PIVOT: Predictive Incremental Variable Ordering Tactic for Efficient Belief Space Planning

Khen Elimelech and Vadim Indelman

I. Motivation

Most autonomous agents share the same fundamental goal – to autonomously plan and execute their actions. These agents are required to account for real-world uncertainty when planning their actions, in order to achieve reliable and robust performance. Also, problems such as long-term autonomous navigation and sensor placement over large areas, often involve optimization of numerous variables. These settings are translated to high-dimensional probabilistic states, known as "beliefs". Belief Space Planning (BSP) is a fundamental problem in artificial intelligence; relevant instantiations include active Simultaneous Localization and Mapping (SLAM), active sensing and robotic arm manipulation. The objective in these planning problems is often to select "safe" actions, which reduce the uncertainty in the belief. Unlike state inference, while planning, one should propagate the belief considering *multiple* candidate courses of action, in order to select the optimal one. Overall, the computational complexity of the problem can turn exceptionally high, thus making it challenging for online systems, or when having a limited processing power.

II. Problem Definition

The BSP problem is usually solved sequentially. At each time-step, the agent maintains a posterior distribution over its current state vector, given the controls and observations until that time. The state vector $\boldsymbol{\chi}_k$ may contain the executed trajectory $\boldsymbol{x}_{1:k}$, with possibly some external variables $\boldsymbol{l}$ (such as landmarks, in SLAM). This distribution, known as the belief, is often modeled as a Guassian distribution:

$$b_k(\boldsymbol{\chi}_k) \doteq P(\boldsymbol{\chi}_k \mid \boldsymbol{u}_{1:k-1}, \boldsymbol{z}_{1:k}) = \mathcal{N}(\overline{\boldsymbol{X}}_k, \boldsymbol{\Lambda}_k^{-1}), \qquad (1)$$

where $\boldsymbol{\Lambda}_k$ is the information matrix and $\overline{\boldsymbol{X}}_k$ is the maximum a posteriori (MAP) state estimate.

We can reason about the updated belief at the next time-steps, after taking some actions (and observations) – a process known as state inference. The information matrix of the belief, after a sequence of T additional controls $\boldsymbol{u} \doteq \boldsymbol{u}_{k:k+T-1}$, is given by $\boldsymbol{\Lambda}_{k+T} = \boldsymbol{\Lambda}_k + \boldsymbol{U}^T\boldsymbol{U}$, where U the *collective Jacobian* of the control sequence $\boldsymbol{u}$ – a stack of the whitened Jacobians of the added constraints.

State of the art inference techniques (e.g. [8], [6]) represent the (Gaussian) belief using the information square root matrix $\boldsymbol{R}_k$, given by the Cholesky factorization, such that $\boldsymbol{R}_k^T\boldsymbol{R}_k = \boldsymbol{\Lambda}_k$. Calculating this root matrix is needed to update the state estimate (for more details see [3]). Conveniently, instead of calculating the root matrix from scratch after each action, it can be incrementally updated, by only calculating the root of the affected block. The variables in this block are the *affected variables*.

In BSP, given an initial belief b_k and a set $\mathcal{U}_k$ of candidate control sequences, we wish to select the control which maximizes the expected accumulated reward, as measured by the objective function $J(b_k, \boldsymbol{u})$, for some reward function. E.g., in information-theoretic BSP, we can use the differential entropy as the reward function, in order to minimize the uncertainty in the posterior belief. To calculate the objective function, the prior belief b_k should be propagated according to the considered control sequence $\boldsymbol{u}$. Overall, at each planning session, we look at the following decision problem:

$$\boldsymbol{u}^* = \underset{\boldsymbol{u}\in\mathcal{U}_k}{\operatorname{argmax}} J(b_k, \boldsymbol{u}). \qquad (2)$$

In our previous work [4], [5], we introduced a novel method for an efficient solution of the aforementioned decision problem; we suggested using a sparse approximation of the initial belief only while planning, without influencing the state inference. This method proved to be very worthwhile. Still, on each planning session, we bared an overhead of calculating the sparsifcation from scratch, as the approximation from the previous step had been used and discarded. In this work, we examine the possibility of incrementally updating the sparse approximation between planning sessions, in order to further improve the planning performance.

III. Approach

The sparsification algorithm (in [5]) for the belief's information matrix can generate different approximations, as it relies on a given subset $\mathcal{S}$ of state variables for sparsification. A particularly interesting case is when $\mathcal{S}$ contains only variables which are *uninvolved* in any of the candidate control actions. Then, the algorithm returns an *action consistent* approximation – one which is guaranteed to not affect the action selection. For each action u, we can identify the set of involved variables marked as $\mathcal{I}nv(u)$; these are the state variables which are directly updated by u, i.e. variables for which the action introduces a new constraint. In the representation presented before, this means that the column corresponding to this variable in the collective Jacobian of the action is non-zero. The remaining variables are uninvolved in this action, and marked $-\mathcal{I}nv(u)$. Accordingly, the variables which are uninvolved in any of the actions $u \in \mathcal{U}$ are marked $-\mathcal{I}nv(\mathcal{U})$.

