Using Treemap as a Generic Least Square Backend for 6-DOF SLAM

Udo Frese

FB 3 Mathematik und Informatik, SFB/TR 8 Spatial Cognition,
Universität Bremen,

Abstract. Treemap is a generic SLAM algorithm that has been successfully used to estimate extremely large 2D maps closing a loop over a million landmarks in 442ms. We are currently working on an open-source implementation that can handle most variants of SLAM. In this paper we show initial results demonstrating 6-DOF feature based SLAM and closing a simulated loop over 106657 3D features in 209ms.

1 Introduction

Simultaneous Localization and Mapping (SLAM) has been a central topic in mobile robotics research for almost two decades by now [1]. Most of the literature is concerned with generating a 2D map with a sensor moving in the plane (3-DOF). Only in the last few years the problem of generating a 3-D map with a sensor moving in 3D space (6-DOF) has received considerable attention [2–5]. Such a system has important applications, for instance rescuing victims from the remains of a collapsed building. So we expect that 6-DOF SLAM will be a growing research area, in particular with the recently emerging 3D cameras.

Many 2D SLAM articles have been concerned with efficiency in estimating large maps ([6] for an overview). In 6-DOF SLAM the efficiency discussion has mainly focused on the first stages of processing in particular on 3D scan matching [7]. 3D maps always contain a lot of data but up to now little attention has been paid to 3D maps, that are by magnitudes larger than the sensor range.

We contributed the treemap algorithm [8] to this discussion in 2D SLAM. It is designed for computing least square estimates for very large maps efficiently. Using treemap we were able to demonstrate closing a simulated loop with one million landmarks in 442ms [9]. On the one hand, the treemap algorithm is sophisticated but also complicated. On the other hand, it is fairly general mainly estimating random varianbles of arbitrary meaning. Hence our current project is to develop an open source implementation that – as an *implementation* – can be used to perform most variants of SLAM including 2D, 3D, features and/or poses, and visual SLAM. This workshop paper reports intermediate results showing a simulated 6-DOF SLAM experiment (3D features, no odometry) that uses the same implementation as our previous million-landmarks (2D features, odometry, marginalized poses) experiment. By building on the efficiency of treemap as a backend, we where able to close a loop over $n = 106657$ 3D features in 209ms.

2 Local and Global Challenges for 6-DOF SLAM

Many challenges currently addressed in 6-DOF SLAM concern the first stages of sensor processing: matching 3D scans, finding reliable features, matching them, rejecting outliers, filtering range images or handling bearing-only initialization. These problems are local in the sense that they affect only that part of the map that is currently observed by the sensor. In contrast there is also the question how this local information and its uncertainty affects the global map. The most prominent situation is certainly closing a loop when the local information that closes the loop leads to back-propagation of the error along the loop. The key point, as we have argued in [6, §12], is that the local uncertainty is small but complex and depends on the actual sensor and the detailed circumstances of observation, whereas the global uncertainty is mostly the composition of local uncertainties, i.e. it is large, rather simple and dominated by the map's geometry.

This insight motivates our treemap approach. In the past it has motivated the design of the treemap algorithm itself that exploits this locality. And now it motivates our idea that many different SLAM variants (2D / 3D, features and/or poses, with/without odometry) can be solved by a specific local preprocessing plus treemap as a global least-square backend. From this perspective a large map is mainly a matter of computation time and hence our goal is:

> Whenever you can formulate your SLAM approach in a least-square framework such that it works for small maps, you can use treemap as a backend to make it work for large maps.

The following section gives the overall idea of the treemap algorithm from a general probabilistic perspective. A more extensive presentation as well as the concrete Gaussian implementation can be found in [9, 8].

3 The Treemap Algorithm

Imagine the robot is in a building that is virtually divided into two parts A and B. Now consider: *If the robot is in part A, what is the information needed about B?* Only few of B's features are involved in observations with the robot in A. All other features are not needed to integrate these observations. So probabilistically speaking, the information needed about B is only the marginal distribution of features of B also observed from A conditioned on observations in B.

The idea can be applied recursively by dividing the building into a binary tree of regions and passing probability distributions along the tree (Fig. 1). The input to treemap are observations assigned to leaves of the tree. They are modeled as distributions $p(X|z_i)$ of the state vector X, i.e. of feature positions and robot poses, given some measurement z_i. With respect to the motivating idea, nodes define local regions and super-regions. Formally, however, a node $\mathbf{n}$ just represents the set of observations assigned to leaves below $\mathbf{n}$ without any explicit geometric definition. During the computation, intermediate distributions $p^M_{\mathbf{n}}$ and $p^C_{\mathbf{n}}$ are passed through the tree and stored at the nodes, respectively.

