}}))} \right) \\
& \approx \text{ReLU} \left(\frac{\log(p(Z_s^{\{t-l\}} = \hat{Z}_s^{\{\tau\}}))}{-\log(p(Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}} | Z_s^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}}))} \right) \\
& \stackrel{\text{def}}{=} P^{\{t-l\}}(\hat{Z}_s^{\{\tau\}}) \tag{5}
\end{aligned}$$

Where $p(Z_s^{\{t-l\}})$ is a marginal of the learning distribution in Eq. (3), $\hat{Z}_s^{\{\tau\}} \sim p(Z_s^{\{\tau\}} | Z_s^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}})$ and $\hat{Z}_s^{\{\tau\}, \{j\}} \sim p(Z_s^{\{\tau\}} | Z_s^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}})$, and we have used the ReLu function in our approximation since $\log\left(\frac{p_1}{p_2}\right) \geq 0$ in general but $D_{\text{KL}}[p_1 || p_2] \geq 0$. The gradient of the log-probability function can be calculated for many commonly used distributions and probabilistic programs composed of these, and thereby also for this approximation. From this approximation we define the distribution over the progress variable for a given state, $Z_s^{\{\tau\}}$, relative to a single of the past states, $Z_s^{\{t-l\}}$, as

$$\begin{aligned}
p(z_p^{\{\tau\}, \{l\}} | Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}) &= \tag{6} \\
\text{Bernoulli} \left(\lambda_p^{\{l\}} \cdot [1 - e^{-\sigma_p \cdot P^{\{t-l\}}(\hat{Z}_s^{\{\tau\}})}] \right)
\end{aligned}$$

where

$$\lambda_p^{\{l\}} = 1 - \frac{(1 - \lambda_{p,\min})(L - 1 - l)}{L - 1} ; L > 1, \quad ; L \leq \underline{T},$$

is a decay variable used to put more emphasis on the oldest states considered, L is the number of old states considered, and σ_p is simply a scaling parameter. Here we have used a trick commonly utilised in probabilistic Reinforcement Learning, and Control [12], where a given reward is converted to a pseudo probability by exponentiation of that reward. Since, $P^{\{t-l\}}(\hat{Z}_s^{\{\tau\}}) \geq 0$ it follows that $e^{-\sigma_p \cdot P^{\{t-l\}}(\hat{Z}_s^{\{\tau\}})} \in [0, 1]$ and thus it can be used as a pseudo probability. Eq. (6) thus state that a state, $Z_s^{\{\tau\}}$, yielding a higher approximated divergence, $P^{\{t-l\}}(\hat{Z}_s^{\{\tau\}})$, has an exponentially higher probability of yielding progress. From Eq. (6) we define the total progress as the combined progress

relative to all of the last $L \leq t - \underline{T}$ past states

$$\begin{aligned}
p(z_p^{\{\tau\}} | Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}) &= \tag{7} \\
\text{Bernoulli} \left(p \left(\bigwedge_{l=0}^{L-1} z_p^{\{\tau\}, \{l\}} = 1 \mid Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}} \right) \right)
\end{aligned}$$

The approximation in Eq. (5) might seem very coarse; however, when used as nested inference inside a stochastic variational inference algorithm, it is evaluated multiple times during inference of the main problem. The effect is thus effectively similar to a mean approximation using many samples.

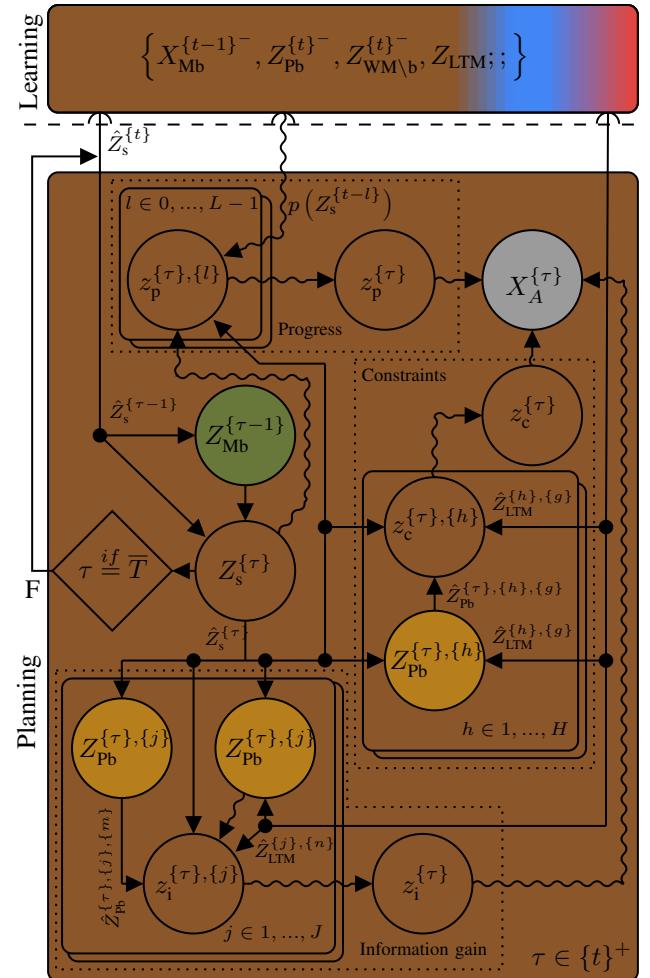


Fig. 2. Illustration of the generative flow of the proposed exploration idiom. Rectangles with rounded corners represent a collection of variables. Two stacked rectangles with rounded corners represent conditional independent collections of variables. Circles indicate a distribution over the variable inside the circle. They thereby constitute potential samples sites in the probabilistic program. Solid arrows indicate samples passed around in the probabilistic program, sampled from the distribution represented by the circle at the origin of the arrow. Wavy arrows indicate an evaluation at a specific point of the parent distribution represented by the circle at the origin of the arrow. Gray-colored circles indicate "observed" variables, while other colors are used to indicate a relation to the different types of memories. The rectangles with dotted borders indicate the variables associated with the three different decision variables.

