Linear-Time Estimation with Tree Assumed Density Filtering and Low-Rank Approximation

Duy-Nguyen Ta and Frank Dellaert

***Abstract*— We present two fast and memory-efficient approximate estimation methods, targeting obstacle avoidance applications on small robot platforms. Our methods avoid a main bottleneck of traditional filtering techniques, which creates densely correlated cliques of landmarks, leading to expensive time and space complexity. We introduce a novel technique to avoid the dense cliques by sparsifying them into a tree structure and maintain that tree structure efficiently over time. Unlike other edge removal graph sparsification methods, our methods sparsify the landmark cliques by introducing new variables to de-correlate them. The first method projects the current density onto a tree rooted at the same variable at each step. The second method improves upon the first one by carefully choosing a new low-dimensional root variable at each step to replace such that the independence and conditional densities of the landmarks given the trajectory are optimally preserved. Our experiments show a significant improvement in time and space complexity of the methods compared to other standard filtering techniques in worst-case scenarios, with small trade-offs in accuracy due to low-rank approximation errors.**

I. Introduction

In the domain of real-time simultaneous localization and map-building (SLAM), filtering [1], [2] and incremental smoothing [3], [4] methods are among the most popular techniques. While filtering methods marginalize out old robot poses to keep the problem size manageable in the long run, incremental smoothing techniques focus the computation only on the variables of interests.

One of the longest-lasting and most challenging bottlenecks of these methods is that marginalizing out old poses results in dense correlations among all landmarks, which correspond to dense fill-in of matrices and lead to expensive computational complexity [2]. For example, the worst-case time complexity of information filters is almost cubic $O(m^3)$ with respect to the number of landmarks m, due to inversion of a dense information matrix [5]. Similarly, the iSAM2 incremental smoothing method also takes cubic time in worst-case scenarios [4]. On the other hand, the Extended Kalman Filter is quadratic $O(m^2)$ in time due to matrix multiplication [5]. Hence, these methods are still unsuitable for small robots with limited memory and processing capabilities, especially in environments with many landmarks.

State-of-the-art research has attempted to eliminate this bottleneck, but not without trade-offs in either inconsistent estimates or non-real-time operations. For example, graph sparsification techniques try to sparsify the dense correlations as much as possible to keep the problem solvable in constant time [8], [9], [6], [7], [10], [11]. However, many of these methods produce inconsistent and over-confident estimates, since they simply remove weak edges by zeroing out small entries in the information matrix [7]. Recent work employs different optimization techniques to search for consistent sparse approximations of the problem [12], [13], [14], or maintains the exact solution by solving for the approximation error with iterative methods [15], [16], [17]. Unfortunately, these methods are not yet ready for real-time applications. Moreover, various re-parameterization schemes have also been proposed to speed up the system [18], [19], [20], [21]. Especially, [22] achieves a constant time update on large graphs using an incremental pose representation, but it is still expensive for real-time purposes.

We present two fast and memory-efficient approximate estimation methods suitable for small and fast maneuvering robot platforms with limited memory and processing power. We favor filtering-based methods, which maintain small memory footprints by marginalizing out old robot poses, over smoothing-based approaches [23], [4], because past poses are unnecessary for high-level controllers and planners to compute an obstacle avoidance policy. Typically, only the pairwise relative geometric relationships between the current robot and each object in the environment at the current time are needed for trajectory planning and execution tasks.

Instead of removing edges, our methods sparsify the dense connections of landmarks by introducing new latent variables to de-correlate them into a tree structure and updating this tree structure incrementally at every step. Intuitively, this is the inverse of the marginalization process: while marginalizing out a variable results in a dense correlation among other variables connected to it, re-introducing a new variable can de-correlate them. By maintaining a tree structure over time, our methods can compute the solution in $O(m)$ time in worst-case scenarios, the best time complexity we can achieve for solving this problem in theory. Furthermore, since trees are the sparsest graph structure, they are memory-efficient and suitable for small robots with limited memory.

Our first method, in Section III, uses techniques from assumed density filtering [24], [25], [26] and tree-dependent component analysis [27] to "project" the current density onto a tree rooted at the same variable at each step. This projection step is efficient; however, similar to other edge removal techniques, it is an inconsistent and lossy approximation.

Our second method, in Section IV, improves upon the first by carefully choosing a new variable at each step such that

The authors are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, Georgia, USA,
{duynguyen, dellaert}@gatech.edu

approximately all landmarks are conditionally independent of each other given this new variable. Because the landmarks are completely independent given the full robot trajectory, the new variables should contain enough information of the trajectory to approximate the landmark conditionals properly.

Instead of finding a low-rank approximation of the full trajectory as done in [28], we directly find new low-dimensional representations of the trajectory that best preserve the landmarks' conditional densities at each step. Under an assumption that the conditional means of nearby landmarks given the trajectory lie on a low-dimensional linear subspace, we propose a fast low-rank approximation scheme to efficiently find the constraints for the new low-dimensional variables to approximate that subspace.

Our approximation scheme is different in nature than other graph sparsification methods. While other methods result in inevitable information loss due to the explicit removal of graph edges or the zeroing out of small entries in the information matrices, our method loses information through the low-rank approximation and the linear subspace assumption. Consequently, as will be shown in Section V, if the problem possesses the low-rank property, our method can provide a lossless solution, while other graph sparsification methods cannot. On the other hand, it might suffer from large approximation errors if the low-rank assumption is poorly satisfied. We note that the linear-time algorithm in [29] also uses a low-rank approximation for the Kalman gain matrix. However, it involves computing the eigenvectors using the Power method, which might be inaccurate with a fixed number of iterations [30].