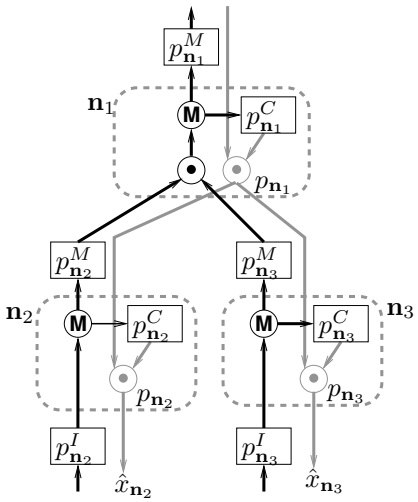

Fig. 1. Data flow view of the probabilistic computations performed by treemap. The leaves store the input $p^I_{\mathbf{n}}$. During updates (black arrows), a node $\mathbf{n}$ integrates ($\odot$) the distributions $p^M_{\mathbf{n}_\swarrow}$ and $p^M_{\mathbf{n}_\searrow}$ passed by its children. The result is factorized (Ⓜ) into a marginal $p^M_{\mathbf{n}}$ passed up and a conditional $p^C_{\mathbf{n}}$ stored at $\mathbf{n}$. To compute an estimate (gray arrows), each node $\mathbf{n}$ receives a distribution $p_{\mathbf{n}_\uparrow}$ from its parent, integrates ($\odot$) it with the conditional $p^C_{\mathbf{n}}$, and passes the result $p_{\mathbf{n}}$ down. In the end, estimates $\hat{x}_{\mathbf{n}}$ are available at the leaves.

A feature is passed in distributions $p^M_{\mathbf{n}}$ from the leaves where it is involved, up to the least common ancestor of all these leaves. There it is marginalized out and finally stored in $p^C_{\mathbf{n}}$. So the distribution $p^M_{\mathbf{n}}$ passed by a node contains those features involved in leaves below $\mathbf{n}$ but also involved above $\mathbf{n}$. An estimate is computed recursively down the tree. A node receives an estimate for the features in $p^M_{\mathbf{n}}$ and combines it with the conditional $P^C_{\mathbf{n}}$ stored at $\mathbf{n}$ to compute an estimate for the features marginalized out there which is in turn passed down.

Figure 2 shows the Bayesian justification for this approach: For each node $\mathbf{n}$ the features $X[\mathbf{n}\colon \not\downarrow\uparrow]$ and measurements $z[\mathbf{n}\colon \not\downarrow\uparrow]$ only above $\mathbf{n}$ are conditionally independent from the features $X[\mathbf{n}\colon \downarrow\not\uparrow]$ and measurements $z[\mathbf{n}\colon \downarrow\not\uparrow]$ only below $\mathbf{n}$ given the features $X[\mathbf{n}\colon \downarrow\uparrow]$ involved at the same time above and below $\mathbf{n}$.

It should be noted, that the process of passing distributions along the tree is exact. The only approximations are linearization when computing the input $p^I_{\mathbf{n}}$ and sparsification needed when old robot poses are marginalized out.

4 Different Variants that could be Supported

In 2D SLAM there are mostly two variants used. The first is *consistent pose estimation* where 3-DOF poses are estimated from 3-DOF links derived from odometry and scan matching. The second is the classical variant with 2D point features (sometimes also 2-DOF lines) and 3-DOF poses where old poses are marginalized out. This requires *sparsification*, an additional approximation to preserve locality during marginalization. We used this variant in our million-landmarks experiment.

In 6-DOF SLAM more variants are possible. Using 3D scan matching [2] one can perform 6-DOF consistent pose estimation. In contrast to 2D SLAM usually no odometry is available. This gives rise to a very simple variant, where poses are marginalized out immediately. Since there is not odometry, sparsity is

