A Sparse Separable SLAM Back-End

Kasra Khosoussi, Shoudong Huang, *Member, IEEE*, and Gamini Dissanayake, *Member, IEEE*

***Abstract*—We propose a scalable algorithm to take advantage of the separable structure of simultaneous localization and mapping (SLAM). Separability is an overlooked structure of SLAM that distinguishes it from a generic nonlinear least-squares problem. The standard relative-pose and relative-position measurement models in SLAM are affine with respect to robot and features' positions. Therefore, given an estimate for robot orientation, the conditionally optimal estimate for the rest of the state variables can be easily computed by solving a sparse linear least-squares 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 in conjunction with any Newton-based solver and is applicable to 2-D/3-D pose-graph and feature-based SLAM. 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 sparse back-ends.**

***Index Terms*—Simultaneous localization and mapping (SLAM) back-end, sparse separable nonlinear least squares, variable projection.**

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) has been investigated for more than two decades [14]. 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 [11]. It was revealed later that the smoothing formulation brings not only accuracy, but also scalability, through exploiting the sparse structure of SLAM. The most distinctive property of modern SLAM algorithms is the exploitation of this natural sparsity [10].

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 reduces to 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 other Newton-based algorithms treat the measurement function (residuals) as a generic smooth nonlinear function of states. However, the widely used measurement model associated with the range-bearing sensors in 2-D/3-D feature-based/pose-graph SLAM has a significant structure. This measurement function is affine in robot/landmark positions. Therefore, given the orientation of robot throughout its trajectory $\boldsymbol{\theta}$, the conditionally optimal estimate—in ML or MAP sense—for positions $\mathbf{p}$ can be computed by solving a (sparse) linear least-squares problem.

In the least-squares literature, NLS problems with partially linear residuals are called separable [16], [3]. In statistical terms, such problems are associated with conditionally linear-Gaussian state-space models [12] since estimating the linear variables, given the nonlinear ones and the measurements corrupted by a Gaussian noise, corresponds to a linear-Gaussian system.

Golub and Pereyra proposed the variable projection (VP) algorithm to solve general separable NLS problems [15]. The main idea behind VP is to explicitly eliminate the linear variables and solve the resulting reduced NLS problem (i.e., the nonlinear core of the problem). This technique has been applied to a wide range of applications. Both theoretically [30] and empirically, it has been shown that compared to solving the full NLS problem, VP techniques typically exhibit a faster convergence rate (see [16] and the references therein).

In this paper, we show how different VP algorithms can be applied to various forms of SLAM without any restrictive assumption on the structure of the noise covariance matrix. Unfortunately, these algorithms—in their existing forms—are incapable of maintaining the sparse structure of SLAM and, therefore, lead to solutions with cubic time and quadratic space complexity. To address this issue, we propose an equivalent VP algorithm that is capable of exploiting both separability and sparsity at the same time.

A. Contributions

The contributions made in this paper are summarized as follows.

1) Investigating an important, yet overlooked, structure of SLAM, as well as establishing the link between SLAM and the vast literature on separable NLS problems.
2) Proposing a new SLAM back-end that combines the advantages of exploiting sparsity and separability.
3) Providing new insights into the link between the separable NLS and conditionally linear-Gaussian problems.

This paper is an extension of our previous work [23]. In this work, we improve our algorithm in [23] by introducing the idea of conditional projection. We use this idea to quantify the gain achieved by exploiting the separable structure of SLAM. This ultimately allows us to decide whether the benefits of leveraging the separable structure of SLAM outweigh the computational

Manuscript received November 28, 2015; revised May 21, 2016; accepted August 9, 2016. Date of publication October 19, 2016; date of current version December 2, 2016. This paper was recommended for publication by Associate Editor M. Kaess and Editor C. Torras upon evaluation of the reviewers' comments. This work was supported by the Australian Research Council Discovery Project DP120102786.

The authors are with the Centre for Autonomous Systems, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: kasra.khosoussi@uts.edu.au; shoudong.huang@uts.edu.au; gamini.dissanayake@uts.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2016.2609394

cost of solving an extra linear system. Furthermore, in this paper, we describe the architecture of our new implementation that requires only a minor modification of the existing sparse back-ends. We also expand our experiments and discussion of the results to identify cases in which exploiting separability is more/less beneficial. The proofs, derivations, and implementation details that were omitted from [23] due to space limitation are included in this paper. Finally, our implementation[1] and Monte Carlo test suite[2] now are publicly available.

B. Outline

In Section II, we review the related works. In Section III, we provide a mathematical formulation of SLAM. In Section IV, we show how the VP algorithm can be applied to exploit the separable structure of SLAM. We propose a new algorithm in Section V to exploit both separability and sparsity. Furthermore, we look at the separable structure of SLAM from a new perspective by factorizing the SLAM posterior. We point out several implementation details in Section VI. The results obtained using both synthetic and real datasets are provided in Section VII. This is followed by the conclusion in Section VIII. Finally, the proofs are provided in the Appendix.

C. Notation

Throughout this paper, bold lowercase and uppercase letters are reserved for vectors and matrices, respectively. Sets are shown by uppercase letters. For convenience, our notation does not distinguish between random variables (e.g., $\mathbf{z}$) and their realizations. $|\mathcal{X}|$ denotes the cardinality of set $\mathcal{X}$. $\mathbf{I}$ and $\mathbf{0}$ are the identity and zero matrices with appropriate sizes, respectively. We use $\mathrm{vec}(\mathbf{q}_1, \ldots, \mathbf{q}_n)$ to denote the column vector obtained by stacking $\{\mathbf{q}_i\}_{i=1}^n$. $\mathbf{S}_1 \succ \mathbf{S}_2$ means $\mathbf{S}_1 - \mathbf{S}_2$ is positive definite. The Kronecker product is denoted by $\otimes$. The Euclidean norm is shown by $\|\cdot\|$. The weighted Euclidean norm of vector $\mathbf{e}$ with matrix $\mathbf{W} \succ \mathbf{0}$ is denoted by $\|\mathbf{e}\|_{\mathbf{W}} \triangleq \sqrt{\mathbf{e}^\top \mathbf{W}\, \mathbf{e}}$. We denote by $\mathrm{diag}(\mathbf{W}_1, \ldots, \mathbf{W}_k)$ the block-diagonal matrix with matrices $\mathbf{W}_1, \ldots, \mathbf{W}_k$ as blocks on its main diagonal. $\dim(\mathbf{x})$ denotes the length of vector $\mathbf{x}$. $\mathrm{range}(\mathbf{X})$ and $\mathrm{null}(\mathbf{X})$ are the column and null spaces of $\mathbf{X}$, respectively. $[n]$ is defined as $[n] \triangleq \{1, 2, \ldots, n\}$. Finally, $\mathrm{SO}(d)$ (special orthogonal group) is defined as

$$\mathrm{SO}(d) \triangleq \{\mathbf{R} \in \mathbb{R}^{d \times d} : \mathbf{R}^\top \mathbf{R} = \mathbf{I}_d, \det(\mathbf{R}) = 1\}.$$

II. Related Works

A theoretical analysis of the convergence properties of VP methods is due to Ruhe and Wedin [30], [31]. An extensive survey of VP applications can be found in [16]. The high-dimensional state space of SLAM is one of its distinctive features in comparison with many other applications of VP. This is the reason why retaining and exploiting sparsity is vital to the scalability of SLAM solvers.

[1]http://github.com/kasra/vp-g2o
[2]http://github.com/kasra/atlas

A common way of approaching conditionally linear-Gaussian state-space models is to use Rao–Blackwellized particle filters (RBPF) (see, e.g., Doucet [12]). RBPF typically uses the standard sequential importance resampling filter to represent $p(\boldsymbol{\theta}|\mathbf{z})$ using p weighted samples $\{\boldsymbol{\theta}^{[i]}\}_{i=1}^p$ drawn independently from a proposal distribution, i.e.

$$p(\boldsymbol{\theta}|\mathbf{z}) \approx \sum_{i=1}^{p} w_i \cdot \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^{[i]}) \tag{1}$$

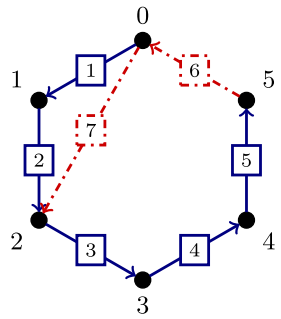
where δ is the Dirac delta function. Then, for each sample, the conditionally optimal estimate for $\mathbf{p}$ is recovered analytically by computing the mean of $p(\mathbf{p}|\boldsymbol{\theta}^{[i]}, \mathbf{z})$ using the Kalman filter (i.e., recursive linear least squares). It is worth noting that this approach naturally leads to the minimum mean-square-error point estimate (i.e., the mean of the posterior instead of its mode).

Zikos and Petridis in L-SLAM [37], [38] utilized this idea to exploit the separable structure of feature-based SLAM. In their RBPF, rather than partitioning the variables into landmarks and poses, which was used in FastSLAM [26], they divide the variables into $\boldsymbol{\theta}$ and $\mathbf{p}$. L-SLAM has two major drawbacks. First and foremost, any sequential Monte Carlo method employed in a high-dimensional state space will eventually suffer from degeneracy and, subsequently, sample impoverishment [13]. Sample impoverishment has a negative impact on estimating $\boldsymbol{\theta}$ due to the loss of sample diversity; after some time $t \gg t_0$, all of the particles will share the same estimate of $\{\theta_i\}_{i=1}^{t_0}$ for some t_0 (i.e., effectively only one sample is drawn from the corresponding region of state space). Furthermore, by partitioning variables based on $\mathbf{p}$ versus $\boldsymbol{\theta}$, the conditional independence property exploited in FastSLAM to gain computational efficiency will be lost.

