Projective Bundle Adjustment from Arbitrary Initialization using the Variable Projection Method

Je Hyeong Hong[1], Christopher Zach[2], Andrew Fitzgibbon[3] and Roberto Cipolla[1]

[1]University of Cambridge, UK ** {jhh37,rc10001}@cam.ac.uk
[2]Toshiba Research Europe, Cambridge, UK christopher.m.zach@gmail.com
[3]Microsoft, Cambridge, UK awf@microsoft.com

Abstract. Bundle adjustment is used in structure-from-motion pipelines as final refinement stage requiring a sufficiently good initialization to reach a useful local mininum. Starting from an arbitrary initialization almost always gets trapped in a poor minimum. In this work we aim to obtain an initialization-free approach which returns global minima from a large proportion of purely random starting points. Our key inspiration lies in the success of the Variable Projection (VarPro) method for affine factorization problems, which have close to 100% chance of reaching a global minimum from random initialization. We find empirically that this desirable behaviour does not directly carry over to the projective case, and we consequently design and evaluate strategies to overcome this limitation. Also, by unifying the affine and the projective camera settings, we obtain numerically better conditioned reformulations of original bundle adjustment algorithms.

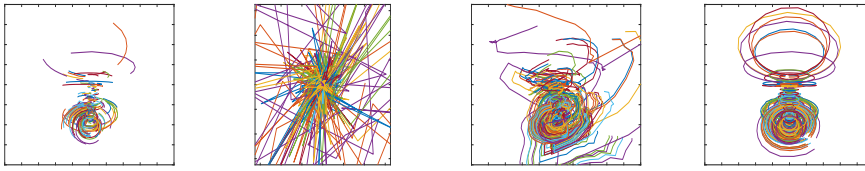
Keywords: projective bundle adjustment, variable projection, nonlinear least squares

1 Introduction

Standard structure from motion (SfM) approaches are typically multi-stage pipelines comprising of feature matching or tracking, initial structure and camera estimation, and final nonlinear refinement stages. While feature matching and tracking and the nonlinear refinement stage have well-established gold standard implementations (most notably matching using SIFT features, tracking via Lucas-Kanade and nonlinear refinement via Levenberg-Marquardt), no elegant and generally accepted framework for estimating the initial poses and 3D structure from feature tracks is known. Even when one has a sensible starting point (initial 3D reconstruction) available, accumulated drift, undetected loop closures, etc., require a large basin of convergence for bundle adjustment to succeed. The essence of this work is widening the convergence basin of bundle adjustment thereby improving SfM systems.

If an affine or weak perspective camera model is given (or assumed), determining pose and 3D structure amounts to solving a matrix factorization problem, which is an easy task if all points are visible in every image. If the visibility pattern is sparse and structured as induced by feature tracking, matrix factorization algorithms employing the Variable Projection (VarPro) method are highly successful (i.e. return a global optimum

** Much of the work was done while the first author was an intern at Toshiba Research Europe.



(a) Observed tracks (b) Initial (377.29) (c) Best affine (9.38) (d) Best proj. (0.84)

Fig. 1. Visualization of the Di2 (see Table 7) tracks recovered using our two-stage meta-algorithms. In each run, each meta-algorithm is initialized from random camera poses and points (Fig. 1b). In the first stage, it performs affine bundle adjustment using either a Linear or Nonlinear VarPro-based algorithms, reaching the best affine optimum (Fig. 1c) in 91% of all runs. The outputs are then used to initialize projective bundle adjustment. Although Di2 has strong perspective effects, our recommended meta-algorithms (TSMA1 and TSMA2) both reach the best projective optimum (Fig. 1d) in 90–98% of all runs.

in a large fraction of runs) even when the poses and the 3D structure are initialized arbitrarily [17, 10]. Thus, the initial SfM computation can be entirely bypassed in the affine case. One obvious question is whether this is also true when using a pinhole camera model. This the main motivation of this work.