B. Information Gain

As the name implies, the information gain variable, z_i , is meant to quantify the amount of information that can potentially be gained from being in a specific state, $Z_s^{\{\tau\}}$, perceiving the environment and thereby obtain new information through the perceptual buffer. The perceptual buffer might contain information from multiple independent perceptual modalities, which we will denote as $Z_{\text{Pb}}^{\{\tau\},\{j\}}$. Each of these perceptual modalities might only relate to a specific part of the long-term memory which we will denote $Z_{\text{LTM}}^{\{j\}}$. To quantify the expected amount of information obtained by being in a specific state, $Z_s^{\{\tau\}}$, we use the Lautum information in Eq. (1). Based on the Lautum information we represent the pseudo probability that the perceptual modality, $Z_{\text{Pb}}^{\{\tau\},\{j\}}$, will yield new knowledge as the distribution

$$p(z_i^{\{\tau\},\{j\}} | Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}) = \text{Bernoulli} \left(1 - e^{-\sigma_I \cdot I_L[Z_{\text{LTM}}^{\{j\}}, Z_{\text{Pb}}^{\{\tau\},\{j\}} | Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}]} \right) \quad (8)$$

where σ_I is a scaling parameter. Maximising the information obtained by each of the perceptual modalities might require wildly different changes to the state, $Z_s^{\{\tau\}}$. Therefore, we focus the attention on the modality providing the most information and define

$$p(z_i^{\{\tau\}} | Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}) = \text{Bernoulli} \left(\max_{j \in [1, J]} p(z_i^{\{\tau\},\{j\}} = 1 | Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}) \right).$$

Calculating the Lautum information inside a probabilistic program also amounts to nested inference. To make the calculation of Lautum information compatible with the use of stochastic variational inference for the main problem, we again make use of the following sample mean estimate

$$\begin{aligned} I_L[x, y | z = \hat{z}] &= E_{\hat{y}} \left[\log(E_{\hat{x}}[p(y = \hat{y} | x = \hat{x}, z = \hat{z})]) - E_{\hat{x}}[\log(p(y = \hat{y} | x = \hat{x}, z = \hat{z}))] \right] \\ &\approx \frac{1}{M} \sum_{m=1}^M \left[\log \left(\frac{1}{N} \sum_{n=1}^N p(y = \hat{y}^{\{m\}} | x = \hat{x}, z = \hat{z}) \right) - \frac{1}{N} \sum_{n=1}^N \log(p(y = \hat{y}^{\{m\}} | x = \hat{x}^{\{n\}}, z = \hat{z})) \right] \\ &= \frac{1}{M} \sum_{m=1}^M \left[\log \left(\sum_{n=1}^N e^{\log(p(y = \hat{y}^{\{m\}} | x = \hat{x}^{\{n\}}, z = \hat{z}))} \right) - \log(N) - \frac{1}{N} \sum_{n=1}^N \log(p(y = \hat{y}^{\{m\}} | x = \hat{x}^{\{n\}}, z = \hat{z})) \right] \end{aligned}$$

where $\hat{x} \sim p(x | z = \hat{z})$ and $\hat{y} \sim p(y | z = \hat{z})$, $\hat{y}^{\{m\}} \sim p(y | z = \hat{z})$ and $\hat{x}^{\{n\}} \sim p(x | z = \hat{z})$. To use the approximation in Eq. (8), we only need to be able to evaluate

$$\log \left(p \left(Z_{\text{Pb}}^{\{\tau\},\{j\}} = \hat{Z}_{\text{Pb}}^{\{\tau\},\{j\},\{m\}} \middle| \begin{array}{l} Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}, \\ Z_{\text{LTM}}^{\{j\}} = \hat{Z}_{\text{LTM}}^{\{j\},\{n\}} \end{array} \right) \right)$$

with

$$\begin{aligned} \hat{Z}_{\text{LTM}}^{\{j\},\{n\}} &\sim p(Z_{\text{LTM}}^{\{j\}}) \\ \hat{Z}_{\text{Pb}}^{\{\tau\},\{j\},\{m\}} &\sim p(Z_{\text{Pb}}^{\{\tau\},\{j\}} | Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}) \end{aligned}$$

where we have assumed that $p(Z_{\text{LTM}}^{\{j\}} | Z_s^{\{\tau\}}) = p(Z_{\text{LTM}}^{\{j\}})$.