Wang *et al.* [36] proposed to explicitly eliminate the linear variables of 2-D feature-based problems with isotropic noise to obtain a smaller optimization problem over $\boldsymbol{\theta}$. This approach is similar to Golub and Pereyra's VP [15], but with numerical differentiation and Newton iterations. Their algorithm is unable to maintain the sparse structure of the problem. Consequently, the algorithm proposed in [36] is computationally much more expensive than the standard sparse NLS solvers.

LAGO [5], [6] uses the separable structure of 2-D pose-graph SLAM to bootstrap GN. First, a refined estimate for $\boldsymbol{\theta}$ is computed by only considering the relative rotational measurements. Using this initial estimate of $\boldsymbol{\theta}$, LAGO then recovers the conditionally optimal estimate for $\mathbf{p}$. The outcome is used to initialize 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 noise components [6], [19]). Unlike our algorithm, LAGO is limited to 2-D pose graphs.

The equivalence between the minima of the original optimization problem and those of the reduced problem makes it possible to study various properties of SLAM by examining the reduced problem. For instance, [34], [35] analyze the number of local minima in some small special cases based on this idea.



$$\tilde{\mathbf{A}} = \left[\begin{array}{ccccc|cc} -1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \end{array}\right]$$

Fig. 1. Graph $\mathcal{G} = ([5] \cup \{0\}, \mathcal{E})$ depicted represents a pose-graph SLAM problem. The blue solid edges correspond to the odometry measurements. The red edges show the loop closures. $\tilde{\mathbf{A}}$ is the incidence matrix of this graph. Each column of $\tilde{\mathbf{A}}$ corresponds to an edge, while its rows correspond to the vertices. The reduced incidence matrix $\mathbf{A}$ after anchoring a vertex $v_\perp$ (e.g., $v_\perp = 0$) is obtained by removing the corresponding row from $\tilde{\mathbf{A}}$.

Similarly, Carlone [4] uses the reduced problem to analyze the convergence of GN in 2-D pose graphs with isotropic noise.

III. Problem Formulation and Preliminaries

A. Graphical Representation of Simultaneous Localization and Mapping

SLAM problems can be naturally represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In this paper, $\mathcal{V} = [n] \cup \{0\}$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and $|\mathcal{E}| = m$. There is a one-to-one correspondence between the vertices of $\mathcal{G}$ and the state variables of the SLAM problem. Each edge represents the relative measurement between the corresponding vertices. Therefore, $(i, j) \in \mathcal{E}$ corresponds to the relative measurement from $\mathbf{x}_i$ to $\mathbf{x}_j$. A simple example is shown in Fig. 1. Due to the relative nature of measurements in SLAM, we must define a global reference frame 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.

Remark 1: It is easy to see that the SLAM problem will be an ill-posed problem if $\mathcal{G}$ is not weakly connected. Any SLAM problem with $c \geq 2$ connected components is essentially equivalent to c separate SLAM problems with respect to c independent reference frames. In such cases, the relative transformation between the frames will be unobservable. Thus, without loss of generality, we can assume that $\mathcal{G}$ is weakly connected.

The reduced incidence matrix of $\mathcal{G}$ after anchoring $\mathbf{x}_0$ (i.e., deleting the corresponding row from the original incidence matrix $\tilde{\mathbf{A}}$) is denoted by $\mathbf{A} \in \{-1, 0, 1\}^{n \times m}$. Let $e_k \triangleq (i_k, j_k) \in \mathcal{E}$ be the kth edge. Then, $\tilde{\mathbf{A}}_{i_k+1,k} = -1$ and $\tilde{\mathbf{A}}_{j_k+1,k} = 1$. The remaining elements of $\tilde{\mathbf{A}}$ are all zero. See Fig. 1 for an example. 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 $\mathbf{A}_\circ$ can be uniquely determined by the number of poses n_p (see the left block of $\tilde{\mathbf{A}}$ in Fig. 1). The ℓ-expansion of $\mathbf{A}$ refers to $\mathbf{A}_\ell \triangleq \mathbf{A} \otimes \mathbf{I}_\ell$ for $\ell \in \mathbb{Z}_{\geq 2}$. The reduced weighted incidence matrix $\mathbf{A}_w$ is defined with respect to a given positive weight function $w : \mathcal{E} \to \mathbb{R}_{>0}$. Let $\mathbf{W} \triangleq \operatorname{diag}(w(e_1), \ldots, w(e_m)) \in \mathbb{R}^{m \times m}$ denote the weight matrix. Then, $\mathbf{A}_w \triangleq \mathbf{A}\mathbf{W}^{1/2}$.

B. Measurement Model

Conventionally, the state vector in SLAM is defined as $\operatorname{vec}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. For reasons that will become clear shortly, we permute the standard state vector and define our state vector as $\mathbf{x} \triangleq \operatorname{vec}(\mathbf{p}, \boldsymbol{\theta})$. In 2-D 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 $\boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{ij})$

$$\mathbf{z}_{ij} = \mathbf{h}_{ij}(\mathbf{x}_i, \mathbf{x}_j) + \boldsymbol{\epsilon}_{ij}. \tag{2}$$

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

$$\mathbf{h}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \begin{bmatrix} \delta x_{ij} \\ \delta y_{ij} \end{bmatrix} = \mathbf{R}(\theta_i)^\top (\mathbf{p}_j - \mathbf{p}_i), & \text{if } j \in \mathcal{S}_f \\ \begin{bmatrix} \delta x_{ij} \\ \delta y_{ij} \\ \delta \theta_{ij} \end{bmatrix} = \begin{bmatrix} \mathbf{R}(\theta_i)^\top (\mathbf{p}_j - \mathbf{p}_i) \\ \operatorname{wrap}(\theta_j - \theta_i) \end{bmatrix}, & \text{if } j \in \mathcal{S}_p \end{cases} \tag{3}$$

where $\mathbf{R}(\theta_i) \in \mathrm{SO}(2)$ is the rotation matrix corresponding to θ_i, $\operatorname{wrap} : \mathbb{R} \to [-\pi, \pi)$ 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

$$\mathbf{R}_\theta \triangleq \operatorname{diag}\left(\mathbf{R}(\theta_{k_1}), \ldots, \mathbf{R}(\theta_{k_m})\right). \tag{4}$$

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 $\mathbf{z} \triangleq \operatorname{vec}(\mathbf{z}_p, \mathbf{z}_\theta)$. Similarly, the stacked vector of noise variables and its covariance matrix are denoted by $\boldsymbol{\epsilon} \triangleq \operatorname{vec}(\boldsymbol{\epsilon}_p, \boldsymbol{\epsilon}_\theta)$ and $\boldsymbol{\Sigma}$, respectively. The measurement model can be expressed in a compact form as

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \tag{5}$$

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mathbf{h}(\mathbf{x}), \boldsymbol{\Sigma}). \tag{6}$$

Remark 2: Note that (3) 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.

Remark 3: As the title suggests, in this paper, we focus on developing a new SLAM back-end to exploit the separable structure of SLAM. Therefore, we assume that the data association is given. Solving the data association problem can be challenging and thus needs to be dealt with separately in the SLAM front-end [17]. It is also common to make the back-end robust in order to deal with any remaining false positive associations (see, e.g.,

[1] and [33]). The separable structure of SLAM is preserved in such robust formulations.

According to (3), the stacked measurement function of 2-D SLAM is given by

$$\mathbf{h}(\mathbf{x}) = \mathbf{H}(\boldsymbol{\theta})\,\mathbf{x} \triangleq \begin{bmatrix} \mathbf{R}_\theta^\top \mathbf{A}_2^\top & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Lambda}^\top \end{bmatrix} \mathbf{x}$$

$$\boldsymbol{\Lambda} = \begin{cases} \mathbf{A}, & \text{pose graph} \\ \mathbf{A}_\circ, & \text{feature based} \end{cases} \tag{7}$$

where $\mathbf{A}_2 \triangleq \mathbf{A} \otimes \mathbf{I}_2$. We assume that the correct regularization terms are applied to the measurements [5]. Note that (7) can be rewritten as $\mathbf{h}(\mathbf{x}) = \mathbf{H}_1(\boldsymbol{\theta})\mathbf{p} + \mathbf{H}_2\boldsymbol{\theta}$, where

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

Although $\mathbf{H}_1(\boldsymbol{\theta})$ depends on $\boldsymbol{\theta}$, hereafter, we drop the argument of $\mathbf{H}_1$ for convenience.

Remark 4: The standard relative pose-pose and pose-point measurement models in 3-D SLAM are also affine with respect to robot's and landmarks' positions. The stacked vector of noise-free translational measurements in 3-D SLAM can be written as $\mathbf{R}_\theta^\top \mathbf{A}_3^\top \mathbf{p}$, where $\mathbf{p}$ is the stacked vector of positions, $\mathbf{A}_3 \triangleq \mathbf{A} \otimes \mathbf{I}_3$, and $\mathbf{R}_\theta$ is a block-diagonal matrix similar to (4) but with 3×3 rotation matrices. Note that rotational measurements do not depend on $\mathbf{p}$. To simplify our notation and without loss of generality, we mainly focus on the 2-D SLAM measurement model in the following sections. We will revisit the case of 3-D SLAM in Remark 6 and explain how our algorithm can be generalized to 3-D pose-graph and feature-based SLAM. Although in this paper we do not consider a specific choice of sensors, the use of inertial sensor in 3-D SLAM is quite common [25]. Therefore, it is worth noting that inertial measurements do not violate the separable structure of SLAM as such measurements are affine in $\mathbf{p}$ (see e.g., [25]).

Remark 5: This paper investigates pose-graph and feature-based SLAM problems with standard relative pose-pose and pose-point measurements. Such measurements can be obtained from range-bearing sensors after a "reparameterization" of the original measurements or matching two scans. These models have become standard choices over the past decade (see, e.g., [17]). Unfortunately, this process often involves a nonlinear transformation of the original data. Thus, the computed covariance matrices are affected by the linearization error, and furthermore, the additive Gaussian noise assumption may not be valid anymore for the new model. In the context of this work, such transformations can be thought as a reparameterization performed to introduce separability. Note that directly using range-bearing measurements does not result in a separable NLS.

