Exploiting the Separable Structure of SLAM

Kasra Khosoussi, Shoudong Huang, Gamini Dissanayake
Centre for Autonomous Systems, University of Technology Sydney
{kasra.khosoussi, shoudong.huang, gamini.dissanayake}@uts.edu.au

Abstract—In this paper we point out an overlooked structure of SLAM that distinguishes it from a generic nonlinear least squares problem. The measurement function in most common forms of SLAM is linear with respect to robot and features’ positions. Therefore, given an estimate for robot orientation, the conditionally optimal estimate for the rest of state variables can be easily obtained by solving a sparse linear-Gaussian estimation problem. We propose an algorithm to exploit this intrinsic property of SLAM by stripping the problem down to its nonlinear core, while maintaining its natural sparsity. Our algorithm can be used together with any Newton-based iterative solver and is applicable to 2D/3D pose-graph and feature-based problems. Our results suggest that iterating solving the nonlinear core of SLAM leads to a fast and reliable convergence as compared to the state-of-the-art back-ends.

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) has been investigated for more than two decades [11]. Mathematically, SLAM is modeled as a high-dimensional nonlinear estimation problem whose goal is to find the “optimal” estimate for robot poses and map using noisy measurements and uncertain priors. The first successful solutions, posed SLAM as a filtering problem [8]. It was revealed later that the smoothing formulation brings not only accuracy, but also scalability, through exploiting the sparse structure of SLAM. Perhaps the most distinctive property of modern SLAM algorithms is the exploitation of this natural sparsity [7].

Under the assumption of Gaussian noise and Gaussian prior, finding the maximum likelihood (ML) and maximum a posteriori (MAP) estimate in the smoothing form of SLAM requires solving a nonlinear least squares (NLS) problem. Given a sufficiently “good” initial guess, this problem can be solved by employing iterative schemes such as Gauss-Newton (GN). In each iteration of GN, the measurement function is approximated by its first-order Taylor expansion around the current estimate, followed by solving the resulting linear least squares problem. GN and many other iterative solvers treat the measurement function as a generic smooth nonlinear function of states. However, the widely-used measurement model associated to range-bearing sensors in 2D/3D feature-based/pose-graph SLAM has a significant structure. This measurement function is linear in robot and landmarks’ positions. Therefore given the robot orientation θ, the optimal choice in ML or MAP sense for positions p can be obtained by solving a sparse linear least squares problem.

In the least squares literature, NLS problems with partially linear residuals are called separable [2, 13]. In statistical terms, such problems are sometimes called conditionally linear-Gaussian state space models [9] since estimating the linear variables, given the nonlinear ones and measurements corrupted by Gaussian noise, corresponds to a linear-Gaussian system.

Golub and Pereyra developed the original Variable Projection (VP) algorithm to solve general separable NLS problems [12]. The main idea behind VP is to explicitly eliminate the linear variables and solve the reduced NLS problem. This technique has been applied to a wide range of applications. Both theoretically [24] and empirically it has been shown that compared to solving the full NLS problem, variable projection techniques exhibit faster or equal convergence rate (see [13] and references therein).

In this paper we show how different variable projection algorithms can be applied to various forms of SLAM without any restrictive assumption. Unfortunately these algorithms, in their existing forms, are incapable of maintaining the sparse structure of SLAM, and therefore lead to solutions with cubic time complexity. To address this issue we propose an equivalent variable projection algorithm that is capable of exploiting both structures (i.e., separability and sparsity) at the same time.

Notation

Throughout this paper bold lower-case and upper-case letters are reserved for vectors and matrices, respectively. Sets are shown by upper-case letters. |X| denotes the cardinality of set X. I and 0 denote the identity and zero matrices with appropriate sizes, respectively. Let vec(q1,...,qn) denote the column vector obtained by stacking q1’s. S1 ⊗ S2 means S1 − S2 is positive-definite. Kronecker product is denoted by ⊗. Euclidean norm is denoted by ∥·∥. The weighted Euclidean norm of vector e with matrix W > 0 is denoted by ∥e∥W ≜ √e⊤We. Finally diag(W1,...,Wk) is the block-diagonal matrix with matrices W1,...,Wk as blocks on its main diagonal.

II. PROBLEM FORMULATION AND PRELIMINARIES

A. Graphical Representation of SLAM

SLAM can naturally be represented by a simple directed graph $G = (\mathcal{V}, \mathcal{E})$. Each state variable is mapped into a vertex $x_i \in \mathcal{V}$, and each relative measurement is represented by an edge $e_k \triangleq (i_k,j_k) \in \mathcal{E}$. Here $(i,j) \in \mathcal{E}$ corresponds to the relative measurement from $x_i$ to $x_j$. Suppose $|\mathcal{V}| = n + 1$ and $|\mathcal{E}| = m$. For pose-graphs we have $n = n_p - 1$ in which $n_p$ denotes the number of poses. Similarly, in feature-based SLAM with $n_f$ features we have $n = n_f + n_p - 1$. Due to
the relative nature of measurements in SLAM and without an informative prior, we must define a global coordinates system and anchor one of the nodes to it. Without loss of generality we can assume \( x_0 \) is the origin of our global coordinates system. The reduced incidence matrix of \( G \) after anchoring \( x_0 \) to the origin (i.e., deleting the corresponding row from the original incidence matrix) is denoted by \( A \in \{−1,0,1\}^{n \times m} \). For the \( k \)th edge \( e_k = (i_k, j_k) \in E \) we have \( A_{i_k,k} = 1 \) and \( A_{j_k,k} = −1 \). The remaining elements of \( A \) are all zero. Similarly, \( A_\circ \in \{−1,0,1\}^{n \times n} \) is the reduced incidence matrix of the subgraph of \( G \) consisting only of robot poses and odometry measurements. Note that \( A_\circ \) has a fixed structure and it is uniquely determined by the number of poses \( n_p \). Finally we define \( A_\ell \defeq A \otimes I_{\ell} \) for \( \ell \in \mathbb{Z}_{\geq 2} \).