C. Constraints

The constraint variable, $z_c^{\{\tau\}}$, is meant to quantify states, $Z_s^{\{\tau\}}$, that should be avoided taking perceived information, $Z_{\text{Pb}}^{\{\tau\}}$, and knowledge stored in long-term memory, Z_{LTM} , into account. Often such constraints can be defined by a set, $A^{\{\tau\},\{h\}}$, that the state, $Z_s^{\{\tau\}}$, should be within. As this set might depend on knowledge stored in the long-term memory, Z_{LTM} , and the expected content of the perceptual buffer, $Z_{\text{Pb}}^{\{\tau\}}$, we assume a set definition of the form

$$A^{\{\tau\},\{h\}} = \{Z_s^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}} | \mathbb{1}_A^{\{\tau\},\{h\}}(Z_s^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}})\}$$

where $\mathbb{1}_A^{\{\tau\},\{h\}}$ is the indicator function of the set $A^{\{\tau\},\{h\}}$. Given that $Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}$ the probability that the constraint defined by the set $A^{\{\tau\},\{h\}}$ is satisfied can then be expressed as

$$\begin{aligned} p(Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}} \in A^{\{\tau\},\{h\}}) \\ = E_{\hat{Z}_{\text{Pb}}^{\{\tau\}}, \hat{Z}_{\text{LTM}}} [\mathbb{1}_A^{\{\tau\},\{h\}}(\hat{Z}_s^{\{\tau\}}, \hat{Z}_{\text{Pb}}^{\{\tau\}}, \hat{Z}_{\text{LTM}})] \end{aligned} \quad (9)$$

where $\hat{Z}_{\text{Pb}}^{\{\tau\}} \sim p(Z_{\text{Pb}}^{\{\tau\}} | \hat{Z}_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}, \hat{Z}_{\text{LTM}} = \hat{Z}_{\text{LTM}})$ and $\hat{Z}_{\text{LTM}} \sim p(Z_{\text{LTM}})$. Based on this we define distribution over the constraint variable for the h 'th constraint at time τ as

$$\begin{aligned} p(z_c^{\{\tau\},\{h\}} | Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}) \\ = \text{Bernoulli} \left(P(Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}} \in A^{\{\tau\},\{h\}}) \right) \end{aligned}$$

and distribution over the combined constraint variable at time τ as

$$\begin{aligned} p(z_c^{\{\tau\}} | Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}) \\ = \text{Bernoulli} \left(P \left(\bigwedge_{h=1}^H z_c^{\{\tau\},\{h\}} = 1 \middle| Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}} \right) \right) \end{aligned}$$

Where H is the number of constraints. Calculating the probability in Eq. (9) also amounts to nested inference, but the discontinuity of the indicator function for the set definition, $\mathbb{1}_A^{\{\tau\},\{h\}}$, also present a problem for calculating the gradients needed for stochastic variational inference. To overcome this, we assume that the indicator function can be specified as

$$\begin{aligned} \mathbb{1}_A^{\{\tau\},\{h\}}(Z_s^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}}) \\ = \begin{cases} 1 & d^{\{\tau\},\{h\}}(Z_s^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}}) > 0 \\ 0 & \text{else} \end{cases} \end{aligned}$$

and make the approximation

$$\mathbb{1}_A^{\{\tau\},\{h\}}(Z_s^{\{\tau\}}) \approx \tilde{\mathbb{1}}_A^{\{\tau\},\{h\}}(d^{\{\tau\},\{h\}}(Z_s^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}}))$$

where $\tilde{\mathbb{1}}_A^{\{\tau\},\{h\}}(x) \in [0, 1]$ is a smooth monotonically increasing function symmetric around $\tilde{\mathbb{1}}_A^{\{\tau\},\{h\}}(0) = 0.5$, e.g., a scaled sigmoid function. From this we make use the sample

mean approximation to obtain the following approximation to the probability in Eq. (9)

$$\begin{aligned} P(Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}} \in A^{\{\tau\}, \{h\}}) \\ \approx \frac{1}{G} \sum_{g=1}^G \mathbb{1}_A^{\{\tau\}, \{h\}} (\hat{Z}_s^{\{\tau\}}, \hat{Z}_{\text{Pb}}^{\{\tau\}, \{g\}}, \hat{Z}_{\text{LTM}}^{\{g\}}) \\ \approx \frac{1}{G} \sum_{g=1}^G \tilde{\mathbb{1}}_A^{\{\tau\}, \{h\}} (d^{\{\tau\}, \{h\}} (\hat{Z}_s^{\{\tau\}}, \hat{Z}_{\text{Pb}}^{\{\tau\}, \{g\}}, \hat{Z}_{\text{LTM}}^{\{g\}})) \end{aligned}$$

where $\hat{Z}_{\text{Pb}}^{\{\tau\}, \{g\}} \sim p(Z_{\text{Pb}}^{\{\tau\}} | \hat{Z}_s^{\{\tau\}}, \hat{Z}_{\text{LTM}})$ and $\hat{Z}_{\text{LTM}}^{\{g\}} \sim p(Z_{\text{LTM}})$.

D. Attention

Finally, the attention variable is meant to summarise the other decision variables and symbolises which states the agent should focus its attention on. Based on the approximate "probabilistic logic" presented in Section II we define.

$$\begin{aligned} p(x_A^{\tau} | Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}}) = & \quad (10) \\ \text{Bernoulli} \left(p \left(\begin{array}{c} \left[z_p^{\{\tau\}} = 1 \right] \\ \vee z_i^{\{\tau\}} = 1 \\ \wedge z_c^{\{\tau\}} = 1 \end{array} \middle| Z_s^{\{\tau\}} = \hat{Z}_s^{\{\tau\}} \right) \right) \end{aligned}$$

Basically, Eq. (10) states that an agent should focus its attention on states that either yields progress, or yield new knowledge, and also satisfies the given constraints.