Formally, we are interested in finding global minimizers of the following nonlinear least squares projective bundle adjustment problem

$$\min_{\{\mathtt{P}_i\}\{\tilde{\mathbf{x}}_j\}} \sum_{\{i,j\}\in\Omega} \|\boldsymbol{\pi}\left(\mathtt{P}_i\tilde{\mathbf{x}}_j\right) - \tilde{\mathbf{m}}_{ij}\|_2^2 \tag{1}$$

without requiring good initial values for the unknowns. In (1) the unknowns are as follows: $\mathtt{P}_i \in \mathbb{R}^{3\times 4}$ is the projective camera matrix for frame i and $\tilde{\mathbf{x}}_j \in \mathbb{R}^4$ is the homogeneous vector of coordinates of point j. $\tilde{\mathbf{m}}_{ij} \in \mathbb{R}^2$ is the observed projection of point j in frame i. Ω denotes a set of visible observations and $\boldsymbol{\pi}\left(\cdot\right)$ is the perspective division such that $\boldsymbol{\pi}([x, y, z]^\top) := [x/z, y/z]^\top$. This division introduces nonlinearity to the objective, and thus we can interpret (1) as a nonlinear matrix factorization problem.

Our quest to solve (1) directly without the help of an initial structure and motion estimation step leads to the following contributions:

+ **Extension of Ruhe and Wedin algorithms:** we extend the separable nonlinear least squares algorithms in [19] to apply to nonseparable problems such as (1).
+ **Unification of affine and projective cases:** we unify affine and projective bundle adjustment as special cases of a more general problem class. As a byproduct we obtain numerically better conditioned formulations for each of the special cases.
+ **Simple two-stage meta-algorithms:** we provide numerical experiments to identify the method yielding the highest success rate overall on real and synthetic datasets. We conclude that each of two winning methods is a Variable Projection (VarPro) method-based two-stage approach, which uses either a traditional or proposed numerically stable affine bundle adjustment algorithm followed by the proposed projective bundle adjustment algorithm.

Conversely, there are limitations of this work: the scope is confined to the L^2-norm projective formulation, and consequently we may encounter new challenges when incoporating robust kernel techniques and/or extending this work to the calibrated case, which is more frequently used in practice. We discuss the iteration complexity of our proposed algorithms but do not include timing measurements as we believe meaningful run-time figures require comparable implementations (and our code in [11] is inefficient as it has not incorporated the speed-up tricks mentioned in [2, 4, 10]).

1.1 Related work

In this section we briefly summarize relevant literature. The first seminal work dates back to Wiberg [24], who investigated the task of matrix factorization under missing data and whose name is today associated with the most successful method to solve this problem. The method is based on the principle of Variable Projection, which rewrites the objective in terms of a reduced set of unknowns by "minimizing out" the remaining ones. This approach is in particular promising, when the dependencies between unknowns forms a bipartite graph (which is the case in the structure from motion setting). In several works it has been experimentally verified that the Wiberg/VarPro method is far superior to naive joint optimization for matrix factorization problems (e.g. [5, 17, 10]), which explains the interest in the more difficult to implement VarPro methods.

In computer vision the connection between matrix factorization and affine structure from motion—but without missing data—was explored in [22]. Solving projective structure from motion via matrix factorization is more difficult, and it requires iterative methods even in the fully observed case (e.g. [21]). One step towards the application of VarPro methods in projective problems is the Nonlinear VarPro extension explored by Strelow [20], which we take as a starting point for our implementation.

All methods mentioned so far are ideally designed not to require a careful initialization for the unknown cameras and 3D structure (or matrix factors in the general case), and VarPro-derived methods work well for matrix factorization tasks even with random values as initializer. In SfM applications strong geometric constraints (we refer to [9] for a comprehensive treatment) can be used to determine sensible initial cameras and 3D structure. This initialization is subsequently used as starting point for nonlinear least-squares optimization (termed bundle adjustment) over all unknowns (see [23] for a review). Determining a good starting point for bundle adjustment is a non-trivial problem and expensive to solve in the general case. Compared to two-view and three-view geometry and to full-scale bundle adjustment this step of finding a good initializer also lacks in theoretical understanding. Hence, it is beneficial to bypass this stage altogether and investigate initialization-free methods for bundle adjustment.

