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 iteratively 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 [12]. 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 [9]. 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 [8].

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/landmark positions. Therefore given the robot orientation $\boldsymbol{\theta}$, the optimal choice in ML or MAP sense for positions $\mathbf{p}$ 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, 14]. In statistical terms, such problems are sometimes called *conditionally linear-Gaussian* state space models [10] 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 [13]. 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 [14] 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. The contributions made in this paper are summarized below.

1) Investigating a significant but overlooked structure of SLAM, as well as establishing the link between SLAM and a vast literature on separable NLS problems.
2) Proposing an effective algorithm to exploit the aforementioned structure without sacrificing the sparsity.
3) Providing new insights into the link between separable and conditionally linear-Gaussian problems.

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. $|\mathcal{X}|$ denotes the cardinality of set $\mathcal{X}$. $\mathbf{I}$ and $\mathbf{0}$ denote the identity and zero matrix with appropriate sizes, respectively. Let $\mathrm{vec}(\mathbf{q}_1, \dots, \mathbf{q}_n)$ denote the column vector obtained by stacking $\mathbf{q}_i$'s. $\mathbf{S}_1 \succ \mathbf{S}_2$ means $\mathbf{S}_1 - \mathbf{S}_2$ is positive-definite. Kronecker product is denoted by $\otimes$. Euclidean norm is denoted by $\|\cdot\|$. The weighted Euclidean norm of vector $\mathbf{e}$ with matrix $\mathbf{W} \succ \mathbf{0}$ is denoted by $\|\mathbf{e}\|_{\mathbf{W}} \triangleq \sqrt{\mathbf{e}^\top \mathbf{W}\, \mathbf{e}}$. Finally $\mathrm{diag}(\mathbf{W}_1, \dots, \mathbf{W}_k)$ is the block-diagonal matrix with matrices $\mathbf{W}_1, \dots, \mathbf{W}_k$ 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 $\mathcal{G} = (\mathcal{V},\mathcal{E})$. Each state variable is mapped into a vertex $\mathbf{x}_i \in \mathcal{V}$, 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 $\mathbf{x}_i$ to $\mathbf{x}_j$. 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 $\mathbf{x}_0$ is the origin of our global coordinates system. The *reduced* incidence matrix of $\mathcal{G}$ after anchoring $\mathbf{x}_0$ to the origin (i.e., deleting the corresponding row from the original incidence matrix) is denoted by $\mathbf{A} \in \{-1,0,1\}^{n\times m}$. For the kth edge $e_k = (i_k,j_k) \in \mathcal{E}$ we have $\mathbf{A}_{i_k,k} = -1$ and $\mathbf{A}_{j_k,k} = 1$. The remaining elements of $\mathbf{A}$ are all zero. Similarly, $\mathbf{A}_\circ \in \{-1,0,1\}^{n\times n}$ is the reduced incidence matrix of the subgraph of $\mathcal{G}$ consisting only of robot poses and odometry measurements. Note that $\mathbf{A}_\circ$ has a fixed structure and it is uniquely determined by the number of poses n_p. Finally we define $\mathbf{A}_\ell \triangleq \mathbf{A} \otimes \mathbf{I}_\ell$ for $\ell \in \mathbb{Z}_{\geq 2}$.

B. Measurement Model

The conventional state vector is usually defined as $\mathrm{vec}(\mathbf{x}_1,\dots,\mathbf{x}_n)$. For reasons that will become clear shortly, we permute the standard state vector and define our state vector as $\mathbf{x} \triangleq \mathrm{vec}(\mathbf{p},\boldsymbol{\theta})$. In 2D SLAM, $\mathbf{p} \in \mathbb{R}^{2n}$ is the vector of x and y coordinates of robot poses and landmark positions, and $\boldsymbol{\theta} \in [-\pi,\pi)^{n_p-1}$ is the vector of robot orientations. Each observation $\mathbf{z}_{ij}$ (from node i to node j) is corrupted by an independently drawn additive Gaussian noise $\boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_{ij})$,

$$\mathbf{z}_{ij} = \mathbf{h}_{ij}(\mathbf{x}_i,\mathbf{x}_j) + \boldsymbol{\epsilon}_{ij}. \tag{1}$$

The measurement function $\mathbf{h}_{ij}(\cdot,\cdot)$ for any $(i,j) \in \mathcal{E}$ has the following form,

$$\mathbf{h}_{ij}(\mathbf{x}_i,\mathbf{x}_j) = \begin{cases} \begin{bmatrix}\delta x_{ij}\\ \delta y_{ij}\end{bmatrix} = \mathbf{R}(\theta_i)^\top(\mathbf{p}_j - \mathbf{p}_i) & \text{if } j \in \mathcal{S}_f, \\ \begin{bmatrix}\delta x_{ij}\\ \delta y_{ij}\\ \delta\theta_{ij}\end{bmatrix} = \begin{bmatrix}\mathbf{R}(\theta_i)^\top(\mathbf{p}_j - \mathbf{p}_i)\\ \mathrm{wrap}\,(\theta_j - \theta_i)\end{bmatrix} & \text{if } j \in \mathcal{S}_p. \end{cases} \tag{2}$$

where $\mathbf{R}(\theta_i) \in \mathrm{SO}(2)$[1] is the rotation matrix corresponding to θ_i, $\mathrm{wrap} : \mathbb{R} \to [-\pi,\pi)$ maps its argument to the equivalent angle in $[-\pi,\pi)$ and $\mathcal{S}_p$ and $\mathcal{S}_f$ are the disjoint sets of indices of robot poses and features, respectively. Let us define

$$\mathbf{R}_\theta \triangleq \mathrm{diag}\left(\mathbf{R}(\theta_{k_1}),\dots,\mathbf{R}(\theta_{k_m})\right). \tag{3}$$

[1]Special orthogonal group.

Here k_i is the index of robot pose making the ith observation. $\mathbf{z}_p$ and $\mathbf{z}_\theta$ denote the stacked vector of δx_{ij} and δy_{ij}, and $\delta\theta_{ij}$ measurements, respectively. We permute the measurement vector accordingly to obtain $\mathbf{z} \triangleq \mathrm{vec}(\mathbf{z}_p,\mathbf{z}_\theta)$. Similarly, the stacked vector of noise variables and its covariance matrix are denoted by $\boldsymbol{\epsilon} \triangleq \mathrm{vec}(\boldsymbol{\epsilon}_p,\boldsymbol{\epsilon}_\theta)$ and $\boldsymbol{\Sigma}$, respectively. Therefore the measurement model can be expressed in a compact form by

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}), \tag{4}$$

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z};\mathbf{h}(\mathbf{x}),\boldsymbol{\Sigma}).^2 \tag{5}$$

