Revisiting the Variable Projection Method for Separable Nonlinear Least Squares Problems

Je Hyeong Hong
University of Cambridge
jhh37@cam.ac.uk

Christopher Zach
Toshiba Research Europe
christopher.m.zach@gmail.com

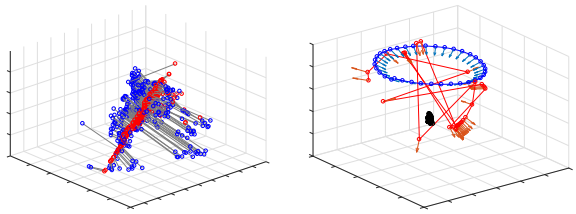
Andrew Fitzgibbon
Microsoft
awf@microsoft.com

Abstract

Variable Projection (VarPro) is a framework to solve optimization problems efficiently by optimally eliminating a subset of the unknowns. It is in particular adapted for Separable Nonlinear Least Squares (SNLS) problems, a class of optimization problems including low-rank matrix factorization with missing data and affine bundle adjustment as instances. VarPro-based methods have received much attention over the last decade due to the experimentally observed large convergence basin for certain problem classes, where they have a clear advantage over standard methods based on Joint optimization over all unknowns. Yet no clear answers have been found in the literature as to why VarPro outperforms others and why Joint optimization, which has been successful in solving many computer vision tasks, fails on this type of problems. Also, the fact that VarPro has been mainly tested on small to medium-sized datasets has raised questions about its scalability. This paper intends to address these unsolved puzzles.

1. Introduction

Optimization methods play an ubiquitous role in computer vision and related fields, and improvements in their performance can enable new capabilities and applications. In recent years, it has been understood that significant improvements in convergence can come from the use of a non-minimal parametrization. Examples include convex relaxations for binary segmentation (e.g. [8]), and lifting methods for MAP inference (e.g. [16, 29]), 3D model fitting [7], and robust costs [31]. In these examples it has been proved theoretically or shown empirically that a non-minimal representation of the unknowns leads to solutions with significantly lower objective values, often because the "non-lifted" optimization stalls far from a good optimum. In contrast, there is one class of problems where the opposite is frequently observed in the literature: using a non-minimal parametrization for low-rank matrix factorization problems



(a) Estimated 3D points (b) Estimated cameras

Figure 1: For an affine bundle adjustment problem, standard Joint optimization (Schur-complement bundle adjustment with inner point iterations) does not reach a useful reconstruction from an arbitrary initialization (Red). In contrast, VarPro (Blue) often finds the best known optimum from random starts. This paper shows how the ostensibly small differences between the two methods give rise to very different convergence properties.

with missing data has notably inferior performance than methods based on Variable Projection (VarPro). Variable Projection optimally eliminates some of the unknowns in an optimization problem, and is therefore especially applicable to separable non-linear least-squares problems described below. Low-rank matrix factorization is a problem class appearing in signal processing (e.g. blind source separation), in machine learning (e.g. factor analysis), but also in 3D computer vision to obtain e.g. affine and non-rigid reconstructions. The success of VarPro methods is often reported in the literature (especially for geometric reconstruction problems), but to our knowledge there is lack of understanding why Variable Projection is so beneficial in this case.

Our work sheds some light on the relation between Variable Projection methods and *Joint optimization* methods using explicit factors for low-rank matrix factorization. It will be revealed in this paper that Joint optimization suffers from an intrinsic numerical ill-conditioning for matrix factorization problems, and therefore is prone to "stalling".

Although we will focus on matrix factorization tasks, our analysis holds also for the more general class of Separable Nonlinear Least Squares problems [10]. A Separable Nonlinear Least Squares (SNLS) problem is defined as minimizing

$$\|\mathtt{G}(\mathbf{u})\mathbf{v} - \mathbf{z}(\mathbf{u})\|_2^2 \tag{1}$$

over $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{v} \in \mathbb{R}^q$ where these two vectors are non-overlapping subsets of system variables and where the functions $\mathtt{G} : \mathbb{R}^p \to \mathbb{R}^{s\times q}$ and $\mathbf{z} : \mathbb{R}^p \to \mathbb{R}^s$ are generally nonlinear in $\mathbf{u}$. This type of problem has a characteristic that its residual vector is linear in at least one set of system parameters. Due to this generality, SNLS problems arise in various parts of engineering and science [11], ranging from exponential fitting [23] to chemistry, mechanical engineering and medical imaging.

A specific class of SNLS problems on which our investigation is focussed is L^2-norm rank-imposed matrix factorization with/out the mean vector, which solves

$$\min_{\mathtt{U},\mathtt{V}} \left\|\mathtt{W} \odot (\mathtt{U}\mathtt{V}^\top - \mathtt{M})\right\|_F^2 \tag{2}$$

where $\mathtt{M} \in \mathbb{R}^{m\times n}$ is the observation matrix, $\mathtt{W} \in \mathbb{R}^{m\times n}$ is the weight matrix, which masks all the missing elements by performing the element-wise (Hadamard) product, $\mathtt{U} \in \mathbb{R}^{m\times r}$ and $\mathtt{V} \in \mathbb{R}^{n\times r}$ are the two low-rank factors and $\|\cdot\|_F$ is the Frobenius norm. If a mean vector is used, then the last column of $\mathtt{V}$ is set to all-1 vector. It is trivial to transform (2) into (1). This branch of problems is visible in several computer vision and machine learning applications; bundle adjustment using affine cameras [27], non-rigid structure-from-motion using basis shapes or point trajectory basis functions [5], photometric stereo assuming ambient light and Lambertian surfaces [2] and recommender systems [3] just to name a few.

This paper presents the following contributions:

+ In §3, we provide an extensive review of the known methods for solving separable nonlinear least squares (SNLS) problems, namely Joint optimization with or without Embedded Point Iterations (EPI) and Variable Projection (VarPro). Unlike previous work we explicitly consider the effect of Levenberg-style damping, without which none of the alternatives perform well.

+ In §4, we unify the aforementioned methods and show that the Joint methods and VarPro effectively share the same algorithmic structure but differ in details.

+ In §5, we provide empirical analysis of how the Joint methods fail while VarPro succeeds despite the small algorithmic difference between the two branches of methods.

+ In §4.3, we propose a simple scalable strategy for VarPro which could be applied to large-scale and potentially dense SNLS problems such as matrix factorization with missing data and affine bundle adjustment.

Conversely, there are limitations of this work: the scope of this paper is confined to L^2-norm minimization. There are still remaining questions to be answered, such as why the Joint methods end up in the observed failure points.

1.1. Related work

Variable Projection (VarPro) was first proposed by Golub and Pereyra [10] for the general SNLS problem, and was applied to principal components analysis (i.e. matrix factorization) by Wiberg [30]. Over the last two decades, the computer vision and machine learning literature has seen a plethora of low-rank matrix factorization algorithms [9] which solves (2). Many of those algorithms were based on the space-efficient alternating least squares algorithm, with extremely poor convergece properties. Buchanan and Fitzgibbon [6] introduced damping with a damped Newton algorithm, but continued to ignore Wiberg. Okatani and Deguchi [21] reconsidered Wiberg, showing its strong convergence properties, and then Okatani et al.[22] combined damping and Wiberg to boost convergence rates to near 100% on some previously-difficult problems. At the same time Gotardo's CSF [12] algorithm showed similar improvements. These rank-r minimization algorithms were later unified [13] to be from the same root of Variable Projection.

Various papers pointed out some structural similarity between Joint optimization and VarPro. Ruhe and Wedin [25] and Okatani et al. [22] pointed out the similarity between the update equations of VarPro and Joint optimization but this was confined to the Gauss-Newton algorithm where no damping is present. Strelow [26] pointed out that VarPro performs additional minimization over the eliminated parameters. The Ceres solver [1], which is a widely-used nonlinear optimization library, also assumes the same. We show that these are not exactly performing VarPro, and removal of damping in some places takes a key role in implementing "pure" VarPro and widening the convergence basin.