B. Measurement Model

The conventional state vector is usually defined as \( \text{vec}(x_1, \ldots, x_n) \). For reasons that will become clear shortly, we permute the standard state vector and define our state vector as \( x \defeq \text{vec}(p, \theta) \). In 2D SLAM, \( p \in \mathbb{R}^2 \) is the vector of \( x \) and \( y \) coordinates of robot poses and landmark positions, and \( \theta \in [−\pi, \pi)^{n_p-1} \) is the vector of robot orientations. Each observation \( z_{ij} \) (from node \( i \) to node \( j \)) is corrupted by an independently drawn additive Gaussian noise \( \epsilon_{ij} \sim \mathcal{N}(0, \Sigma_{ij}) \),

\[
z_{ij} = h_{ij}(x_i, x_j) + \epsilon_{ij}. \tag{1}
\]

The measurement function \( h_{ij}(:, :) \) for any \( (i,j) \in E \) has the following form,

\[
h_{ij}(x_i, x_j) = \begin{bmatrix} \delta x_{ij} \\ \delta y_{ij} \end{bmatrix} = R(\theta_i)\top(p_j − p_i) \quad \text{if } j \in S_f,
\]

\[
h_{ij}(x_i, x_j) = \begin{bmatrix} \delta x_{ij} \\ \delta y_{ij} \end{bmatrix} = \begin{bmatrix} R(\theta_i)\top(p_j − p_i) \\ \text{wrap}(\theta_j − \theta_i) \end{bmatrix} \quad \text{if } j \in S_p.
\]

where \( R(\theta_i) \in \text{SO}(2) \)\(^1\) is the rotation matrix corresponding to \( \theta_i \), \( \text{wrap} : \mathbb{R} \rightarrow [−\pi, \pi) \) maps its argument to the equivalent angle in \([−\pi, \pi) \) and \( S_p \) and \( S_f \) are the disjoint sets of indices of robot poses and features, respectively. Let us define

\[
R_\theta \defeq \text{diag}(R(\theta_{k_1}), \ldots, R(\theta_{k_n})).
\]

(3)

Here \( k_p \) and \( k_\theta \) is the index of robot pose making the \( i \)th observation. \( z_p \) and \( z_\theta \) denote the stacked vector of \( \delta x_{ij} \) and \( \delta y_{ij} \), and \( \delta \theta_{ij} \) measurements, respectively. We permute the measurement vector accordingly to obtain \( z \defeq \text{vec}(z_p, z_\theta) \). Similarly, the stacked vector of noise variables and its covariance matrix are denoted by \( \epsilon \defeq \text{vec}(\epsilon_p, \epsilon_\theta) \) and \( \Sigma \), respectively. Therefore the measurement model can be expressed in a compact form by

\[
z = h(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \Sigma), \tag{4}
\]

\[
p(z|x) = \mathcal{N}(z; h(x), \Sigma). \tag{5}
\]

\(^1\)Special orthogonal group.

\(^2\)To simplify our notations we denote random variables (e.g., \( z \)) and their realisation in the same way.

\[\text{Remark 1.} \text{ Note that (2) admits both pose-graph and feature-based SLAM problems as special cases. In pose-graph SLAM } S_f = \emptyset, \text{ while in feature-based SLAM relative pose measurements are limited to odometry measurements.}\]

According to (2), the stacked measurement function of planar SLAM is given by

\[
h(x) = H(\theta)x \defeq \begin{bmatrix} R_\theta \top A_2 & 0 \\ 0 & \Lambda \end{bmatrix} x,
\]

\[
A = \begin{cases} A \text{ pose-graph,} \\ A_\circ \text{ feature-based.} \end{cases}
\]

Here we assume the correct regularization terms are applied to the measurements [4]. In order to avoid redundancy in our formulas we use this unified measurement function (6) in the rest of the paper. Note that (6) can be rewritten as

\[
h_1(x) = H_1(\theta)p + H_2(\theta)θ \text{ in which,}
\]

\[
H_1(\theta) \defeq \begin{bmatrix} R_\theta \top A_2 \\ 0 \end{bmatrix}, \quad H_2 \defeq \begin{bmatrix} 0 \\ \Lambda \end{bmatrix}.
\]

(7)

It is important to note that unlike \( H_2 \), \( H_1(\theta) \) depends on \( \theta \). Nevertheless, we drop the argument of \( H_1 \) for the sake of simplicity of our notation.

\[\text{Remark 2.} \text{ For the purpose of this paper, the measurement model in 3D SLAM with SE(3)\(^3\) measurements is fairly similar to that of planar SLAM. With a little abuse of notation, the measurement function for 3D SLAM can be expressed as } \]

\[
h_3(x) = H_3(\theta)p + H_4(\theta)ψ(\theta) \text{ in which,}
\]

\[
H_3(\theta) \defeq \begin{bmatrix} R_\theta \top A_3 \\ 0 \end{bmatrix}, \quad H_4(\theta) \defeq \begin{bmatrix} 0 \\ \psi(\theta) \end{bmatrix}.
\]