2 Known methods for bivariate least-squares optimization

Bivariate least-squares solves

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

where $\mathbf{u}$ and $\mathbf{v}$ are sets of model parameters and $\boldsymbol{\varepsilon}$ is the residual vector. We can solve this by using various methods, namely Joint optimization, Variable Projection (Linear [6] and Nonlinear VarPro [20]) and Alternating least-squares (ALS), which is equivalent to RW3 (see §2.2 and §3).

The key to implementing all these methods is the use of the Levenberg-Marquardt (LM) algorithm [13, 15], which is a widely used trust-region strategy.

The Levenberg-Marquardt algorithm (LM)

LM [13, 15] is an extension of the Gauss-Newton algorithm, which minimizes $\|\boldsymbol{\varepsilon}(\mathbf{x})\|_2^2$ by iteratively solving its linearization and updating $\mathbf{x}$ accordingly. At each iteration, the Gauss-Newton update $\boldsymbol{\Delta}\mathbf{x}$ is the solution of the linearized problem

$$\underset{\boldsymbol{\Delta}\mathbf{x}}{\arg\min} \|\boldsymbol{\varepsilon}(\mathbf{x}) + \mathtt{J}(\mathbf{x})\boldsymbol{\Delta}\mathbf{x}\|_2^2 \tag{3}$$

where $\mathbf{x}$ denotes the parameter values from the previous iteration and $\mathtt{J}(\mathbf{x}) := \partial\boldsymbol{\varepsilon}(\mathbf{x})/\partial\mathbf{x}$. This step is likely to lead to a lower objective if the local cost surface about $\mathbf{x}$ resembles a quadratic model, but otherwise may lead to a higher objective. To overcome this, LM incorporates a regularizer to control the step size and find the augmented solution

$$\underset{\boldsymbol{\Delta}\mathbf{x}}{\arg\min} \|\boldsymbol{\varepsilon}(\mathbf{x}) + \mathtt{J}(\mathbf{x})\boldsymbol{\Delta}\mathbf{x}\|_2^2 + \lambda\|\boldsymbol{\Delta}\mathbf{x}\|_2^2 \tag{4}$$

where λ is known as the damping factor which we tune to decrease the cost. This parameter indicates the size of the trust region — the smaller the value of λ, the larger the region that can be "trusted" as quadratic.

2.1 Joint optimization

Joint optimization solves for all parameters simultaneously. This is achieved by stacking into the vector $\mathbf{x} = [\mathbf{u}; \mathbf{v}]$ and using a Newton-like solver such as LM. In general, the update at iteration k is

$$[\mathbf{u}_{k+1}; \mathbf{v}_{k+1}] = \mathbf{x}_{k+1} = \mathbf{x}_k - (\mathtt{H}(\mathbf{x}_k) + \lambda\mathtt{I})^{-1}\mathbf{g}(\mathbf{x}_k) \tag{5}$$

where $\mathtt{H}(\mathbf{x}_k)$ is the Hessian (or its approximation) of $\|\boldsymbol{\varepsilon}(\mathbf{x})\|_2^2$ at $\mathbf{x}_k$, $\mathbf{g}(\mathbf{x}_k)$ is the gradient $\nabla_{\mathbf{x}}\|\boldsymbol{\varepsilon}(\mathbf{x}_k)\|_2^2$ and λ is the damping factor. A widely used Hessian approximation, which is also used by LM (and throughout this paper), is the Gauss-Newton matrix $2\mathtt{J}(\mathbf{x}_k)^\top\mathtt{J}(\mathbf{x}_k)$ where $\mathtt{J}(\mathbf{x}_k) := \partial\boldsymbol{\varepsilon}(\mathbf{x}_k)/\partial\mathbf{x}$.