Remark 1. Note that (2) admits both pose-graph and feature-based SLAM problems as special cases. In pose-graph SLAM $\mathcal{S}_f = \varnothing$, 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

$$\mathbf{h}(\mathbf{x}) = \mathbf{H}(\boldsymbol{\theta})\,\mathbf{x} \triangleq \begin{bmatrix}\mathbf{R}_\theta^\top\mathbf{A}_2^\top & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Lambda}^\top\end{bmatrix}\mathbf{x},$$
$$\boldsymbol{\Lambda} = \begin{cases}\mathbf{A} & \text{pose-graph,}\\ \mathbf{A}_\circ & \text{feature-based.}\end{cases} \tag{6}$$

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 $\mathbf{h}(\mathbf{x}) = \mathbf{H}_1(\boldsymbol{\theta})\mathbf{p} + \mathbf{H}_2\boldsymbol{\theta}$ in which,

$$\mathbf{H}_1(\boldsymbol{\theta}) \triangleq \begin{bmatrix}\mathbf{R}_\theta^\top\mathbf{A}_2^\top\\ \mathbf{0}\end{bmatrix},\ \mathbf{H}_2 \triangleq \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\Lambda}^\top\end{bmatrix}. \tag{7}$$

It is important to note that unlike $\mathbf{H}_2$, $\mathbf{H}_1(\boldsymbol{\theta})$ depends on $\boldsymbol{\theta}$. Nevertheless, we drop the argument of $\mathbf{H}_1$ for the sake of simplicity of our notation.

Remark 2. For the purpose of this paper, the measurement model in 3D SLAM wtih $\mathrm{SE}(3)$[3] measurements is fairly similar to that of planar SLAM. For instance, with a little abuse of notation, the measurement function for 3D SLAM can be expressed as $\mathbf{h}^\circ(\mathbf{x}) = \mathbf{H}_1^\circ(\boldsymbol{\theta})\mathbf{p} + \mathbf{h}_2^\circ(\boldsymbol{\theta})$ in which

$$\mathbf{H}_1^\circ(\boldsymbol{\theta}) \triangleq \begin{bmatrix}\mathbf{R}_\theta^\top\mathbf{A}_3^\top\\ \mathbf{0}\end{bmatrix},\ \mathbf{h}_2^\circ(\boldsymbol{\theta}) \triangleq \begin{bmatrix}\mathbf{0}\\ \boldsymbol{\psi}(\boldsymbol{\theta})\end{bmatrix}. \tag{8}$$

Similar to our formulation of 2D SLAM, here $\mathbf{p}$ is the stacked vector of robot positions, $\boldsymbol{\theta}$ is the vector of robot orientation (parametrised by quaternions or Euler angles), $\mathbf{R}_\theta$ is a block diagonal matrix composed of $\mathbf{R}(\boldsymbol{\theta}_i) \in \mathrm{SO}(3)$ and $\boldsymbol{\psi}(\boldsymbol{\theta})$ denotes relative robot orientations. The measurement function and the corresponding error term (see e.g., [15]) are therefore linear in $\mathbf{p}$. Without loosing any generality, for simplicity of notation and due to space limitation here we mainly focus on 2D SLAM.

[2]To simplify our notaions we denote random variables (e.g., $\mathbf{z}$) and their realisation in the same way.

[3]Special Euclidean group.

Remark 3. This paper investigates SLAM problems with standard $\mathrm{SE}(2)$ and $\mathrm{SE}(3)$ 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 reparameterization 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 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 $\mathbf{p}$ (see e.g., [19]).

C. Point Estimation Criterion

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

$$p(\mathbf{x}|\mathbf{z}) \propto p(\mathbf{z}|\mathbf{x})\ p(\mathbf{x}). \tag{9}$$

Therefore the MAP estimate $\hat{\mathbf{x}}_{\text{MAP}}$ is given by:

$$\hat{\mathbf{x}}_{\text{MAP}} = \arg\min_{\mathbf{x}} \left(\|\mathbf{z} - \mathbf{h}(\mathbf{x})\|^2_{\boldsymbol{\Sigma}^{-1}} + \|\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}\|^2_{\boldsymbol{\Sigma}_{\mathbf{x}}^{-1}} \right). \tag{10}$$

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

$$\hat{\mathbf{x}}_{\text{ML}} = \arg\min_{\mathbf{x}} \|\mathbf{z} - \mathbf{h}(\mathbf{x})\|^2_{\boldsymbol{\Sigma}^{-1}}. \tag{11}$$

In the following sections we mainly focus on $\hat{\mathbf{x}}_{\text{ML}}$ as it is fairly rare to have an informative prior over $\mathbf{x}$ 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 $c^\star$. Here $c^\star$ 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 *nonlinear* least squares problems. By further inspection of (7) one can see that the nonlinearity is caused by the rotation matrices. Hence given the robot orientations $\boldsymbol{\theta}$, measurements are linear in robot and features positions $\mathbf{p}$. Therefore given $\boldsymbol{\theta}$, the ML and MAP estimates for $\mathbf{p}$ are obtained by solving *linear* least squares problems. Such problems are often called *separable* NLS [14].

[4] In this case, $\mathbf{x}$ is treated as the vector of unknown deterministic parameters in the measurement model.

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 [13]. 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 [13, Theorem 2.1]). Here we explain how their approach can be applied to SLAM.

I. Find $\mathbf{p}^\star(\boldsymbol{\theta})$, the closed-form expression for $\mathbf{p}$ as a function of $\boldsymbol{\theta}$ that minimizes the original cost function in $\mathbf{p}$.
II. Replace $\mathbf{p}$ with $\mathbf{p}^\star(\boldsymbol{\theta})$ in the original problem and minimize the new objective function in $\boldsymbol{\theta}$ to obtain $\boldsymbol{\theta}^\star$. After this step the optimal $\mathbf{p}^\star = \mathbf{p}^\star(\boldsymbol{\theta}^\star)$ can be recovered instantly.

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

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} \|\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_1\mathbf{p} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}\|^2, \tag{12}$$

in which $\tilde{\mathbf{z}} \triangleq \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{z}$, $\tilde{\mathbf{H}}_1 \triangleq \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{H}_1$ and $\tilde{\mathbf{H}}_2 \triangleq \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{H}_2$. The fact that the reduced incidence matrix is full rank in (weakly) connected graphs results in the following lemma.

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

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

$$\mathbf{p}^\star(\boldsymbol{\theta}) \triangleq \arg\min_{\mathbf{p}} \|\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_1\mathbf{p} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}\|^2 \tag{13}$$

$$= \tilde{\mathbf{H}}_1^\dagger(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}), \tag{14}$$

in which $\tilde{\mathbf{H}}_1^\dagger \triangleq (\tilde{\mathbf{H}}_1^\top\tilde{\mathbf{H}}_1)^{-1}\tilde{\mathbf{H}}_1^\top$ is the Moore-Penrose pseudoinverse of $\tilde{\mathbf{H}}_1$.