II. Problem Formulation

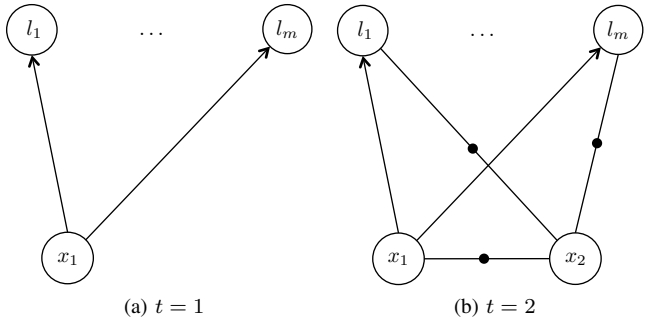
As a standard in the literature [1], [23], SLAM is formulated as finding the maximum a posterior (MAP) estimate of the unknown robot poses $X = \{x_i\}_{i=1}^n$ and landmarks $L = \{l_j\}_{j=1}^m$, given the set of all measurements $Z = \{z_{ij}\}$, with z_{ij} the measurement of the landmark l_j viewed from the pose x_i, and the set of all odometry measurements $U = \{u_i\}$, with u_i the odometry measurement between the camera poses $i-1$ and i:

$$\begin{aligned} X^*, L^* &= \underset{X,L}{\operatorname{argmax}}\, p(X, L, Z, U) \\ &= \underset{X,L}{\operatorname{argmax}}\, p(x_1) \prod_{i=2}^{m} p(x_i \,|\, x_{i-1}, u_i) \prod_{i,j} p(z_{ij} \,|\, x_i, l_j) \end{aligned} \tag{1}$$

In the following, since Z, U are constant measurements, we will omit them for clarity whenever their existence is clear from the context.

We are interested in filtering-based SLAM methods which marginalize out past poses to keep the problem size manageable, since we target robot platforms with limited memory and past poses are unnecessary for high-level planners and controllers. Specifically, we are concerning with the following joint density of the last pose and the landmarks:

$$p(x_t, L) = \int_{X_{1:t-1}} p(X, L)$$



(a) $t = 1$ (b) $t = 2$

Fig. 1: The tree structure at $t = 1$, and the graph at $t = 2$ with odometry (x_1, x_2) and new measurements from x_2

Graph representations of SLAM as well as the connections between optimization methods and graph inference techniques have been thoroughly discussed [9], [7], [31], [23]. Basically, the variable elimination method factorizes the joint density into a conditional and a new factor on the remaining variables: $p(X, L) = p(X_{1:t-1}|L, x_t)p(L, x_t)$. Marginalization is essentially a by-product of a variable elimination, which retains only the new factor and ignores the conditional: $\int_{X_{1:t-1}} p(X, L) = \int_{X_{1:t-1}} p(X_{1:t-1}|L, x_t)p(L, x_t) = p(L, x_t)$. In this view, information filtering methods [5] are special cases of incremental smoothing techniques [3], [4] with a special variable elimination ordering which eliminates all the past poses first.

Eliminating past poses produces the new factor $p(x_t, L)$ that links all landmarks together. This is essentially the large clique that leads to $O(m^3)$ time complexity for naïve implementations of information filters [6] and iSAM2 [4]. As discussed in Section I, much work focuses on avoiding this large clique by removing edges in the graph, or zeroing out small entries in the information matrix, but they are not yet ready for real-time applications.

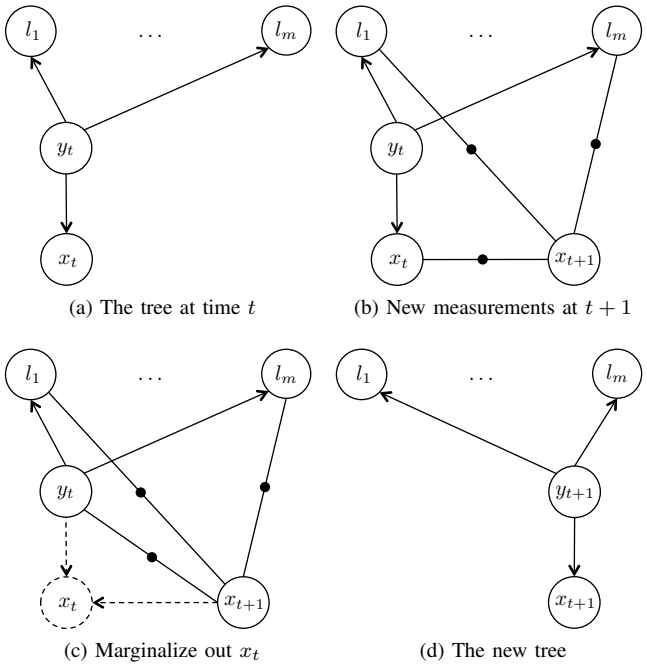
We avoid the large clique of landmarks and achieve linear-time estimation by always keeping a tree structure at time t rooted at a new variable y_t as shown in the Bayes net in Fig. 2a. The Bayes net encodes the tree factorization of the following joint density:

$$q(L, x_t, y_t) = q(y_t)q(x_t|y_t) \prod_{j=1}^{m} q(l_j|y_t)$$

The new variable y_t and its related densities must be chosen such that the joint density $q(L, x_t)$, after y_t is marginalized out from $q(L, x_t, y_t)$, matches the original joint density $p(L, x_t)$ that we want to estimate:

$$q(L, x_t) = \int_{y_t} q(L, x_t, y_t) \approx p(L, x_t) \tag{2}$$

Fortunately, it turns out that if we can find an efficient tree update scheme at every step, this condition (2) will be inductively satisfied by construction. This is because we always start out with a tree structure at time $t = 1$ where $y_1 = x_1$ (Fig. 1a), and the problem of finding the tree structure at time $t = 2$ with the first odometry and the second