2.2 Linear Variable Projection (Linear VarPro)

Linear VarPro [6] is an approach for solving separable nonlinear least-squares [19], which is a subset of bivariate optimization problems and has a property that the residual vector is linear in at least one of two variables, e.g.

$$\min_{\mathbf{u},\mathbf{v}} \|\boldsymbol{\varepsilon}(\mathbf{u}, \mathbf{v})\|_2^2 = \min_{\mathbf{u},\mathbf{v}} \|\mathtt{A}(\mathbf{u})\mathbf{v} - \mathbf{b}\|_2^2 \tag{6}$$

where $\mathbf{u}$ and $\mathbf{v}$ are sets of model parameters, $\boldsymbol{\varepsilon}$ is the residual vector, $\mathtt{A}(\mathbf{u})$ is a linear operator which depends on $\mathbf{u}$ and $\mathbf{b}$ is a constant vector. Since $\boldsymbol{\varepsilon}$ is linear in $\mathbf{v}$, we have a direct solution for $\mathbf{v}$ that minimizes (6) given $\mathbf{u}$ which we call

$$\mathbf{v}^*(\mathbf{u}) := \arg\min_{\mathbf{v}} \|\mathtt{A}(\mathbf{u})\mathbf{v} - \mathbf{b}\|_2^2 = \mathtt{A}^\dagger(\mathbf{u})\mathbf{b}. \tag{7}$$

Substituting $\mathbf{v}^*(\mathbf{u})$ for $\mathbf{v}$ in (6) yields

$$\min_{\mathbf{u}} \|\boldsymbol{\varepsilon}^*(\mathbf{u})\|_2^2 := \min_{\mathbf{u}} \|\mathtt{A}(\mathbf{u})\mathbf{v}^*(\mathbf{u}) - \mathbf{b}\|_2^2 = \min_{\mathbf{u},\mathbf{v}} \| \left(\mathtt{A}(\mathbf{u})\mathtt{A}^\dagger(\mathbf{u}) - \mathtt{I}\right)\mathbf{b}\|_2^2 \tag{8}$$

which is a nonlinear reduced problem in $\mathbf{u}$ that can be solved using LM.

In [7], the authors claim that this reduced objective is almost always better conditioned than the original one. Although no formal proof is provided, we can find empirical evidence in matrix factorization [10].

Deriving the Jacobian of the reduced problem First, we write the Jacobian of the original problem (6) as

$$\mathtt{J}(\mathbf{u},\mathbf{v}) := \left[\frac{\partial\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v})}{\partial\mathbf{u}} \; \frac{\partial\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v})}{\partial\mathbf{v}}\right] = \left[\frac{\partial\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v})}{\partial\mathbf{u}} \;\; \mathtt{A}(\mathbf{u})\right] =: \left[\mathtt{J}_{\mathbf{u}}(\mathbf{u},\mathbf{v}) \; \mathtt{J}_{\mathbf{v}}(\mathbf{u})\right]. \tag{9}$$

We then express (using the chain rule) the Jacobian of the reduced problem (8) as

$$\mathtt{J}^*(\mathbf{u}) := \frac{d\boldsymbol{\varepsilon}^*(\mathbf{u})}{d\mathbf{u}} = \frac{d\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{d\mathbf{u}} = \frac{\partial\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{\partial\mathbf{u}} + \frac{\partial\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{\partial\mathbf{v}}\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} \tag{10}$$

$$= \mathtt{J}_{\mathbf{u}}(\mathbf{u},\mathbf{v}^*(\mathbf{u})) + \mathtt{J}_{\mathbf{v}}(\mathbf{u})\frac{d}{d\mathbf{u}}\left[\mathtt{A}(\mathbf{u})^\dagger\mathbf{b}\right] \tag{11}$$

$$= \mathtt{J}_{\mathbf{u}}(\mathbf{u},\mathbf{v}^*(\mathbf{u})) + \mathtt{J}_{\mathbf{v}}(\mathbf{u})\frac{d}{d\mathbf{u}}\left[\mathtt{J}_{\mathbf{v}}(\mathbf{u})^\dagger\right]\mathbf{b}. \tag{12}$$

If $\mathbf{v}^*(\mathbf{u})$ is differentiable, (12) is analytically tractable. (see [11].)