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 $\hat{\mathbf{x}}_{\text{MAP}}$ is the maximizer of the posterior density

$$\hat{\mathbf{x}}_{\text{MAP}} = \arg\max_{\mathbf{x}} \; p(\mathbf{x}|\mathbf{z}) = \arg\max_{\mathbf{x}} \; p(\mathbf{z}|\mathbf{x})\; p(\mathbf{x}). \tag{9}$$

For convenience, we minimize the negative log-posterior

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

In the absence of an informative prior over $\mathbf{x}$,[3] 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) results in $\hat{\mathbf{x}}_{\text{ML}}$

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

Our emphasis in this paper is mostly 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 $\hat{c}_{\text{MAP}}$. Similarly, we denote the (full) NLS cost function by $f(\mathbf{p}, \boldsymbol{\theta})$, where f could be the cost function appeared either in (10) or (11).

IV. Exploiting Separability in Simultaneous Localization and Mapping

In the previous section, we showed that, under some standard assumptions, finding the ML/MAP estimates in SLAM boils down to solving an NLS problem. By further inspection of (8), one can see that the nonlinearity is due to the rotational components. Hence, given the robot orientation $\boldsymbol{\theta}$ throughout its trajectory, measurements are affine with respect to the robot and features' positions $\mathbf{p}$. Consequently, the ML and MAP estimates for $\mathbf{p}$ can be computed by solving linear least-squares problems. Such problems are often called separable NLS [16]. This special structure distinguishes SLAM from a generic NLS problem. In this section, we show how this structure can be exploited in SLAM.

A. Variable Projection

VP is an algorithm proposed by Golub and Pereyra to exploit this structure [15]. Here, we explain how their approach can be applied to SLAM. Golub and Pereyra 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 [15], Th. 2.1).

1) Find $\mathbf{p}^\star(\boldsymbol{\theta})$, the conditionally optimal estimate for $\mathbf{p}$ as a function of $\boldsymbol{\theta}$ by minimizing the original cost function in $\mathbf{p}$.
2) 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.

1) Phase I—$\mathbf{p}^\star(\boldsymbol{\theta})$: Consider 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 Euclidean

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

norm minimization in (11) as the following unweighted least squares:

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

where $\tilde{\mathbf{z}} \triangleq \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{z}$, $\tilde{\mathbf{H}}_1 \triangleq \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{H}_1$, and $\tilde{\mathbf{H}}_2 \triangleq \boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{H}_2$. Lemma 1 follows directly from the fact that the incidence matrix of a weakly connected directed graph is full rank.

Lemma 1: For the measurement models defined in Section III, if the corresponding graph is weakly connected, $\tilde{\mathbf{H}}_1$ is full column rank regardless of $\boldsymbol{\theta}$.

Proof: See Appendix A for the proof. ■

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

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

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

where $\tilde{\mathbf{H}}_1^\dagger \triangleq (\tilde{\mathbf{H}}_1^\top\tilde{\mathbf{H}}_1)^{-1}\tilde{\mathbf{H}}_1^\top$ is the Moore–Penrose pseudoinverse of $\tilde{\mathbf{H}}_1$ (see, e.g., [3]).

2) Phase II—Reduced Nonlinear Least Squares: By substituting $\mathbf{p}$ in the original objective function (12) with $\mathbf{p}^\star(\boldsymbol{\theta})$ in (14) and solving the 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 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 $\mathbf{r}_{\mathrm{vp}} \triangleq \mathbf{P}_\theta^\perp(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})$.

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., $\mathbf{J} \triangleq \frac{\partial}{\partial\mathbf{x}}\mathbf{r}$, where $\mathbf{r} \triangleq \mathbf{z} - \mathbf{h}(\mathbf{x})$. Although it is possible to approximate $\mathbf{J}_{\mathrm{vp}}$ using finite differences (see [16] and the references therein), here, we use the exact analytical expression for $\mathbf{J}_{\mathrm{vp}}$, which is based on the work of [15]. The only difference between (15) and the case that was originally examined by [15] is the extra linear term $-\tilde{\mathbf{H}}_2\,\boldsymbol{\theta}$.

Theorem 1: The jth column of $\mathbf{J}_{\mathrm{vp}}$ is given by

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

Proof: See Appendix B. ■

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 more expensive than GN iterations on the full problem. Hence, solving the reduced problem does not necessarily improve the overall runtime.

Algorithm 1: SLAM Solver Based on [22].

1: **repeat**
2: Compute the full QR factorization of $\tilde{\mathbf{H}}_1$ (18)
3: Recover $\mathbf{p}^\star$ by solving
$$\tilde{\mathbf{R}}_1\,\mathbf{p}^\star(\boldsymbol{\theta}_{(i)}) = \mathbf{Q}_1^\top(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}_{(i)})$$
4: Compute the modified residual and Jacobian (21)
5: Construct the normal equations for the reduced problem
6: Solve the normal equations to obtain $\delta\boldsymbol{\theta}_{(i)}$
7: $\boldsymbol{\theta}_{(i+1)} \leftarrow \boldsymbol{\theta}_{(i)} + \delta\boldsymbol{\theta}_{(i)}$
8: **until** convergence
9: $\mathbf{p}^\star \leftarrow \mathbf{p}^\star(\boldsymbol{\theta}^\star)$ according to (14)

B. Kaufman's Algorithm

Kaufman [22] proposed to approximate the jth column of $\mathbf{J}_{\mathrm{vp}}$ according to

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

In small-residual problems, the impact of the term neglected in $\mathbf{J}_{\mathrm{vp}}^{\mathrm{K}}$ on the convergence rate of the algorithm will be negligible, as shown in [22]. This is rigorously investigated in [30] and has been verified empirically in various domains [16]. In the case of SLAM, the residuals will be "small" if the measurements are "sufficiently consistent" with each other, or equivalently, when the realized noise is "sufficiently small." It is worth noting that even GN, as an approximation of Newton's method, relies on a small-residual assumption [3].

Kaufman's simplification reduces the time per iteration of VP up to 25% [30]. As a result, Kaufman's method has become the preferred algorithm for solving separable NLS problems [16]. As we will see shortly, in sparse separable NLS problems (such as SLAM), Kaufman's modification, in a slightly different form, plays an even more crucial role by enabling us to preserve the sparse structure of the problem.

Algorithm 1 summarizes a SLAM solver based on an efficient implementation of VP using Kaufman's modification. In the rest of this section, we discuss some important details about implementing an efficient version of VP with Kaufman's modification using the QR decomposition of $\tilde{\mathbf{H}}_1$. First, we need to compute the full QR factorization of $\tilde{\mathbf{H}}_1$

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

The sparse structure of $\tilde{\mathbf{H}}_1$ can be exploited in this stage by using optimized software packages such as SuiteSparseQR [9]. Define $d_z \triangleq \dim(\mathbf{z})$ and $d_p \triangleq \dim(\mathbf{p})$. Then, $\mathbf{Q}_1 \in \mathbb{R}^{d_z \times d_p}$ and $\mathbf{Q}_2 \in \mathbb{R}^{d_z \times (d_z - d_p)}$ are orthogonal matrices, and $\tilde{\mathbf{R}}_1 \in \mathbb{R}^{d_p \times d_p}$ is an upper triangular matrix. 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. 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)

$$\mathbf{r}_{\mathrm{vp}} = \mathbf{P}_{\boldsymbol{\theta}}^{\perp}(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) = \mathbf{Q}_2\mathbf{Q}_2^{\top}(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}). \tag{19}$$

To evaluate the reduced NLS cost function g, we can use the QR decomposition of $\tilde{\mathbf{H}}_1$ in (18) 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 $\mathbf{J}_{\mathrm{vp}}^{\mathrm{K}}$ is given by

$$[\mathbf{J}_{\mathrm{vp}}^{\mathrm{K}}]_{\cdot,j} = -\mathbf{Q}_2\mathbf{Q}_2^{\top}\left(\frac{\partial\tilde{\mathbf{H}}_1}{\partial\theta_j}\mathbf{p}^{\star}(\boldsymbol{\theta}) + [\tilde{\mathbf{H}}_2]_{\cdot,j}\right). \tag{20}$$

Note that for a given $\boldsymbol{\theta}$, $\mathbf{p}^{\star}(\boldsymbol{\theta})$ can be easily computed by solving $\tilde{\mathbf{R}}_1\,\mathbf{p}^{\star}(\boldsymbol{\theta}) = \mathbf{Q}_1^{\top}(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})$ 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 (19) and (20) by $\mathbf{Q}_2^{\top}$ from left

$$\begin{aligned}\mathbf{Q}_2^{\top}\mathbf{r}_{\mathrm{vp}} &= \mathbf{Q}_2^{\top}(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})\\ \left[\mathbf{Q}_2^{\top}\mathbf{J}_{\mathrm{vp}}^{\mathrm{K}}\right]_{\cdot,j} &= -\mathbf{Q}_2^{\top}\left(\frac{\partial\tilde{\mathbf{H}}_1}{\partial\theta_j}\mathbf{p}^{\star}(\boldsymbol{\theta}) + [\tilde{\mathbf{H}}_2]_{\cdot,j}\right).\end{aligned} \tag{21}$$

Note that $\mathbf{Q}_2$ cannot be "eliminated" in the original VP (i.e., without using Kaufman's simplification of the Jacobian).

V. Sparse Variable Projection

At first glance, it may seem that applying Algorithm 1 can be computationally advantageous, as it reduces the size of the normal equations from $(2n + n_p)$ to only n_p. However, the linear system that emerges from the reduced problem in SLAM is generally dense. Hence, VP as implemented in Algorithm 1 does not lead to a scalable algorithm, as we would need at least $O(n_p^2)$ space and $O(n_p^3)$ time per iteration to solve the resulting dense linear system. Hence, it is sensible to ask ourselves whether we can take advantage of separability without giving up the intrinsic sparse structure of SLAM.