Remark 4. Although we do not require any special structure in $\boldsymbol{\Sigma}$, 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 $\boldsymbol{\Sigma}_p^i$. An interesting special case emerges when $\boldsymbol{\Sigma}_p^i$ is *spherical*, i.e., $\boldsymbol{\Sigma}_p^i = {\sigma_p^i}^2\mathbf{I}_2$. Noting that $\mathbf{R}_\theta$ is orthogonal, it is easy to show that in such SLAM problems $\tilde{\mathbf{H}}^\top\tilde{\mathbf{H}}$ is proportional to $\mathbf{L}_2 \triangleq \mathbf{L} \otimes \mathbf{I}_2 = \mathbf{A}_2\mathbf{A}_2^\top$ in which $\mathbf{L}$ is the reduced Laplacian matrix of graph $\mathcal{G}$. A similar structure exists in 3D SLAM problems with spherical noise covariance matrices.

Phase II – Reduced NLS: By substituting $\mathbf{p}$ in the original objective function (12) with $\mathbf{p}^\star(\boldsymbol{\theta})$ in (14) and solving the

Algorithm 1 SLAM solver based on [17]

repeat
  Compute the full QR factorization of $\tilde{\mathbf{H}}_1$ (18)
  Recover $\mathbf{p}^\star$ by solving $\mathbf{R}_1\,\mathbf{p}^\star(\boldsymbol{\theta}_{(i)}) = \mathbf{Q}_1^\top(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}_{(i)})$
  Compute the modified Jacobian matrix (37) (Appendix C)
  Construct the normal equations for the reduced problem
  Solve the normal equations to obtain $\delta\boldsymbol{\theta}_{(i)}$
  $\boldsymbol{\theta}_{(i+1)} \leftarrow \boldsymbol{\theta}_{(i)} + \delta\boldsymbol{\theta}_{(i)}$
until convergence
$\mathbf{p}^\star \leftarrow \mathbf{p}^\star(\boldsymbol{\theta}^\star)$ according to (14)

resulting optimization problem we obtain $\boldsymbol{\theta}^\star \triangleq \arg\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta})$ where

$$g(\boldsymbol{\theta}) \triangleq \|(\mathbf{I} - \tilde{\mathbf{H}}_1\tilde{\mathbf{H}}_1^\dagger)(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})\|^2. \tag{15}$$

Note that $g(\cdot)$ is a function of only robot headings $\boldsymbol{\theta}$, while the original optimization problem (12) was over both $\mathbf{p}$ and $\boldsymbol{\theta}$. Therefore we have reduced the parameter space from $\mathbb{R}^{2n}\times[-\pi,\pi)^{n_p-1}$ to $[-\pi,\pi)^{n_p-1}$. $\mathbf{P}_\theta \triangleq \tilde{\mathbf{H}}_1\tilde{\mathbf{H}}_1^\dagger$ is the orthogonal projection onto $\mathrm{range}(\tilde{\mathbf{H}}_1)$, while $\mathbf{P}_\theta^\perp \triangleq \mathbf{I} - \tilde{\mathbf{H}}_1\tilde{\mathbf{H}}_1^\dagger$ is its orthogonal complement. Let us define $\mathbf{r}_{\mathrm{vp}} \triangleq \mathbf{P}_\theta^\perp(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})$. In order to solve (15) using Newton-based NLS solvers we need to compute the Jacobian matrix of $\mathbf{r}_{\mathrm{vp}}$, i.e., $\mathbf{J}_{\mathrm{vp}} \triangleq \frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{r}_{\mathrm{vp}}$. Computing $\mathbf{J}_{\mathrm{vp}}$ requires differentiating pseudoinverses ($\tilde{\mathbf{H}}_1^\dagger$), and therefore is more complex than computing the Jacobian matrix of the original full problem, i.e., $\mathbf{J} \triangleq \frac{\partial}{\partial\mathbf{x}}\mathbf{r}$ in which $\mathbf{r} \triangleq \mathbf{z} - \mathbf{h}(\mathbf{x})$. One way to avoid this complexity is to approximate $\mathbf{J}_{\mathrm{vp}}$ using finite differences (see [14] and the references therein). Here we use the exact analytical expression for $\mathbf{J}_{\mathrm{vp}}$ which was derived by Golub and Pereyra [13].

Computing $\mathbf{J}_{\mathrm{vp}}$: First note that (15) has a slightly more general form compared to the case which was originally considered by [13] in that we have an additional term, linear with respect to $\boldsymbol{\theta}$ in our residual, i.e., $-\tilde{\mathbf{H}}_2\,\boldsymbol{\theta}$. We can show that the jth column of $\mathbf{J}_{\mathrm{vp}}$ is given by,

$$[\mathbf{J}_{\mathrm{vp}}]_{\cdot,j} = -\left((\mathbf{P}_{\boldsymbol{\theta}}^\perp\frac{\partial\tilde{\mathbf{H}}_1}{\partial\theta_j}\tilde{\mathbf{H}}_1^\dagger) + (\mathbf{P}_{\boldsymbol{\theta}}^\perp\frac{\partial\tilde{\mathbf{H}}_1}{\partial\theta_j}\tilde{\mathbf{H}}_1^\dagger)^\top\right)(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\,\boldsymbol{\theta}) - \mathbf{P}_{\boldsymbol{\theta}}^\perp[\tilde{\mathbf{H}}_2]_{\cdot,j}. \tag{16}$$

After finding $\boldsymbol{\theta}^\star \triangleq \arg\min g(\boldsymbol{\theta})$ using an iterative NLS solver such as GN, we can recover the optimal $\mathbf{p}^\star$ according to (14), i.e., by solving $(\tilde{\mathbf{H}}_1^\top\tilde{\mathbf{H}}_1)\mathbf{p}^\star = \tilde{\mathbf{H}}_1^\top(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}^\star)$. In general, VP iterations are 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 [17] Kaufman proposed to approximate the jth column of $\mathbf{J}_{\mathrm{vp}}$ according to

$$[\mathbf{J}_{\mathrm{vp}}^{\mathrm{K}}]_{\cdot,j} \triangleq -\left(\mathbf{P}_{\boldsymbol{\theta}}^\perp\frac{\partial\tilde{\mathbf{H}}_1}{\partial\theta_j}\tilde{\mathbf{H}}_1^\dagger\right)(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\,\boldsymbol{\theta}) - \mathbf{P}_{\boldsymbol{\theta}}^\perp[\tilde{\mathbf{H}}_2]_{\cdot,j}. \tag{17}$$

The term ignored in $\mathbf{J}_{\mathrm{vp}}^{\mathrm{K}}$ has been theoretically and practically shown to be negligible in small-residual problems in terms of the convergence rate of GN [17, 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 [14]. 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 $\tilde{\mathbf{H}}_1$,

$$\tilde{\mathbf{H}}_1 = \mathbf{Q}\mathbf{R} = \begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix}\begin{bmatrix}\mathbf{R}_1\\ \mathbf{0}\end{bmatrix}, \tag{18}$$

in which the columns of $\mathbf{Q}_1$ and $\mathbf{Q}_2$ form orthonormal bases for $\mathrm{range}(\tilde{\mathbf{H}}_1)$ and $\mathrm{null}(\tilde{\mathbf{H}}_1^\top)$, respectively.