(a) The tree at time t (b) New measurements at $t+1$

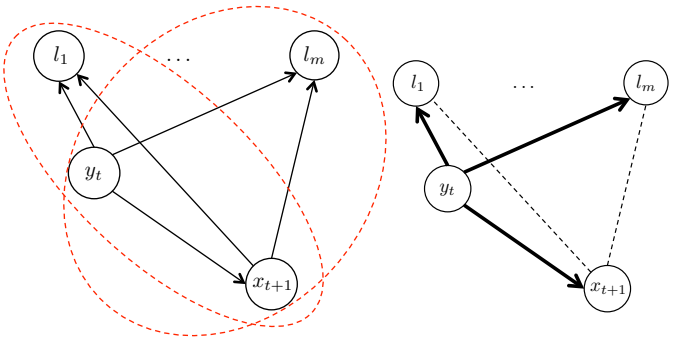
(c) Marginalize out x_t (d) The new tree

Fig. 2: Our general tree filtering scheme: (a) The tree at time t. (b) New measurements at time $t+1$ break the tree structure. (c) Marginalizing out x_t is efficient and does not produce the dense clique of landmarks. (d) The new tree we want to find at time $t+1$.

set of landmark measurements (Fig. 1b), is the same problem of finding the tree structure at time $t+1$ after we marginalize out x_t at any future time step (Fig. 2c).

Our general tree filtering scheme is as follows (Fig. 2). Assuming we already have a tree structure at time t (Fig. 2a), the new odometry and landmark measurements from x_{t+1} break this tree structure as shown in Fig. 2b. After marginalizing out x_t, we obtain the new graph in Fig. 2c, which has the same structure with the graph at time $t=2$ (Fig. 1b). We note that due to the tree structure at time t, marginalizing out x_t is efficient and does not result in a dense clique of landmarks as what happen with other standard filtering methods. Our goal is to find a new tree at time $t+1$ as in Fig. 2d that approximates the density in Fig. 2c.

The rest of this paper will focus on efficient methods to turn a graph in Fig. 2c into a tree in Fig. 2d with a new variable y_{t+1} that approximately satisfies the condition in Eq. (2) at time $t+1$. We present two methods to find such trees. The first method, based on assumed density filtering "projects" the new density onto the same tree propagated from the previous step after the previous pose is marginalized out. This method is simple and fast, but suffers from information loss and leads to inconsistent estimates, similar to other edge removal graph sparsification techniques [7]. The second method improves upon the first one with different new variables which better satisfy (2) at every step.



(a) Bayes net to compute the marginals (b) The new tree

Fig. 3: Tree Assumed Density Filtering scheme reusing the root of the previous step: (a) Elimination process to compute the root and conditional marginals for the new tree. Ellipses denote the three-variable cliques to compute the pairwise marginals $p(l_j, y_t)$ efficiently. (b) The new tree is bolded; dash lines are edges from the original graph (Fig. 2c).

III. Tree Assumed Density Filtering

An immediate solution for a tree at time $t+1$ is to reuse the tree of the previous time t. After marginalizing out the previous pose x_t, we use an idea similar to Assumed Density Filtering (ADF) technique [25], [26] to approximately "project" the current density onto the same tree structure T of the previous step, rooted at the same variable $y_{t+1} \equiv y_t$, as highlighted in Fig. 3b. In general, ADF and the related Expectation Propagation technique find an approximate density $q(x)$ of the original distribution $p(x)$. In our case, the approximate density $q(x)$ is limited to be in the family D^T of densities encoded by the tree T.

The best tree approximation $p_T(x) \in D^T$ of $p(x)$ that minimizes the KL-divergence $KL(p||q)$ over all $q \in D^T$ has been derived in the context of tree-dependent component analysis in [27]. The optimal tree approximation $p_T(x)$ of an arbitrary density $p(x)$ is factorized as follows:

$$\begin{aligned} q(x) = p_T(x) &= \prod_{u,v \in E} \frac{p(x_u, x_v)}{p(x_u)p(x_v)} \prod_{u \in V} p(x_u) \\ &= p(x_0) \prod_{u,v \in E} p(x_v | x_u) \end{aligned}$$

where V and E are the set of vertices and edges of our tree, rooted at x_0, and x_v is a child of x_u in the tree.

Hence, the new root prior $q(x_0)$ is simply the marginal $p(x_0)$, and the new conditionals $q(x_v|x_u)$ are the "marginal" conditionals $p(x_v|x_u) = p(x_v, x_u)/p(x_u)$, which can be computed from the marginals $p(x_u)$ and the pairwise marginals $p(x_v, x_u)$ easily.

In our case, we would like to approximate the density $p(y_t, x_{t+1}, L)$ at time $t+1$ in Fig. 3a with a tree Bayes net rooted at the same y_t in Fig. 3b:

$$\begin{aligned} q(y_t, x_{t+1}, L) &= q(y_t) q(x_{t+1} | y_t) \prod_j q(l_j | y_t) \\ &\approx p(y_t, x_{t+1}, L) \end{aligned}$$

Using the above result, we need to compute the marginal $p(y_t) = q(y_t)$, and the conditionals $p(x_{t+1}|\, y_t)$ and $p(l_j|\, y_t)$, which might be obtained from the pairwise marginals $p(x_{t+1}, y_t)$ and $p(l_j, y_t)$.

With our special tree structure at time t, these marginals, conditionals and pairwise marginals can be computed efficiently. As shown in Fig. 3a, we first eliminate all landmark variables l_j before eliminating x_{t+1} to obtain a Bayes net

$$p(y_t, x_{t+1}, L) = p(y_t)p(x_{t+1}|y_t) \prod_j p(l_j|y_t, x_{t+1}). \quad (3)$$

As results of the elimination, the root marginal $p(y_t)$ and the conditional $p(x_{t+1}|y_t)$ is ready from the Bayes net. The other pairwise marginals $p(l_j, y_t)$ can be obtained by marginalizing out x_{t+1} from the three-variable clique of y_t, x_{t+1}, and l_j (red ellipses in Fig. 3a): $p(l_j, y_t) = \int_{\neg\{l_j, y_t\}} p(y_t)p(x_{t+1}|y_t) \prod_k p(l_k|y_t, x_{t+1}) = \int_{x_{t+1}} p(y_t)p(x_{t+1}|y_t)p(l_j|y_t, x_{t+1})$. We then compute the conditionals $p(l_j|y_t)$ from $p(l_j, y_t)$ and the marginal $p(y_t)$.