Barham and Drane [2] proposed an intuitive algorithm to solve separable NLS problems. Unlike the VP algorithm, in their approach, the linear variables are not entirely eliminated from the optimization problem. Instead, they use the Schur complement in each iteration to solve the normal equations of the original full NLS only for the nonlinear variables. Then, instead of back-substituting the resulting estimate into the normal equations, they exploit the separable structure of the problem by computing the conditionally optimal estimate for the linear variables. This process is repeated until convergence. This intuitive algorithm may first seem like a simple heuristic. However, it has been shown that this procedure leads to the same iterations as the Kaufman's algorithm [28].

Now, if the original system of normal equations is sparse (as in SLAM), computing the Schur complement, in general, can destroy the sparsity. Nevertheless, instead of computing the Schur complement, we can simply solve the normal equations for both linear and nonlinear variables and then replace the resulting estimate for the linear ones with the conditionally optimal values (14). By doing so, the sparse structure of the problem is preserved in the process of exploiting separability. In what follows, we explain how this idea can be applied to SLAM.

A. Retaining Sparsity

Consider the normal equations of the original problem

$$\tilde{\mathbf{J}}^{\top}\tilde{\mathbf{J}}\,\delta\mathbf{x}_{(i)} = -\tilde{\mathbf{J}}^{\top}\tilde{\mathbf{r}}. \tag{22}$$

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 $\mathbb{I} \triangleq \tilde{\mathbf{J}}^{\top}\tilde{\mathbf{J}}$ be the approximated Hessian. Note that $\tilde{\mathbf{J}}$ can be divided into two blocks: $\tilde{\mathbf{J}} = [\tilde{\mathbf{J}}_p \;\; \tilde{\mathbf{J}}_\theta]$, where $\tilde{\mathbf{J}}_p \triangleq \frac{\partial}{\partial\mathbf{p}}\tilde{\mathbf{r}}$ and $\tilde{\mathbf{J}}_\theta \triangleq \frac{\partial}{\partial\boldsymbol{\theta}}\tilde{\mathbf{r}}$. Therefore, we can expand (22) as

$$\begin{bmatrix}\mathbb{I}_p & \mathbb{I}_{p,\theta}\\ \mathbb{I}_{p,\theta}^{\top} & \mathbb{I}_\theta\end{bmatrix}\begin{bmatrix}\delta\mathbf{p}_{(i)}\\ \delta\boldsymbol{\theta}_{(i)}\end{bmatrix} = \begin{bmatrix}-\tilde{\mathbf{J}}_p^{\top}\tilde{\mathbf{r}}\\ -\tilde{\mathbf{J}}_\theta^{\top}\tilde{\mathbf{r}}\end{bmatrix}. \tag{23}$$

Barham and Drane [2] proposed to eliminate $\delta\mathbf{p}_{(i)}$ from (23) using the Schur complement of $\mathbb{I}$ with respect to $\mathbb{I}_p$

$$\left(\mathbb{I}_\theta - \mathbb{I}_{p,\theta}^{\top}\mathbb{I}_p^{-1}\mathbb{I}_{p,\theta}\right)\delta\boldsymbol{\theta}_{(i)} = \left(-\tilde{\mathbf{J}}_\theta^{\top} + \mathbb{I}_{p,\theta}^{\top}\mathbb{I}_p^{-1}\tilde{\mathbf{J}}_p^{\top}\right)\tilde{\mathbf{r}}. \tag{24}$$

Solving (24) results in $\delta\boldsymbol{\theta}_{(i)}$. Back-substituting this solution into (23) and recovering $\delta\mathbf{p}_{(i)}$ leads to the standard GN direction for the original cost function. Instead, Barham and Drane proposed to pair $\boldsymbol{\theta}_{(i+1)} = \boldsymbol{\theta}_{(i)} + \delta\boldsymbol{\theta}_{(i)}$ with the conditionally optimal estimate for $\mathbf{p}_{(i+1)}$ [2]. This estimate can be computed by solving the following sparse linear system:

$$(\tilde{\mathbf{H}}_1^{\top}\tilde{\mathbf{H}}_1)\,\mathbf{p}_{(i+1)} = \tilde{\mathbf{H}}_1^{\top}\left(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}_{(i+1)}\right). \tag{25}$$

It is important to note that here $\tilde{\mathbf{H}}_1$ is evaluated at $\boldsymbol{\theta}_{(i+1)}$. Repeating this procedure until convergence leads to a sequence of steps along which the cost function (11) has a zero gradient with respect to $\mathbf{p}$. This algorithm is summarized in Algorithm 2.

As mentioned earlier, the Schur complement in the reduced linear system (24) is generally dense. Needless to say, in each iteration of Algorithm 2, the solution of the reduced system (24) is identical to the $\delta\boldsymbol{\theta}_{(i)}$ obtained by solving the full system (23). Unlike the Schur complement, the full system of normal equations is sparse. Hence, instead of eliminating linear variables based on the Schur complement, the same result can be achieved by solving the sparse full system (23), discarding the obtained $\delta\mathbf{p}_{(i)}$ and instead computing the conditionally optimal $\mathbf{p}_{(i+1)}$ according to (25). This procedure is summarized in Algorithm 3.

It is of utmost importance to note that our proposed algorithm produces (mathematically) identical steps to those of Algorithms 1 and 2 (see [28] for the equivalence between Kaufman's method and the algorithm proposed by Barham and Drane [2]). However, unlike those algorithms, Algorithm 3 only requires solving two sparse linear systems in each iteration, which leads to a crucial computational benefit:

1) *Full step*: The first sparse linear system is the normal equations of the original full NLS problem (23). Solving this system results in $\delta\boldsymbol{\theta}_{(i)}$ and $\boldsymbol{\theta}_{(i+1)}$ consequently.

TABLE I
SUMMARY OF RELATED ALGORITHMS SORTED BY THEIR COST PER ITERATION IN DESCENDING ORDER

Algorithm	Separability	Sparsity
Algorithm 1 [22]	✓	✗
Algorithm 2 [2]	✓	✗
Algorithm 3	✓	✓
Full Least Squares	✗	✓

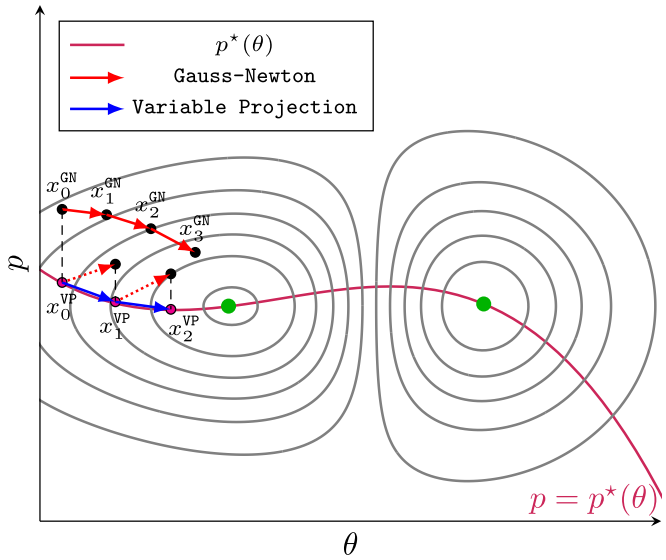


Fig. 2. In this figure, we see the contour lines of a simplified version of SLAM cost function. The green points show the local minima. The magenta curve is $p^\star(\theta)$, a function that maps any given θ to the corresponding conditionally optimal estimate for p. The blue vector shows the ith step of Algorithm 1—which is identical to the ith step of Algorithms 2 and 3. The red vector is the GN step obtained by starting from x_i^{GN}. The dashed red vectors are the intermediate GN steps obtained by starting from x_i^{VP}. 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 quadratic space and cubic time complexity. By contrast, our indirect approach in Algorithm 3 enables us to preserve the sparse structure of the original problem.

2) *Projection*: In the second sparse linear system, we recover the conditionally optimal estimate $\mathbf{p}_{(i+1)}$ by solving (25). This is where we exploit the separable structure of SLAM.

Our algorithm benefits from the advantages of both Algorithms 1 and 2. First and foremost, the equivalence between Algorithms 1 and 3 (through their equivalence to Algorithm 2 [28]) provides a rigorous justification for our approach and connects us to the rich literature on VP and the performance of Kaufman's algorithm. Furthermore, the equivalence between Algorithms 2 and 3, besides providing an intuitive interpretation for Kaufman's approximation, enables us to preserve the sparse structure of the problem. Algorithm 3 can be easily implemented by a simple modification of the existing state-of-the-art back-ends: we only need to solve an extra sparse linear system (25) (e.g., using a sparse Cholesky solver). Table I provides a summarized comparison between these algorithms, sorted by their cost per iteration in descending order. An efficient implementation of our algorithm is discussed in Section VI. Fig. 2 pictorially illustrates the proposed algorithm.