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 $\mathcal{O}(n_p^3)$ 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,

$$\tilde{\mathbf{J}}^\top\tilde{\mathbf{J}}\,\delta\mathbf{x}_{(i)} = -\tilde{\mathbf{J}}^\top\tilde{\mathbf{r}}. \tag{19}$$

Here $\delta\mathbf{x}_{(i)} \triangleq \mathrm{vec}(\delta\mathbf{p}_{(i)}, \delta\boldsymbol{\theta}_{(i)})$ denotes the ith GN direction, $\tilde{\mathbf{r}}$ is the residual vector $\tilde{\mathbf{r}} \triangleq \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{r}$ and $\tilde{\mathbf{J}} \triangleq \frac{\partial}{\partial\mathbf{x}}\tilde{\mathbf{r}}$, both evaluated at $\mathbf{x}_{(i)}$. Let $\boldsymbol{\mathcal{H}} \triangleq \tilde{\mathbf{J}}^\top\tilde{\mathbf{J}}$ be the approximated Hessian. Note that $\tilde{\mathbf{J}}$ can be divided into two blocks, $\tilde{\mathbf{J}} = [\tilde{\mathbf{J}}_p \;\; \tilde{\mathbf{J}}_\theta]$ in which $\tilde{\mathbf{J}}_p \triangleq \frac{\partial}{\partial\mathbf{p}}\tilde{\mathbf{r}}$ and $\tilde{\mathbf{J}}_\theta \triangleq \frac{\partial}{\partial\boldsymbol{\theta}}\tilde{\mathbf{r}}$. Therefore we can expand (19) as

$$\begin{bmatrix}\boldsymbol{\mathcal{H}}_p & \boldsymbol{\mathcal{H}}_{p,\theta}\\ \boldsymbol{\mathcal{H}}_{p,\theta}^\top & \boldsymbol{\mathcal{H}}_\theta\end{bmatrix}\begin{bmatrix}\delta\mathbf{p}_{(i)}\\ \delta\boldsymbol{\theta}_{(i)}\end{bmatrix} = \begin{bmatrix}-\tilde{\mathbf{J}}_p^\top\tilde{\mathbf{r}}\\ -\tilde{\mathbf{J}}_\theta^\top\tilde{\mathbf{r}}\end{bmatrix}. \tag{20}$$

One can eliminate $\delta\mathbf{p}_{(i)}$ from (20) to obtain a smaller linear system in terms of $\delta\boldsymbol{\theta}_{(i)}$. This can be achieved using the Schur complement of $\boldsymbol{\mathcal{H}}$ with respect to $\boldsymbol{\mathcal{H}}_p$.

$$\left(\boldsymbol{\mathcal{H}}_\theta - \boldsymbol{\mathcal{H}}_{p,\theta}^\top\boldsymbol{\mathcal{H}}_p^{-1}\boldsymbol{\mathcal{H}}_{p,\theta}\right)\delta\boldsymbol{\theta}_{(i)} = \left(-\tilde{\mathbf{J}}_\theta^\top + \boldsymbol{\mathcal{H}}_{p,\theta}^\top\boldsymbol{\mathcal{H}}_p^{-1}\tilde{\mathbf{J}}_p^\top\right)\tilde{\mathbf{r}}. \tag{21}$$

Solving (21) gives us $\delta\boldsymbol{\theta}_{(i)}$. It should be clear that using this solution to recover $\delta\mathbf{p}_{(i)}$ from (20) leads to the standard GN direction for the original cost function. Barham and Drane proposed to ignore $\delta\mathbf{p}_{(i)}$ of (20) and instead compute the conditionally optimal $\mathbf{p}_{(i+1)}$ given $\boldsymbol{\theta}_{(i+1)} = \boldsymbol{\theta}_{(i)} + \delta\boldsymbol{\theta}_{(i)}$ using (14) [1]. This can be achieved by solving the following *sparse* linear system,

$$(\tilde{\mathbf{H}}_1^\top\tilde{\mathbf{H}}_1)\,\mathbf{p}_{(i+1)} = \tilde{\mathbf{H}}_1^\top\left(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}_{(i+1)}\right). \tag{22}$$

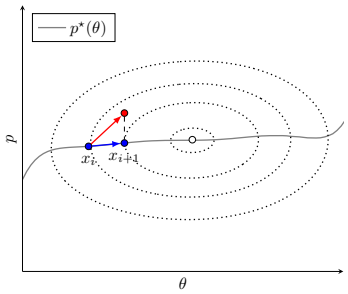


Fig. 1. In this figure we see the contour lines of a simplified version of SLAM cost function. The point in the middle is the optimal solution. The gray curve is $p^\star(\theta)$; a function that maps any given θ to the corresponding conditionally optimal estimate for p. The cost function has zero gradient with respect to p along this curve. The blue vector shows the ith step of Algorithm 1—which is identical to the ith step of Algorithm 2 and Algorithm 3. The red vector is the GN step obtained by starting from x_i. Algorithm 3 corrects this intermediate step by projecting the obtained solution back on $p^\star(\theta)$ (dashed line). Note that Algorithm 1 computes the blue vector directly by performing a GN iteration on the *reduced* problem (15) with cubic time complexity. However our indirect approach in Algorithm 3 enables us to retain the sparse structure of the original problem.

Repeating this procedure until convergence leads to a sequence of steps for which the cost function (11) has zero gradient with respect to $\mathbf{p}$ along the path. This algorithm is summarized in Algorithm 2. The equivalence between Algorithm 1 and 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 (21) is dense in general. But note that in each iteration of Algorithm 2, the solution of the reduced system (21) is identical to the $\delta\boldsymbol{\theta}_{(i)}$ obtained from solving the full system (20). Therefore, instead of eliminating variables using Schur complement, the same outcome can be achieved by solving the sparse full system (20), discarding the obtained $\delta\mathbf{p}_{(i)}$ and instead computing the conditionally optimal $\mathbf{p}_{(i+1)}$ according to (22). 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 (22) 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 $\mathbf{x}$ is available. 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 $p(\mathbf{p},\boldsymbol{\theta}|\mathbf{z})$ can be factored according to

$$p(\mathbf{p},\boldsymbol{\theta}|\mathbf{z}) = p(\boldsymbol{\theta}|\mathbf{z})\, p(\mathbf{p}|\boldsymbol{\theta},\mathbf{z}). \tag{23}$$