Although this tree assumed density filtering scheme is efficient to compute, it incurs inevitable information loss. This is because it removes the conditional links between x_{t+1} and l_js, which is, similar to other graph sparsification techniques, equivalent to zeroing out the corresponding (x_{t+1}, l_j) entries in the information matrix. However, we found approximation errors are small in our experiments. Furthermore, the algorithm has $O(m)$ time complexity, since it only loops over the landmarks to compute the marginals.

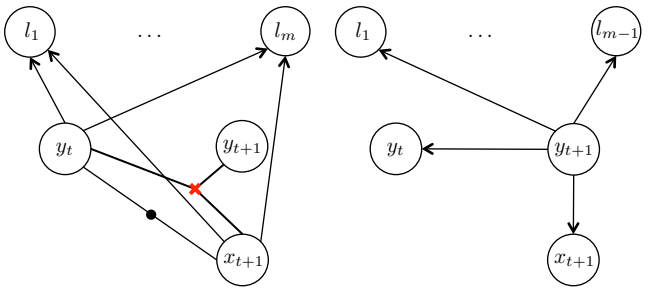
IV. Incremental Tree Filtering

To derive a better tree approximation for the new density at time $t+1$ satisfying the condition (2), we base ourselves on the fact that all landmarks are conditionally independent given the full trajectory $X_{1:t+1}$. In fact, in the full SLAM formulation (1), if instead of the poses we eliminate the landmark variables first, we obtain the following factorization: $p(X_{1:t+1}, L) = p(X_{1:t}, x_{t+1}) \prod_j p(l_j|X_{1:t+1})$, meaning that l_j are conditionally independent given the full trajectory $X_{1:t+1}$. Hence, a trivial choice for y_{t+1} that exactly satisfies (2) is $X_{1:t+1}$, but is too expensive a choice, naturally.

An immediate solution is to find a low-rank approximation of the full trajectory, as done in [28]. Unfortunately, this technique does not fit in the context of filtering-based SLAM, because it requires knowledge of all the past poses. Furthermore, a low-rank approximation of the trajectory is not our main interest, since the conditional independence of the landmarks might not be guaranteed given this approximation.

A more direct goal is to find a low-dimensional replacement y_{t+1} for the full trajectory $X_{1:t+1}$ such that the landmark conditional densities are optimally approximated, i.e. $p(l_j|\, y_{t+1}) \approx p(l_j|\, X_{1:t+1}), \forall j$. In a filtering context, $\{y_t, x_{t+1}\}$ plays the same role as the full trajectory $X_{1:t+1}$ in smoothing, in the sense that, at time $t+1$, all landmarks are conditionally independent given these two variables (Fig. 3a): $p(L, y_t, x_{t+1}) = p(y_t, x_{t+1}) \prod_j p(l_j|y_t, x_{t+1})$.

Equivalently, in filtering, we would like to find a low-dimensional variable y_{t+1} as a re-parameterization of $\{y_t, x_{t+1}\}$, such that it can "replace" $\{y_t, x_{t+1}\}$ in the conditionals $p(l_j|y_t, x_{t+1})$, and approximately generate the same conditional densities on l_js as $\{y_t, x_{t+1}\}$ do. As discussed above, the trivial re-parameterization $y_{t+1} = \{y_t, x_{t+1}\}$ will not gain us any computational benefits, because the dimension of the new variables will quickly increase, incorporating all information of the full trajectory into a high dimensional vector y_{t+1} at each step.



(a) Adding y_{t+1} and its constraint (b) The new tree

Fig. 4: Incremental Tree Filtering scheme. We find a new low-dimensional variable y_{t+1} as a reparameterization of $\{y_t, x_{t+1}\}$ such that the conditional densities of landmarks given the new variable in the new tree (b) best approximate the original conditional densities given $\{y_t, x_{t+1}\}$ in (a).

Using the moment-matching and low-rank approximation techniques, detailed in Section IV-A and IV-B, we find the best low-dimensional re-parameterization y_{t+1} of the original variables $\{y_t, x_{t+1}\}$, represented as a hard equality constraint among these three variables. The constraint guarantees that the conditional densities $p(l_j|\, y_t, x_{t+1})$ are best approximated by $p(l_j|y_{t+1})$ for all j, and given y_{t+1}, all l_js are approximately independent of each other: $p(L|y_t, x_{t+1}) = \prod_j p(l_j|y_t, x_{t+1}) \approx p(L|y_{t+1}) = \prod_j p(l_j|y_{t+1})$.

To find the new tree, we first add a constrained factor representing the found constraint between y_{t+1} and $\{y_t, x_{t+1}\}$ to the original graph (Fig. 4a), then apply the tree assumed density filtering technique in Section III to project the original density onto the new tree rooted at y_{t+1}, as shown in Fig. 4b. This procedure guarantees our original condition (2) to be satisfied at time $t+1$, i.e. after marginalizing out y_{t+1} from the new density $q(L, y_t, x_{t+1}, y_{t+1})$ with the hard constraint included, we obtain the same original density on $\{L, y_t, x_{t+1}\}$ as before. This is because intuitively a hard constraint can be seen as a delta distribution with zero information on the constrained variables; hence, adding it to the graph will not add more information nor change the density of the original variables. Furthermore, since y_{t+1} is specially chosen to approximate $\{y_t, x_{t+1}\}$ in the conditionals $p(l_j|y_t, x_{t+1})$, the tree assumed density filtering step will not incur much information loss, depending on how well $p(l_j|y_{t+1})$ can approximate $p(l_j|y_t, x_{t+1})$ in our low-rank approximation scheme. We note that due to our previous tree structure at time t, the marginals and pairwise marginals needed for our new tree can also be computed efficiently in $O(m)$ time as already discussed in Section III.