Remark 6: As mentioned in Remark 4, our approach can be easily extended to 3-D pose-graph and feature-based SLAM with the standard relative pose-pose and pose-point measurement models. Without loss of generality, let us assume that the translational and rotational noise components are uncorrelated. With a little abuse of notation, the cost function in these problems can be expressed as

$$f(\mathbf{p}, \boldsymbol{\theta}) = f_p(\mathbf{p}, \boldsymbol{\theta}) + f_\theta(\boldsymbol{\theta}) \tag{26}$$

where $f_p(\mathbf{p}, \boldsymbol{\theta}) \triangleq \|\mathbf{z}_p - \mathbf{R}_\theta^\top \mathbf{A}_3^\top \mathbf{p}\|^2_{\boldsymbol{\Sigma}_p^{-1}}$ and

$$f_\theta(\boldsymbol{\theta}) \triangleq \sum_{(i,j)\in\mathcal{E}_p} \|\text{Log}(\mathbf{R}_{z_{ij}}^\top \mathbf{R}_i^\top \mathbf{R}_j)\|^2_{\boldsymbol{\Sigma}_{\theta_{ij}}^{-1}} \tag{27}$$

where $\text{Log} : \text{SO}(3) \to \mathbb{R}^3$ maps a 3×3 rotation matrix to the corresponding rotation vector (i.e., axis-angle representation), $\mathcal{E}_p \subseteq \mathcal{E}$ is the subset of edges that correspond to pose-pose measurements, $\mathbf{R}_{z_{ij}}$ is the observed rotational measurement (corrupted by noise), $\boldsymbol{\Sigma}_{\theta_{ij}}$ is the corresponding covariance matrix, and $\mathbf{R}_i \in \text{SO}(3)$ is the rotation matrix of the ith robot pose. The state-of-the-art sparse back-ends use a minimal parameterization (e.g., rotation vector) for $\boldsymbol{\theta}$ in each iteration of the standard iterative schemes such as GN. Let $\boldsymbol{\theta}_{(k+1)}$ be the estimate obtained at the $(k+1)$th iteration of GN using the update rule $\boldsymbol{\theta}_{(k+1)} = \boldsymbol{\theta}_{(k)} \boxplus \delta\boldsymbol{\theta}_{(k)}$. Here, $\boxplus$ generalizes $+$ and can be defined based on, for example, the exponential map for $\text{SO}(3)$

Algorithm 2: SLAM Solver Based on [2].

Repeat
  Construct the normal equations (23)
  Eliminate $\delta\mathbf{p}_{(i)}$ using the Schur complement (24)
  Solve (24) to obtain $\delta\boldsymbol{\theta}_{(i)}$
  $\boldsymbol{\theta}_{(i+1)} \leftarrow \boldsymbol{\theta}_{(i)} + \delta\boldsymbol{\theta}_{(i)}$
  until convergence
$\mathbf{p}_{(i+1)} \leftarrow \mathbf{p}^\star(\boldsymbol{\theta}_{(i+1)})$ according to (25)

[18]. Now, given $\boldsymbol{\theta}_{(k+1)}$, the conditionally optimal estimate for $\mathbf{p}$ that minimizes (26) is the solution of

$$\tilde{\mathbf{H}}_1^{\circ\top} \tilde{\mathbf{H}}_1^\circ \, \mathbf{p}_{(k+1)} = \tilde{\mathbf{H}}_1^{\circ\top} \mathbf{z}_p \tag{28}$$

where $\tilde{\mathbf{H}}_1^\circ \triangleq \boldsymbol{\Sigma}_p^{-\frac{1}{2}} \mathbf{R}_\theta^\top \mathbf{A}_3^\top$, in which $\mathbf{R}_\theta$ is evaluated at $\boldsymbol{\theta}_{(k+1)}$. The proof of Lemma 1 can be easily extended to show that $\tilde{\mathbf{H}}_1^\circ$ is always full rank if the underlying graph is weakly connected.

Algorithm 3: Sparse Variable Projection.

repeat
  Construct the normal equations (23)
  Use the sparse Cholesky solver to solve (23)
  $\boldsymbol{\theta}_{(i+1)} \leftarrow \boldsymbol{\theta}_{(i)} + \delta\boldsymbol{\theta}_{(i)}$
  $\mathbf{p}_{(i+1)} \leftarrow \mathbf{p}^\star(\boldsymbol{\theta}_{(i+1)})$ according to (25) using the sparse Cholesky solver
until convergence

B. Projection Gain

Algorithm 3 and the conventional approach can be seen as the two ends of a spectrum of solvers. On one end, conventional methods provide cheap sparse iterations without exploiting the separable structure of the problem. On the other end of this spectrum, Algorithm 3 performs an extra projection step at the end of every iteration, which increases the effectiveness of each iteration at the cost of solving an extra (but smaller) sparse linear system (25) per iteration. Our empirical observations indicate that performing projection is crucial in the first few iterations, where it also exhibits a bootstrapping effect [19]. After few iterations and upon getting sufficiently close to a local minimizer, the effectiveness of f-iterations become similar to that of VP iterations. This is where allocating resources to the projection step is not justified anymore, and one can safely switch back to cheap f-iterations.

To design a switching scheme, first, it is necessary to quantify the concept of projection gain. At the ith iteration, we have

$$f_i^\star \triangleq f(\mathbf{p}^\star(\boldsymbol{\theta}_{(i)}), \boldsymbol{\theta}_{(i)}) = \min_{\mathbf{p}} f(\mathbf{p}, \boldsymbol{\theta}_{(i)}) \leq f(\mathbf{p}_{(i)}, \boldsymbol{\theta}_{(i)}) \triangleq f_i^\circ. \tag{29}$$

Here, $\mathbf{p}_{(i)}$ and $\boldsymbol{\theta}_{(i)}$ specify the solution obtained after performing a GN iteration at iteration i on the full cost function (see the black point connected to, e.g., x_0^{VP} via the red-dashed vector in Fig. 2) and f_i° is the value of (the original) cost function at this point. Similarly, $\mathbf{p}^\star(\boldsymbol{\theta}_{(i)})$ and $\boldsymbol{\theta}_{(i)}$ specify the solution obtained after performing the projection step (see x_1^{VP} in Fig. 2), whose cost is denoted by $f_i^\star$. To quantify the projection gain, we define the relative gain as

$$\gamma_i \triangleq (f_i^\circ - f_i^\star)/f_i^\circ. \tag{30}$$

By definition, $0 \leq \gamma_i \leq 1$. Note that computing the projection gain only requires an extra evaluation of the cost function f before performing the projection step. At $i = 0$, γ_i is usually close to 1 (which indicates a high projection gain) unless the initial guess is already close to a local minimum. Generally speaking, γ_i exhibits a decreasing (but not necessarily monotonic) trend. Upon convergence to a solution, the gain will diminish, and therefore, $\gamma_i = 0$ (see Fig. 3). Therefore, a sensible switching heuristic is to perform projections only, while γ_i is larger than a threshold, γ_T.

C. Maximum a Posteriori Estimation

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 perspective 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

$$p(\mathbf{p}, \boldsymbol{\theta}|\mathbf{z}) = p(\boldsymbol{\theta}|\mathbf{z})\, p(\mathbf{p}|\boldsymbol{\theta}, \mathbf{z}). \tag{31}$$

Using the Bayes rule as shown in Appendix C, we have $p(\mathbf{p}|\boldsymbol{\theta}, \mathbf{z}) \propto p(\mathbf{z}|\mathbf{x})\, p(\mathbf{p}|\boldsymbol{\theta})$. To simplify our notation and without losing 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 $\boldsymbol{\Sigma}_{\mathbf{p}}$. The innovation covariance is defined as

$$\mathbf{S} \triangleq \mathbf{H}_1 \boldsymbol{\Sigma}_{\mathbf{p}} \mathbf{H}_1^\top + \boldsymbol{\Sigma}. \tag{32}$$

For any given $\boldsymbol{\theta}$, $p(\mathbf{p}|\boldsymbol{\theta}, \mathbf{z}) = \mathcal{N}\big(\mathbf{p}; \boldsymbol{\mu}(\boldsymbol{\theta}), \boldsymbol{\Sigma}(\boldsymbol{\theta})\big)$, where

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \left(\mathbf{H}_1^\top \boldsymbol{\Sigma}^{-1} \mathbf{H}_1 + \boldsymbol{\Sigma}_{\mathbf{p}}^{-1}\right)^{-1} \tag{33}$$

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \boldsymbol{\Sigma}(\boldsymbol{\theta})\left(\mathbf{H}_1^\top \boldsymbol{\Sigma}^{-1}(\mathbf{z} - \mathbf{H}_2\boldsymbol{\theta}) + \boldsymbol{\Sigma}_{\mathbf{p}}^{-1}\boldsymbol{\mu}_{\mathbf{p}}\right) \tag{34}$$

$$= \boldsymbol{\mu}_{\mathbf{p}} + \boldsymbol{\Sigma}_{\mathbf{p}} \mathbf{H}_1^\top \mathbf{S}^{-1}\left(\mathbf{z} - \mathbf{H}_2\boldsymbol{\theta} - \mathbf{H}_1\boldsymbol{\mu}_{\mathbf{p}}\right) \tag{35}$$

in which we have used the matrix inversion lemma. Therefore, 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 (31)

$$\mathbf{x}^\star = \arg\max_{\mathbf{p}, \boldsymbol{\theta}} \; p(\boldsymbol{\theta}|\mathbf{z})\, p(\mathbf{p}|\boldsymbol{\theta}, \mathbf{z}). \tag{36}$$

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

$$\mathbf{p}^\star(\boldsymbol{\theta}) = \arg\max_{\mathbf{p}} \; p(\mathbf{p}|\boldsymbol{\theta}, \mathbf{z}). \tag{37}$$

As in any Gaussian density, the mean of $p(\mathbf{p}|\boldsymbol{\theta}, \mathbf{z})$ is equal to its mode, and therefore, $\boldsymbol{\mu}(\boldsymbol{\theta})$ is the solution of (37). Maximizing $p(\mathbf{p}, \boldsymbol{\theta}|\mathbf{z})$ subject to $\mathbf{p} = \boldsymbol{\mu}(\boldsymbol{\theta})$ reduces to an NLS problem that can be solved like before. After obtaining the MAP estimate for $\boldsymbol{\theta}$, we can recover $\mathbf{p}^\star$ by evaluating (37) at $\boldsymbol{\theta}^\star$. In practice, we should use the approach taken in Algorithm 3 in order to retain the sparse structure of SLAM.