(8)

Similar to our formulation of 2D SLAM, here \( p \) is the stacked vector of robot positions, \( \theta \) is the vector of robot orientation (parametrised by quaternions or Euler angles), \( R_\theta \) is a block-diagonal matrix composed of \( R(\theta_i) \in \text{SO}(3) \) and \( \psi(\theta) \) denotes relative robot orientations. The measurement function and the corresponding error term (see e.g., [14]) are therefore linear in \( p \). Without losing any generality, for simplicity of notation and due to space limitation here we mainly focus on 2D SLAM.

\[\text{Remark 3.} \text{ This paper investigates SLAM problems with standard SE(2) and SE(3) measurements for pose-graphs and pose-point measurements in 2D and 3D for feature-based problems. Such measurements can be obtained from range-bearing sensors after a reparameterization of original measurements (e.g., after performing scan-matching). These models have become standard choices in the past few years. However, such reparameterizations often involve nonlinear transformation of original data which could introduce some error in computing covariance matrices and even affect the validity of the assumption of additive Gaussian noise. In the context of our work, such transformations can be thought as reparameterization performed to introduce separability. Note}\]

\(^3\)Special Euclidean group.
that directly using range-bearing measurements does not lead to a separable NLS. Although here we do not consider a specific choice of sensors, the use of inertial sensors in 3D SLAM is common [19]. Therefore it is worth noting that inertial measurements does not violate the separable structure of SLAM as such measurements are linear in p (see e.g., [19]).

C. Point Estimation Criterion

In the Bayesian approach to SLAM x is modeled as a random vector with prior \( p(x) = \mathcal{N}(x; \mu_x, \Sigma_x) \). The MAP estimate is the maximizer of the posterior density (or the minimizer of its negative log),

\[
p(x|z) \propto p(z|x) \ p(x).
\]

Therefore the MAP estimate \( \hat{x}_{\text{MAP}} \) is given by:

\[
\hat{x}_{\text{MAP}} = \arg\min_x \left( \|z - h(x)\|^2_{\Sigma^{-1}} + \|x - \mu_x\|^2_{\Sigma_x^{-1}} \right).
\] (9)

In the absence of an informative prior over x,\(^4\) one may seek the maximizer of the likelihood function \( p(z|x) \) in order to obtain the ML estimate \( \hat{x}_{\text{ML}} \). Dropping the prior in (10) gives \( \hat{x}_{\text{ML}} \).

\[
\hat{x}_{\text{ML}} = \arg\min_x \|z - h(x)\|^2_{\Sigma^{-1}}.
\] (11)

In the following sections we mainly focus on \( \hat{x}_{\text{ML}} \) as it is fairly rare to have an informative prior over x in real applications. Nevertheless, our approach can be straightforwardly generalized to the Bayesian formulation (10) as well. To simplify our notation we denote the optimal estimate for any variable like c with \( c^* \). Here \( c^* \) is either \( \hat{c}_{\text{ML}} \) or \( \hat{c}_{\text{MAP}} \).

III. EXPLOITING SEPARABILITY IN SLAM

In the previous section we formulated the MAP and ML estimation problems in SLAM as non-linear least squares problems. By further inspection of (7) one can see that the non-linearity is caused by the rotation matrices. Hence given the robot orientations \( \theta \), measurements are linear in robot and features’ positions p. Therefore given \( \theta \), the ML and MAP estimates for p are obtained by solving linear least squares problems. Such problems are often called separable NLS [13]. This special structure distinguishes SLAM from the general NLS. Our main goal in this paper is to exploit this structure in order to improve the performance of the current SLAM algorithms.

A. Variable Projection

Variable projection (VP) is an algorithm proposed by Golub and Pereyra to exploit this structure [12]. They proved that under some regularity conditions, the solution of the original problem (10) or (11) (for general separable NLS problems) can be obtained using the following procedure (see [12, Theorem 2.1]). Here we explain how their approach can be applied to SLAM.

\(^4\)In this case, x is treated as the vector of unknown deterministic parameters in the measurement model.

I. Find \( p^*(\theta) \), the closed-form expression for p as a function of \( \theta \) that minimizes the original cost function in p.

II. Replace p with \( p^*(\theta) \) in the original problem and minimize the new objective function in \( \theta \) to obtain \( \theta^* \). After this step the optimal \( p^* = p^*(\theta^*) \) can be recovered instantly.

Phase I – \( p^*(\theta) \): Let us start with the symmetric positive definite square root of the noise precision matrix \( \Sigma^{-\frac{1}{2}} \succ 0 \). To simplify our notation, we express the weighted \( \ell^2 \) norm minimization in (11) as the following unweighted least squares,

\[
x^* = \arg\min_x \|\tilde{z} - \tilde{H}_1 p - \tilde{H}_2 \theta\|^2_x,
\] (12)

in which \( \tilde{z} = \Sigma^{-\frac{1}{2}} z \), \( \tilde{H}_1 = \Sigma^{-\frac{1}{2}} H_1 \) and \( \tilde{H}_2 = \Sigma^{-\frac{1}{2}} H_2 \). The fact that the reduced incidence matrix is full rank in (weakly) connected graphs results in the following lemma [17].