Ruhe and Wedin algorithms for Linear VarPro In [19], Ruhe and Wedin proposed three Newton-like algorithms each of which uses an approximation to the Hessian. The first algorithm, RW1, simply uses Gauss-Newton ($2\mathtt{J}^\top\mathtt{J}$). The second algorithm, RW2, approximates $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u}$ in the Jacobian such that the approximated Gauss-Newton matrix is orthogonal to the column space of $\mathtt{J}_{\mathbf{v}}(\mathbf{u})$. Finally, RW3 assumes independence of the two variables by setting $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u} = 0$, leading to alternation.

Although Ruhe and Wedin did not associate any trust region strategy with the above algorithms, we can easily incorporate this by using LM.

2.3 Nonlinear Variable Projection (Nonlinear VarPro)

The approach in §2.2 can be applied only to separable nonlinear least-squares, where the objective is bivariate and linear in at least one of two variables. Strelow [20] extended this to apply to nonseparable problems, which can be expressed as

$$\min_{\mathbf{u},\mathbf{v}} \|\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v})\|_2^2 = \min_{\mathbf{u},\mathbf{v}} \|\mathbf{f}(\mathbf{u},\mathbf{v}) - \mathbf{b}\|_2^2. \tag{13}$$

Similar to §2.2, we wish to find $\mathbf{v}^*(\mathbf{u}) := \arg\min_{\mathbf{v}} \|\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v})\|_2^2$ and solve

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

In this case, $\mathbf{v}^*(\mathbf{u})$ may not have a closed form solution as the residual vector is nonlinear in both $\mathbf{u}$ and $\mathbf{v}$. Instead, we apply a second-order iterative solver (e.g. LM) to approximately solve $\arg\min_{\mathbf{v}} \|\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v})\|_2^2$ and store the final solution in $\hat{\mathbf{v}}_0^*$.

Now, assuming that $\hat{\mathbf{v}}_0^*$ has converged, we define $\hat{\mathbf{v}}^*(\mathbf{u})$ as the quantity obtained by performing one additional Gauss-Newton iteration over $\mathbf{v}$ from $(\mathbf{u}, \hat{\mathbf{v}}_0^*)$. i.e.

$$\hat{\mathbf{v}}^*(\mathbf{u}) := \hat{\mathbf{v}}_0^* + \underbrace{\arg\min_{\boldsymbol{\Delta}\mathbf{v}} \|\boldsymbol{\varepsilon}(\mathbf{u},\hat{\mathbf{v}}_0^*) + \mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)\boldsymbol{\Delta}\mathbf{v}\|_2^2}_{\text{Additional Gauss-Newton step}} = \hat{\mathbf{v}}_0^* - \mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)^{\dagger}\boldsymbol{\varepsilon}(\mathbf{u},\hat{\mathbf{v}}_0^*). \tag{15}$$

Above expression implicitly assumes that $\boldsymbol{\varepsilon}(\mathbf{u},\mathbf{v})$ is locally linear in $\mathbf{v}$ about $\boldsymbol{\varepsilon}(\mathbf{u},\hat{\mathbf{v}}_0^*)$. This approximation allows us to estimate $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u}$ by computing $d\hat{\mathbf{v}}^*(\mathbf{u})/d\mathbf{u}$:

$$\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} \approx \frac{d\hat{\mathbf{v}}^*(\mathbf{u})}{d\mathbf{u}} = -\frac{\partial}{\partial \mathbf{u}}[\mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)^{\dagger}\boldsymbol{\varepsilon}(\mathbf{u},\hat{\mathbf{v}}_0^*)]. \tag{16}$$

Combining the results of (10) and (16) yields

$$\begin{aligned}\tilde{\mathtt{J}}^*(\mathbf{u}) &:= \mathtt{J}_{\mathbf{u}}(\mathbf{u},\hat{\mathbf{v}}_0^*) - \mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)\frac{\partial}{\partial \mathbf{u}}\left[\mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)^{\dagger}\boldsymbol{\varepsilon}(\mathbf{u},\hat{\mathbf{v}}_0^*)\right] &(17)\\ &= \left(\mathtt{I} - \mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)\mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)^{\dagger}\right)\mathtt{J}_{\mathbf{u}}(\mathbf{u},\hat{\mathbf{v}}_0^*) \\ &\quad - \mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)\frac{\partial}{\partial \mathbf{u}}\left[\mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)^{\dagger}\right]\boldsymbol{\varepsilon}(\mathbf{u},\hat{\mathbf{v}}_0^*) &(18)\end{aligned}$$

where $\tilde{\mathtt{J}}^*(\mathbf{u})$ is the approximate Jacobian used by Nonlinear VarPro. This expression can be further simplified using the differentiation rule for matrix pseudo-inverses [6].