E. Variational distribution

As stated in Section II, a parameterized unconditional variational distribution, $q(Z_{\text{WM}}^{\{t\}+})$, needs to be specified to utilize stochastic variational inference for approximate inference. Most of the factors in Eq. (4) are assumed to be known and thus fixed. Therefore, only the distribution over the variables in the motor buffer can be considered a free distribution, and thus we define

$$\begin{aligned} q(Z_{\text{WM}}^{\{t\}+}) &= q(Z_{\text{WM}\setminus\text{b}}^{\{t\}+}, Z_{\text{Mb}}^{\{t-1\}+}) \\ &\stackrel{\text{def}}{=} p(Z_s^{\{t+1\}} | \check{Z}_s^{\{t\}}, Z_{\text{Mb}}^{\{t\}}) q_{\phi_{\text{Mb}}^{\{\tau-1\}}} (Z_{\text{Mb}}^{\{t\}} | Z_s^{\{t\}}) \\ &\cdot \prod_{\tau=t+2}^{t+\bar{T}} \left[p(Z_s^{\{\tau\}} | Z_s^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}}) \cdot q_{\phi_{\text{Mb}}^{\{\tau-1\}}} (Z_{\text{Mb}}^{\{\tau-1\}} | Z_s^{\{\tau-1\}}) \right] \end{aligned}$$

where $\phi_{\text{Mb}}^{\{\tau\}}$ are the parameters that need to be found by stochastic variational inference.

F. Summary

So far, general functionality that could potentially be utilised for multiple problems has been described and thus could be considered an idiom. This idiom is implemented as an abstract class utilising the probabilistic programming language Pyro [13] developed on top of PyTorch and python. The class contains the following abstract methods that need to be implemented

$q_{\phi_{\text{Mb}}^{\{\tau-1\}}} (Z_{\text{Mb}}^{\{\tau-1\}} Z_s^{\{\tau-1\}})$	$p(Z_{\text{Pb}}^{\{\tau\}, \{j\}} Z_s^{\{\tau\}}, Z_{\text{LTM}}^{\{j\}})$
$p(Z_{\text{Mb}}^{\{\tau-1\}} Z_s^{\{\tau-1\}})$	$d^{\{\tau\}, \{h\}} (Z_s^{\{\tau\}}, Z_{\text{Pb}}^{\{\tau\}}, Z_{\text{LTM}})$
$p(Z_s^{\{\tau\}} Z_s^{\{\tau-1\}}, Z_{\text{Mb}}^{\{\tau-1\}})$	$\tilde{\mathbb{1}}_A^{\{\tau\}, \{h\}} (d)$
$p(Z_{\text{Pb}}^{\{\tau\}, \{j\}} Z_s^{\{\tau\}})$	$p(Z_{\text{LTM}}^{\{j\}})$

The abstract methods representing probability functions need to be implemented as compatible probabilistic programs utilising Pyro. Besides the abstract methods users also need to provide $p(Z_s^{\{t\}})$ as a probabilistic program. Besides the above necessary methods, the class also specifies two additional methods that can be used to control the subsampling of the long-term memory and perceptual buffer for use in the calculation of information gain and constraint violations. With these methods implemented users can call the class method "makePlan(...)" which via stochastic variational inference finds an approximate optimal set of parameters, $q_{\phi_{\text{Mb}}^{\{\tau\}}, *}$ to the variational inference problem

$$\min_{\phi_{\text{WM}}^{\{t\}+}} D \left[p \left(Z_{\text{WM}}^{\{t\}+} \middle| \check{Z}_s^{\{t\}}, \check{Z}_{\text{LTM}}, \dots \right) \middle| \begin{array}{l} q_{\phi_{\text{WM}}^{\{t\}+}} \left(Z_{\text{WM}}^{\{t\}+} \middle| \check{Z}_s^{\{t\}}, \check{Z}_{\text{LTM}} \right) \\ \cdot p(Z_{\text{WM}}^{\{t\}-}, Z_{\text{LTM}}) \end{array} \right] \cdot p(Z_{\text{WM}}^{\{t\}-}, Z_{\text{LTM}})$$

where the user can specify the divergence measure and optimiser used. The optimal set of parameters is used to draw samples of the future motor buffer

$$Z_{\text{Mb}}^{\{\tau\}} \sim q_{\phi_{\text{Mb}}^{\{\tau\}}, *} (Z_{\text{Mb}}^{\{\tau\}} | Z_s^{\{\tau\}}) \quad ; \tau \in \{t, \dots, \bar{T} - 1\}.$$

These samples constitute potential future optimal actions needed to optimise information gain or progress while satisfying constraints. Finally, the "makePlan(...)" method either returns these samples or a sample mean hereof. The code is available through [14].