**Lemma 1.** In any SLAM problem with the measurement models defined in Section II (including 2D/3D feature-based or pose-graph) \( \tilde{H}_1 \) is full column rank regardless of \( \theta \).

As mentioned before, given \( \theta \), (12) is a linear least squares problem in p. Lemma 1 assures us that for any given \( \theta \), the optimal choice for p as a function of \( \theta \) is uniquely given by

\[
p^*(\theta) = \arg\min_p \|\tilde{z} - \tilde{H}_1 p - \tilde{H}_2 \theta\|^2_p
\] (13)

In which \( \tilde{H}_1^\dagger = (\tilde{H}_1^\top \tilde{H}_1)^{-1}\tilde{H}_1^\top \) is the Moore-Penrose pseudoinverse of \( \tilde{H}_1 \).

**Remark 4.** Although we do not require any special structure in \( \Sigma \), here we point out an interesting property of spherical noise covariance matrices. Let us denote the covariance matrix of the translational component of the ith measurement with \( \Sigma_p^i \). An interesting special case emerges when \( \Sigma_p^i \) is spherical, i.e., \( \Sigma_p^i = \sigma_p^2 I_2 \). Noting that \( R_0 \) is orthogonal, it is easy to show that in such SLAM problems \( \tilde{H}_1^\top \tilde{H}_1 \) is proportional to \( L_2 \equiv L \otimes I_2 = A_2 A_2^\top \) in which \( L \) is the reduced Laplacian matrix of graph \( G \). A similar structure exists in 3D SLAM problems with spherical noise covariance matrices.

Phase II – Reduced NLS: By substituting p in the original objective function (12) with \( p^*(\theta) \) in (14) and solving the resulting optimization problem we obtain \( \theta^* = \arg\min_{\theta} g(\theta) \) where

\[
g(\theta) = \| (I - \tilde{H}_1 \tilde{H}_1^\top)(\tilde{z} - \tilde{H}_2 \theta)\|^2_2.
\] (15)

Note that \( g(\cdot) \) is a function of only robot headings \( \theta \), while the original optimization problem (12) was over both p and \( \theta \). Therefore we have reduced the parameter space from \( \mathbb{R}^{2n} \times [-\pi, \pi)^{n_p-1} \) to \( [-\pi, \pi)^{n_p-1} \). \( P_\theta \equiv \tilde{H}_1 \tilde{H}_1^\top \) is its orthogonal projection onto range(\( \tilde{H}_1 \)), while \( P_\theta^\perp \equiv I - \tilde{H}_1 \tilde{H}_1^\top \) is its orthogonal complement. Let us define \( r_{vp} \equiv P_\theta^\top (\tilde{z} - \tilde{H}_2 \theta) \). In order to solve (15) using Newton-based NLS solvers we need to compute the Jacobian matrix of \( r_{vp}, i.e., J_{vp} = \frac{\partial}{\partial \theta} r_{vp} \). Computing \( J_{vp} \) requires differentiating pseudoinverses (\( \tilde{H}_1^\top \)), and therefore is more complex than computing the Jacobian.
Algorithm 1 SLAM solver based on [16]

1: repeat
2: Compute the full QR factorization of $\tilde{H}_1$ (18)
3: Recover $p^*$ by solving $R_1 \ p^*(\theta_i) = Q_1 (\tilde{z} - H_2 \theta_i)$
4: Compute the modified Jacobian matrix (19) (see [17])
5: Construct the normal equations for the reduced problem
6: Solve the normal equations to obtain $\delta \theta_i$
7: $\theta_{i+1} \leftarrow \theta_i + \delta \theta_i$
8: until convergence
9: $p^* \leftarrow p^*(\theta^*)$ according to (14) given below,

$$Q_2^T r_{vp} = Q_2^T (\tilde{z} - H_2 \theta)$$

$$[Q_2^T J_{vp}]_{i,j} = -Q_2^T \left( \frac{\partial H_1}{\partial \theta} p^*(\theta) + [H_2]_{i,j} \right).$$

IV. SPARSE VARIABLE PROJECTION

In the previous section we explained how Algorithm 1 exploits the separable structure of SLAM. At first glance, applying this algorithm to SLAM may appear to be computationally advantageous as it reduces the size of the normal equations from $(2n + n_p)$ to only $n_p$. However, the normal equations of the reduced problem in SLAM is dense in general. Hence VP, as implemented in Algorithm 1, does not lead to a scalable algorithm as we would need $O(n^3_p)$ time per iteration just for solving the resulting dense linear system. Therefore it is sensible to ask whether we can exploit separability without giving up the intrinsic sparse structure of SLAM?