In summary, one iteration of Nonlinear VarPro amounts to solving one inner minimization over $\mathbf{v}$ given $\mathbf{u}$, which outputs $\hat{\mathbf{v}}_0^* \approx \mathbf{v}^*(\mathbf{u})$, followed by one outer minimization over $\mathbf{u}$, which is achieved by linearizing the residual vector in $\mathbf{v}$ about $\boldsymbol{\varepsilon}(\mathbf{u},\hat{\mathbf{v}}_0^*)$.

For separable problems, the residual vector is always linear in $\mathbf{v}$, and therefore (18) becomes the exact Jacobian.

3 Ruhe and Wedin algorithms for Nonlinear VarPro

We acquire the nonlinear extensions of the original Ruhe and Wedin algorithms [19] as follows: since original RW1 applies the Gauss-Newton algorithm on the reduced problem (8), we propose that Nonlinear RW1 employs the Gauss-Newton algorithm using the Jacobian derived in (18). (This is essentially the same as Strelow's General Wiberg [20]). Original RW2 projects the exact Jacobian of original RW1 to the left nullspace of $\mathtt{J}_{\mathbf{v}}(\mathbf{u})$, resulting in an approximated Gauss-Newton matrix. For Nonlinear RW2, we project the Jacobian of Nonlinear RW1 to the left nullspace of $\mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)$, which is equivalent to discarding the latter term of (18). Lastly, original RW3 assumes $\mathbf{u}$ and $\mathbf{v}$ to be independent. For the nonlinear case, this translates to $d\hat{\mathbf{v}}^*(\mathbf{u})/d\mathbf{u} = 0$, yielding $\mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}_0^*)$ as the approximate Jacobian of Nonlinear RW3.

Table 1. A list of approximate Jacobians used by our nonlinear extension of the Ruhe and Wedin algorithms. Nonlinear RW1 applies the Gauss-Newton (GN) algorithm on the reduced problem (14), Nonlinear RW2 makes an approximation to the Jacobian as described in §3 and Nonlinear RW3 makes a further approximation which turns it into alternation. $\hat{\mathbf{v}}_0^*$ is obtained using a second-order iterative solver (see §2.3).

Algorithm	Approximate Jacobian used
Nonlinear RW3 (ALS)	$\mathtt{J}_{RW3} := \mathtt{J}_{\mathbf{u}}(\mathbf{u}, \hat{\mathbf{v}}_0^*)$
Nonlinear RW2	$\mathtt{J}_{RW2} := \mathtt{J}_{RW3} - \mathtt{J}_{\mathbf{v}}(\mathbf{u}, \hat{\mathbf{v}}_0^*)\mathtt{J}_{\mathbf{v}}(\mathbf{u}, \hat{\mathbf{v}}_0^*)^\dagger \mathtt{J}_{\mathbf{u}}(\mathbf{u}, \hat{\mathbf{v}}_0^*)$
Nonlinear RW1 (GN)	$\mathtt{J}_{RW1} := \mathtt{J}_{RW2} - \mathtt{J}_{\mathbf{v}}(\mathbf{u}, \hat{\mathbf{v}}_0^*) \frac{\partial}{\partial \mathbf{u}} \left[\mathtt{J}_{\mathbf{v}}(\mathbf{u}, \hat{\mathbf{v}}_0^*)^\dagger\right] \boldsymbol{\varepsilon}(\mathbf{u}, \hat{\mathbf{v}}_0^*)$

3.1 The sparsity of the Hessian approximations

Okatani et al. [17] and Strelow [20] pointed out structural similarity between the Schur complement reduced system for Joint optimization and Linear VarPro. Analysing the exact differences between the two methods is a research question on its own, and therefore in this paper we just confirm numerically that the Hessian approximations of both RW1 and RW2 (linear and nonlinear) have the same sparsity pattern but are not equal to the Schur complement reduced system for Gauss-Newton based Joint optimization. Furthermore, we found (through code implementation) that the iteration complexity of our proposed nonlinear extensions is similar (same for RW2, higher for RW1) to standard bundle adjustment with embedded point iterations [12]. Hence, one LM iteration of both standard bundle adjustment and our method will take roughly similar amount of time.