Using the Bayes rule we have $p(\mathbf{p}|\boldsymbol{\theta},\mathbf{z}) \propto p(\mathbf{z}|\mathbf{x})\, p(\mathbf{p}|\boldsymbol{\theta})$. From the fact that the measurement function is linear in $\mathbf{p}$ it readily follows that a Gaussian prior over $\mathbf{x}$ results in a Gaussian $p(\mathbf{p}|\boldsymbol{\theta},\mathbf{z}) = \mathcal{N}(\mathbf{p};\boldsymbol{\mu}^\circ_{\boldsymbol{\theta}},\boldsymbol{\Sigma}^\circ_{\boldsymbol{\theta}})$. The mean and covariance matrix of this distribution can be found in Appendix E. Although in general the original posterior distribution $p(\mathbf{p},\boldsymbol{\theta}|\mathbf{z})$ may be far from being Gaussian (e.g., multi-modal, skewed, etc), recovering the optimal $\mathbf{p}$ given $\boldsymbol{\theta}$ 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 (23),

$$\mathbf{x}^\star = \arg\max_{\mathbf{p},\boldsymbol{\theta}} p(\boldsymbol{\theta}|\mathbf{z})\, p(\mathbf{p}|\boldsymbol{\theta},\mathbf{z}). \tag{24}$$

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

$$\mathbf{p}^\star(\boldsymbol{\theta}) = \arg\max_{\mathbf{p}} p(\mathbf{p}|\boldsymbol{\theta},\mathbf{z}) = \boldsymbol{\mu}^\circ_{\boldsymbol{\theta}}. \tag{25}$$

As in any Gaussian density, the mean of $p(\mathbf{p}|\boldsymbol{\theta},\mathbf{z})$ is equal to its mode, and therefore $\boldsymbol{\mu}^\circ_{\boldsymbol{\theta}}$ is the solution of (25). Maximizing $p(\mathbf{p},\boldsymbol{\theta}|\mathbf{z})$ subject to $\mathbf{p} = \boldsymbol{\mu}^\circ_{\boldsymbol{\theta}}$ is equivalent to a NLS problem that can be solved like before. After obtaining the MAP estimate for $\boldsymbol{\theta}$ we can recover $\mathbf{p}^\star$ by evaluating (25) at $\boldsymbol{\theta}^\star$. 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)[5] 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 90° rotation per step. The valid sensor range for scan matching is between 1 and 5 meters within the 135° 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 10^4 poses per dataset. Each test suite corresponds to a fixed noise level $\alpha \in \{1,\dots,5\}$. Noise covariance for each suite is $\boldsymbol{\Sigma}_\alpha = (0.01\alpha)^2\mathbf{I}$. 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

[5]In this section VP refers to our proposed algorithm, not to be confused with the original or Kaufman's VP algorithms.

Algorithm 2 SLAM solver based on [1]

1: **repeat**
2: Construct the normal equations (20)
3: Eliminate $\delta\mathbf{p}_{(i)}$ using the Schur complement (21)
4: Solve (21) to obtain $\delta\boldsymbol{\theta}_{(i)}$
5: $\boldsymbol{\theta}_{(i+1)} \leftarrow \boldsymbol{\theta}_{(i)} + \delta\boldsymbol{\theta}_{(i)}$
6: $\mathbf{p}_{(i+1)} \leftarrow \mathbf{p}^{\star}(\boldsymbol{\theta}_{(i+1)})$ according to (22)
7: **until** convergence

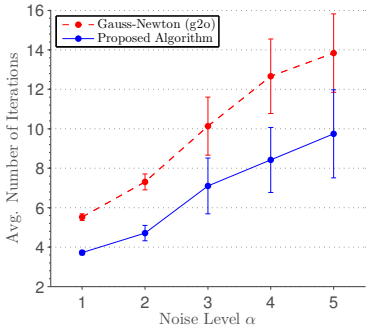


Fig. 2. Average number of iterations performed to converge to the global minimum (ML estimate) under different noise levels (α). For each noise level, 100 random datasets with 10,000 poses has been generated. The error bars show the 95% confidence interval. The increasing length of error bars is partly 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).

minimizer) is found within 50 iterations, or (iii) **Not Converged**: the solver has not converged to a solution before 50 iterations.

Table I 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 α 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 [14], 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 [16]. 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

Algorithm 3 Proposed Algorithm

1: **repeat**
2: Construct the normal equations (20)
3: Use the sparse Cholesky solver to solve (20)
4: $\boldsymbol{\theta}_{(i+1)} \leftarrow \boldsymbol{\theta}_{(i)} + \delta\boldsymbol{\theta}_{(i)}$
5: $\mathbf{p}_{(i+1)} \leftarrow \mathbf{p}^{\star}(\boldsymbol{\theta}_{(i+1)})$ according to (22) using the sparse Cholesky solver
6: **until** convergence

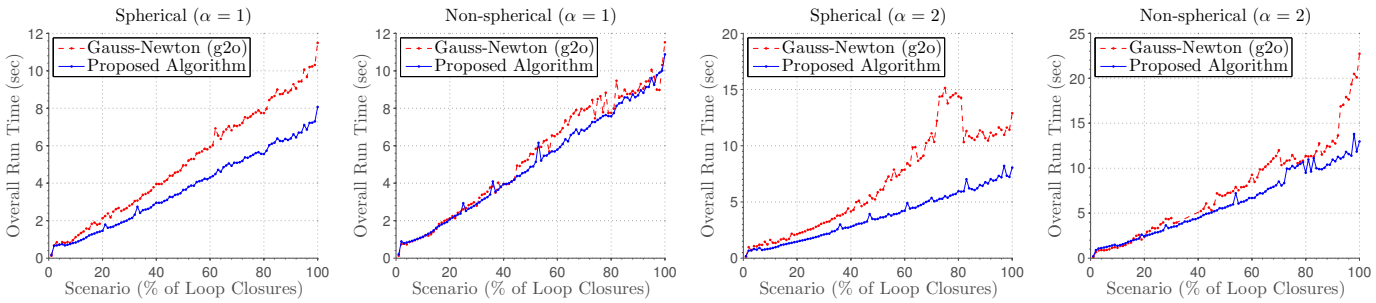
observe a statistically significant difference in the tendency of VP and GN for converging to local minima.

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 g^{2}o'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). This observation is consistent with numerous reports from other researchers who apply variable projection to separable NLS problems in other contexts [14].

Reducing the number of iterations does not necessarily reduce the total computation time as each VP iteration is usually more costly than that of GN. More specifically for SLAM, original VP [13], Kaufman's approach [17] (Algorithm 1) and Barham and Drane'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 5×10^4 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



(a) $\alpha = 1$ (spherical). (b) $\alpha = 1$ (non-spherical). (c) $\alpha = 2$ (spherical). (d) $\alpha = 2$ (non-spherical).

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 $|\mathcal{V}| = 50{,}000$ and $|\mathcal{E}| = 173{,}000$ are included—not the complete graph.

based on each simulated dataset. The ith 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 $\tilde{\mathbf{H}}_1^\top \tilde{\mathbf{H}}_1$ in (22). 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 $\mathbf{p}$. 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.

TABLE I
OUTCOME (%) OF GN AND VP AFTER 50 ITERATIONS UNDER DIFFERENT NOISE LEVELS.