Barham and Drane [1] proposed an intuitive algorithm to solve separable NLS problems. It has been shown that their algorithm and Kaufman’s modification in VP produce identical steps (assuming infinite-precision arithmetic) [22, 24]. Although Barham and Drane’s algorithm does not preserve sparsity, it can be restructured to do so as shown below. Consider the normal equations of the original problem,

$$\tilde{J}^T \delta x(i) = -\tilde{J}^T \tilde{r}.$$ (20)

Here $\delta x(i) \triangleq \text{vec}(\delta p(i), \delta \theta(i))$ denotes the $i$th GN direction, $\tilde{r}$ is the residual vector $\tilde{r} \triangleq \Sigma^{-\frac{1}{2}} r$ and $\tilde{J} \triangleq \frac{\partial}{\partial \theta^T} \tilde{r}$, both evaluated at $x(i)$. Let $U \triangleq \tilde{J}^T \tilde{J}$ be the approximated Hessian. Note that $U$ can be divided into two blocks, $\tilde{J} = [\tilde{J}_p, \tilde{J}_\theta]$ in which $\tilde{J}_p \triangleq \frac{\partial}{\partial p^T} \tilde{r}$ and $\tilde{J}_\theta \triangleq \frac{\partial}{\partial \theta^T} \tilde{r}$. Therefore we can expand (20) as

$$\begin{bmatrix} H_p & H_p, \theta \\ H_p, \theta & \theta \end{bmatrix} \begin{bmatrix} \delta p(i) \\ \delta \theta(i) \end{bmatrix} = \begin{bmatrix} -\tilde{J}_p^T \tilde{r} \\ -\tilde{J}_\theta^T \tilde{r} \end{bmatrix}.$$ (21)

One can eliminate $\delta p(i)$ from (21) to obtain a smaller linear system in terms of $\delta \theta(i)$. This can be achieved using the Schur complement of $U$ with respect to $H_p$.

$$(H_p - H_p, \theta H_p^{-1} H_p, \theta) \delta \theta(i) = \left( -\tilde{J}_\theta^T + H_p, \theta H_p^{-1} \tilde{J}_p \right) \tilde{r}.$$ (22)

Solving (22) gives us $\delta \theta(i)$. It should be clear that using this solution to recover $\delta p(i)$ from (21) leads to the standard GN direction for the original cost function. Barham and Drane proposed to ignore $\delta p(i)$ of (21) and instead compute the conditionally optimal $p_{i+1}$ given $\theta_{i+1} = \theta_i + \delta \theta(i)$ using (14) [1]. This can be achieved by solving the following sparse linear system,

$$\tilde{H}^T \tilde{H} \ p_{i+1} = \tilde{H}^T \tilde{r}.$$ (23)

Repeating this procedure until convergence leads to a sequence of steps for which the cost function (11) has zero gradient with respect to $p$ along the path. This algorithm is summarized in Algorithm 2. The equivalence between Algorithm 1 and

matrix of the original full problem, i.e., $J \triangleq \frac{\partial}{\partial p} r$ in which $r \triangleq z - h(x)$. One way to avoid this complexity is to approximate $J_{vp}$ using finite differences (see [13] and the references therein). Here we use the exact analytical expression for $J_{vp}$ which was derived by Golub and Pereyra [12].

Computing $J_{vp}$: First note that (15) has a slightly more general form compared to the case which was originally considered by [12] in that we have an additional term, linear with respect to $\theta$ in our residual, i.e., $-H_2 \theta$. We can show that the $j$th column of $J_{vp}$ is given by,

$$[J_{vp}]_{i,j} = - \left( P_{\theta} \left( \frac{\partial \tilde{H}_1}{\partial \theta} H_1 \right) + \left( P_{\theta} \left( \frac{\partial \tilde{H}_1}{\partial \theta} H_1 \right) \right)^T \right) (\tilde{z} - H_2 \theta) - P_{\theta} [H_2]_{i,j}. $$ (16)

After finding $\theta^* \triangleq \arg \min g(\theta)$ using an iterative NLS solver such as GN, we can recover the optimal $p^*$ according to (14). However, finding $\theta^*$ is computationally much more expensive than GN iterations on the full problem. Therefore directly applying VP to SLAM does not lead to an efficient solver.

B. Kaufman’s Algorithm

In [16] Kaufman proposed to approximate the $j$th column of $J_{vp}$ according to

$$[J^K_{vp}]_{i,j} \triangleq - \left( P_{\theta} \left( \frac{\partial \tilde{H}_1}{\partial \theta} H_1 \right) \right) (\tilde{z} - H_2 \theta) - P_{\theta} [H_2]_{i,j}. $$ (17)

The term ignored in $J^K_{vp}$ has been theoretically and practically shown to be negligible in small-residual problems in terms of the convergence rate of GN [16, 24]. Kaufman’s simplification reduces the time per iteration of VP up to 25% [24]. Consequently Kaufman’s method has become the standard way of approaching separable NLS problems in many fields [13]. Algorithm 1 summarizes a SLAM solver based on an efficient implementation of VP using Kaufman’s modification. This algorithm relies on the full QR decomposition of $\tilde{H}_1$,

$$\tilde{H}_1 = QR = [Q_1 \ Q_2] \begin{bmatrix} R_1 \\ 0 \end{bmatrix}, $$ (18)

in which the columns of $Q_1$ and $Q_2$ form orthonormal bases for range($\tilde{H}_1$) and null($\tilde{H}_1$), respectively. In [17] we show that the normal equations associated to the residual-Jacobian ordered pair $(\tilde{r}_{vp}, J^K_{vp})$ is identical to that of $(Q_2^T \tilde{r}_{vp}, Q_2^T J^K_{vp})$.
Algorithm 2 can be verified by expanding the normal equations of the reduced problem associated to Kaufman’s Jacobian (17). Now recall that our goal is to find a way to simultaneously benefit from the sparsity and separability of SLAM. Unfortunately, the reduced linear system (i.e., the Schur complement) in (22) is dense in general. But note that in each iteration of Algorithm 2, the solution of the reduced system (22) is identical to the $\delta \theta_{(i)}$ obtained from solving the full system (21). Therefore, instead of eliminating variables using Schur complement, the same outcome can be achieved by solving the sparse full system (21), discarding the obtained $\delta p_{(i)}$ and instead computing the conditionally optimal $p_{(i+1)}$ according to (23). Our proposed algorithm is summarized in Algorithm 3.