4 A unified notation for uncalibrated camera models

In this section, we present a unified notation for uncalibrated (affine and projective) cameras which allows a modularized compilation of bundle adjustment algorithms.

Affine and projective cameras are widely-used uncalibrated camera models which can be expressed in the homogeneous or the inhomogeneous form. We can incorporate both models and forms into a single camera matrix by defining

$$\mathtt{P}_i := \mathtt{P}(\mathbf{p}_i, \mathbf{q}_i, s_i, \mu_i) := \begin{bmatrix} p_{i1} & p_{i2} & p_{i3} & p_{i4} \\ p_{i5} & p_{i6} & p_{i7} & p_{i8} \\ \mu_i q_{i1} & \mu_i q_{i2} & \mu_i q_{i3} & s_i \end{bmatrix} =: \begin{bmatrix} \mathbf{p}_{i1:}^\top \\ \mathbf{p}_{i2:}^\top \\ \left[\mu_i \mathbf{q}_i^\top \; s_i\right] \end{bmatrix} \tag{19}$$

where $\mathbf{p}_i = [\mathbf{p}_{i1:}^\top, \mathbf{p}_{i2:}^\top]^\top = [p_{i1}, \cdots, p_{i8}]^\top$ and $\mathbf{q}_i = [q_{i1}, q_{i2}, q_{i3}]^\top$ are the projective camera parameters for frame i, $\mu_i \in [0, 1]$ indicates the degree of "projectiveness" of frame i, and s_i is the scaling factor of the i-th camera.

Now each point is typically parametrized as

$$\tilde{\mathbf{x}}_j := \tilde{\mathbf{x}}(\mathbf{x}_j, t_j) := \left[\mathbf{x}_j^\top \; t_j\right] := \left[x_{j1} \; x_{j2} \; x_{j3} \; t_j\right]^\top \tag{20}$$

where $\mathbf{x}_j = \left[x_{j1}, x_{j2}, x_{j3}\right]^\top$ is the vector of unscaled inhomogeneous coordinates of point j and and $\tilde{\mathbf{x}}_j$ is the vector of homogeneous coordinates of point j. (19) and (20) lead to a unified projection function

$$\boldsymbol{\pi}_{ij} := \boldsymbol{\pi}(\mathtt{P}_i, \tilde{\mathbf{x}}_j) = \boldsymbol{\pi}(\mathtt{P}(\mathbf{p}_i, \mathbf{q}_i, s_i, \mu_i), \tilde{\mathbf{x}}(\mathbf{x}_j, t_j)) = \frac{1}{\mu_i \mathbf{q}_i^\top \mathbf{x}_j + s_i t_j} \begin{bmatrix} \mathbf{p}_{i1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i2:}^\top \tilde{\mathbf{x}}_j \end{bmatrix}. \quad (21)$$

We show in Table 2 that the affine and the projective models (in both homogeneous and inhomogeneous forms) are specific instances of the unified model described above.

Table 2. A summary of uncalibrated camera models using the unified notation. $\mathtt{H} \in \mathbb{R}^{4\times4}$ is an arbitrary invertible matrix, and $\mathtt{A} \in \mathbb{R}^{4\times4}$ is an arbitrary invertible matrix with the last row set to $[0, 0, 0, 1]$. $\alpha_i, \beta_j \in \mathbb{R}$ is an arbitrary scale factor.