With regards to scalable implementation of VarPro, Boumal et al.'s RTRMC [4] is in principle indirectly solving the VarPro reduced problem, which is what we propose. However, their algorithm is based on the regularized problem so their algorithm performs well for machine learning recommender systems and other random matrices but suffers from numerical instability when performed on SfM problems [13], where the regularizer is not a good idea because it essentially puts priors on Us and Vs. We provide a numerically stable scalable VarPro algorithm which is tested and works well on matrix factorization problems of various sizes and densities.

1.2. Notations

Throughout this paper, we make use of the following definitions and rule for any real thin matrix $\mathtt{A}$:

$$\mathtt{A}^{-\lambda} := (\mathtt{A}^\top \mathtt{A} + \lambda \mathtt{I})^{-1} \mathtt{A}^\top \tag{3}$$

$$\mathtt{A}^\dagger := (\mathtt{A}^\top \mathtt{A})^{-1} \mathtt{A}^\top = \mathtt{A}^{-0} \tag{4}$$

2. The Levenberg-Marquardt (LM) algorithm

We start by illustrating the Levenberg-Marquardt (LM) algorithm [19, 20], which is widely used for solving general nonlinear least squares problems, and it also forms the basis of the Joint optimization and Variable Projection (VarPro) methods for solving separable nonlinear least squares (SNLS) problems.

LM is an iterative algorithm for solving general nonlinear least squares problems (of which SNLS is a subset). It aims to solve

$$\min_{\mathbf{x}} \|\boldsymbol{\varepsilon}(\mathbf{x})\|_2^2 , \tag{5}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a vector of variables and $\boldsymbol{\varepsilon} : \mathbb{R}^n \to \mathbb{R}^s$ is the residual vector. A solution obtained using this algorithm is at best guaranteed to be a local minimum.

Given a current solution $\mathbf{x}_k$ the quantity of interest is the update $\boldsymbol{\Delta}\mathbf{x}_k$ to improve upon $\mathbf{x}_k$, forming $\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\Delta}\mathbf{x}_k$. Linearizing the residual yields

$$\boldsymbol{\varepsilon}(\mathbf{x}_{k+1}) = \boldsymbol{\varepsilon}(\mathbf{x}_k + \boldsymbol{\Delta}\mathbf{x}_k) \approx \boldsymbol{\varepsilon}_k + \mathtt{J}_k \boldsymbol{\Delta}\mathbf{x}_k, \tag{6}$$

where $\boldsymbol{\varepsilon}_k := \boldsymbol{\varepsilon}(\mathbf{x}_k)$ and $\mathtt{J}_k := \mathtt{J}(\mathbf{x}_k) := \partial \boldsymbol{\varepsilon}(\mathbf{x}_k)/\partial \mathbf{x}$, which is the Jacobian at $\mathbf{x}_k$. The Gauss-Newton (GN) step is obtained by solving the unregularized subproblem

$$\arg\min_{\boldsymbol{\Delta}\mathbf{x}} \|\boldsymbol{\varepsilon}_k + \mathtt{J}_k \boldsymbol{\Delta}\mathbf{x}\|_2^2 , \tag{7}$$

Solving (7) assumes that the cost is locally quadratic in $\boldsymbol{\Delta}\mathbf{x}_k$, and therefore $\mathbf{x}_k + \boldsymbol{\Delta}\mathbf{x}_k$ may not necessarily decrease the true objective $\|\boldsymbol{\varepsilon}(\mathbf{x})\|_2^2$. LM regularizes the update by adding a penalty term with a damping parameter $\lambda_k \in \mathbb{R}$,

$$\boldsymbol{\Delta}\mathbf{x}_k = \arg\min_{\boldsymbol{\Delta}\mathbf{x}} \|\boldsymbol{\varepsilon}_k + \mathtt{J}_k \boldsymbol{\Delta}\mathbf{x}\|_2^2 + \lambda_k \|\boldsymbol{\Delta}\mathbf{x}\|_2^2 . \tag{8}$$

The key intuition behind this augmentation is that the added term makes the quadratic assumption to be valid near $\mathbf{x}_k$ only. λ_k controls the size of the region which can be trusted as quadratic. To elaborate, if the step $\boldsymbol{\Delta}\mathbf{x}_k$ improves the actual cost, the update is accepted and λ_{k+1} is decreased, making (8) closer to the GN update in (7). Otherwise, the update is rejected and λ_k is increased, forcing the algorithm to behave more like gradient descent. Pseudocode for a straightforward implementation is given in the supplementary material.

The solution of (8) is explicitly given by

$$\begin{aligned}
\boldsymbol{\Delta}\mathbf{x}_k &= \arg\min_{\boldsymbol{\Delta}\mathbf{x}} \left\| \begin{bmatrix} \boldsymbol{\varepsilon}_k \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathtt{J}_k \\ \sqrt{\lambda_k}\mathtt{I} \end{bmatrix} \boldsymbol{\Delta}\mathbf{x} \right\|_2^2 \\
&= - \begin{bmatrix} \mathtt{J}_k \\ \sqrt{\lambda_k}\mathtt{I} \end{bmatrix}^\dagger \begin{bmatrix} \boldsymbol{\varepsilon}_k \\ \mathbf{0} \end{bmatrix} \\
&= -(\mathtt{J}_k^\top \mathtt{J}_k + \lambda_k \mathtt{I})^{-1} \mathtt{J}_k^\top \boldsymbol{\varepsilon}_k = {\mathtt{J}_k}^{-\lambda_k} \boldsymbol{\varepsilon}_k .
\end{aligned} \tag{9}$$

Computing $\boldsymbol{\Delta}\mathbf{x}_k$ can either be achieved by solving (9) directly using a matrix decomposition algorithm such as QR or Cholesky, or via an iterative method such as preconditioned conjugate gradients (PCG).

3. Review of methods for solving separable nonlinear least squares (SNLS) problems

In this section, we review each of the Joint optimization and Variable Projection (VarPro) methods in detail. These are re-illustrated with consistent notation to allow easier comparison between the methods and provide a comprehensive build-up to our contributions in the forthcoming sections.

We additionally define the following terms specific to the type of SNLS problem:

$$\boldsymbol{\varepsilon}(\mathbf{u}, \mathbf{v}) := \mathtt{G}(\mathbf{u})\mathbf{v} - \mathbf{z}(\mathbf{u}) \tag{10}$$

$$\mathtt{J}_{\mathbf{u}}(\mathbf{u}, \mathbf{v}) := \frac{\partial \boldsymbol{\varepsilon}(\mathbf{u}, \mathbf{v})}{\partial \mathbf{u}} = \frac{d[\mathtt{G}(\mathbf{u})]\mathbf{v}}{d\mathbf{u}} - \frac{d\mathbf{z}(\mathbf{u})}{d\mathbf{u}} \tag{11}$$

$$\mathtt{J}_{\mathbf{v}}(\mathbf{u}) := \frac{\partial \boldsymbol{\varepsilon}(\mathbf{u}, \mathbf{v})}{\partial \mathbf{v}} = \mathtt{G}(\mathbf{u}) \tag{12}$$

$$\mathtt{Q}_{\mathbf{v}}(\mathbf{u}) := \mathtt{I} - \mathtt{J}_{\mathbf{v}}(\mathbf{u}) \mathtt{J}_{\mathbf{v}}(\mathbf{u})^\dagger . \tag{13}$$

3.1. Joint optimization

Joint optimization uses the Gauss-Newton approximation with respect to both variables $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{v} \in \mathbb{R}^q$. The unknowns $\mathbf{u}$ and $\mathbf{v}$ are stacked to form $\mathbf{x} := [\mathbf{u}; \mathbf{v}] \in \mathbb{R}^{p+q}$, and LM (see §2) is applied to solve

$$\min_{\mathbf{x}} \|\boldsymbol{\varepsilon}(\mathbf{x})\|_2^2 := \min_{\mathbf{x}=[\mathbf{u};\mathbf{v}]} \|\boldsymbol{\varepsilon}(\mathbf{u}, \mathbf{v})\|_2^2 . \tag{14}$$