A. Moment-Matching of Gaussian Conditionals

As discussed above, we would like to find a new variable y_{t+1} such that the conditional $p(l_j|y_t, x_{t+1})$ can be approximated by $p(l_j|y_t)$. Assuming Gaussian densities, this is a special case of a more general "Gaussian conditional matching" problem as follows.

Gaussian Conditional Matching Problem: *What are the conditions on x and y such that the Gaussian conditional densities $p(l|x)$ and $q(l|y)$ match with each other, i.e. $p(l|x) = q(l|y)$?*

We assume the Gaussian conditionals p and q have the following forms:

$$\begin{aligned} p(l|x) &\propto \exp -\frac{1}{2} \|Rl - Sx - d\|^2, \text{ and} \\ q(l|y) &\propto \exp -\frac{1}{2} \|Pl - Ty - e\|^2, \end{aligned}$$

which satisfy the properties of Gaussian conditional distributions – their means are linear functions on the conditioned variables, and their information matrices are independent of these variables [32, pg. 90-91]. For example, the mean of $p(l|x)$ in this form is $(R^{-1}Sx + R^{-1}d)$, a linear function on x, and its information matrix, $R^T R$, is independent of x.

The necessary conditions for these two conditionals to match are $R = P$, $Sx = Ty$ and $d = e$. This is because for every pair of x and y generating the same conditional densities on l, we must have $\|Rl - Sx - d\|^2 = \|Pl - Ty - e\|^2, \forall l$, and the conditions follow.

Since setting $P = R$ and $e = d$ is trivial, we will focus on the other condition $Sx = Ty$. The condition $Sx = Ty$ must be satisfied for *all* possible pairs of x and y, such that the linear subspace generated by Ty must be the same as the linear subspace generated by Sx. Intuitively, this means that the *linear* space of all possible conditional means of the distribution $q(l|y)$, generated by all realizations of y, must be the same as that of the original distribution $p(l|x)$.

B. Low-rank Approximation

Applying the conditional matching results to our problem with $l = L, x = \{y_t, x_{t+1}\}$, and $y = y_{t+1}$, we would like to find a new variable y_{t+1} such that the two conditionals

$$\begin{aligned} p(L|y_t, x_{t+1}) &\propto \exp -\frac{1}{2} \left\| RL - S \begin{bmatrix} y_t \\ x_{t+1} \end{bmatrix} - d \right\|^2, \text{ and} \\ q(L|y_{t+1}) &\propto \exp -\frac{1}{2} \|RL - Ty_{t+1} - d\|^2 \end{aligned}$$

match with each other. Using the above result for matching Gaussian conditionals, we need to choose T and y such that Ty can generate the same linear subspace as Sx does. The condition $Sx = Ty$ gives us a hard equality constraint between $x = \{y_t, x_{t+1}\}$ and $y = y_{t+1}$:

$$Ty_{t+1} - S_1 y_t - S_2 x_{t+1} = 0. \tag{4}$$

where S_1 and S_2 are columns of S corresponding to y_t and x_{t+1} respectively. As discussed above, the trivial choice $y = x$, i.e. $y_{t+1} = \{y_t, x_{t+1}\}$ and $T = S$, increases the size of the new variables y_{t+1} at each step and is expensive for computation. Hence, we want y_{t+1} to be low-dimensional.

To maintain the low computational complexity, we enforce the dimensions of the new variables to be the same at every step, i.e. $r = \dim(y_t) = \dim(y_{t+1})$. Let $h = \dim(l_j)$, $k = \dim(x_{t+1})$, the size of S is $mh \times (r+k)$, and of T is $mh \times r$, and we assume that $mh \gg r$.

The condition for Sx and Ty to generate the same subspace can only hold if both S and T have the same rank. As Sx and Ty are the linear combinations of S's and T's columns respectively, the r columns of T must be independent vectors in the r-dimensional subspace spanned by columns of S.

We can choose T by doing a low-rank approximation on S using SVD decomposition: $S = UDV^T$, and T can be chosen from the r columns of U corresponding to the r largest singular values in D. This well-known technique guarantees the best low-rank approximation for S. The SVD decomposition of S, with size $mh \times (r+k)$, can be done in $O(mh(r+k)^2)$ time [30, Lecture 31], and because $h(r+k)^2$ is a constant, it is linear in the number of features m.

We also experiment with a much faster approximation for T by simply choosing r independent columns from S to be the columns of T. If S has exactly rank-r, these r-independent columns will generate the whole subspace for Sx exactly, and Ty will generate the same subspace as Sx does. Otherwise, Ty will generate an approximate subspace of Sx. Although this is not the best subspace approximation for Sx, we found it is good enough in our experiments. A better subspace approximation might be found by carefully ranking S's columns according to their pairwise dot products and choosing the columns that maximize them.

We finally select only r independent rows of T and S to form the constraint in (4), instead of using all mh rows, which is expensive since it depends on the number of landmarks. If S is not exactly rank-r, the full mh rows of the above equality constraint cannot all be satisfied, leading to an overdetermined system. On the other hand, if S is exactly rank-r, only the first r independent rows is enough to constrain the whole system.

V. Experiments

A. Simulated datasets

We first study the performance of our two proposed algorithms, the simple Tree Assumed Density Filtering (TADF) algorithm in Section III and the better low-rank approximation Incremental Tree Filtering (ITF) algorithm in Section IV, on simulated datasets reflecting the worst-case scenarios for SLAM in obstacle avoidance context.