Noise Level	Solver	Global Min	Local Min	Not Converged
$\alpha = 1$	GN	100	0	0
	VP	100	0	0
$\alpha = 2$	GN	91	8	1
	VP	94	6	0
$\alpha = 3$	GN	76	16	8
	VP	78	19	3
$\alpha = 4$	GN	56	36	8
	VP	57	41	2
$\alpha = 5$	GN	37	50	13
	VP	39	60	1

VI. Related Works

An extensive survey of VP applications can be found in [14]. A theoretical analysis on the convergence properties of variable projection methods is due to Ruhe and Wedin [24, 25]. The high-dimensional state space of SLAM is one of its distinctive features in comparison with many other applications of VP.

A common way of approaching conditionally linear-Gaussian state space models is to use Rao-Blackwellized particle filters (RBPF), see e.g., [10]. In these methods first N i.i.d. samples $\{\boldsymbol{\theta}^{[i]}\}_{i=1}^N$ are generated according to

TABLE II
OUTCOME (%) OF LM AND VP-LM AFTER 50 & 100 ITERATIONS UNDER DIFFERENT NOISE LEVELS.

Noise Level	Solver	Global Min		Local Min		Not Converged	
		50 iter.	100 iter.	50 iter.	100 iter.	50 iter.	100 iter.
$\alpha = 1$	LM	53	79	10	7	37	14
	VP-LM	97	97	3	3	0	0
$\alpha = 2$	LM	17	41	22	13	61	46
	VP-LM	90	90	10	10	0	0
$\alpha = 3$	LM	9	18	18	25	73	57
	VP-LM	72	73	27	27	1	0
$\alpha = 4$	LM	1	8	15	24	84	68
	VP-LM	48	48	49	52	3	0
$\alpha = 5$	LM	2	6	16	24	82	0
	VP-LM	32	33	63	66	5	1

TABLE III
SUMMARY OF RESULTS FOR SOME OF THE PUBLICLY AVAILABLE REAL AND SYNTHETIC DATASETS.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Solver	# Iter.	Time (s)
City10K	10,000	20,678	GN	7	0.45023
			VP	4	0.31845
Manhattan	3,500	5,598	GN	6	0.08757
			VP	4	0.07687
Intel	943	1,837	GN	3	0.01759
			VP	2	0.01713
UTM Downtown	14,549	16,365	GN	10	0.28154
			VP	4	0.23555
Sphere2500	2,500	9,799	GN	5	0.96145
			VP	4	0.82080
New College	52,480	52,577	GN	8	2.19560
			VP	6	2.00967

$\boldsymbol{\theta}^{[i]} \sim p(\boldsymbol{\theta}|\mathbf{z})$ (e.g., using importance sampling). Then for *each* sample the optimal $\mathbf{p}$ is recovered analytically by finding the mean of $p(\mathbf{p}|\boldsymbol{\theta}^{[i]},\mathbf{z})$ using e.g., the Kalman filter. The latter stage is equivalent to (25). It is 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., $\boldsymbol{\theta}$ vs. $\mathbf{p}$, 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 [11]. Particle depletion has a direct negative effect on estimating $\boldsymbol{\theta}$: at time step $t \gg t_0$, eventually all of the particles share the same estimate for $\{\theta_i\}_{i=1}^{t-t_0}$ 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 $\boldsymbol{\theta}$. This approach is similar to Golub and Pereyra's VP [13], 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 $\boldsymbol{\theta}$ is computed by only considering relative measurements of robot heading. Using this initial estimate of $\boldsymbol{\theta}$, LAGO then recovers the conditionally-optimal estimate for $\mathbf{p}$. 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 LAGO's bootstrapping approach, without the initial phase of approximating $\boldsymbol{\theta}$ (which makes our algorithm robust to strong correlations between the components of noise [5, 16]). Unlike our algorithm, LAGO is limited to 2D pose-graphs. The equivalence between the minima of the original optimization problem and those of the reduced problem (15) have allowed researchers to study various properties of SLAM by looking at the reduced problem. For instance in [27, 28] we analyze the number of local minima is in some small special cases using this idea. Similarly in [3] the reduced problem is used to analyze the convergence of GN in 2D pose-graphs with spherical noise covariance.

VII. CONCLUSION AND FUTURE WORK

In this paper we proposed a scalable and efficient algorithm to take advantage of the separable structure of SLAM. It was shown that by exploiting this structure, we can achieve faster and more reliable convergence than the state-of-the-art solvers. A key contribution of this work comes from establishing the link to a less-known but vast literature on separable NLS problems. In particular, recognizing the equivalence between Algorithm 1 and Algorithm 2 was the missing link that enabled us to retain sparsity while exploiting the separable structure of problem through Algorithm 3. This link also provides a firm theoretical justification for the proposed algorithm.

The proposed algorithm can be applied to the most common forms of SLAM (2D/3D feature-based and pose-graphs) without any restrictive assumption. Our algorithm is not limited to a particular type of NLS solver and the benefits it brings along are orthogonal to those of other possible improvements such as a more efficient implementation (of e.g., GN) or using different Newton-based solvers, trust-region or line search techniques. As an advantage, our final algorithm can be easily adopted by the existing back-ends (e.g., LM, Powell's Dog-leg [23], etc) without any major modification. By stripping down SLAM to its nonlinear core and recovering the conditionally optimal estimate for linear variables, our approach yields more effective and reliable iterations than solving the full NLS problem. The number of iterations required for solving the reduced problem (15) was shown to be less than that of 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 `g2o` 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 our algorithm, 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. This idea requires further investigation.

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 g^{2}o [18]. This work was supported by the Australian Research Council Discovery Project DP120102786.

References

[1] RH Barham and Wanzer Drane. An algorithm for least squares estimation of nonlinear parameters when some of the parameters are linear. *Technometrics*, 14(3):757–766, 1972.

[2] Ake Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[3] Luca Carlone. Convergence analysis of pose graph optimization via gauss-newton methods. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2013.

[4] Luca Carlone, Rosario Aragues, Jose Castellanos, and Basilio Bona. A linear approximation for graph-based simultaneous localization and mapping. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[5] Luca Carlone, Rosario Aragues, José A Castellanos, and Basilio Bona. A fast and accurate approximation for planar pose graph optimization. *The International Journal of Robotics Research*, pages 965–987, 2014.

[6] Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22, 2008.

[7] Timothy A Davis. Algorithm 915, suitesparseqr: Multifrontal multithreaded rank-revealing sparse QR factorization. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):8, 2011.

[8] Frank Dellaert and Michael Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[9] Gamini Dissanayake, Paul Newman, Steve Clark, Hugh Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, 2001.

[10] Arnaud Doucet. On sequential simulation-based methods for bayesian filtering. Technical Report CUED/F-INFENG/TR. 310, Cambridge University Department of Engineering, 1998.

[11] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.

[12] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine*, 13(2): 99–110, 2006.

[13] Gene Golub and Victor Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on numerical analysis*, 10(2):413–432, 1973.

[14] Gene Golub and Victor Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Inverse problems*, 19(2):R1, 2003.

[15] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *Intelligent Transportation Systems Magazine, IEEE*, 2(4): 31–43, 2010.