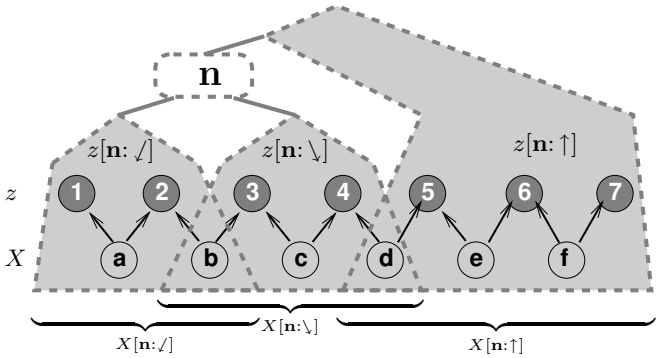

Fig. 2. Bayesian View. In this example, observations $z_{1...7}$ provide information about the features $X_{\mathsf{a}...\mathsf{f}}$. The arrows and circles show this probabilistic input as a Bayes net with observed nodes in gray. The dashed outlines illustrate the view of a single node $\mathbf{n}$. It divides the tree into three parts, *left-below* $\swarrow$, *right-below* $\searrow$ and *above* $\uparrow$. Hence the observations z are disjointly divided into $z[\mathbf{n}{:}\swarrow] = z_{1...2}$, $z[\mathbf{n}{:}\searrow] = z_{3...4}$ and $z[\mathbf{n}{:}\uparrow] = z_{5...7}$. The corresponding features $x[\mathbf{n}{:}\swarrow] = X_{\mathsf{a}...\mathsf{b}}$, $x[\mathbf{n}{:}\searrow] = X_{\mathsf{b}...\mathsf{d}}$ and $x[\mathbf{n}{:}\uparrow] = X_{\mathsf{d}...\mathsf{f}}$ however overlap ($X[\mathbf{n}{:}\swarrow\searrow] = X_{\mathsf{b}}$, $X[\mathbf{n}{:}\uparrow\searrow] = X_{\mathsf{d}}$). The key insight is that $X[\mathbf{n}{:}\downarrow\uparrow] = X[\mathbf{n}{:}\swarrow\uparrow \vee \searrow\uparrow] = X_{\mathsf{d}}$ separates the observations $z[\mathbf{n}{:}\downarrow]$ and features $X[\mathbf{n}{:}\downarrow\not\uparrow]$ below $\mathbf{n}$ from the observations $z[\mathbf{n}{:}\uparrow]$ and features $X[\mathbf{n}{:}\not\downarrow\uparrow]$ above $\mathbf{n}$, so both are conditionally independent given $X[\mathbf{n}{:}\downarrow\uparrow]$. The notation follows [8].

maintained and no sparsification is needed. Essentially, this means, that each set of observations is converted into relative information on the involved 3D point features. This variant is presented in the experiments here.

It has a major limiation. Without odometry a small sensor blackout or too little overlap between observations will disintegrate the map because no information links the involved two poses anymore. Inertial sensors can help by providing relative orientation (gyros) and absolute inclination (accelerometers). This is the pendant of classic SLAM with 3D point features and 6-DOF poses marginalized out later. Still, with orientation-odometry only, consecutive observations must share one feature. Yet another variant uses the accelerometers as translation-odometry. But when acceleration is integrated the result is relative velocity not relative position, so the poses must be augmented by 3D velocity (9-DOF total).

With a monocular camera [4], no distance can be measured. So, while consistent pose estimation can use the 5-DOF links arising from matching two images [10], additional information is needed for the overall scale. In a feature based approach this leads to the corresponding problem of bearing-only initialization.

5 Towards an Open Source Implementation

We believe that these different variants can all be based on treemap as the same least square backend. The main difference lies in the size of the vectors representing features and poses, in different Jacobians for linearization, and in the way an initial estimate can be computed for linearization. Another difference

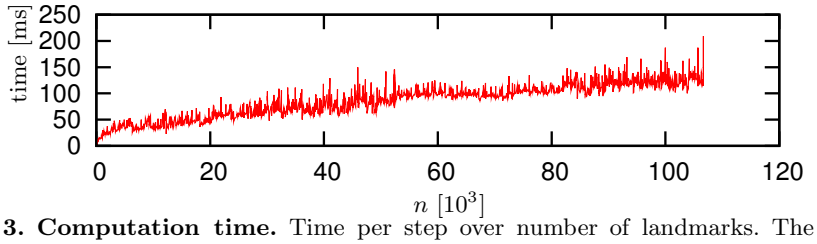

Fig. 3. Computation time. Time per step over number of landmarks. The final increase in computation time happens after closing the loop.

is the control policy: Which observations are combined in one leaf? Are old poses marginalized out? When is sparsification used for the sake of efficiency? When are Jacobians recomputed from the original nonlinear observations?