Our datasets simulate a robot moving in 2D and observing an object with many features that it needs to avoid. A worst-case scenario in SLAM happens when the robot observes all features of the object at every time step. In this case, the full graph is densely connected, and there is no variable elimination order exists that can avoid the $O(m^3)$ time complexity for information filters and iSAM2. We note that this scenario is common in practice, for example, when

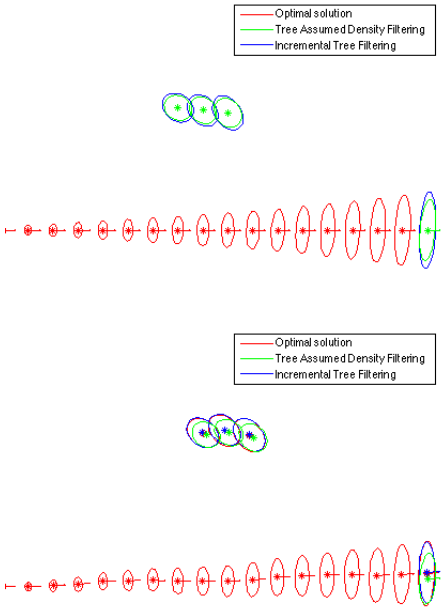


Fig. 5: Results of our tree filtering schemes compared with the optimal solution for the noise-free (top) and noisy (bottom) datasets. Three landmarks are at the top, while the robot is moving in a straight line.

a robot uses a laser-scanner to obtain a large number of data point observations at every step, or when it observes a textured object with many visual features by cameras.

We use a simple measurement model in our experiments, where we assume that the robot can observe the relative 2D position of the landmarks in its coordinate frame. Many other measurement models, for example, bearing-range sensors, or stereo cameras, can be easily transformed into this form.

We first study the accuracy of our algorithms by comparing their results with the best optimal solution obtained from solving the full graph at the last time step. We experiment with two sets of measurements, an ideal noise-free set to study the theoretical amount of information loss, and a noisy set corrupted with additive Gaussian noise. The noise-free measurement set satisfies our low-rank assumption exactly, whereas the noisy set is approximately low-rank.

Fig. 5 shows the estimation results of the two methods, TADF and ITF, compared with the optimal solutions in a simple case with three landmarks for both types of noise-free and noisy datasets. As expected, TADF estimates are inconsistent and overconfident with smaller marginal covariance ellipses over time. On the other hand, ITF achieves the exact results in the noise-free dataset, and approximates very well with the optimal solutions in the noisy one.

To better understand their performances, we compare the KL-divergence of the approximate densities estimated by our methods with the optimal densities. Fig. 6 plots the KL-divergence results. ITF achieves the exact densities with zero KL-divergence in the noise-free dataset, so we only report results in the noisy case. Whereas TADF accumulates its

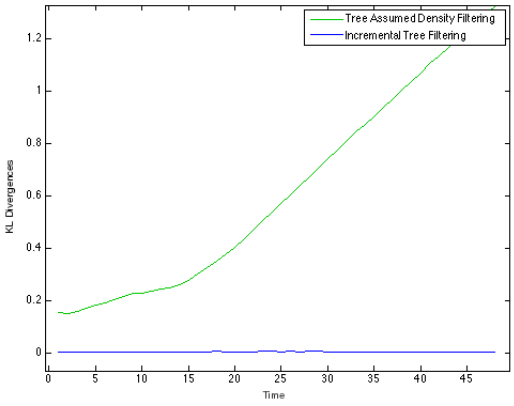


Fig. 6: Comparing the KL-divergences of TADF (green) and ITF (blue) with respect to the optimal densities over time.

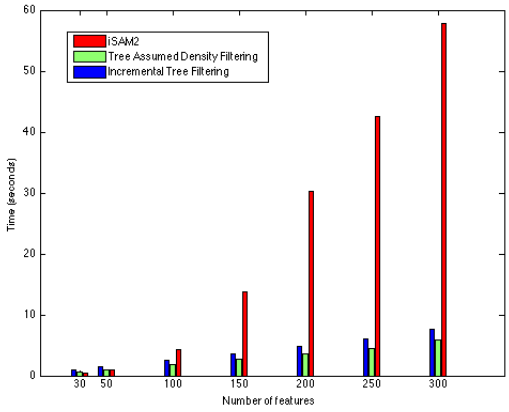


Fig. 7: Timing comparison among iSAM2, TADF and ITF.

approximation errors, ITF's errors are very small and do not increase over time.

To study the time complexity of TADF and ITF, we compare their speeds with iSAM2 [4] using a series of datasets with 100 poses and increasing numbers of landmark features from 30 to 300. As clearly shown in Fig. 7, the processing time of our methods is linear, whereas iSAM2's processing time grows in a polynomial order with respect to the number of landmarks. Especially, with 300 features every frame, our methods are 10 time faster than iSAM2.

The memory requirements for our methods are much cheaper compared to iSAM2. For the same dataset with 100 poses and 300 landmarks, our implementation requires only around 13MB for ITF and 9MB for TADF, whereas iSAM2 needs almost 500MB. We notice that this is a bias comparison, however, since iSAM2 retains the full graph with all the past poses in the memory whereas our filtering schemes marginalize them out. Nevertheless, this reflects the fact that our methods are more ready than iSAM2 for small robots with limited memory capacity.