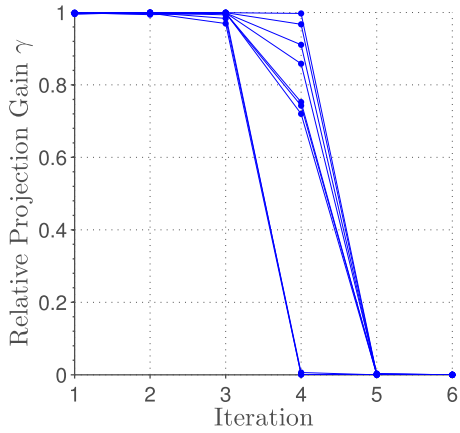


Fig. 3. Evolution of the projection gain $\gamma_i \triangleq (f_i^\circ - f_i^\star)/f_i^\circ$ in ten simulated datasets. The projection gain is maximum when γ is close to one. At the sixth iteration, $\gamma = 0$ upon convergence to the optimal estimate.

VI. Implementation

In this section, we outline an efficient implementation of sparse VP that requires minor modification of existing SLAM solvers such as `g2o` [24]. In each iteration of sparse VP, we need to construct and solve the sparse linear system in (25).

First note that there is a close relation between the coefficient matrix in the left-hand side of (25) and the top-left block ($\mathbb{I}_p$) in the full approximated Hessian (23). It can be easily verified that these two terms are the same expression evaluated at different points. The former is evaluated at $\boldsymbol{\theta}_{(i+1)}$, whereas the latter is computed at $\boldsymbol{\theta}_{(i)}$. In other words

$$\tilde{\mathbf{H}}_1^\top \tilde{\mathbf{H}}_1 = \sum_{k=1}^{|\mathcal{E}|} \frac{\partial \mathbf{r}_k}{\partial \mathbf{p}}^\top \boldsymbol{\Sigma}_k^{-1} \frac{\partial \mathbf{r}_k}{\partial \mathbf{p}}. \tag{38}$$

The Jacobians appearing in (38) are already available in any conventional Newton-based SLAM solver. Our implementation benefits from this insight by relying on `g2o` for computing the coefficient matrix of (25). Similar to `g2o`, we exploit the block structure of the coefficient matrix, which enables us to process the information terms for edges in parallel. The right-hand side of (25) can also be computed using the existing routines for computing the right-hand side of the full normal equations (23). Any of the existing sparse linear solvers can be used to efficiently solve (25).

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

$$\tilde{\mathbf{H}}_1^\top \tilde{\mathbf{H}}_1 = \mathbf{L}_w \otimes \mathbf{I}_2$$

where $\mathbf{L}_w \triangleq \mathbf{A}_w \mathbf{A}_w{}^\top$ is the reduced weighted Laplacian matrix of graph $\mathcal{G}$ with the following weight function:

$$\begin{aligned} w : \mathcal{E} &\to \mathbb{R}_{>0} \\ e_i &\mapsto \sigma_{p_i}^{-2}. \end{aligned} \tag{39}$$

A similar structure emerges from the 3-D SLAM problems with spherical noise covariance matrices. Therefore, in such cases, the coefficient matrix of (25) remains constant throughout the iterations and needs to be factorized only once at the first iteration. In such cases, the following VP iterations only need to solve sparse triangular linear systems with different right-hand sides using backward and forward substitutions. As shown in Section VII, exploiting this structure significantly reduces the extra cost of projection in VP.

VII. Results and Experiments

We performed experiments on both real and synthetic datasets to evaluate the performance of the proposed algorithm. We have used `g2o`'s implementation of GN [24] (without line search). CHOLMOD [8] is used as the linear solver with the approximate minimum degree ordering. Our Algorithm 3 (VP)[4] is implemented in `C++` as a `g2o` solver. Our fully integrated code is available at `http://github.com/kasra/vp-g2o`. An Intel Core i5-2400 CPU operating at 3.1 GHz is used for all of the experiments in this paper. We verified the equivalence between the iterations of Algorithms 1–3 numerically.

We used `g2o`'s 2-D simulator to generate Manhattan-like [27] pose-graph datasets. This simulator generates a random walk in plane with either 1 m forward motion or 90° rotation per step. The valid sensor range for scan matching is between 1 and 5 m 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. For our Monte Carlo analysis, we created Atlas, a collection of 500 simulated datasets available at `http://github.com/kasra/atlas`. Atlas consists of five test suites, each of which is composed of 100 randomly generated pose-graph datasets with 10^4 poses per dataset. Each test suite corresponds to a fixed noise level $\alpha \in \{1, \ldots, 5\}$. Noise covariance for each suite is $\boldsymbol{\Sigma}_\alpha = (0.01\alpha)^2 \mathbf{I}$.

A. Convergence

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 following.

1) *Global Min*: The global minimizer is found within 50 iterations.
2) *Local Min*: A local minimizer (other than the global minimizer) is found within 50 iterations.
3) *Not Converged*: The solver has not converged to a solution before 50 iterations.

We treat the solution of GN initialized with the ground truth as the global minimizer and use this to verify if an obtained solution is the global minimizer.[5] If the absolute difference between the cost of the last two iterations is larger than a threshold, we categorize that case as an instance of **Not Converged**. Similarly, if the absolute difference between the final cost and the minimum of the NLS cost function is larger than a threshold, we categorize that case as an instance of **Local Min**.

Table II summarizes the outcome under different noise levels. Although the two algorithms exhibit comparable performances in terms of converging to the optimal solution, our algorithm significantly outperforms GN in avoiding divergence or extremely slow convergence. Therefore, as reported in many other domains [16], VP iterations lead to a faster and more reliable convergence than solving the full NLS problem. 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. This can be avoided by using a "sufficiently good" initial guess [19]. 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 synthetic datasets used in Table 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

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

[5]This strategy may fail especially when the realized noise is large.

TABLE II
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

TABLE III
OUTCOME (%) OF LM AND VP-LM AFTER 50 AND 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	70
	VP-LM	32	33	63	66	5	1

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 II, we performed another experiment by generating random initial guesses through sampling uniformly from the surface of the hypersphere centered at the global minimum with varying radii. As in Table II, we did not observe a statistically significant difference in the tendency of VP and GN for converging to local minima.

Table III shows the results obtained by Levenberg–Marquardt (LM) and a trust-region version of Algorithm 3 using LM (VP-LM). Note that VP-LM is equivalent to performing LM on the reduced cost function under Kaufman's approximation. For both solvers, we use g2o'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 III, LM exhibits extremely slow convergence, while the performance of VP-LM within 50 iterations is comparable to that of GN and VP. Nevertheless, the success rates of GN and VP are slightly higher than VP-LM. It is also remarkable how VP-LM, due to its trust-region mechanism, avoids divergence after a sufficient number of iterations.

Fig. 4 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

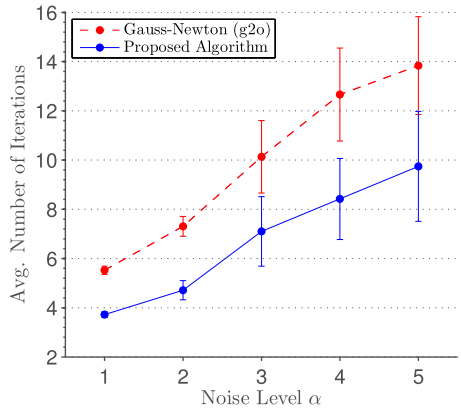


Fig. 4. Average number of iterations performed to converge to the global minimum (ML estimate) under different noise levels (α) in the Atlas datasets. For each noise level, 100 random datasets with 10000 poses has been generated. The error bars show the 95% confidence interval. The increasing length of error bars is partly due to the decreasing number of successful samples (see Table II). 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).

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 have applied VP to separable NLS problems in other contexts [16].

B. Runtime

Reducing the number of iterations does not necessarily reduce the total computation time since each VP iteration is more costly than that of GN. In fact, for SLAM, original VP [15], Kaufman's approach [22] (see Algorithm 1) and Barham and Drane's method [2] (see Algorithm 2) are all significantly slower than the state-of-the-art SLAM solvers, since they are all incapable of exploiting sparsity. By contrast, 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, Fig. 4 suggests 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.

To compare the overall runtimes, we generated ten large-scale Manhattan-like random walks, each with 10^5 poses. Relative measurements are corrupted by additive Gaussian noise with a nonspherical covariance matrix. The marginal variances are equal to those of $\alpha = 1$ noise level.[6] Each problem was solved iteratively using GN, VP, and VP with the relative projection gain threshold $\gamma_T = 0.2$ (see Section V-B). The average runtimes are listed in Table IV. In all but one of the datasets,

[6]Given the large number of poses, higher noise levels lead to convergence failure (for both GN and VP) and hence are not considered here.

TABLE IV
TOTAL RUNTIME FOR TEN RANDOM WALKS WITH 10^5 POSES AND NONSPHERICAL NOISE WITH MARGINAL VARIANCES OF $\alpha = 1$ NOISE LEVEL

#	$\lvert\mathcal{E}\rvert$	GN (sec)	VP$_{\gamma_T=0.2}$ (sec)	Saving versus GN (%)	VP (sec)	Saving versus GN (%)
1	344 364	15.14	14.99	1.01	15.53	−2.58
2	343 436	15.99	12.60	21.17	13.00	18.70
3	346 019	31.52	25.93	17.73	26.85	14.82
4	345 249	24.63	20.41	17.15	20.33	17.44
5	343 864	–	91.53	–	95.41	–
6	345 842	29.45	24.35	17.32	24.56	16.59
7	345 864	22.41	22.74	−1.51	23.51	−4.94
8	343 642	18.52	16.68	9.95	17.19	7.21
9	344 730	24.84	18.45	25.72	19.15	22.92
10	343 685	18.31	16.93	7.51	17.76	3.00

all algorithms converged to the optimal solution. In all of the datasets, except two cases, VP with γ_T exhibits the fastest performance, while VP outperforms GN in eight datasets. VP with γ_T outperforms GN because of its more effective iterations on the nonlinear core of SLAM (i.e., projection steps). On the other hand, as expected, it outperforms VP by avoiding unnecessary costly projection steps when the gain is insignificant.