By definition, for each action, the involved variables are a subset of the affected variables. The affected variables

can contain also uninvolved variables "trapped" between the involved ones, that should only be considered in the update due to the order of variables. Had they been positioned before the first involved variable, they would have been unaffected. When looking at the sparse information matrix, for each sparsified variable, the sparsification algorithm replaces the factors tied to this variable, with a single adjusted diagonal entry. So, when applying the action, we are able to minimize the computational cost inflicted by these "trapped" uninvolved variables on the update process.

Now, consider by separating the variables before each planning session under the order

$$\boldsymbol{P}_k = \begin{bmatrix} -\mathcal{I}nv(\mathcal{U}_k) \\ \mathcal{I}nv(\mathcal{U}_k) \end{bmatrix}. \tag{3}$$

Then, all the uninvolved variables $-\mathcal{I}nv(\mathcal{U})$ would appear before the involved variables $\mathcal{I}nv(\mathcal{U})$. Thus, for all the candidate actions, these variables are necessarily also unaffected. Hence, we now come to realize that by keeping the uninvolved variables first, we do not actually need to sparsify them. Since they are not a part of the update calculation, for all candidate actions in the planning session, their sparsification would not influence the performance anyway. Meaning, instead of sparsifying the trapped variables, the suggested order improves the cost of belief updates in planning by pushing backwards trapped variables, and reducing the size of affected block for each action. Hence, using this order, we can achieve the computational benefits in planning previously gained by applying sparsification. Since the identification of involved variables, and thus this order, is based on the predicted updates induced by the candidate action, we refer to $\boldsymbol{P}_k$ as *predictive variable order*.

Thus far, after solving each planning session, the (reordered and) sparsified belief was discarded, and the selected action was applied on the original belief. Hence, at each planning session, we applied a full "batch" sparsification of the root matrix (which required a full refactorization). However, the classification (un/involved) of variables usually does not change much between planning sessions. We thus propose to enforce this variable order on the state inference as well, instead of using the altered belief only for planning. This way, the order applied at the previous planning session is kept, and can be incrementally updated, according to the change in variable classification. We, hence, refer to the approach as *PIVOT – Predictive Incremental Variable Ordering Tactic*. When the reordering required between sessions is small, the original cost of applying the sparsification, can be reduced significantly. Still, we should make sure the efficiency of the inference process is not compromised.

IV. Initial Results

As a proof of concept, we applied PIVOT in the solution of an active-SLAM scenario, in which a ground agent autonomously navigats to a specified goal in an unknown environment, with random landmarks scattered. The agent simulates having a radar/lidar sensor. Using the Probabilistic RoadMmap (PRM) algorithm [9], the agent generates a set of candidate trajectories to the goal: one direct trajectory, and indirect trajectories which pass through known loop closure, in order to reduce uncertainty. Overall, the agent's state consists of the entire executed trajectory and positions of observed landmarks (i.e. full-SLAM). Our approach is highly relevant in this case; as the state size grows quickly from both the past poses and the mapping of the environment, planning becomes more computationally-challenging as the agent progresses. Moreover, as the agent progresses towards the goal, previously observed landmarks are expected to become uninvolved. Under various settings, the improvement in planning time is substantial, with almost no influence on the state inference process time.

V. Related Work

Our previous work [4], [5], [7] was the first attempt to utilize sparsification exclusively for planning. This concept now evolved to PIVOT.

As explained, to solve the problem, we examine the propagated belief's triangular square root information matrix. It is widely known that the state variables' order can have a dramatic impact on the number of non-zeros in the root matrix, and by such on the factorization efficiency. Thus, fill-reducing variable ordering tactics are very commonly used in the solution of the state inference problem, especially for SLAM [1]. State of the art inference techniques, such as iSAM2 [8] and SLAM++ [6], apply such orders incrementally, as part of the belief updates. However, variable ordering specifically for planning, as presented here, has (almost) not been considered. The only related idea is presented in [2], and suggests to constrain variable order in the Bayes tree, to allow reuse of calculations for similar candidate actions and successive planning iterations. Still, this scheme is conceptually different than the one presented here.

References

[1] Pratik Agarwal and Edwin Olson. Variable reordering strategies for slam. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3844–3850. IEEE, 2012.

[2] Stephen M Chaves and Ryan M Eustice. Efficient planning with the Bayes tree for active SLAM. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4664–4671. IEEE, 2016.

[3] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research*, 25(12):1181–1203, Dec 2006.

[4] K. Elimelech and V. Indelman. Consistent sparsification for efficient decision making under uncertainty in high dimensional state spaces. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2017.

[5] K. Elimelech and V. Indelman. Scalable sparsification for efficient decision making under uncertainty in high dimensional state spaces. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, September 2017.

[6] Viorela Ila, Lukas Polok, Marek Solony, and Pavel Svoboda. SLAM++ - a highly efficient and temporally scalable incremental slam framework. *Intl. J. of Robotics Research*, 36(2):210–230, 2017.

[7] V. Indelman. No correlations involved: Decision making under uncertainty in a conservative sparse information space. *IEEE Robotics and Automation Letters (RA-L)*, 1(1):407–414, 2016.

[8] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31:217–236, Feb 2012.

[9] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, 1996.