Hence, the update equations for Joint optimization follow (9) with $\boldsymbol{\Delta}\mathbf{x}_k := [\boldsymbol{\Delta}\mathbf{u}_k; \boldsymbol{\Delta}\mathbf{v}_k]$, $\boldsymbol{\varepsilon}_k := \boldsymbol{\varepsilon}(\mathbf{u}_k, \mathbf{v}_k)$ and $\mathtt{J}_k = [\mathtt{J}_{\mathbf{u}}(\mathbf{u}_k, \mathbf{v}_k) \ ; \ \mathtt{J}_{\mathbf{v}}(\mathbf{u}_k)] =: [\mathtt{J}_{\mathbf{u}_k} \ ; \ \mathtt{J}_{\mathbf{v}_k}]$. i.e.

$$\begin{bmatrix} \mathtt{J}_{\mathbf{u}_k}^\top \mathtt{J}_{\mathbf{u}_k} + \lambda_k \mathtt{I} & \mathtt{J}_{\mathbf{u}_k}^\top \mathtt{J}_{\mathbf{v}_k} \\ \mathtt{J}_{\mathbf{v}_k}^\top \mathtt{J}_{\mathbf{u}_k} & \mathtt{J}_{\mathbf{v}_k}^\top \mathtt{J}_{\mathbf{v}_k} + \lambda_k \mathtt{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta}\mathbf{u}_k \\ \boldsymbol{\Delta}\mathbf{v}_k \end{bmatrix} = - \begin{bmatrix} \mathtt{J}_{\mathbf{u}_k}^\top \boldsymbol{\varepsilon}_k \\ \mathtt{J}_{\mathbf{v}_k}^\top \boldsymbol{\varepsilon}_k \end{bmatrix} . \tag{15}$$

Schur complement reduced system The Schur complement is often suggested to solve (15) efficiently (e.g. [28]): instead of solving for a $(p+q) \times (p+q)$ system matrix,

$$\left(\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-\lambda_k})\mathtt{J}_{\mathbf{u}_k}+\lambda_k\mathtt{I}\right)\boldsymbol{\Delta}\mathbf{u}_k=-\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-\lambda_k})\boldsymbol{\varepsilon}_k \qquad \text{Joint} \qquad (17)$$

$$\left(\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-\lambda_k})\mathtt{J}_{\mathbf{u}_k}+\lambda_k\mathtt{I}\right)\boldsymbol{\Delta}\mathbf{u}_k=-\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-0})\boldsymbol{\varepsilon}_k=-\mathtt{J}_{\mathbf{u}_k}^\top\boldsymbol{\varepsilon}_k \qquad \text{Joint+EPI} \qquad (22)$$

$$\left(\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-0})\mathtt{J}_{\mathbf{u}_k}+\lambda_k\mathtt{I}\right)\boldsymbol{\Delta}\mathbf{u}_k=-\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-0})\boldsymbol{\varepsilon}_k=-\mathtt{J}_{\mathbf{u}_k}^\top\boldsymbol{\varepsilon}_k \qquad \text{VarPro} \qquad (33)$$

Figure 2: The key equations for $\boldsymbol{\Delta}\mathbf{u}_k$ according to the three SNLS optimization approaches. The apparently small differences in where damping is applied give rise to very different convergence properties.

the Schur complement reduces the problem to two subproblems of size $p \times p$ and $q \times q$, respectively,

$$\mathtt{S}_{\lambda_k} := \mathtt{I} - \mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-\lambda_k} \qquad (16)$$

$$\boldsymbol{\Delta}\mathbf{u}_k = -\left(\mathtt{J}_{\mathbf{u}_k}^\top\mathtt{S}_{\lambda_k}\mathtt{J}_{\mathbf{u}_k}+\lambda_k\mathtt{I}\right)^{-1}\mathtt{J}_{\mathbf{u}_k}^\top\mathtt{S}_{\lambda_k}\boldsymbol{\varepsilon}_k \qquad (17)$$

$$\boldsymbol{\Delta}\mathbf{v}_k = -\mathtt{J}_{\mathbf{v}_k}^{-\lambda_k}\left(\boldsymbol{\varepsilon}_k+\mathtt{J}_{\mathbf{u}_k}\boldsymbol{\Delta}\mathbf{u}_k\right) \qquad (18)$$

More importantly, the Schur complement matrix $\mathtt{J}_{\mathbf{u}_k}^\top\mathtt{S}_{\lambda_k}\mathtt{J}_{\mathbf{u}_k}$ reveals the local quadratic model assumed by Joint optimization solely in terms of $\boldsymbol{\Delta}\mathbf{u}$ and will play the central role in § 4.

3.2. Embedded Point Iterations (EPI)

Embedded point iterations (EPI) is a method proposed to accelerate classical bundle adjustment [17] by using a nested optimization approach: after computing the standard Gauss-Newton or LM updates, one set of unknowns (w.l.o.g. $\mathbf{v}$) are optimized with $\mathbf{u}$ fixed. EPI derives its name from how it is used in bundle adjustment, where $\mathbf{v}$ represents the 3D point structure. Since $\mathbf{v}$ is optimized in each iteration with respect to the current value of $\mathbf{u}$, it can be interpreted as a variant of Variable Projection. Consequently, it is sometimes identified with actual VarPro [1] but the difference to VarPro is that the Joint optimization model is used to update $\mathbf{u}$ (i.e. using (17) instead of (33)).

For SNLS problems, due to the residual $\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v})$ being linear in $\mathbf{v}$, the optimal iterate $\mathbf{v}_{k+1}$ can be computed in closed form given $\mathbf{u}_{k+1} := \mathbf{u}_k + \boldsymbol{\Delta}\mathbf{u}_k$,

$$\begin{aligned}\mathbf{v}_{k+1} &= \arg\min_{\mathbf{v}} \|\boldsymbol{\varepsilon}(\mathbf{u}_{k+1},\mathbf{v})\|_2^2 \\ &= \arg\min_{\mathbf{v}} \|\mathtt{G}(\mathbf{u}_{k+1})\mathbf{v}-\mathbf{z}(\mathbf{u}_{k+1})\|_2^2 \\ &= \mathtt{G}(\mathbf{u}_{k+1})^\dagger\mathbf{z}(\mathbf{u}_{k+1}).\end{aligned} \qquad (19)$$

Note that $\mathbf{v}_{k+1}$ is independent of the previous value of $\mathbf{v}$, and therefore (18) can be bypassed altogether in this case. The fact that the previous iterate $\mathbf{v}_k$ is optimal for $\|\boldsymbol{\varepsilon}(\mathbf{u}_k,\mathbf{v})\|_2^2$ implies that

$$0 = \mathtt{J}_{\mathbf{v}}(\mathbf{u}_k)^\top\boldsymbol{\varepsilon}(\mathbf{u}_k,\mathbf{v}_k) = \mathtt{J}_{\mathbf{v}_k}^\top\boldsymbol{\varepsilon}_k. \qquad (20)$$

Hence, (15) simplifies to

$$\begin{bmatrix}\mathtt{J}_{\mathbf{u}_k}^\top\mathtt{J}_{\mathbf{u}_k}+\lambda_k\mathtt{I} & \mathtt{J}_{\mathbf{u}_k}^\top\mathtt{J}_{\mathbf{v}_k}\\ \mathtt{J}_{\mathbf{v}_k}^\top\mathtt{J}_{\mathbf{u}_k} & \mathtt{J}_{\mathbf{v}_k}^\top\mathtt{J}_{\mathbf{v}_k}+\lambda_k\mathtt{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Delta}\mathbf{u}_k\\ \boldsymbol{\Delta}\mathbf{v}_k\end{bmatrix} = -\begin{bmatrix}\mathtt{J}_{\mathbf{u}_k}^\top\boldsymbol{\varepsilon}_k\\ 0\end{bmatrix} \qquad (21)$$

and using the Schur complement we finally obtain

$$\boldsymbol{\Delta}\mathbf{u}_k = -\left(\mathtt{J}_{\mathbf{u}_k}^\top\mathtt{S}_{\lambda_k}\mathtt{J}_{\mathbf{u}_k}+\lambda_k\mathtt{I}\right)^{-1}\mathtt{J}_{\mathbf{u}_k}^\top\boldsymbol{\varepsilon}_k. \qquad (22)$$