We also test our methods in more challenging scenarios. As shown in Fig. 8, we replicate a real RC-car racing track

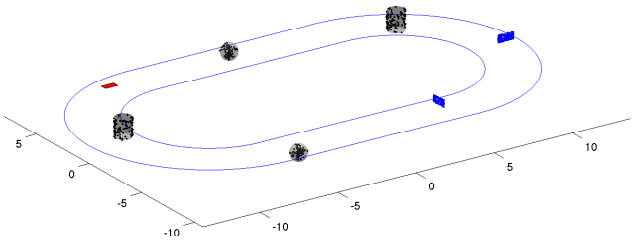


Fig. 8: A simulated RC car (red) and different 3D objects

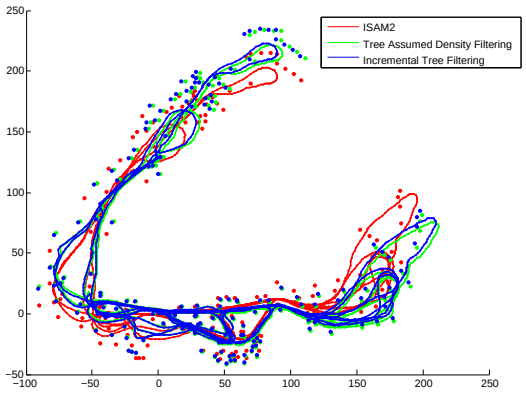


Fig. 10: Results on Victoria Park dataset

(~30m×16m) in simulation, and consider three types of 3D objects with different structures: planar panels, cylinder barrels and transparent spheres, each of which has 100 randomly generated features. In the experiments in Fig. 9, we consider only one object at a specific location, and assume the car can measure the relative 3D position of each feature in its local coordinate frame. We also conduct experiments for both noise-free and noisy measurements, assuming zero mean Gaussian noise with 0.1m standard deviation in all x,y and z. In the noise-free experiments, ITF has no information loss. Hence, we only report results for the noisy cases in Fig. 9. Each column of Fig. 9 shows the top view of the test scenarios with the trajectory and the tested object, and plots the KL-divergence results of TADF and ITF. As can be seen, ITF is has less information loss than TADF in these cases.

B. Victoria Park dataset

We next study the performance of our methods on the well-known SLAM Victoria Park dataset. This dataset does not reflect the scenarios we assume in this paper, i.e. for short term obstacle avoidance applications instead of exact map building. However, the results of our filtering methods still approximate well with the full optimal solution obtained from iSAM2 as shown in Fig. 10. Moreover, whereas we assume a dense graph with many landmark observations on the same object at each time step, Victoria Park dataset is very sparse with only a few landmark measurements, one per object, at each time. This sparsity also breaks the low-rank assumption of our ITF method that all landmarks should be observed from each robot pose in *the same way*, because at each pose, only a few landmarks are observed and the rest are not. Consequently, a good low-rank approximation of the trajectory to generate the space of the landmark conditional means does not exist. In those cases, the results of ITF are similar to TADF's as shown in Fig. 10.

VI. Conclusion

We have presented two tree filtering methods which significantly improve upon the speed of the traditional filtering schemes in worst-case scenarios. Our methods achieve linear-time complexity $O(m)$ with respect to the number of landmarks m, whereas traditional methods EKF and information filters take $O(m^2)$ and $O(m^3)$ time respectively, due to the dense correlations of landmarks resulted from marginalizing out old robot poses. Hence, our methods are suitable for small robots with limited memory and processing power.

Our key idea to avoid the problem of dense cliques in filtering-based SLAM is to maintain an approximate tree structure of the full density at every time step by finding new low-dimensional variables to de-correlate them. Intuitively, these new variables capture the essential information of the entire robot trajectory such that given them the landmarks are conditionally independent of each other and the conditional densities of the landmarks are best approximated. We use techniques from tree assumed density filtering and low-rank approximation to keep the size of the new variables small and achieve linear-time updates at every step.

There are several important questions that need to be further addressed in our future work to gain more insights about the methods. First, our low-rank approximation scheme is based on the assumption that the conditional means of landmarks given the trajectory lie in a low-dimensional subspace. Although our experiments show cases where this assumption is valid, further studies need to be done to understand when this assumption can be applied. Obviously, it depends on the object structure as well as the measurement models of the sensors. Another related questions are how good the approximation is when this low-rank assumption is violated and what the optimal choice for the dimension of the new variables is to capture enough essential information. This parameter is a trade-off between performance and accuracy. Our future work will also focus on implementing and studying these tree filtering schemes on real robot platforms with different sensors and measurement models, and further improving on limitations of the low-rank assumption.

Acknowledgements

Duy-Nguyen Ta and Frank Dellaert are supported by an ARO MURI grant, award number W911NF-11-1-0046.

References

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robotics & Automation Magazine*, Jun 2006.

[2] T. Bailey and H. Durrant-Whyte, "Simultaneous localisation and mapping (SLAM): Part II state of the art," *Robotics & Automation Magazine*, Sep 2006.