It is of utmost importance to note that our proposed algorithm produces (mathematically) identical steps to those of Algorithm 1 and Algorithm 2. However, unlike Algorithm 1 and Algorithm 2, we only need to solve sparse linear systems in each iteration of our method which leads to a crucial computational benefit. Also note that Algorithm 3 can be easily implemented by a simple modification of existing state-of-the-art back-ends. In fact, we only need to solve (23) using e.g., a sparse Cholesky solver. Figure 1 illustrates how our proposed algorithm works.

**Remark 5.** Note that (10) is also a separable NLS problem. Therefore Algorithm 3 can be easily modified to find the MAP estimate assuming a Gaussian prior over $x$. Nevertheless, here we address the Bayesian formulation from a slightly different angle that gives us new insights into the structure of SLAM. The posterior density $p(p, \theta | z)$ can be factored according to

$$
p(p, \theta | z) = p(\theta | z) p(p | \theta, z). \quad (24)
$$

Using the Bayes rule we have $p(p, \theta | z) \propto p(z | x) p(p | \theta)$. From the fact that the measurement function is linear in $p$ it readily follows that a Gaussian prior over $x$ results in a Gaussian $p(p | \theta, z) = \mathcal{N}(p; \mu_\theta, \Sigma_\theta)$. The mean and covariance matrix of this distribution are computed in [17]. Although in general the original posterior distribution $p(p, \theta | z)$ may be far from being Gaussian (e.g., multi-modal, skewed, etc), recovering the optimal $p$ given $\theta$ reduces to a simple linear-Gaussian estimation problem. Such problems are often called *conditionally* linear-Gaussian. By definition the MAP estimate is the maximizer of the posterior distribution (24),

$$x^* = \arg \max_{p, \theta} p(\theta | z) p(p | \theta, z). \quad (25)$$

It is easy to see that for any given $\theta$, maximizing the product above implies maximizing $p(p | \theta, z)$ with respect to $p$.

$$p^*(\theta) = \arg \max_{p} p(p | \theta, z) = \mu_\theta. \quad (26)$$

As in any Gaussian density, the mean of $p(p | \theta, z)$ is equal to its mode, and therefore $\mu_\theta$ is the solution of (26). Maximizing $p(p, \theta | z)$ subject to $p = \mu_\theta$ is equivalent to a NLS problem that can be solved like before. After obtaining the MAP estimate for $\theta$ we can recover $p^*$ by evaluating (26) at $\theta^*$. In practice we should use the approach taken in Algorithm 3 in order to retain the sparsity of SLAM.

V. RESULTS

In this section we report the performance of the proposed algorithm on both synthetic and real datasets. We used $g^2$o’s implementation of GN [18], CHOLMOD [6] is used as the linear solver with the default choice of fill-reducing permutation. We implemented Algorithm 3 (VP)

\footnote{In this section VP refers to our proposed algorithm, not to be confused with the original or Kaufman’s VP algorithms.} in C++. An Intel Core i5-2400 CPU operating at 3.1GHz is used for all of the experiments in this paper. We verified the equivalence between the iterations of Algorithm 1, Algorithm 2 and Algorithm 3 numerically. We used $g^2$o’s 2D simulator to create Manhattan-like pose-graph datasets. This simulator creates a random walk in plane with 1 meter forward motion or 90o rotation per step. The valid sensor range for scan matching is between 1 and 5 meters within the 135o field of view. In reality, scan matching is an expensive operation. Therefore extracting each and every (potential) loop closure is practically intractable. We imitate this practical limitation by imposing an upper bound on the degree of each vertex in the simulator.

We generated five test suites, each of which is composed of 100 randomly generated datasets with 109 poses per dataset. Each test suite corresponds to a fixed noise level $\alpha \in \{1, \ldots, 5\}$. Noise covariance for each suite is $\Sigma_\alpha = (0.01 \alpha)^2 I$. For each dataset we performed 50 iterations of different solvers. Solvers are initialized using odometry data. The outcome of each run is one of the followings. (i) **Global Min:** the global minimizer is found within 50 iterations, (ii) **Local Min:** a local minimizer (other than the global minimizer) is found within 50 iterations, or (iii) **Not Converged:** the solver has not converged to a solution before 50 iterations.

Table 1 summarizes the results obtained under different noise levels. Although in few instances VP outperformed GN, in general the two algorithms exhibit comparable performances.
in terms of converging to the optimal solution. As expected, both algorithms tend to converge to local minima as $\alpha$ increases; a “good” initial guess is crucial for converging to the optimal solution. Nevertheless, according to Table I our algorithm significantly outperforms GN in avoiding divergence (or extremely slow convergence). Therefore, as reported by other researchers in various fields [13], VP iterations lead to a faster and more reliable convergence than solving the full NLS problem. As any other iterative solver, using VP can lead to a local minimizer other than MLE. This can be avoided by using a “sufficiently good” initial guess [15]. It is worth noting that in the case of converging to local minima, the results obtained by both solvers were generally inaccurate and far from the optimal estimate. Out of the 500 simulations used in Table I and II, there are 89 cases for which both GN and VP converge to local minima. In 38 of those instances, GN and VP converge to the same local minimum. Furthermore, in 74 of those (89) cases, the local minima found by GN and VP are both at least 10% larger than the true minimum. In addition to the results shown in Table I, we performed another experiment by generating random initial guesses in a neighbourhood of MLE. Random initial guesses were generated by sampling uniformly from the surface of hyperspheres centered at the global minimum with different radii. As in Table I, we did not observe a statistically significant difference in the tendency of VP and GN for converging to local minima.