All 6-DOF SLAM variants share the problem of parameterizing 3D orientation. We extend a technique by Castellanos [11] to 3D and use the product

$$Q = Q_0 \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \approx Q_0 \begin{pmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{pmatrix}$$

of a fixed orientation Q_0 and three Euler rotations the angles of which are the random variables estimated. Q_0 is initialized with the current estimate so the Euler angles only parameterize the small *perturbation* of the orientation and are far from singularity. Hence they are always linearized at $\alpha = \beta = \gamma = 0$ and the linearization has the simple form shown above.

This technique also allows to reduce the linearization error caused by error in the robot orientation. The distributions passed from a nodes children are rotated according to the current estimate before multiplying them (Fig. 1, $\odot$) [8].

The goal of our current project is to implement all the different SLAM variants. The treemap backend is already finished with a sufficiently generic interface so we were able to implement a driver for feature based 2D SLAM [9] in 2100 lines of C++ code and a driver for feature based 6-DOF SLAM without odometry in 1200 lines of code. The following section shows results for the latter.

6 Experimental Results

In our simulated experiment the robot moves through a 20 story building with features on the rooms walls (Fig. 4). Then it crosses a bridge on the 19^{th} floor into another 20 story building and maps that building too. Finally it returns to the starting position and closes a loop over all feature. The overall map has $n = 106657$ features and $m = 5319956$ observations from $p = 488289$ poses. Poses are not represented in the map. Computation time was at most 209ms (Fig. 3).

7 Conclusion and Outlook

We have demonstrated that that the treemap algorithm in the same generic implementation can be used to solve both 2D and 3D feature based SLAM

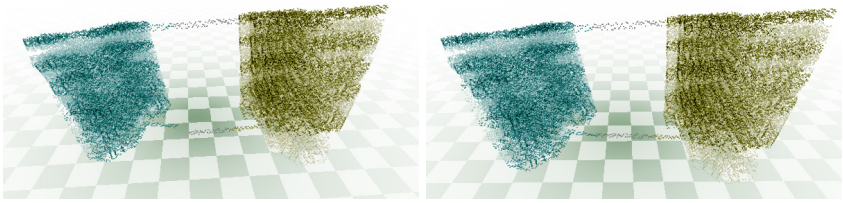

Fig. 4. 6-DOF SLAM map. before and after closing the large loop (between the two building on the ground level) over all $n = 106657$ features. A 3D animations of the growing 3D map and the previous 2D million-landmarks simulation can be downloaded from our web site `www.informatik.uni-bremen.de/~ufrese/`.

(without odometry) with high efficiency. Future work includes implementing the remaining SLAM variants, integrating a solution to the bearing-only initialization problem and implementing a 3D variant of the rotation technique used to reduce linearization error.

We then plan to publish the implementation as an open source library.

References

1. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005)
2. Surmann, H., Nüchter, A., Lingemann, K., Hertzberg, J.: 6D SLAM - preliminary report on closing the loop in six dimensions. In: Proceedings of the 5th Symposium on Intelligent Autonomous Vehicles, Lissabon. (2004)
3. Miro, J.V., Dissanayake, G., Zhou, W.: Vision-based SLAM using natural features in indoor environments. In: Proceedings of the 2005 IEEE International Conference on Intelligent Networks, Sensor Networks and Information Processing. (2005)
4. Davison, A., Cid, Y., Kita, N.: Real time SLAM with wide angle. In: Proc. IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon. (2004)
5. Ohno, K., Tadokoro, S.: Dense 3D map building based on LRF data and color image fusion. In: Proceedings of the International Conference on Intelligent Robots and Systems. (2005) 1774–1779
6. Frese, U.: A discussion of simultaneous localization and mapping. Autonomous Robots **20**(1) (2006) 25–42
7. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling, Quebec City. (2001) 145 – 152
8. Frese, U.: Treemap: An $O(\log n)$ algorithm for indoor simultaneous localization and mapping. Autonomous Robots (2006) to appear.
9. Frese, U., Schröder, L.: Closing a million-landmarks loop. In: Proceedings of the IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems, Beijing. (2006)
10. Eustice, R., Singh, H., Leonard, J., Walter, M., Ballard, R.: Visually navigating the rms titanic with slam information filters. In: Proceedings of Robotics Science and Systems, Boston. (2005)
11. Castellanos, J., Montiel, J., Neira, J., Tardós, J.: The SPmap: A probablistic framework for simultaneous localization and map building. IEEE Transactions on Robotics and Automation **15**(5) (1999) 948 – 952