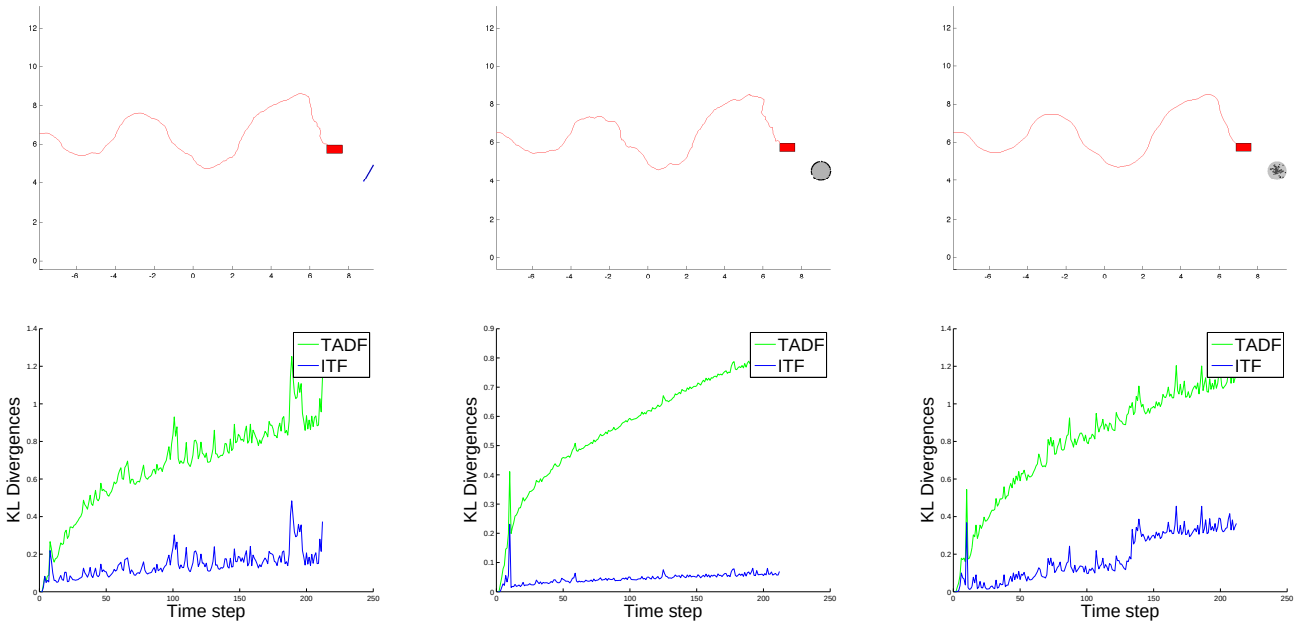


Fig. 9: KL-divergence results of TADF (green) and ITF (blue) with a complicated trajectory and different object structures

[3] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1365–1378, Dec 2008.

[4] 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*, vol. 31, pp. 217–236, Feb 2012.

[5] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[6] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *Intl. J. of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, 2004.

[7] R. Eustice, M. Walter, and J. Leonard, "Sparse extended information filters: Insights into sparsification," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Aug 2005, pp. 3281–3288.

[8] S. Julier, "A sparse weight Kalman filter approach to simultaneous localisation and map building," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. 3, 2001, pp. 1251 – 1256.

[9] M. Paskin, "Thin junction tree filters for simultaneous localization and mapping," in *Intl. Joint Conf. on AI (IJCAI)*, 2003.

[10] R. Eustice, H. Singh, and J. Leonard, "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Trans. Robotics*, vol. 22, no. 6, pp. 1100–1114, Dec 2006.

[11] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based SLAM," *Intl. J. of Robotics Research*, vol. 26, no. 4, pp. 335–359, April 2007.

[12] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact Pose SLAM," *IEEE Trans. Robotics*, vol. 26, no. 1, 2010, In press. [Online]. Available: http://dx.doi.org/10.1109/TRO.2009.2034435

[13] N. Carlevaris-Bianco and R. M. Eustice, "Generic factor-based node marginalization and edge sparsification for pose-graph slam," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5748–5755.

[14] G. Huang, M. Kaess, and J. Leonard, "Consistent sparsification for graph optimization," in *Proc. of the European Conference on Mobile Robots (ECMR)*, 2012.

[15] F. Dellaert, J. Carlson, V. Ila, K. Ni, and C. Thorpe, "Subgraph-preconditioned conjugate gradient for large scale slam," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[16] Y.-D. Jian, D. Balcan, and F. Dellaert, "Generalized subgraph preconditioners for large-scale bundle adjustment," in *Intl. Conf. on Computer Vision (ICCV)*, 2011.

[17] Y.-D. Jian, D. Balcan, I. Panageas, P. Tetali, and F. Dellaert, "Support-theoretic subgraph preconditioners for large-scale slam," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 9–16.

[18] M. Csorba and H. F. Durrant-Whyte, "New approach to map building using relative position estimates," pp. 115–125, 1997. [Online]. Available: http://dx.doi.org/10.1117/12.277214

[19] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2006, pp. 2262–2269.

[20] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Robotics: Science and Systems (RSS)*, Jun 2007.

[21] G. Sibley, C. Mei, I. Reid, and P. Newman, "Adaptive relative bundle adjustment," in *Robotics: Science and Systems (RSS)*, 2009.

[22] ——, "Vast scale outdoor navigation using adaptive relative bundle adjustment," *Intl. J. of Robotics Research*, vol. 29, no. 8, pp. 958–980, 2010.

[23] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, Dec 2006.

[24] T. Minka, "A family of algorithms for approximate bayesian inference," Ph.D. dissertation, MIT Media Lab, MIT, 2001.

[25] T. P. Minka, "Expectation propagation for approximate Bayesian inference," in *Proc.* 17^{th} *Conf. on Uncertainty in AI (UAI)*, Seattle, WA, August 2001, pp. 362–369.

[26] T. Minka and Y. Qi, "Tree-structured approximations by expectation propagation," *Advances in Neural Information Processing Systems (NIPS)*, vol. 16, p. 193, 2004.

[27] F. R. Bach and M. I. Jordan, "Beyond independent components: trees and clusters," *The Journal of Machine Learning Research*, vol. 4, pp. 1205–1233, 2003.

[28] D. Steedly, I. Essa, and F. Dellaert, "Spectral partitioning for structure from motion," in *Intl. Conf. on Computer Vision (ICCV)*, 2003.

[29] E. D. Nerurkar and S. I. Roumeliotis, "Power-slam: a linear-complexity, anytime algorithm for slam," *The International Journal of Robotics Research*, vol. 30, no. 6, pp. 772–788, 2011.

[30] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.

[31] J. Folkesson, P. Jensfelt, and H. Christensen, "Graphical SLAM using vision and the measurement subspace," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Aug 2005.

[32] C. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Secaucus, NJ, USA: Springer-Verlag, 2006.