IV. AUTONOMOUS ROBOT EXPLORATION

To exemplify the utility of the proposed idiom, we have used it to implement an algorithm for autonomous robot exploration. The code for this can be found through [14]. The goal of the implementation is for a robot to explore an environment represented by a grid map autonomously, consider the problem at a high level, and define the state to be the current position in the XY-plane, $Z_s^{\{\tau\}} = [z_x^{\{\tau\}}, z_y^{\{\tau\}}]^T$, and use the simple transition model as

$$p(Z_s^{\{\tau+1\}} | Z_s^{\{\tau\}}, Z_{\text{Mb}}^{\{\tau\}}) = N(Z_s^{\{\tau\}} + A(Z_{\text{Mb}}^{\{\tau\}}), \sigma_a)$$

where $Z_{\text{Mb}}^{\{\tau\}}$ is the relative position scaled to be in the interval $[0, 1]$, $A(\dots)$ is a linear scaling of the relative position to be in the range $[\Delta Z_a, \bar{\Delta Z}_a]$, and σ_a is the covariance of the error allowed in the movement. Since the robot should have no prior preference of its movement we define

$$p(Z_{\text{Mb}}^{\{\tau\}} | Z_s^{\{\tau\}}) = U \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right). \quad (11)$$

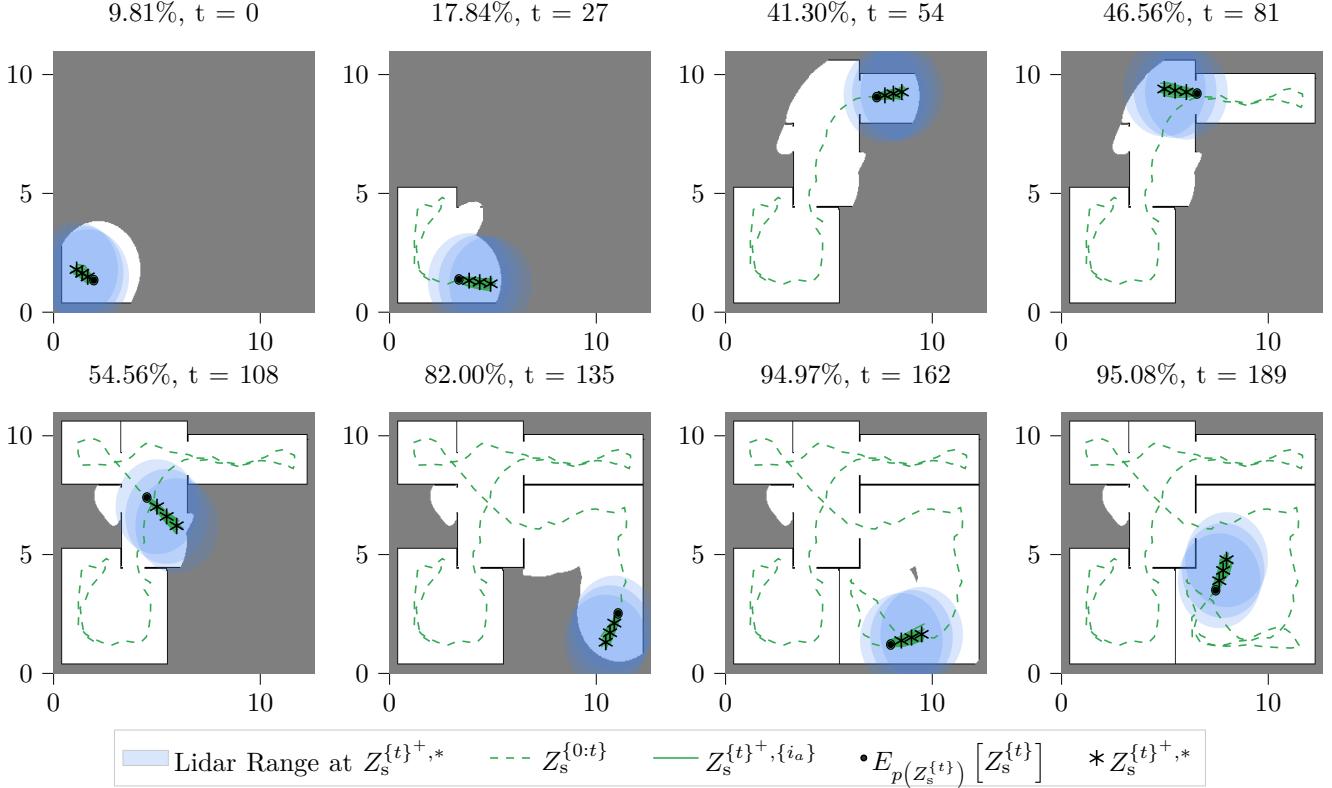


Fig. 3. The progress of a simulation for the map with ID "7fb9c9203cb8c4404f4af1781f1c6999" after each of the timesteps $t = \{0, 27, 54, 81, 108, 135, 162, 189\}$. For each timestep the previous positions, $Z_s^{\{0:t\}}$, is shown by a green dashed line, the mean of the current position, $Z_s^{\{t\}}$, is shown by a black dot, samples of future positions, $Z_s^{\{\tau\}}{}^{+, \{i_a\}}$, is shown by solid green lines, and the mean of these samples corresponding to the optimal future positions based on $Z_a^{\{t\}}{}^{+, *}$ are shown by black asterisks. The simulation where terminated after $t = 189$ since the exploration percentage where above 95%.