3.3. Variable Projection (VarPro)

VarPro reduces the problem of minimizing (1) over $\mathbf{u}$ and $\mathbf{v}$ into solving a nonlinear problem over $\mathbf{u}$ only. First, observe that the optimal value of $\mathbf{v}$ given $\mathbf{u}$ is

$$\mathbf{v}^*(\mathbf{u}) := \arg\min_{\mathbf{v}} \|\mathtt{G}(\mathbf{u})\mathbf{v}-\mathbf{z}(\mathbf{u})\|_2^2 = \mathtt{G}(\mathbf{u})^\dagger\mathbf{z}(\mathbf{u}). \qquad (23)$$

Inserting (23) into (1) yields the reduced problem,

$$\min_{\mathbf{u}} \|\boldsymbol{\varepsilon}^*(\mathbf{u})\|_2^2 := \min_{\mathbf{u}} \|\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v}^*(\mathbf{u}))\|_2^2 \qquad (24)$$

$$= \min_{\mathbf{u}} \left\|\left(\mathtt{G}(\mathbf{u})\mathtt{G}(\mathbf{u})^\dagger-\mathtt{I}\right)\mathbf{z}(\mathbf{u})\right\|_2^2. \qquad (25)$$

(25) can also be viewed as problem defined in a reduced subspace since we can reformulate the residual in (25) as

$$\begin{aligned}\boldsymbol{\varepsilon}^*(\mathbf{u}) &= \left(\mathtt{I}-\mathtt{G}(\mathbf{u})\mathtt{G}(\mathbf{u})^\dagger\right)\left(\mathtt{G}(\mathbf{u})\mathbf{v}-\mathbf{z}(\mathbf{u})\right)\\ &= \left(\mathtt{I}-\mathtt{J}_{\mathbf{v}}(\mathbf{u})\mathtt{J}_{\mathbf{v}}(\mathbf{u})^\dagger\right)\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v})\\ &= \mathtt{Q}_{\mathbf{v}}(\mathbf{u})\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v})\end{aligned} \qquad (26)$$

for any value of $\mathbf{v}$, where $\mathtt{Q}_{\mathbf{v}}(\mathbf{u})$ is the orthogonal projector defined in (13). Since $\mathbf{v}$ is projected out, the reduced model solely in terms of $\mathbf{u}$ is orthogonal, i.e. "agnostic", to perturbations of $\mathbf{v}$.

VarPro uses LM (see §2) to solve (25) and therefore requires the Jacobian of the reduced residual $\boldsymbol{\varepsilon}^*(\mathbf{u})$. The total derivative of (24) reads as

$$\begin{aligned}\mathtt{J}_{\mathbf{u}}^*(\mathbf{u}) &:= \frac{d\boldsymbol{\varepsilon}^*(\mathbf{u})}{d\mathbf{u}} = \frac{\partial\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{\partial\mathbf{v}}\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}}+\frac{\partial\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{\partial\mathbf{u}}\\ &= \mathtt{J}_{\mathbf{v}}(\mathbf{u},\mathbf{v}^*(\mathbf{u}))\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}}+\mathtt{J}_{\mathbf{u}}(\mathbf{u},\mathbf{v}^*(\mathbf{u})),\end{aligned} \qquad (27)$$

where $\mathtt{J}_{\mathbf{u}}$ and $\mathtt{J}_{\mathbf{v}}$ are the Jacobians of the original residual (10). $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u}$ can be derived analytically by using the differentiation rule of pseudo-inverse matrices in (4) as follows.

Computing $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u}$ and its approximations Differentiating $\mathbf{v}^*(\mathbf{u})$ using the product rule yields

$$\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} = \frac{d\left[\mathtt{G}(\mathbf{u})^\dagger\right]\mathbf{z}(\mathbf{u})}{d\mathbf{u}} + \mathtt{G}(\mathbf{u})^\dagger\frac{d\mathbf{z}(\mathbf{u})}{d\mathbf{u}}. \tag{28}$$

By noting that $\mathtt{G}(\mathbf{u}) = \mathtt{J}_\mathbf{v}(\mathbf{u})$ and applying the differentiation rule for a pseudo-inverse matrix, we obtain the following result (see [14] for details):

$$\begin{aligned}\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} &= -\mathtt{J}_\mathbf{v}(\mathbf{u})^\dagger\mathtt{J}_\mathbf{u}(\mathbf{u},\mathbf{v}^*(\mathbf{u})) \\ &\quad - (\mathtt{J}_\mathbf{v}(\mathbf{u})^\top\mathtt{J}_\mathbf{v}(\mathbf{u}))^{-1}\frac{d[\mathtt{J}_\mathbf{v}(\mathbf{u})]^\top\boldsymbol{\varepsilon}^*(\mathbf{u})}{d\mathbf{u}}.\end{aligned} \tag{29}$$

Inserting (29) into (27) yields

$$\mathtt{J}^*_\mathbf{u}(\mathbf{u}) = \mathtt{Q}_\mathbf{v}(\mathbf{u})\mathtt{J}_\mathbf{u}(\mathbf{u},\mathbf{v}^*(\mathbf{u})) - \mathtt{J}_\mathbf{v}(\mathbf{u})^{\dagger\top}\frac{d[\mathtt{J}_\mathbf{v}(\mathbf{u})]^\top\boldsymbol{\varepsilon}^*(\mathbf{u})}{d\mathbf{u}}. \tag{30}$$

Note that (29) (and therefore (30)) contains a contains a second order derivative of the residual (via $d[\mathtt{J}_\mathbf{v}(\mathbf{u})]/d\mathbf{u}$), and consequently approximations have been proposed to reduce the computation cost. One option is to use the coarse approximation $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u} \approx 0$, which is termed RW3 (following the taxonomy of Ruhe and Wedin [25]). The underlying assumption is, that $\mathbf{u}$ and $\mathbf{v}$ are indepedent, and the resulting method is essentially a block-coordinate method (which has shown generally poor performance for matrix factorization problems [6, 22, 12, 13]).

Another approximation, called RW2 (Ruhe and Wedin Algorithm 2), discards the second term in (29), leading to

$$\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} \approx -\mathtt{J}_\mathbf{v}(\mathbf{u})^\dagger\mathtt{J}_\mathbf{u}(\mathbf{u},\mathbf{v}^*(\mathbf{u})) \tag{31}$$

$$\Rightarrow\ \mathtt{J}^*_\mathbf{u}(\mathbf{u}) \approx \mathtt{Q}_\mathbf{v}(\mathbf{u})\mathtt{J}_\mathbf{u}(\mathbf{u},\mathbf{v}^*(\mathbf{u})). \tag{32}$$

Despite the naming convention, RW2 was first proposed by Kaufman [18] as an efficient way to implement VarPro. There is significant empirical evidence [18, 25, 11, 12, 13] over the past 40 years that RW2-VarPro has similar convergence property to the fully-derived VarPro while benefiting from reduced computational complexity. Consequently, we will focus on the RW2-approximated version of VarPro and and assume that VarPro refers to RW2-VarPro unless otherwise stated.