We conducted another experiment to compare the overall runtime of VP and GN under varying edge density. For this purpose, first, we generated a Manhattan-like random walk with 10^5 poses. Two pose-graph datasets were generated based on this random walk by simulating noisy measurements for the following noise models: 1) $\alpha = 1$; and 2) nonspherical noise covariance matrix with marginal variances similar to $\alpha = 1$. To study the effect of edge density on the overall runtime, we generated 100 scenarios 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. Fig. 5 shows the overall runtime as a function of loop closure density for each noise model. According to Fig. 5, in the vast majority of cases, VP has been faster than GN. This shows that reducing the number of iterations has paid off the additional cost paid for each iteration. There are few cases where GN is slightly faster than VP. This situation happens mainly in extremely trivial (sparse) scenarios, in which the initial guess based on odometry is already close to the maximum likelihood estimate (MLE), and thus, GN can find the solution immediately. Such scenarios are often too sparse to be considered as realistic cases. Once again, our results indicate that taking the (expected) projection gain into account is generally beneficial as it helps to avoid unnecessary projection steps.

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 (25). As explained in Section VI, this term is constant (i.e., independent of the current estimate) when the noise covariance matrix is spherical. Thus, in such cases, the Cholesky factorization has to be computed only once (i.e., in the first iteration). In the rest of iterations, the conditionally optimal estimate for $\mathbf{p}$ can be recovered by solving sparse triangular linear systems via forward and backward substitutions. As illustrated in Fig. 5, this can reduce the cost per iteration and overall runtime of our algorithm.

We also used a number of publicly available datasets to evaluate the performance of the proposed algorithm. Table V provides the number of iterations performed to find the optimal solution, as well as the average of total computation time over ten runs. The datasets listed in Table V span the most common forms of SLAM (2-D/3-D 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 [4]. In such cases, exploiting the separable structure of SLAM can be less effective. This conclusion is consistent with what we witnessed earlier in Fig. 5.

For the datasets listed in Table V, both algorithms are able to converge to MLE starting from the odometry initial guess. This is true for most of the existing real datasets, although as seen in Table II for synthetic datasets, a bad initial guess can cause VP and GN to converge to (different) local minimizers of the cost function. For example, this is the case for the Victoria Park dataset. Starting from the same initial guess and after 30 iterations, VP and GN achieve $f_{\text{VP}} = 13\,659$ and $f_{\text{GN}} = 30\,611$, respectively. Similar to what we observed before in our Monte Carlo simulations, even without using line search, the objective value resulting from VP directions follows a stable decreasing trend, while in the first few iterations, GN steps cause the cost function to increase.

VIII. DISCUSSION AND CONCLUSION

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 a faster and more reliable convergence than the state-of-the-art solvers. A key contribution of this work comes from establishing the link to the rich literature on separable NLS problems. In particular, recognizing the equivalence between Algorithms 1 and 2 enabled us to retain sparsity while exploiting the separable structure through Algorithm 3. This link also provides a solid theoretical justification for the proposed algorithm. Moreover, we demonstrated how one can avoid performing insignificant projections by taking the projection gain into account.

The proposed algorithm can be applied to the most common forms of SLAM (2-D/3-D feature based and pose graphs) without any restrictive assumption on the structure of the noise covariance matrix. Our algorithm is not limited to a particular type

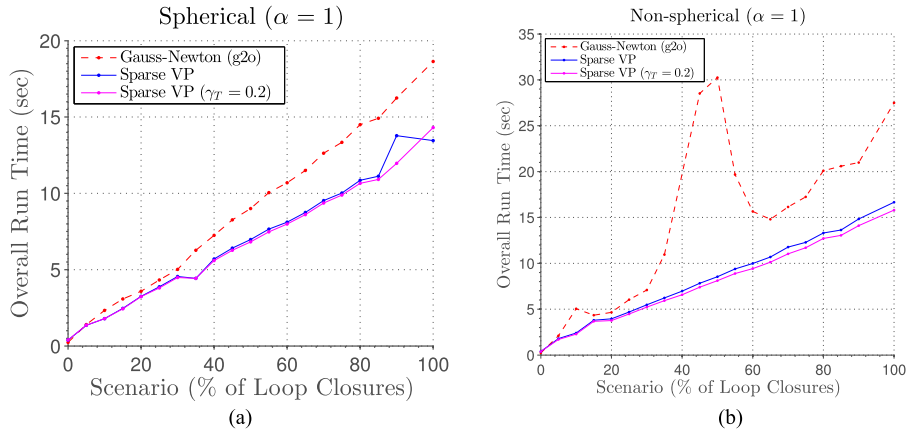


Fig. 5. Overall runtime for converging to MLE as a function of edge density under different noise models. In (b), GN failed to converge to MLE in one scenario (40). Note that the "100% loop closure density" refers to the 100th scenario, in which all of the loop closures of a realistic sparse SLAM problem with $|\mathcal{V}| = 10^5$ and $|\mathcal{E}| \approx 342\,000$ are included—not the complete graph. (a) $\alpha = 1$ (spherical). (b) $\alpha = 1$ (nonspherical).

TABLE V
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	0.41068
			VP	4	0.31617
Manhattan	3500	5598	GN	6	0.07907
			VP	4	0.07200
Intel Research Lab	943	1837	GN	3	0.01463
			VP	2	0.01607
UTM Downtown	14 549	16 365	GN	10	0.24754
			VP	4	0.17719
Sphere2500	2500	9799	GN	5	0.96145
			VP	4	0.82080
New College	52 480	52 577	GN	8	2.19560
			VP	6	2.00967

In all cases, both algorithms converge to the ML estimate.

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 backends (e.g., LM, Powell's Dog-leg [29], etc.) without any major modification. By stripping down SLAM to its nonlinear core and recovering the conditionally optimal estimate for the 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 any other Newton-based solver) iterations are relatively costly and/or when it takes more than few iterations to solve the full NLS problem.

In this paper, we mainly considered batch solvers. Incrementally solving SLAM [20], [21], [29] is more suitable for online applications. The separable structure of SLAM is preserved in the incremental formulation of SLAM and can be exploited by the same principles and techniques introduced in this paper. As seen earlier, our results indicate that exploiting separability, through increasing the convergence rate, is mostly beneficial if the initial guess is not already "too close" to the solution. One advantage of incrementally solving SLAM is that one can use the ML estimate using the data collected up to time $t-1$ for finding the ML estimate at time t. This initial guess is usually good, and therefore, exploiting separability may not be useful in such cases. An important exception is when the most recent measurements (measurements collected at time t) lead to a significant update of robot trajectory. For instance, this situation could arise if, for a period of time, the robot has only access to noisy odometry data (dead reckoning) and small loop-closures and then suddenly closes a larger loop. Incremental solvers that are capable of exploiting separability (using the solver proposed in this paper) can, therefore, benefit from its faster convergence in such scenarios. Recall that if the initial guess is already close to the solution, the projection gain γ will be small. Therefore, immediately after the first projection step and by computing the projection gain, the proposed switching scheme in Section V-B can automatically detect whether or not exploiting separability is beneficial. In the worst case, after performing a single projection step, our algorithm may switch to GN steps. However, if the projection gain happens to be significant (e.g., in the case of closing larger loops), an incremental solver that is capable of exploiting separability will exhibit faster convergence. We plan to investigate the case of iterative solvers in our future work.

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 (8), $\mathbf{H}_1$ is full column rank if and only if $\mathbf{R}_\theta^\top \mathbf{A}_2^\top$ (and $\mathbf{R}_\theta^\top \mathbf{A}_3^\top$ in 3-D SLAM) is full column rank. It is obvious that $\mathbf{R}_\theta$ is nonsingular. The reduced incidence matrix $\mathbf{A}$ is full (row) rank if and only if the corresponding graph is (weakly) connected [7], which is the case in well-defined SLAM problem (see the discussion in Section III-A). Consequently, $\mathbf{H}_1$ is full (column) rank, regardless of the value of $\boldsymbol{\theta}$. ■

APPENDIX B
PROOF OF THEOREM 1

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

$$\begin{aligned}[\mathbf{J}_{\text{vp}}]_{\cdot,j} &= \frac{\partial \mathbf{r}_{\text{vp}}}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}[\mathbf{P}_{\boldsymbol{\theta}}^{\perp}\,(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})] \\ &= \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}^{\perp}}{\partial \theta_j}\,(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) + \mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta})}{\partial \theta_j} \\ &= \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}^{\perp}}{\partial \theta_j}\,(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) - \mathbf{P}_{\boldsymbol{\theta}}^{\perp}[\tilde{\mathbf{H}}_2]_{\cdot,j} \\ &= -\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\,(\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) - \mathbf{P}_{\boldsymbol{\theta}}^{\perp}[\tilde{\mathbf{H}}_2]_{\cdot,j}. \end{aligned} \tag{40}$$

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 [31]. First note that $\mathbf{P}_{\boldsymbol{\theta}}$ is (1) idempotent, i.e., $(\mathbf{P}_{\boldsymbol{\theta}})^2 = \mathbf{P}_{\boldsymbol{\theta}}$, and 2) symmetric. Also note that $\mathbf{P}_{\boldsymbol{\theta}}\tilde{\mathbf{H}}_1 = \tilde{\mathbf{H}}_1$. Therefore, we have

$$\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}\tilde{\mathbf{H}}_1}{\partial \theta_j} = \frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j} = \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\tilde{\mathbf{H}}_1 + \mathbf{P}_{\boldsymbol{\theta}}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}. \tag{41}$$

Therefore

$$\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\tilde{\mathbf{H}}_1 = (\mathbf{I} - \mathbf{P}_{\boldsymbol{\theta}})\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j} = \mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}. \tag{42}$$