Table II shows the results obtained by Levenberg-Marquardt (LM) and a trust-region version of Algorithm 3 using LM (VP-LM). For both solvers we use $\sigma_0$’s default settings (e.g., strategy to update the damping parameter). Due to the slow progress of LM, here we report the results after both 50 and 100 iterations. According to Table II, LM exhibits extremely slow convergence while the performance of VP-LM is comparable to that of GN and VP. That being said, the success rates of GN and VP are slightly higher than VP-LM.

Figure 2 shows the average number of iterations performed to converge to the optimal solution under different noise levels. It clearly indicates that the proposed algorithm can converge to the optimal solution in less number of iterations than GN. To correctly interpret this result, it is crucial to note that there is a one-to-one correspondence between iterations of Algorithm 3 and those of Algorithm 1 performed on the reduced problem (15).

Algorithm 2 SLAM solver based on [1]

1: repeat
2: Construct the normal equations (21)
3: Eliminate $\delta p_i$ using the Schur complement (22)
4: Solve (22) to obtain $\delta \theta_i$
5: $\theta_{i+1} \leftarrow \theta_i + \delta \theta_i$
6: $p_{i+1} \leftarrow p_i^* (\theta_{i+1})$ according to (23)
7: until convergence

Algorithm 3 Proposed Algorithm

1: repeat
2: Conduct the normal equations (21)
3: Use the sparse Cholesky solver to solve (21)
4: $\theta_i \leftarrow \theta_i + \delta \theta_i$
5: $p_{i+1} \leftarrow p_i^* (\theta_{i+1})$ according to (23) using the sparse Cholesky solver
6: until convergence

Fig. 2. Average number of iterations performed to converge to the global minimum (ML estimate) under different noise levels ($\alpha$). For each noise level, 100 random datasets with $10^5$ poses has been generated. The error bars show the 95% confidence interval. The increasing length of error bars is partly due to the decreasing number of successful samples (see Table I). Note that there is a one-to-one correspondence between iterations of Algorithm 3 and those of Algorithm 1 performed on the reduced problem (15).

Fig. 3. Overall run time for converging to MLE as a function of edge density under different noise models. In Figure 3d, GN failed to converge to MLE in 8 scenarios (33 – 40). Note that the “100% loop closure density” refers to the 100th scenario in which all of the loop closures of a realistic sparse SLAM problem with $|V| = 50,000$ and $|E| = 173,000$ are included—not the complete graph.
original VP [12]. Kaufman’s approach [16] (Algorithm 1) and Barham and Drake’s method [1] (Algorithm 2) are all significantly slower than the state-of-the-art SLAM solvers since they are all incapable of exploiting sparsity. Unlike these algorithms, Algorithm 3 is designed to retain the sparse structure of SLAM. Nevertheless, recall that in each iteration of our algorithm, compared to GN, we have an additional (but smaller) sparse linear system to solve. Therefore each iteration of our algorithm is still slightly more expensive than that of GN. Informally speaking, Figure 2 indicates that by exploiting the separable structure of SLAM we can achieve more effective iterations at the cost of solving an additional sparse linear system in each iteration.

Consequently we conducted another experiment to compare the overall run time of VP and GN. For this purpose, first we created a Manhattan-like random walk with $5 \times 10^4$ poses. Four pose-graph datasets were created based on this random walk by simulating noisy measurements for the following noise models: (i) $\alpha = 1$, (ii) $\alpha = 2$, (iii) Non-spherical noise covariance matrix with standard deviations similar to $\alpha = 1$ and (iv) Non-spherical noise covariance matrix with standard deviations similar to $\alpha = 2$. To study the effect of edge density on the overall run time we created 100 scenarios based on each simulated dataset. The $i$th scenario contains odometry edges plus $i$ percent of the loop closures of the original simulation (including the ones that were included in the $(i-1)$th scenario). Note that the original simulation is a realistic sparse SLAM problem with a density similar to commonly used benchmarks. Loop closures are selected randomly (i.e., with no particular order) to achieve realistic and balanced scenarios.

Figure 3 shows the total run time as a function of loop closure density for each noise model. According to Figure 3, in the vast majority of cases, VP has been faster than GN. This shows that in those cases, reducing the number of iterations has paid off the additional cost paid for each iteration. In a small fraction of scenarios we observe minor difference between the run times. There are also a limited number of individual cases where GN is slightly faster than VP. This situation happens mainly in extremely sparse scenarios, in which the initial guess based on odometry is already close to MLE and thus GN can find the solution in 2-3 iterations. Such scenarios are often too sparse to be considered as realistic cases. Figure 3 also indicates that using VP becomes more beneficial (in terms of the overall run time) as noise level increases. This conclusion is consistent with what we saw earlier in Figure 2.