[16] Gibson Hu, Kasra Khosoussi, and Shoudong Huang. Towards a reliable SLAM back-end. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 37–43. IEEE, 2013.

[17] Linda Kaufman. A variable projection method for solving separable nonlinear least squares problems. *BIT Numerical Mathematics*, 15(1):49–57, 1975.

[18] Rainer Kuemmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[19] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *Robotics, IEEE Transactions on*, 28(1):61–76, 2012.

[20] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the National conference on Artificial Intelligence*, pages 593–598. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[21] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.

[22] Teresa Anne Parks. *Reducible nonlinear programming problems (separable least squares)*. PhD thesis, Rice University, 1985.

[23] David M. Rosen, Michael Kaess, and John J. Leonard. RISE: An incremental trust-region method for robust online sparse least-squares estimation. *IEEE Trans. on Robotics, TRO*, 30(5):1091–1108, Oct 2014.

[24] Axel Ruhe and Per Åke Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Review*, 22(3): 318–337, 1980.

[25] George A. F. Seber and C. J. Wild. *Nonlinear Regression*. Wiley-Interscience, 1989.

[26] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *The International Journal of Robotics Research*, 28(5):595–599, May 2009. ISSN 0278-3649. doi: http://dx.doi.org/10.1177/0278364909103911. URL http://www.robots.ox.ac.uk/NewCollegeData/.

[27] Heng Wang, Gibson Hu, Shoudong Huang, and Gamini Dissanayake. On the structure of nonlinearities in pose graph SLAM. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[28] Heng Wang, Shoudong Huang, Udo Frese, and Gamini Dissanayake. The nonlinearity structure of point feature

SLAM problems with spherical covariance matrices. *Automatica*, 49(10):3112–3119, 2013.

[29] Heng Wang, Shoudong Huang, Kasra Khosoussi, Udo Frese, Gamini Dissanayake, and Bingbing Liu. Dimensionality reduction for point feature SLAM problems with spherical covariance matrices. *Automatica*, 51(0): 149 – 157, 2015.

[30] Nikos Zikos and Vassilios Petridis. L-SLAM: Reduced dimensionality FastSLAM with unknown data association. In *Robotics and Automation (ICRA), International Conference on*, pages 4074–4079. IEEE, 2011.

[31] Nikos Zikos and Vassilios Petridis. 6-DoF low dimensionality SLAM (L-SLAM). *Journal of Intelligent & Robotic Systems*, pages 1–18, 2014.

APPENDIX A
PROOF OF LEMMA 1

Proof: First note that $\mathbf{\Sigma}^{-\frac{1}{2}}$ is full rank. Then using the rank-nullity theorem it is easy to verify that $\tilde{\mathbf{H}}_1 = \mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{H}_1$ is full column rank if and only if $\mathbf{H}_1$ is full column rank. According to (7), $\mathbf{H}_1$ is full column rank if and only if $\mathbf{R}_\theta^\top \mathbf{A}_2^\top$ and $\mathbf{R}_\theta^\top \mathbf{A}_3^\top$ are full column rank. It is obvious that $\mathbf{R}_\theta$ is non-singular. The reduced incidence matrix $\mathbf{A}$ is full (row) rank if and only if the corresponding graph is (weakly) connected, which is the case in SLAM datasets. Consequently $\mathbf{H}_1$ is full (column) rank, regardless of the value of $\boldsymbol{\theta}$. ∎

APPENDIX B
COMPUTING $\mathbf{J}_{\text{vp}}$

The jth column of $\mathbf{J}_{\text{vp}}$ is given by:

$$\begin{aligned}
[\mathbf{J}_{\text{vp}}]_{\cdot,j} &= \frac{\partial \mathbf{r}_{\text{vp}}}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}[\mathbf{P}_{\boldsymbol{\theta}}^{\perp}(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})] \\
&= \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}^{\perp}}{\partial \theta_j}(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) + \mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})}{\partial \theta_j} \\
&= \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}^{\perp}}{\partial \theta_j}(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) - \mathbf{P}_{\boldsymbol{\theta}}^{\perp}[\tilde{\mathbf{H}}_2]_{\cdot,j} \\
&= -\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) - \mathbf{P}_{\boldsymbol{\theta}}^{\perp}[\tilde{\mathbf{H}}_2]_{\cdot,j} \qquad (26)
\end{aligned}$$

Now we only need to compute $\partial \mathbf{P}_{\boldsymbol{\theta}}/\partial\theta_j$. This term was first computed by Golub and Pereyra. We briefly mention their proof as stated in [25]. First note that $\mathbf{P}_{\boldsymbol{\theta}}$ is (i) idempotent, i.e., $(\mathbf{P}_{\boldsymbol{\theta}})^2 = \mathbf{P}_{\boldsymbol{\theta}}$, and (ii) symmetric. Also note that $\mathbf{P}_{\boldsymbol{\theta}}\tilde{\mathbf{H}}_1 = \tilde{\mathbf{H}}_1$. Therefore we have:

$$\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}\tilde{\mathbf{H}}_1}{\partial \theta_j} = \frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j} = \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\tilde{\mathbf{H}}_1 + \mathbf{P}_{\boldsymbol{\theta}}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j} \qquad (27)$$

Therefore:

$$\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\tilde{\mathbf{H}}_1 = (\mathbf{I} - \mathbf{P}_{\boldsymbol{\theta}})\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j} = \mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j} \qquad (28)$$

Multiplying both sides by $\tilde{\mathbf{H}}_1^{\dagger}$ from right we get:

$$\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\mathbf{P}_{\boldsymbol{\theta}} = \mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\tilde{\mathbf{H}}_1^{\dagger}. \qquad (29)$$

Also note that:

$$(\mathbf{P}_{\boldsymbol{\theta}}\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j})^\top = \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\mathbf{P}_{\boldsymbol{\theta}}. \qquad (30)$$

Now we use the properties of $\mathbf{P}_{\boldsymbol{\theta}}$ as an orthogonal projection:

$$\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j} = \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}^2}{\partial \theta_j} = \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\mathbf{P}_{\boldsymbol{\theta}} + \mathbf{P}_{\boldsymbol{\theta}}\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}, \qquad (31)$$

which can be simplified using (29) and (30):

$$\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j} = \mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\tilde{\mathbf{H}}_1^{\dagger} + (\mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\tilde{\mathbf{H}}_1^{\dagger})^\top \qquad (32)$$

Plugging (32) into (26) completes the proof:

$$\begin{aligned}
[\mathbf{J}_{\text{vp}}]_{\cdot,j} = &-\left[\mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\tilde{\mathbf{H}}_1^{\dagger} + (\mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\tilde{\mathbf{H}}_1^{\dagger})^\top\right](\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) \\
&- \mathbf{P}_{\boldsymbol{\theta}}^{\perp}[\tilde{\mathbf{H}}_2]_{\cdot,j}. \qquad (33)
\end{aligned}$$