Update equations By feeding the approximated Jacobian from (32) into (9), we obtain the update equation for VarPro at iteration k:

$$\boldsymbol{\Delta}\mathbf{u}_k = -(\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I} - \mathtt{J}_{\mathbf{v}_k}\mathtt{J}^\dagger_{\mathbf{v}_k})\mathtt{J}_{\mathbf{u}_k} + \lambda_k\mathtt{I})^{-1}\mathtt{J}_k^\top\boldsymbol{\varepsilon}_k \tag{33}$$

where $\mathtt{J}_{\mathbf{u}_k} := \mathtt{J}_\mathbf{u}(\mathbf{u}_k,\mathbf{v}^*(\mathbf{u}_k))$, $\mathtt{J}_{\mathbf{v}_k} := \mathtt{J}_\mathbf{v}(\mathbf{u}_k)$ and $\boldsymbol{\varepsilon}_k := \boldsymbol{\varepsilon}(\mathbf{u}_k,\mathbf{v}_k) = \boldsymbol{\varepsilon}(\mathbf{u}_k,\mathbf{v}^*(\mathbf{u}_k))$. The above derivation uses the property that $\mathtt{Q}^2_\mathbf{v}(\mathbf{u}_k) = (\mathtt{I} - \mathtt{J}_{\mathbf{v}_k}\mathtt{J}^\dagger_{\mathbf{v}_k})^2 = \mathtt{I} - \mathtt{J}_{\mathbf{v}_k}\mathtt{J}^\dagger_{\mathbf{v}_k}$. Once $\mathbf{u}$ is updated, $\mathbf{v}$ is solved in closed form to be optimal for the new $\mathbf{u}$.

Improving numerical stability In (33), computing $\mathtt{J}_{\mathbf{v}_k}\mathtt{J}^\dagger_{\mathbf{v}_k} = \mathtt{J}_{\mathbf{v}_k}(\mathtt{J}^\top_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k})^{-1}\mathtt{J}^\top_{\mathbf{v}_k}$ accurately can be difficult if $\mathtt{J}_{\mathbf{v}_k}$ is ill-conditioned. One solution is to use the economy-size QR decomposition to form $\mathtt{J}_{\mathbf{v}_k} = \mathtt{J}_{\mathbf{v}_{\mathtt{Q},k}}\mathtt{J}_{\mathbf{v}_{\mathtt{R},k}}$, where $\mathtt{J}_{\mathbf{v}_{\mathtt{Q},k}}$ forms an orthonormal basis of $\mathrm{col}(\mathtt{J}_{\mathbf{v}_k})$ and $\mathtt{J}_{\mathbf{v}_{\mathtt{R},k}}$ is a square upper triangular matrix, then compute $\mathtt{J}_{\mathbf{v}_k}\mathtt{J}^\dagger_{\mathbf{v}_k} = \mathtt{J}_{\mathbf{v}_{\mathtt{Q},k}}\mathtt{J}^\top_{\mathbf{v}_{\mathtt{Q},k}}$. For matrix factorization problems, $\mathtt{J}_{\mathbf{v}_k}$ is block-diagonal, and therefore $\mathtt{J}_{\mathbf{v}_{\mathtt{Q},k}}$ can be obtained by performing the QR decomposition on each sub-block.

4. Unifying methods

In this section, we show how the Joint optimization and Variable Projection (VarPro) methods, which were separately reviewed in §3, are exactly related. We specifically compare between Joint optimization (Joint), Joint optimization with Embedded Point Iterations (Joint+EPI) and Variable Projection with RW2 approximation (VarPro).

4.1. Comparing initial conditions

Given an arbitrary initial point $(\mathbf{u}_0,\mathbf{v}_0)$, Joint and Joint+EPI start from $(\mathbf{u}_0,\mathbf{v}_0)$ whereas VarPro begins from $(\mathbf{u}_0,\mathbf{v}^*(\mathbf{u}_0))$ since the reduced residual $\boldsymbol{\varepsilon}^*(\mathbf{u}_0) = \boldsymbol{\varepsilon}(\mathbf{u}_0,\mathbf{v}^*(\mathbf{u}_0))$ does not incorporate the initial value of $\mathbf{v}$.

To show that this is not the major cause of the performance difference between the methods, we will assume that all methods are initialized from $(\mathbf{u}_0,\mathbf{v}_0)$, where $\mathbf{v}_0 = \mathbf{v}^*(\mathbf{u}_0)$, such that the initial conditions are identical.

4.2. Comparing update equations

In light of (17), (22), and (33) we are now in the position to directly compare the updates for $\boldsymbol{\Delta}\mathbf{u}_k$ induced by the different methods in Fig. 2. We also made use of the following relations to emphasize the connection between the various update rules: $\mathtt{J}^\dagger_{\mathbf{v}_k} = \mathtt{J}^{-0}_{\mathbf{v}_k}$, and $\boldsymbol{\varepsilon}_k = (\mathtt{I} - \mathtt{J}_{\mathbf{v}_k}\mathtt{J}^{-0}_{\mathbf{v}_k})\boldsymbol{\varepsilon}_k$ when $\mathbf{v}_k = \mathbf{v}^*(\mathbf{u}_k)$. using (26). It is apparent in Fig. 2 that the only difference between three methods for SNLS, which often behave very differently in practice, is the role of the damping parameter λ_k: Joint optimization enables damping of $\boldsymbol{\Delta}\mathbf{v}_k$ via λ_k in both the system matrix on the l.h.s. and in the reduced residual on the r.h.s., Joint optimization with EPI disables damping of $\boldsymbol{\Delta}\mathbf{v}_k$ in the reduced residual, and VarPro disables damping of $\boldsymbol{\Delta}\mathbf{v}_k$ entirely.

In addition to how $\boldsymbol{\Delta}\mathbf{u}$ is determined in each iteration, the three algorithms also differ in the update for $\boldsymbol{\Delta}\mathbf{v}$: Joint optimization uses the locally linear model to obtain the next iterate $\mathbf{v}_k$, whereas Joint+EPI and VarPro fully optimize $\mathbf{v}$ given the new value $\mathbf{u}_{k+1} = \mathbf{u}_k + \boldsymbol{\Delta}\mathbf{u}_k$.

The simple observations in particular regarding the updates of $\mathbf{u}$ have several important consequences:

1. First, they establish that VarPro for SNLS is in terms of derivation and implementation related to (but different

from) the more familiar Joint optimization approach combined with a Schur complement strategy. As a consequence, numerical implementations of VarPro should be comparable in terms of run-time to regular Joint optimization. We will discuss this topic in §4.3.

2. Second, it allows us to reason about the differences between Joint optimization and VarPro. In §5 we analyze the impact of damping of $\boldsymbol{\Delta}\mathbf{v}$ in matrix factorization problems, and how it distorts the update directions unfavorably in Joint optimization.
3. Finally, it is straightforward to unify these algorithms and to choose between them. In summary, there are two independent decisions: (i) is EPI enabled? (ii) is the damping paramete for $\boldsymbol{\Delta}\mathbf{v}$, which we denote by $\lambda_{\mathbf{v}}$, initialized to 0 (and remains at 0 during the iterations)? This gives rise to four algorithms: Joint, Joint+EPI, VarPro, and a Joint optimization method with unequal damping ($\lambda_{\mathbf{u}} \neq 0$, $\lambda_{\mathbf{v}} = 0$) on the unknowns as fourth alternative (see also Table 1). The steps in these algorithms are presented in [14].

4.3. A scalable algorithm for VarPro

Since there is little difference in implementation between the Joint and VarPro approaches, it should be theoretically possible to adapt any large-scale implementation for Joint optimization to use Variable Projection (VarPro). For large and dense problems, using a conjugate gradient-based algorithm to indirectly solve (33) would be preferred.

However, this alone may not replicate VarPro's large convergence basin. For matrix factorization problems, even though Boumal et al.'s RTRMC [4] indirectly solves the VarPro problem using a preconditioned conjugate gradient solver, it shows poor performance on several SfM datasets [13]. We believe that this is due to the ill-conditioned nature of these datasets, and therefore maintaining some degree of numerical stability is crucial in widening the convergence basin on this type of problem.

Our strategy comes down to solving a numerically more-stable QR-factorized reduced system in §3.3 with the MINRES solver [24], which is a conjugate gradient-type method for solving

$$\min_{\mathbf{x}} \|\mathtt{A}\mathbf{x} - \mathbf{b}\|_2^2 \tag{34}$$

where $\mathtt{A}$ is a symmetric matrix which can be definite, indefinite or singular.

We later demonstrate that the convergence basin of VarPro (using a direct solver) is mostly carried over to our strategy for affine bundle adjustment, which can be formulated as a matrix factorization problem.