Datasets with spherical noise covariance matrices possess an additional structure that can be exploited to significantly reduce the cost per iteration of Algorithm 3. Recall that in each iteration of our algorithm we need to solve an additional linear system to recover the conditionally-optimal estimate of linear variables. The cost of this extra step is dominated by the Cholesky factorization of $H_1^\top H_1$ in (23). According to Remark 4, this term is constant (i.e., independent of the current estimate) when the noise covariance matrix is spherical. Thus, the Cholesky factorization needs to be done only once in such problems (i.e., for the first iteration). In the rest of iterations we only need to solve sparse triangular linear systems by forward and backward substitutions in order to recover the conditionally-optimal $p$. As illustrated by Figure 3, this trick can significantly reduce the cost per iteration and overall run time of our algorithm.

We also used a number of publicly available datasets to evaluate the performance of the proposed algorithm. Table III provides the number of iterations performed to find the optimal solution, as well as the average of total computation time over 10 runs. Starting from the odometry initial guess, both algorithms are able to converge to MLE. This is true for most of the existing real datasets, although as seen in Table I for simulated datasets, a bad initial guess can cause VP and GN to converge to (different) local minimizers of the cost function. The datasets listed in Table III span the most common forms of SLAM (2D/3D real/synthetic pose-graphs). As expected, VP converges to the optimal estimate in less number of iterations than GN (up to 50%). In most cases VP also outperforms GN in terms of the total computation time (up to 30%). Small datasets and accurate measurements make the SLAM problem less challenging in terms of convergence [3]. In such cases, the computational benefits of exploiting the separable structure of SLAM can be less than more challenging scenarios. This conclusion is consistent with what we saw earlier in Figure 3.

VI. RELATED WORKS

An extensive survey of VP applications can be found in [13, 25]. A theoretical analysis on the convergence properties of variable projection methods is due to Ruhe and Wedin [24]. The high-dimensional state space of SLAM is one of its distinctive features in comparison with many other applications of VP.
Then for each sample the optimal depletion has a direct negative effect on estimating degeneracy and consequently, particle depletion [10]. Particle a high-dimensional state space will eventually suffer from [20, 21]. Any sequential Monte Carlo method employed on particle filters (RBPF), see e.g., [9]. In these methods first $N$ Gaussian state space models is to use Rao-Blackwellized $\theta$ vs. the separable structure of feature-based SLAM by employing instead of its mode). L-SLAM [30, 31] uses this idea to exploit the minimum worth noting that this approach naturally leads to the minimum mean-square error point estimate (i.e., mean of the posterior instead of its mode). L-SLAM [30, 31] uses this idea to exploit the separable structure of feature-based SLAM by employing a RBPF based on a clever partitioning of state variables, i.e., $\theta$ vs. $p$, instead of FastSLAM's choice of poses vs. features [20, 21]. Any sequential Monte Carlo method employed on a high-dimensional state space will eventually suffer from degeneracy and consequently, particle depletion [10]. Particle depletion has a direct negative effect on estimating $\theta$: at time step $t \gg t_0$, eventually all of the particles share the same estimate for $\{\theta_i\}_{i=1}^t$ for some $t_0$ (i.e., effectively only one sample is drawn from the corresponding region).

In our previous work [29] linear variables of 2D feature-based problems with spherical noise are explicitly eliminated to obtain a smaller optimization problem over $\theta$. This approach is similar to Golub and Pereyra's VP [12], but with numerical differentiation and Newton iterations. This method is computationally beneficial only in extremely noisy problems with dense graphs. LAGO [4, 5] uses the separable structure of SLAM to bootstrap GN. First a refined estimate for $\theta$ is computed by only considering relative measurements of robot heading. Using this initial estimate of $\theta$, LAGO then recovers the conditionally-optimal estimate for $p$. Finally the result is used as the initial guess in GN. From this perspective, our algorithm can be roughly interpreted as a constant use of GN or (other Newton-based solver) iterations are relatively costly and/or when it takes more than few iterations to solve the full NLS problem. Exploiting separability is especially beneficial when GN (or other Newton-based solver) iterations are relatively costly and/or when it takes more than few iterations to solve the full NLS problem. Datasets with relatively high measurement noise and bad initial guess are among those cases.

Our current implementation relies on $\hat{g}^o$ to compute the intermediate GN step. Identifying and merging some of the terms shared between VP and GN to achieve a more tightly integrated implementation can further improve the computational benefits of the proposed method, especially in non-spherical problems. According to our empirical observations, using VP seems to be more crucial in the first few iterations. A hybrid strategy would involve an “iteration management” stage in which the back-end can select the next iteration (e.g., VP or GN) based on the expected gain and cost. Such a strategy may impose negligible extra cost since GN step is already computed in each iteration of Algorithm 3. We plan to investigate this idea in our future work.

### ACKNOWLEDGMENTS

We would like to thank C. Stachniss et al. and the EUROPA project for providing the UTM-Downtown dataset, D. Haehnel for the Intel dataset, E. Olson for the Manhattan dataset, M. Kaess for City10K and Sphere2500 datasets, M. Smith et al. for the New College dataset [26], and R. Kuemmerle et al. for $\hat{g}^o$ [18]. This work was supported by the Australian Research Council Discovery Project DP120102786.
REFERENCES

[28] Heng Wang, Shoudong Huang, Udo Frese, and Gamini Dissanayake. The nonlinearity structure of point feature SLAM problems with spherical covariance matrices. Au-