APPENDIX C
ALGORITHM 1 (KAUFMAN'S MODIFICATION)

In this section we discuss some important details about implementing an efficient version of VP with Kaufman's modification using the QR decomposition of $\tilde{\mathbf{H}}_1$. These details are summarized in Algorithm 1. First we need to compute the *full* QR factorization of $\tilde{\mathbf{H}}_1$. Sparsity of $\tilde{\mathbf{H}}_1$ can be exploited in this stage by using optimized software packages such as SuiteSparseQR [7].

$$\tilde{\mathbf{H}}_1 = \mathbf{Q}\mathbf{R} = \begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix}\begin{bmatrix}\mathbf{R}_1 \\ \mathbf{0}\end{bmatrix}. \qquad (34)$$

Here $\mathbf{Q}_1 \in \mathbb{R}^{n_z \times n_p}$ and $\mathbf{Q}_2 \in \mathbb{R}^{n_z \times (n_z - n_p)}$ are orthogonal matrices, and $\mathbf{R}_1 \in \mathbb{R}^{n_p \times n_p}$ is an upper triangular matrix. The columns of $\mathbf{Q}_1$ and $\mathbf{Q}_2$ form orthonormal bases for $\text{range}(\tilde{\mathbf{H}}_1)$ and $\text{null}(\tilde{\mathbf{H}}_1^\top)$, respectively. Consequently $\mathbf{P}_{\boldsymbol{\theta}} = \mathbf{Q}_1\mathbf{Q}_1^\top$ and $\mathbf{P}_{\boldsymbol{\theta}}^{\perp} = \mathbf{Q}_2\mathbf{Q}_2^\top$. The residual vector of VP functional can be simplified using (18),

$$\mathbf{r}_{\text{vp}} = \mathbf{P}_{\boldsymbol{\theta}}^{\perp}(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) = \mathbf{Q}_2\mathbf{Q}_2^\top(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}). \qquad (35)$$

To evaluate the reduced NLS cost function $g(\cdot)$ we can use the QR decomposition of $\tilde{\mathbf{H}}_1$ in (34) and the fact that the Euclidean norm is invariant under orthogonal transformations. Therefore we have $g(\boldsymbol{\theta}) = \|\mathbf{Q}_2^\top(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})\|^2$. According to (17), the jth column of $\mathbf{J}_{\text{vp}}^{\text{K}}$ is given by

$$[\mathbf{J}_{\text{vp}}^{\text{K}}]_{\cdot,j} = -\mathbf{Q}_2\mathbf{Q}_2^\top\left(\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\mathbf{p}^\star(\boldsymbol{\theta}) + [\tilde{\mathbf{H}}_2]_{\cdot,j}\right). \qquad (36)$$

Note that for a given $\boldsymbol{\theta}$, $\mathbf{p}^\star(\boldsymbol{\theta})$ can be easily computed by solving $\mathbf{R}_1\,\mathbf{p}^\star(\boldsymbol{\theta}) = \mathbf{Q}_1^\top(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})$ by back substitution. It can be easily verified that multiplying both the residual vector and Jacobian matrix from left by an orthogonal matrix does not change the GN direction. Hence we can eliminate $\mathbf{Q}_2$ by multiplying (35) and (36) by $\mathbf{Q}_2^\top$ from left.

$$\begin{aligned}
\mathbf{Q}_2^\top\mathbf{r}_{\text{vp}} &= \mathbf{Q}_2^\top(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) \\
[\mathbf{Q}_2^\top\mathbf{J}_{\text{vp}}^{\text{K}}]_{\cdot,j} &= -\mathbf{Q}_2^\top\left(\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\mathbf{p}^\star(\boldsymbol{\theta}) + [\tilde{\mathbf{H}}_2]_{\cdot,j}\right). \qquad (37)
\end{aligned}$$

Note that $\mathbf{Q}_2$ could not be "eliminated" without using Kaufman's simplification.

APPENDIX D
DERIVATION OF $p(\mathbf{p} \mid \boldsymbol{\theta},\mathbf{z})$

$$\begin{aligned} p(\mathbf{p}|\boldsymbol{\theta},\mathbf{z}) &= \frac{p(\mathbf{p},\boldsymbol{\theta},\mathbf{z})}{p(\boldsymbol{\theta},\mathbf{z})} \\ &= \frac{p(\mathbf{z}|\mathbf{x})\,p(\mathbf{x})}{p(\boldsymbol{\theta},\mathbf{z})} \\ &= \frac{p(\mathbf{z}|\mathbf{x})\,p(\mathbf{p}|\boldsymbol{\theta})\,p(\boldsymbol{\theta})}{p(\mathbf{z}|\boldsymbol{\theta})\,p(\boldsymbol{\theta})} \\ &= \frac{p(\mathbf{z}|\mathbf{x})\,p(\mathbf{p}|\boldsymbol{\theta})}{p(\mathbf{z}|\boldsymbol{\theta})} \end{aligned} \tag{38}$$

APPENDIX E
$\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\circ}$ AND $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{\circ}$

From the fact that the measurement function is linear in $\mathbf{p}$ it readily follows that a Gaussian prior over $\mathbf{p}$ results in a Gaussian $p(\mathbf{p}|\boldsymbol{\theta},\mathbf{z}) = \mathcal{N}(\mathbf{p};\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\circ},\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{\circ})$. To simplify our notation and without loosing any generality let us assume $p(\mathbf{p}|\boldsymbol{\theta}) = p(\mathbf{p})$. As mentioned earlier, this prior is assumed to be Gaussian with mean $\boldsymbol{\mu}_{\mathbf{p}}$ and covariance $\boldsymbol{\Sigma}_{\mathbf{p}}$. Then we have,

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}^{\circ} = \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{\circ}\left(\mathbf{H}_1^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{z}-\mathbf{H}_2\boldsymbol{\theta}) + \boldsymbol{\Sigma}_{\mathbf{p}}^{-1}\boldsymbol{\mu}_{\mathbf{p}}\right) \tag{39}$$

$$= \boldsymbol{\mu}_{\mathbf{p}} + \boldsymbol{\Sigma}_{\mathbf{p}}\mathbf{H}_1^{\top}\mathbf{S}^{-1}\left(\mathbf{z}-\mathbf{H}_2\boldsymbol{\theta}-\mathbf{H}_1\boldsymbol{\mu}_{\mathbf{p}}\right) \tag{40}$$

in which we have used the matrix inversion lemma and $\mathbf{S}$ is the so-called innovation covariance,

$$\begin{aligned} \mathbf{S} &\triangleq \mathbf{H}_1\boldsymbol{\Sigma}_{\mathbf{p}}\mathbf{H}_1^{\top} + \boldsymbol{\Sigma}, \\ \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{\circ} &= \left(\mathbf{H}_1^{\top}\boldsymbol{\Sigma}^{-1}\mathbf{H}_1 + \boldsymbol{\Sigma}_{\mathbf{p}}^{-1}\right)^{-1}. \end{aligned} \tag{41}$$