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

$$\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\mathbf{P}_{\boldsymbol{\theta}} = \mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\tilde{\mathbf{H}}_1^{\dagger}. \tag{43}$$

Also note that

$$\left(\mathbf{P}_{\boldsymbol{\theta}}\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\right)^{\top} = \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\mathbf{P}_{\boldsymbol{\theta}}. \tag{44}$$

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

$$\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j} = \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}^2}{\partial \theta_j} = \frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j}\mathbf{P}_{\boldsymbol{\theta}} + \mathbf{P}_{\boldsymbol{\theta}}\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j} \tag{45}$$

which can be simplified using (43) and (44)

$$\frac{\partial \mathbf{P}_{\boldsymbol{\theta}}}{\partial \theta_j} = \mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\tilde{\mathbf{H}}_1^{\dagger} + \left(\mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\tilde{\mathbf{H}}_1^{\dagger}\right)^{\top}. \tag{46}$$

Plugging (46) into (40) completes the proof, i.e.,

$$\begin{aligned}[\mathbf{J}_{\text{vp}}]_{\cdot,j} = &-\left[\mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\tilde{\mathbf{H}}_1^{\dagger} + (\mathbf{P}_{\boldsymbol{\theta}}^{\perp}\frac{\partial \tilde{\mathbf{H}}_1}{\partial \theta_j}\tilde{\mathbf{H}}_1^{\dagger})^{\top}\right](\tilde{\mathbf{z}} - \tilde{\mathbf{H}}_2\boldsymbol{\theta}) \\ &- \mathbf{P}_{\boldsymbol{\theta}}^{\perp}[\tilde{\mathbf{H}}_2]_{\cdot,j}. \end{aligned} \tag{47}$$

■

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

$$\begin{aligned} p(\mathbf{p}|\boldsymbol{\theta}, \mathbf{z}) &= \frac{p(\mathbf{p}, \boldsymbol{\theta}, \mathbf{z})}{p(\boldsymbol{\theta}, \mathbf{z})} \\ &= \frac{p(\mathbf{z}|\mathbf{x})\,p(\mathbf{x})}{p(\boldsymbol{\theta}, \mathbf{z})} \\ &= \frac{p(\mathbf{z}|\mathbf{x})\,p(\mathbf{p}|\boldsymbol{\theta})\,p(\boldsymbol{\theta})}{p(\mathbf{z}|\boldsymbol{\theta})\,p(\boldsymbol{\theta})} \\ &= \frac{p(\mathbf{z}|\mathbf{x})\,p(\mathbf{p}|\boldsymbol{\theta})}{p(\mathbf{z}|\boldsymbol{\theta})}. \end{aligned} \tag{48}$$

ACKNOWLEDGMENT

The authors would like to thank T. Zhang and the reviewers for their valuable comments. They are also grateful to Stachniss *et al.* and the EUROPA project for providing the UTM-Downtown dataset, D. Haehnel for the Intel Research Lab dataset, E. Olson for the Manhattan dataset, M. Kaess for the City10K and Sphere2500 datasets, Smith *et al.* for the New College dataset [32], J. Guivant and E. Nebot for the Victoria Park dataset, and Kuemmerle *et al.* for `g2o` [24].

REFERENCES

[1] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 62–69.
[2] R. H. Barham and W. Drane, "An algorithm for least squares estimation of nonlinear parameters when some of the parameters are linear," *Technometrics*, vol. 14, no. 3, pp. 757–766, 1972.
[3] A. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, PA, USA: SIAM, 1996.
[4] L. Carlone, "Convergence analysis of pose graph optimization via Gauss-Newton methods," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 965–972.
[5] L. Carlone, R. Aragues, J. Castellanos, and B. Bona, "A linear approximation for graph-based simultaneous localization and mapping," in *Proc. Robot.: Sci. Syst. Conf.*, Los Angeles, CA, USA, Jun. 2011, pp. 41–48.
[6] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona, "A fast and accurate approximation for planar pose graph optimization," *Int. J. Robot. Res.*, vol. 33, no. 7, pp. 965–987, Jun. 2014.
[7] W.-K. Chen, *Applied Graph Theory*, vol. 13. Amsterdam, The Netherlands: North Holland, 1971.
[8] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate," *ACM Trans. Math. Softw.*, vol. 35, no. 3, pp. 22, 2008.
[9] T. A. Davis, "Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization," *ACM Trans. Math. Softw.*, vol. 38, no. 1, 2011, Art. no. 8.
[10] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1203, 2006.
[11] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 229–241, Jun. 2001.
[12] A. Doucet, "On sequential simulation-based methods for Bayesian filtering," Dept. Eng., Univ. Cambridge, Cambridge, U.K., *Tech. Rep. CUED/F-INFENG/TR.310*, 1998.
[13] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *The Oxford Handbook of Nonlinear Filtering*. London, U.K.: Oxford Univ. Press, 2011, pp. 656–704.

[14] H. Durrant-Whyte andT. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, 2006.
[15] G. Golub and V. Pereyra, "The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate," *SIAM J. Numer. Anal.*, vol. 10, no. 2, pp.413–432, 1973.
[16] G. Golub and V. Pereyra, "Separable nonlinear least squares: The variable projection method and its applications," *Inverse Probl.*, vol. 19, no. 2, 2003, Art. no. R1.
[17] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, 2010.
[18] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Inf. Fusion*, vol. 14, no. 1, pp. 57–77, 2013.
[19] G. Hu, K. Khosoussi, and S. Huang, "Towards a reliable SLAM back-end," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2013, pp. 37–43.
[20] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.
[21] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Int. J. Robot. Res.*, vol. 31, no. 2, pp. 216–235, 2012.
[22] L. Kaufman, "A variable projection method for solving separable nonlinear least squares problems," *BIT Numer. Math.*, vol. 15, no. 1, pp. 49–57, 1975.
[23] K. Khosoussi, S. Huang, and G. Dissanayake, "Exploiting the separable structure of SLAM," in *Proc. Robot.: Sci. Syst. Conf.*, Rome, Italy, Jul. 2015.
[24] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3607–3613.
[25] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Trans. Robot.*, vol. 28, no. 1, pp. 61–76, Feb. 2012.
[26] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proc. 18th Int. Joint Conf. Artif. Intell.*, Acapulco, Mexico, 2003, pp. 1151–1156.
[27] E. Olson, "Robust and efficient robotic mapping," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, Jun. 2008.
[28] T. A. Parks, "Reducible nonlinear programming problems (separable least squares)," Ph.D. dissertation, Rice Univ., Houston, TX, USA, 1985.
[29] D. M. Rosen, M. Kaess, and J. J. Leonard, "RISE: An incremental trust-region method for robust online sparse least-squares estimation," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1091–1108, Oct. 2014.
[30] A. Ruhe and P. Å. Wedin, "Algorithms for separable nonlinear least squares problems," *SIAM Rev.*, vol. 22, no. 3, pp. 318–337, 1980.
[31] G. A. F. Seber and C. J. Wild, *Nonlinear Regression*. New York, NY, USA: Wiley-Interscience, 1989.
[32] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *Int. J. Robot. Res.*, vol. 28, no. 5, pp. 595–599, May 2009. [Online]. Available: http://www.robots.ox.ac.uk/NewCollegeData/
[33] N. Sunderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Oct. 2012, pp. 1879–1884.
[34] H. Wang, G. Hu, S. Huang, and G. Dissanayake, "On the structure of nonlinearities in pose graph SLAM," in *Proc. Robot.: Sci. Syst. Conf.*, Sydney, Australia, Jul. 2012.
[35] H. Wang, S. Huang, U. Frese, and G. Dissanayake, "The nonlinearity structure of point feature SLAM problems with spherical covariance matrices," *Automatica*, vol. 49, no. 10, pp. 3112–3119, 2013.
[36] H. Wang, S. Huang, K. Khosoussi, U. Frese, G. Dissanayake, and B. Liu, "Dimensionality reduction for point feature SLAM problems with spherical covariance matrices," *Automatica*, vol. 51, pp. 149–157, 2015.
[37] N. Zikos and V. Petridis, "L-SLAM: Reduced dimensionality FastSLAM with unknown data association," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 4074–4079.
[38] N. Zikos and V. Petridis, "6-DoF low dimensionality SLAM (L-SLAM)," *J. Intell. Robot. Syst.*, vol. 79, pp. 55–72, 2015.

Kasra Khosoussi received the B.Sc. degree in computer engineering from the Department of Electrical and Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran, in 2011. He is currently working toward the Ph.D. degree in robotics with the Centre for Autonomous Systems, University of Technology Sydney, Sydney, Australia.

From 2008 to 2012, he was a member of the Advanced Robotics and Automated Systems. In 2015 and 2016, he was a Visiting Scholar with the Department of Computer Science, University of Southern California, Los Angeles, CA, USA. His research interests include the science of autonomy and robotics, with an emphasis on perception and decision making.

Shoudong Huang (M'04) received the bachelor's and master's degrees in mathematics and the Ph.D. degree in automatic control from Northeastern University, Shenyang, China, in 1987, 1990, and 1998, respectively.

He is currently an Associate Professor with the Centre for Autonomous Systems, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, Australia. His research interests include nonlinear control systems and mobile robots simultaneous localization and mapping, exploration, and navigation.

Gamini Dissanayake (M'05) graduated in mechanical/production engineering from University of Peradeniya, Peradeniya, Sri Lanka, and received the B.Sc. (Eng), M.Sc. degree in machine tool technology and the Ph.D. degree in mechanical engineering (robotics) from University of Birmingham, Birmingham, U.K., in 1981 and 1985, respectively.

He is the James N. Kirby Professor of mechanical and mechatronic engineering with University of Technology Sydney (UTS), Sydney, Australia. His research interests include localization and map building for mobile robots, navigation systems, dynamics, and control of mechanical systems, cargo handling, optimization, and path planning. He leads the Centre for Autonomous Systems, UTS.