5. Early stopping of Joint optimization

In this section we outline why Joint optimization is prone to early stopping (or stalling) for SNLS (for example, see Figure 3). This is in particular the case for matrix factorization problems, where "flat-lining" of the objective is frequently observed when using Joint optimization. It is tempting to assume that in such cases the Joint optimization method has reached a suboptimal local solution (or at least a stationary point), but the analysis below will reveal that in general this is not the case. Recalling Fig. 2, we can write the update equation for $\boldsymbol{\Delta}\mathbf{u}$ as follows,

	$\lambda_{\mathbf{v}} \neq 0$	$\lambda_{\mathbf{v}} = 0$
EPI off	Joint (4%)	(Joint+zero $\lambda_{\mathbf{v}}$) (0%)
EPI on	Joint+EPI (24%)	**VarPro** (94 %)

Table 1: A taxonomy of methods based on the findings of §4 and the corresponding average probabilities of reaching the best optimum on the *trimmed dinosaur* sequence in Table 2.

$$\left(\mathtt{J}_{\mathbf{u}}^\top(\mathtt{I} - \mathtt{J}_{\mathbf{v}}\mathtt{J}_{\mathbf{v}}^{-\lambda_{\mathbf{v}}})\mathtt{J}_{\mathbf{u}} + \lambda_{\mathbf{u}}\mathtt{I}\right)\boldsymbol{\Delta}\mathbf{u} = \mathbf{b}, \tag{35}$$

where $\lambda_{\mathbf{u}} > 0$, $\lambda_{\mathbf{v}} \geq 0$ and $\mathbf{b}$ is one of the r.h.s. in Fig. 2. Let the singular value decomposition of $\mathtt{J}_{\mathbf{v}}$ be given by

$$\mathtt{J}_{\mathbf{v}} = \begin{bmatrix}\mathcal{U} & \tilde{\mathcal{U}}\end{bmatrix}\begin{bmatrix}\Sigma \\ 0\end{bmatrix}\mathcal{V}^\top \tag{36}$$

with $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_q)$ and $\tilde{\mathcal{U}} = \mathrm{null}(\mathtt{J}_{\mathbf{v}}^\top)$. Then,

$$\begin{aligned}\mathtt{J}_{\mathbf{v}}^{-\lambda_{\mathbf{v}}} &= \left(\mathtt{J}_{\mathbf{v}}^\top\mathtt{J}_{\mathbf{v}} + \lambda_{\mathbf{v}}\mathtt{I}\right)^{-1}\mathtt{J}_{\mathbf{v}}^\top = \mathcal{V}(\Sigma^2 + \lambda_{\mathbf{v}}\mathtt{I})^{-1}\mathcal{V}^\top\mathtt{J}_{\mathbf{v}}^\top \\ &= \mathcal{V}(\Sigma^2 + \lambda_{\mathbf{v}}\mathtt{I})^{-1}\Sigma\mathcal{U}^\top. \end{aligned} \tag{37}$$

Consequently,

$$\begin{aligned}\mathtt{I} - \mathtt{J}_{\mathbf{v}}\mathtt{J}_{\mathbf{v}}^{-\lambda_{\mathbf{v}}} &= \left[\mathcal{U},\tilde{\mathcal{U}}\right]\left[\mathcal{U},\tilde{\mathcal{U}}\right]^\top - \mathcal{U}\Sigma^2(\Sigma^2 + \lambda_{\mathbf{v}}\mathtt{I})^{-1}\mathcal{U}^\top \\ &= \tilde{\mathcal{U}}\tilde{\mathcal{U}}^\top + \mathcal{U}\left(\mathtt{I} - \Sigma^2(\Sigma^2 + \lambda_{\mathbf{v}}\mathtt{I})^{-1}\right)\mathcal{U}^\top \\ &= \tilde{\mathcal{U}}\tilde{\mathcal{U}}^\top + \mathcal{U}\tilde{\Sigma}_{\lambda_{\mathbf{v}}}^2\mathcal{U}^\top, \end{aligned} \tag{38}$$

where $\tilde{\Sigma}_{\lambda_{\mathbf{v}}}$ is defined as $\mathrm{diag}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_q)$, in which $\tilde{\sigma}_i := \sqrt{\lambda_{\mathbf{v}}/(\sigma_i^2 + \lambda_{\mathbf{v}})}$ for $i = 1, \ldots, q$. Observe that (35) is also the first order optimality condition for

$$\begin{aligned}&\min_{\boldsymbol{\Delta}\mathbf{u}} \left\|\left(\tilde{\mathcal{U}} + \mathcal{U}\tilde{\Sigma}_{\lambda_{\mathbf{v}}}\right)^\top\mathtt{J}_{\mathbf{u}}\boldsymbol{\Delta}\mathbf{u}\right\|_2^2 + \lambda_{\mathbf{u}}\|\boldsymbol{\Delta}\mathbf{u}\|^2 - 2\mathbf{b}^\top\boldsymbol{\Delta}\mathbf{u} \\ &= \min_{\boldsymbol{\Delta}\mathbf{u}} \left\|\begin{bmatrix}\tilde{\mathcal{U}}^\top \\ \tilde{\Sigma}_{\lambda_{\mathbf{v}}}\mathcal{U}^\top\end{bmatrix}\mathtt{J}_{\mathbf{u}}\boldsymbol{\Delta}\mathbf{u}\right\|_2^2 + \lambda_{\mathbf{u}}\|\boldsymbol{\Delta}\mathbf{u}\|^2 - 2\mathbf{b}^\top\boldsymbol{\Delta}\mathbf{u} \end{aligned} \tag{39}$$

since $\tilde{\mathcal{U}}^\top\mathcal{U} = 0$. (39) reveals the local quadratic model of the least squares objective w.r.t. $\boldsymbol{\Delta}\mathbf{u}$ used by the algorithm. If $\lambda_{\mathbf{v}} = 0$, i.e. trust-region damping on $\mathbf{v}$ is deactivated, then the leading quadratic term models the objective only in the null-space of $\mathtt{J}_{\mathbf{v}}^\top$.

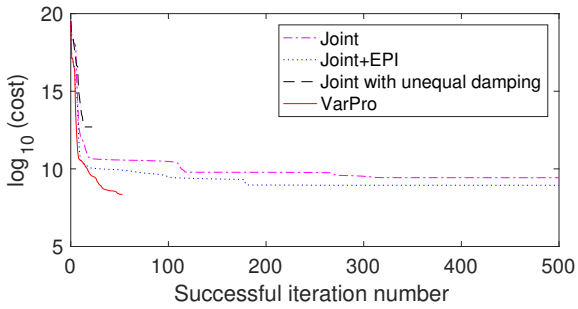


Figure 3: Convergence plots for each algorithm. For this example, VarPro converges to the best known optimum (4.23×10^3) in less than 100 iterations whereas Joint and Joint+EPI both exhibit flat-lining behaviours. Joint with unequal damping terminates quickly at a bad minimum.

If all singular values σ_i are relatively large compared to the current value of $\lambda_{\mathbf{v}}$, then $\tilde{\sigma}_i \approx 0$, and the perturbations in the linear model (and in the update direction $\boldsymbol{\Delta}\mathbf{u}$) are negligible. If in contrast $\lambda_{\mathbf{v}} > 0$ and one or several singular values σ_i are (close to) zero for some i, then $\tilde{\sigma}_i \approx 1$. In the limit $\lambda \to \infty$, we have $\tilde{\Sigma}_{\lambda_{\mathbf{v}}} = \mathtt{I}$, and due to $[\tilde{\mathcal{U}}, \mathcal{U}]$ being a rotation matrix, (39) degenerates to a block-coordinate method for $\mathbf{u}$, which is known to perform poorly on matrix factorization problems [22, 13]).