$q_{\phi_{\text{Mb}}^{\{\tau\}}} (Z_{\text{Mb}}^{\{\tau\}} | Z_s^{\{\tau\}})$ should have the same support as Eq. (11), but should also be flexible enough to represent preferences in the relative position. Thus, we define

$$q_{\phi_{\text{Mb}}^{\{\tau\}}} (Z_{\text{Mb}}^{\{\tau\}} | Z_s^{\{\tau\}}) = \text{Beta} (\alpha_a^{\{\tau\}}, \beta_a^{\{\tau\}})$$

since the beta distribution subsumes the uniform distribution, but also can represent a single mode. Thus, we have $\phi_{\text{Mb}}^{\{\tau\}} = \{\alpha_a^{\{\tau\}}, \beta_a^{\{\tau\}}\}$. We consider the grid map to be the long-term memory. That is, $Z_{\text{LTM}} = \{z_m^{\{1\}}, \dots, z_m^{\{I_m\}}\}$ where $z_m^{\{i_m\}}$ is each of the cells in the grid map, and make the common assumption that

$$p(Z_{\text{LTM}}) = \prod_{i_m=1}^I p(z_m^{\{i_m\}})$$

where

$$p(z_m^{\{i_m\}}) = \text{Bernoulli} (P_m^{\{i_m\}})$$

and $P_m^{\{i_m\}}$ is the probability of the i_m 'th grid cell being occupied. We assume that the environment is perceived through a lidar with 360° field of view and evenly spaced

lidar beams with 1° spacing, and define

$$p(Z_{\text{pb}}^{\{\tau\}} | Z_s^{\{\tau\}}, Z_{\text{LTM}}) = \prod_{i_r=1}^{360} p(z_{r,d}^{\{\tau\}, \{i_r\}} | Z_s^{\{\tau\}}, Z_{\text{LTM}}^{\{\tau\}, \{i_r\}})$$

where $z_{r,d}^{\{\tau\}, \{i_r\}}$ is the distance measured by the i_r 'th laser beam at time τ given the current position and grid map, and

$$Z_{\text{LTM}}^{\{\tau\}, \{i_r\}} = \{z_m^{\{i_m\}} \in Z_{\text{LTM}} | \text{ray } i_r \text{ intersects cell } i_m\}$$

is the cell in the grid map that the i_r 'th laser beam intersects. We obtain the set, $Z_{\text{LTM}}^{\{\tau\}, \{i_r\}}$, through ray-tracing. The distribution $p(z_{r,d}^{\{\tau\}, \{i_r\}} | Z_s^{\{\tau\}}, Z_{\text{LTM}}^{\{\tau\}, \{i_r\}})$ is implemented according to the laser beam model in [15]. Without taking the map into consideration the robot have no prior knowledge on the distance measured by the lidar, and thus we define

$$p(z_{r,d}^{\{\tau\}, \{i_r\}} | Z_s^{\{\tau\}}) = U(0, \bar{z}_{r,d})$$

where $\bar{z}_{r,d}$ is the max range of the lidar beams. We furthermore want the robot to keep a minimum distance, d_{\min} , to occupied cells in the map and thus define the constraints via the logistic function

$$\begin{aligned} \tilde{\mathbb{1}}_A^{\{\tau\}, \{h\}} (d^{\{\tau\}, \{h\}} (Z_s^{\{\tau\}}, Z_{\text{pb}}^{\{\tau\}}, Z_{\text{LTM}})) \\ = \frac{1}{1 + e^{-\sigma_c \cdot (z_{r,d}^{\{\tau\}, \{h\}} - d_{\min})}} \end{aligned}$$

where σ_c determines the steepness of the logistic function. With the above definitions, we have $J = H = 360$. Calculating the information gain and constraint violation based on all 360 lidar beams is computationally intractable in the current implementation. Therefore, for each timestep, τ , we sub-sample the number of lidar beams taken into account by randomly picking $\tilde{J} \ll 360$ and $\tilde{H} \ll 360$ lidar beams for calculating the information gain and constraint violation, respectively. In our implementation, we have furthermore chosen to use Pyro's build-in "ClippedAdam" optimizer with the standard D_{KL} divergence measure. Finally, the next action that the robot should take, $Z_a^{\{\tau\},*}$, is calculated as the sample mean of optimal actions

$$Z_a^{\{\tau\},*} = \frac{1}{I_a} \sum_{i_a=1}^{I_a} A(Z_{Mb}^{\{\tau\},\{i_a\}}) \quad (12)$$

where $Z_{Mb}^{\{\tau\},\{i_a\}} \sim q_{\phi_{Mb}^{\{\tau\},*}}(Z_{Mb}^{\{\tau\}} | Z_s^{\{\tau\}})$. The calculated $Z_a^{\{\tau\},*}$ is considered the optimal action for the robot to take in order to maximize progress or the information obtained.

A. Simulation

To test the algorithm implemented for autonomous robot exploration, we performed simulations on the 35,126 2D floor plans available in the HouseExpo dataset utilising a modified version of the accompanying PseudoSLAM simulator [5]. The PseudoSLAM simulator is made to efficiently generate occupancy grid maps directly from 2D floor plans, without the computational burden of running a real SLAM algorithm. The simulator also calculates the percentage of the map that has been explored and keeps a count of the number of crashes. Thereby, the simulator is suitable for large-scale simulation studies.