We can focus in the following on the analysis of the block-coordinate method, since if $\sigma_i \ll \lambda_{\mathbf{v}}$ only for some i, then $\tilde{\Sigma}_{\lambda_{\mathbf{v}}} \approx \operatorname{diag}(1, \ldots, 1, 0, \ldots, 0)$ and solving (39) corresponds essentially to a block-coordinate approach. Now we assume that $\mathtt{J}_{\mathbf{v}}$ is rank deficient. For simplicity we will make the even stronger assumption that $\mathtt{J}_{\mathbf{v}} \approx 0$ (and therefore $\sigma_i \ll \lambda_{\mathbf{v}}$ and $\tilde{\Sigma}_{\lambda_{\mathbf{v}}} \approx \mathtt{I}$).

To illustrate an intuitive idea, we focus on the updates $\boldsymbol{\Delta}\mathbf{v}$ computed by VarPro and Joint optimization in their respective linear systems. Note that for VarPro and Joint+EPI, these updates are not actually used in updating $\mathbf{v}$ as EPI takes care of it, but they still play a key role in determining the updates $\boldsymbol{\Delta}\mathbf{u}$ since $\mathbf{u}$ and $\mathbf{v}$ are correlated in SNLS problems. As written in (18), the Joint optimization family of algorithms compute

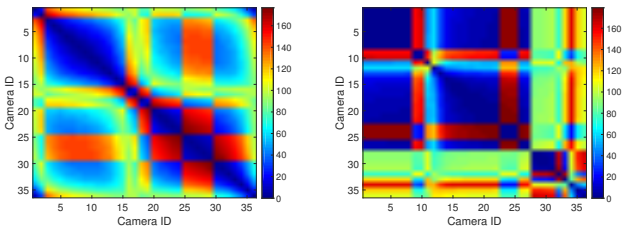
$$\boldsymbol{\Delta}\mathbf{v}_{\text{joint}} = -\mathtt{J}_{\mathbf{v}}^{-\lambda_{\mathbf{v}}} (\boldsymbol{\varepsilon} + \mathtt{J}_{\mathbf{u}}\boldsymbol{\Delta}\mathbf{u}) \approx -\frac{1}{\lambda_{\mathbf{v}}} \mathtt{J}_{\mathbf{v}}^{\top} (\boldsymbol{\varepsilon} + \mathtt{J}_{\mathbf{u}}\boldsymbol{\Delta}\mathbf{u})$$

and therefore

$$\|\boldsymbol{\Delta}\mathbf{v}_{\text{joint}}\| \approx \frac{1}{\lambda_{\mathbf{v}}} \left\|\mathtt{J}_{\mathbf{v}}^{\top} (\boldsymbol{\varepsilon} + \mathtt{J}_{\mathbf{u}}\boldsymbol{\Delta}\mathbf{u})\right\|$$

using our assumption $\sigma_i \ll \lambda_{\mathbf{v}}$ for all i. On the other hand, our analysis in §4 shows that VarPro has no damping on $\mathbf{v}$, and therefore its corresponding update $\boldsymbol{\Delta}\mathbf{v}$ is

$$\boldsymbol{\Delta}\mathbf{v}_{\text{varpro}} = -\mathtt{J}_{\mathbf{v}}^{\dagger} (\boldsymbol{\varepsilon} + \mathtt{J}_{\mathbf{u}}\boldsymbol{\Delta}\mathbf{u})$$



(a) VarPro (best optimum) (b) Joint+EPI (failure)

Figure 4: Angles between the directions of output affine cameras from the *trimmed dinosaur* dataset in the projective frame. (b) shows that some neighbouring cameras (e.g. between ID 1 to 8) are closely aligned together when it fails to reach the best known optimum value of 4.23×10^3.

leading to

$$\|\boldsymbol{\Delta}\mathbf{v}_{\text{varpro}}\| \geq \frac{1}{\bar{\sigma}^2} \left\|\mathtt{J}_{\mathbf{v}}^{\top} (\boldsymbol{\varepsilon} + \mathtt{J}_{\mathbf{u}}\boldsymbol{\Delta}\mathbf{u})\right\|,$$

where $\bar{\sigma} = \max_i \sigma_i$. Consequently, we obtain that $\|\boldsymbol{\Delta}\mathbf{v}_{\text{joint}}\| / \|\boldsymbol{\Delta}\mathbf{v}_{\text{varpro}}\| \approx \bar{\sigma}^2/\lambda_{\mathbf{v}} \ll 1$ under our assumptions. Hence, the update $\boldsymbol{\Delta}\mathbf{v}_{\text{joint}}$ will be much smaller than the update $\boldsymbol{\Delta}\mathbf{v}_{\text{varpro}}$. In the more general setting with $\mathtt{J}_{\mathbf{v}}$ being near singular instead of close to the zero matrix we obtain that $\boldsymbol{\Delta}\mathbf{v}_{\text{joint}}$ will be much shorter than $\boldsymbol{\Delta}\mathbf{v}_{\text{varpro}}$ in the certain directions. The lack of update $\boldsymbol{\Delta}\mathbf{v}_{\text{joint}}$ (in certain directions) is reflected in the local quadratic model (39) for $\boldsymbol{\Delta}\mathbf{u}$: reducing residuals is entirely the responsibility of $\boldsymbol{\Delta}\mathbf{u}$.

Note, that if $\mathtt{J}_{\mathbf{v}}$ is far from being singular, $\mathtt{J}_{\mathbf{v}}^{-\lambda_{\mathbf{v}}}$ is close to $\mathtt{J}_{\mathbf{v}}^{\dagger}$ and $\boldsymbol{\Delta}\mathbf{v}_{\text{joint}} \approx \boldsymbol{\Delta}\mathbf{v}_{\text{varpro}}$. Thus, in this case Joint and VarPro optimization behave similarly.

To see how this affects the algorithm performance, we resort to an example of affine bundle adjustment, where $\mathbf{u}$ is a set of camera parameters and $\mathbf{v}$ is a set of 3D points. For this problem, nearly-singular $\mathtt{J}_{\mathbf{v}}$ can arise when a bundle of rays corresponding to a 3D point is almost collinear. In such a case, the Joint optimization submodel fixes $\boldsymbol{\Delta}\mathbf{v}$ (the point update) in the depth direction, and consequently this places more burden on the camera parameters to reduce the objective. On the other hand, VarPro allows unconstrained point updates $\boldsymbol{\Delta}\mathbf{v}$, allowing camera updates $\boldsymbol{\Delta}\mathbf{u}$ to make more adventurous moves.

6. Experimental results

To verify our analysis in §4 and §5, we conducted two experiments solving affine bundle adjustment, which can be formulated as a matrix factorization problem [27]. It has been shown empirically [15] that the obtained affine solutions could be used to bootstrap projective bundle adjustment.

In the first experiment, we tested our VarPro-MINRES strategy against Joint optimization (Joint), Joint optimiza-

Dataset	f	n	Missing (%)	Joint	Joint+EPI	VarPro	VarPro-MinRes*
Blue teddy bear (trimmed)	196	827	80.7	10 (238)	20 (155)	**88 (22.3)**	76 (21.9)
Corridor	11	737	50.2	40 (8.71)	4 (14.8)	100 (1.07)	**100 (0.78)**
Dinosaur (trimmed)	36	319	76.9	4 (5.95)	24 (9.38)	**94 (1.55)**	**99 (3.96)**
Dinosaur including outliers	36	4983	90.8	0 (28.6)	0 (62.1)	**100 (13.9)**	36 (38.9)
House	10	672	57.7	44 (4.90)	8 (9.71)	**100 (0.30)**	**100 (0.41)**
Road scene #47	11	150	47.1	44 (1.88)	32 (3.00)	**100 (0.16)**	**100 (0.17)**
Stockholm Guildhall (trimmed)	43	1000	18.0	92 (45.1)	48 (35.7)	100 (22.8)	**100 (3.12)**
Wilshire	190	411	60.7	38 (409)	94 (9.90)	100 (7.64)	**100 (1.96)**
Ladybug (skeleton)	49	7776	91.6	0 (77.3)	0 (155)	**50 (49.7)**	0 (155)
Trafalgar Square (skeleton)	21	11315	84.7	0 (76.2)	0 (160)	**100 (14.7)**	100 (56.4)
Dubrovnik (skeleton)	16	22106	76.3	38 (159)	0 (346)	**100 (23.6)**	100 (32.9)
Venice (skeleton)	52	64053	89.6	0 (913)	0 (1495)	**80 (123)**	60 (329)

Table 2: Experimental results for affine bundle adjustment on various datasets. For each dataset and each algorithm, the percentages of runs which converged to the best known optimum of that dataset is reported with corresponding median runtime in seconds inside the parentheses. *We have a comparatively less efficient implementation of VarPro-MINRES while other algorithms are based on our patched version of the Ceres Solver [1] library.

tion with Embedded Point Iterations (Joint+EPI) and Variable Projection (VarPro) on a variety of SfM datasets. VarPro-MINRES was less efficiently implemented in MATLAB while the other methods were implemented within the Ceres Solver framework [1]. As our code analysis showed that the current Ceres version implements Joint+EPI rather than VarPro, we patched Ceres to properly support VarPro (without MINRES) based on the unification work from §4.

For each run on each algorithm, we sampled each element of $\mathbf{u}_0$ from $\mathcal{N}(0,1)$ and then used $\mathbf{u}_0$ to generate $\mathbf{v}_0 = \mathbf{v}^*(\mathbf{u}_0)$ in order to ensure equal initial conditions across all algorithms. On each dataset, we ran each algorithm for a fixed number of times and reported the fraction of runs reaching the best known optimum of the dataset. For some datasets, the best optimum values are known (e.g. dinosaur and trimmed dinosaur), but for others we used the best objective values we observed in all runs across all implemented algorithms. We set the function tolerance to 10^{-9} and the maximum number of successful iterations to 300. For VarPro-MINRES, we set the relative tolerance to 10^{-6} and the maximum number of inner iterations to 300. The reported objective values in Fig. 3 and Fig. 4 are half of the values computed from (2).

Table 2 shows that VarPro-MINRES mostly retains the large basin of convergence observed for standard VarPro. Note that its slower speed for larger sparse dataset may be due to to its comparatively inefficient implementation.

In the second experiment, we observed the behaviours of the four algorithms described in §4 on the *trimmed dinosaur* dataset [6] from a random starting point. This circular motion-derived sequence consists of 36 reasonably weak-perspective cameras and 319 inlier point tracks. 76.9% of the elements are missing and exhibit a banded occlusion pattern without a loop closure. Table 1 shows that the use of EPI improves the success rate on its own but must be accompanied by the removal of the damping factor $\lambda_{\mathbf{v}}$ (i.e. switch to VarPro) to dramatically boost the algorithm performance.

Fig. 3 illustrates the typical "stalling" behaviour shared by Joint and Joint+EPI. In §5, we claimed that this behaviour is observed when a batch of camera rays are nearly collinear in affine bundle adjustment. This statement is verified in Fig. 4 and Fig. 1, which shows that the angles between some affine camera directions (e.g. a set of cameras from ID 1 to ID 8) are very small at the point of failure for the Joint optimization-based algorithm. Such collinear alignment of rays is not observed in the optimum reached by VarPro.

7. Conclusions

In this paper, we showed that Joint optimization and Variable Projection (VarPro), which are two apparently very different methods of solving separable nonlinear least squares problems, can be unified. The most important difference between Joint optimization and VarPro is the unbalanced trust-region assumption in the latter method. The revealed connection between the two methods shows that VarPro can be in principle implemented as efficiently as standard Joint optimization, which allows VarPro to be demonstrated on significantly larger datasets than reported in the literature. We also tackled the question why VarPro has much higher success rates than Joint optimization for certain problem classes important in computer vision.

Acknowledgements The work was supported by Microsoft and Toshiba Research Europe. We further thank Roberto Cipolla for additional funding support.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org, 2014. 2, 4, 8

[2] P. N. Belhumeur and D. Kriegman. What is the set of images of an object under all possible lighting conditions? In *1996 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 270–277, Jun 1996. 2

[3] J. Bennett and S. Lanning. The Netflix prize. In *Proceedings of 2007 KDD Cup and Workshop*, pages 3–6, 2007. 2

[4] N. Boumal and P.-A. Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, pages 406–414. 2011. 2, 6

[5] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *2000 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 690–696, 2000. 2

[6] A. M. Buchanan and A. W. Fitzgibbon. Damped Newton algorithms for matrix factorization with missing data. In *2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 316–322, 2005. 2, 5, 8

[7] T. J. Cashman and A. W. Fitzgibbon. What shape are dolphins? building 3D morphable models from 2D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):232–244, 2013. 1

[8] T. F. Chan and S. Esedoglu. Aspects of total variation regularized L^1 function approximation. *SIAM Journal on Applied Mathematics*, 65(5):1817–1837, 2004. 1

[9] P. Chen. Optimization algorithms on subspaces: Revisiting missing data problem in low-rank matrix. *International Journal of Computer Vision (IJCV)*, 80(1):125–142, 2008. 2

[10] G. H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on Numerical Analysis (SINUM)*, 10(2):413–432, 1973. 2

[11] G. H. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its applications. In *Proceedings of Inverse Problems*, pages 1–26, 2002. 2, 5

[12] P. F. Gotardo and A. M. Martinez. Computing smooth time trajectories for camera and deformable shape in structure from motion with occlusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(10):2051–2065, Oct 2011. 2, 5

[13] J. H. Hong and A. W. Fitzgibbon. Secrets of matrix factorization: Approximations, numerics, manifold optimization and random restarts. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4130–4138. 2, 5, 6, 7

[14] J. H. Hong, C. Zach, and A. W. Fitzgibbon. Revisiting the variable projection method for separable nonlinear least squares problems: Supplementary document, 2017. https://github.com/jhh37/varpro. 5, 6

[15] J. H. Hong, C. Zach, A. W. Fitzgibbon, and R. Cipolla. Projective bundle adjustment from arbitrary initialization using the variable projection method. In *14th European Conference on Computer Vision (ECCV)*, pages 477–493, 2016. 7

[16] H. Ishikawa. Exact optimization for Markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(10):1333–1336, 2003. 1

[17] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I. S. Kweon. Pushing the envelope of modern methods for bundle adjustment. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1474–1481, 2010. 4

[18] L. Kaufman. A variable projection method for solving separable nonlinear least squares problems. *BIT Numerical Mathematics*, 15(1):49–57, 1975. 5

[19] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quarterly of Applied Mathmatics*, 2(2):164–168, 1944. 3

[20] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. 3

[21] T. Okatani and K. Deguchi. On the Wiberg algorithm for matrix factorization in the presence of missing components. *International Journal of Computer Vision (IJCV)*, 72(3):329–337, 2007. 2

[22] T. Okatani, T. Yoshida, and K. Deguchi. Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 842–849, 2011. 2, 5, 7

[23] D. P. O'Leary and B. W. Rust. Variable projection for nonlinear least squares problems. *Computational Optimization and Applications*, 54(3):579–593, Apr 2013. 2

[24] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975. 6

[25] A. Ruhe and P. Å. Wedin. Algorithms for separable nonlinear least squares problems. *SIAM Review (SIREV)*, 22(3):318–337, 1980. 2, 5

[26] D. Strelow. General and nested Wiberg minimization: L2 and maximum likelihood. In *12th European Conference on Computer Vision (ECCV)*, pages 195–207. 2012. 2

[27] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision (IJCV)*, 9(2):137–154, 1992. 2, 7

[28] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - A modern synthesis. In *International Workshop on Vision Algorithms: Theory and Practice*, 1999 IEEE International Conference on Computer Vision (ICCVW), pages 298–372, 2000. 3

[29] Y. Weiss, C. Yanover, and T. Meltzer. MAP estimation, linear programming and belief propagation with convex free energies. In *Uncertainty in Artificial Intelligence*, 2007. 1

[30] T. Wiberg. Computation of principal components when data are missing. In *2nd Symposium of Computational Statistics*, pages 229–326, 1976. 2

[31] C. Zach. Robust bundle adjustment revisited. In *13th European Conference on Computer Vision (ECCV)*, pages 772–787, 2014. 1