Unfortunately, the original PseudoSLAM simulator only allowed for the three fixed discrete movements: turn θ degrees to the left, turn θ degrees to the right, and move X meters forward, where θ and X are fixed variables. Thus, the original simulator was not suitable for the continuous movements calculated by Eq. (12). Therefore, modifications were made to allow for such continuous movements in the simulator. Furthermore, it was found that the function "*measure_ratio()*" build into the PseudoSLAM simulator, meant to quantify the percentage of the map explored, counter-intuitively could return values greater than 1. Thus, we also modified this function. The modified PseudoSLAM simulator is available through [14].

For our simulations, we adopted the simulation procedure used in [5]. One simulation with a random initial position was performed for each of the 35,126 2D floor plans. The simulations were limited to 200 time-steps. They were terminated if the "*measure_ratio()*" function returned more than 0.95, corresponding to more than 95% of the map had been explored. As an example, the result of one of the simulations is illustrated in Fig. 3.

From Fig. 4 it is seen that for the smallest floor plans in the data set, the robot manages to explore most of its environment. As the size of the floor plans increases,

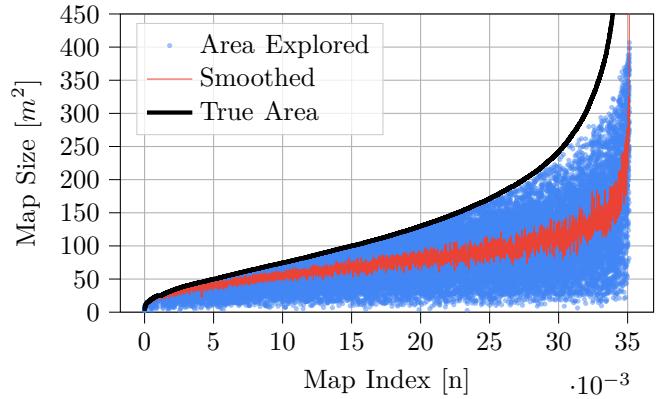


Fig. 4. The area explored for each of the 35126 simulations performed with the indices sorted in ascending order by the true area of the map. The red curve shows a moving average with a window size of 20.

a smaller percentage of the environment is explored on average. This is expected behavior since there is a limit to how much of a map the robot can explore in a fixed amount of time steps. However, Fig. 5 might reveal another cause.

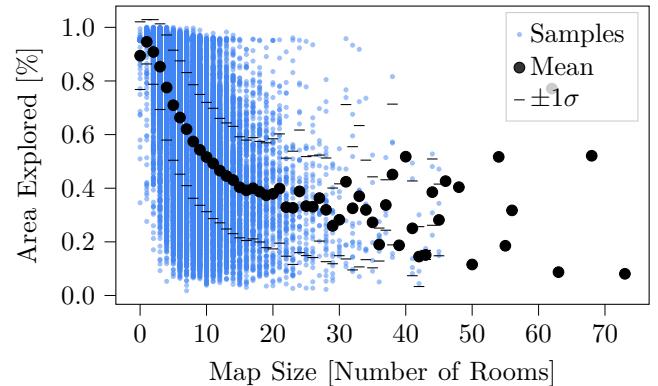


Fig. 5. The percentage of area explored in each of the 35126 simulations performed compared to the number of rooms in each of the maps.

From Fig. 5 there seems to be a clear relationship between the number of rooms in the environment, and the percentage of the environment that the robot manages to explore. A possible cause of this could be that for the robot to explore multiple rooms it often has to pass through narrow doorways. Passing through narrow doorways presents a high risk of constraint violation. In many situations, there will be alternative paths away from doorways that still yield progress. Therefore, if the paths going through the doorway does not yield a high probability of information gain, the presented idiom will prefer actions away from such doorways. This means that the robot could spend more time-steps than necessary in rooms that are fully explored.

As an example consider the simulation illustrated in Fig. 6. In this simulation, the robot starts in "room 1" and passes through a doorway to "room 2" already after a few time-steps. After passing through the doorway, the robot quickly explores the entire "room 2". However, since the area in "room 1" in close vicinity to the doorway is already ex-

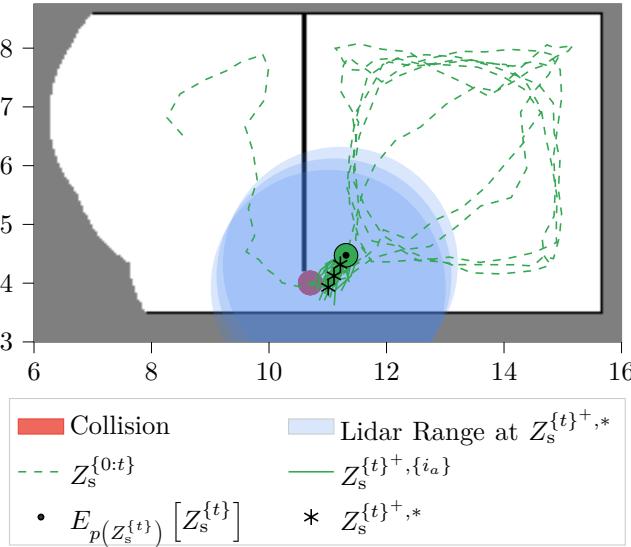


Fig. 6. The progress of a simulation for the map with ID "23e99dac3228ee2d371c5a627c49e415" after 200 time-steps. In this simulation, the robot spends a lot of time-steps driving around in the same room without getting out of it even though it is fully explored. The figure also shows an example of a collision in a doorway.

plored, the probability of information gain for paths passing back through the doorway is low due to the limited lidar range used to define $p(Z_{\text{pb}}^{\{\tau\}} | Z_s^{\{\tau\}}, Z_{\text{LTM}})$. Therefore, the robots keep driving around in "room 2" driven purely by progress. Overcoming this behavior would require some kind of memory about from which of the previous states the robot could obtain more knowledge, and some additional decision variables to guide the robot back to these states.

Besides guiding an agent towards new knowledge the idiom is also supposed to avoid constraints. In the implemented robot exploration algorithm, the only constraint is to prevent collisions with the robots surrounding. A total of 1617 unique collisions were recorded in 1253 different maps during the 6469065 time-steps simulated in all of the 35126 2D floor plans. Thus, only 0,25 % of the time-steps resulted in collisions. Nearly all of these collisions were registered near corners or doorways, like the collision shown in Fig. 6. Given that the idiom currently only supports checking constraints at discrete states, such behavior is to be expected, since the constraint can be satisfied at two consecutive states but not in between. Furthermore, for the specific application of robot exploration, this small probability of collision would probably be deemed tolerable, since in many cases would have to be a low-level collision avoidance system anyway. If this cannot be tolerated, the idiom would have to modified to include checking of constraint in between the discrete states.

Everything considered the ability of the idiom to guide an agent towards new knowledge while avoiding constraints seems to be as should be expected.

V. DISCUSSION

In this paper, we have shown how to develop a generally applicable probabilistic programming idiom for the

problem of making decisions under uncertainty to obtain new knowledge about an environment. We based our idiom on the memory structure of the Standard model of mind, and other ideas from research in cognitive architectures. We furthermore show how this idiom can be used for the specific problem of active mapping and robot exploration. Based on an extensive simulation study of this problem, it is concluded that the idiom works as could be expected. The simulation also indicated that the idiom probably would benefit from additional memory of old states in which more knowledge can be obtained. Furthermore, the simulation also indicated that the idiom for some application could benefit from checking constraints in between states.

REFERENCES

- [1] I. Lluvia, E. Lazcano, and A. Ansuategi, "Active mapping and robot exploration: A survey," *Sensors*, vol. 21, no. 7, 2021. [Online]. Available: <https://www.mdpi.com/1424-8220/21/7/2445>
- [2] A. Topiwala, P. Inani, and A. Kathpal, "Frontier based exploration for autonomous robot," 2018.
- [3] E. Uslu, F. Çakmak, M. Balcılar, A. Akinci, M. F. Amasyali, and S. Yavuz, "Implementation of frontier-based exploration algorithm for an autonomous robot," in *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, 2015, pp. 1–7.
- [4] D. A. Perkasa and J. Santoso, "Improved frontier exploration strategy for active mapping with mobile robot," in *2020 7th International Conference on Advance Informatics: Concepts, Theory and Applications (ICAICTA)*, 2020, pp. 1–6.
- [5] L. Tingguang, H. Danny, L. Chenming, Z. Delong, W. Chaoqun, and M. Q.-H. Meng, "Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots," *arXiv preprint arXiv:1903.09845*, 2019.
- [6] J. E. Laird, C. Lebiere, and P. S. Rosenbloom, "A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics," *AI Magazine*, vol. 38, no. 4, pp. 13–26, Dec. 2017. [Online]. Available: <https://ojs.aaai.org/index.php/aimagazine/article/view/2744>
- [7] M. R. Damgaard, R. Pedersen, and T. Bak, "Toward an idiomatic framework for cognitive robotics," 2021.
- [8] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, no. 4, pp. 1303–1347, 2013. [Online]. Available: <http://jmlr.org/papers/v14/hoffman13a.html>
- [9] D. Palomar and S. Verdú, "Lautum information," in *2006 IEEE Information Theory Workshop - ITW '06 Punta del Este*, 2006, pp. 1–5.
- [10] T. Minka, "Divergence measures and message passing," Microsoft, Tech. Rep. MSR-TR-2005-173, January 2005. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/divergence-measures-and-message-passing/>
- [11] P. S. Rosenbloom, J. Gratch, and V. Ustun, "Towards emotion in sigma: From appraisal to attention," in *Artificial General Intelligence*, J. Bieger, B. Goertzel, and A. Potapov, Eds. Cham: Springer International Publishing, 2015, pp. 142–151. [Online]. Available: https://doi.org/10.1007/978-3-319-21365-1_15
- [12] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *CoRR*, vol. abs/1805.00909, 2018. [Online]. Available: <http://arxiv.org/abs/1805.00909>
- [13] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *J. Mach. Learn. Res.*, vol. 20, pp. 28:1–28:6, 2019. [Online]. Available: <http://jmlr.org/papers/v20/18-403.html>
- [14] M. R. Damgaard, "probmind," Jan. 2022. [Online]. Available: <https://doi.org/10.5281/zenodo.5841292>
- [15] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*, ser. Intelligent robotics and autonomous agents. MIT Press, 2005.