Form	Variable	Affine model ($\mu_i = 0$)	Projective model ($\mu_i = 1$)
Homogeneous	Camera ($\mathtt{P}_i$)	$\mathtt{P}(\mathbf{p}_i, 0, s_i, 0)$	$\mathtt{P}(\mathbf{p}_i, \mathbf{q}_i, s_i, 1)$
	Point ($\tilde{\mathbf{x}}_j$)	$\tilde{\mathbf{x}}(\mathbf{x}_j, t_j)$	$\tilde{\mathbf{x}}(\mathbf{x}_j, t_j)$
	Inverse depth	$s_i t_j$	$\mathbf{q}_i^\top \mathbf{x}_j + s_i t_j$
	Model property	Nonlinear in $\mathtt{P}_i$ and $\tilde{\mathbf{x}}_j$	
	Gauge freedom	$\mathtt{P}_i\tilde{\mathbf{x}}_j = (\mathtt{P}_i\mathtt{H})(\mathtt{H}^{-1}\tilde{\mathbf{x}}_j)$	
	Scale freedom	$\boldsymbol{\pi}(\mathtt{P}_i, \tilde{\mathbf{x}}_j) = \boldsymbol{\pi}(\alpha_i\mathtt{P}_i, \beta_j\mathbf{x}_j)$	
Inhomogeneous	Camera ($\mathtt{P}_i$)	$\mathtt{P}(\mathbf{p}_i, 0, 1, 0)$	$\mathtt{P}(\mathbf{p}_i, \mathbf{q}_i, 1, 1)$
	Point ($\tilde{\mathbf{x}}_j$)	$\tilde{\mathbf{x}}(\mathbf{x}_j, 1)$	$\tilde{\mathbf{x}}(\mathbf{x}_j, 1))$
	Inverse depth	1	$\mathbf{q}_i^\top \mathbf{x}_j + 1$
	Model property	linear in $\mathtt{P}_i$ and $\tilde{\mathbf{x}}_j$	nonlinear in $\mathtt{P}_i$ and $\tilde{\mathbf{x}}_j$
	Gauge freedom	$\mathtt{P}_i\tilde{\mathbf{x}}_j = (\mathtt{P}_i\mathtt{A})(\mathtt{A}^{-1}\tilde{\mathbf{x}}_j)$	$\mathtt{P}_i\tilde{\mathbf{x}}_j = (\mathtt{P}_i\mathtt{H})(\mathtt{H}^{-1}\tilde{\mathbf{x}}_j)$
	Scale freedom	None	

5 Compilation of affine/projective bundle adjustment algorithms

In this section, we present the building blocks of our bundle adjustment algorithms for uncalibrated cameras which stem from §2, §3 and §4. To simplify notations, we stack the variables introduced in §4 across all cameras or points by omitting the corresponding subscript, e.g. $\mathbf{p} = [\mathbf{p}_1^\top, \cdots, \mathbf{p}_f^\top]^\top$ and $\mathbf{x} = [\mathbf{x}_1^\top, \cdots, \mathbf{x}_n^\top]^\top$, where f is the number of frames and n is the number of points in the dataset used. We also define $\tilde{\mathbf{p}}$ to be the collection of the camera parameters $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{s}$. (Note that $\boldsymbol{\mu}$ is not included.) We can now rewrite (1) as

$$\min_{\mathbf{p},\mathbf{q},\mathbf{s},\mathbf{x},\mathbf{t}} \|\boldsymbol{\varepsilon}(\mathbf{p}, \mathbf{q}, \mathbf{s}, \mathbf{x}, \mathbf{t}, \boldsymbol{\mu})\|_2^2. \quad (22)$$

In this paper, we assume that $\boldsymbol{\mu}$ (the projectiveness vector) is fixed during optimization. (Finding an optimal way to adjust $\boldsymbol{\mu}$ at each iteration is future work.) Our algorithms first eliminate points ($\tilde{\mathbf{x}}$), generating a reduced problem over camera poses ($\tilde{\mathbf{p}}$), but we could reverse the order to eliminate poses first as described in §6.1 of [23].

5.1 Required derivatives

We only need three types of derivatives to implement all the algorithms mentioned in §2 irrespective of the camera model used.

The first two derivatives are the Jacobian with respect to camera poses ($\mathrm{J}_{\tilde{\mathbf{p}}}$) and the Jacobian with respect to feature points ($\mathrm{J}_{\tilde{\mathbf{x}}}$) which are the first order derivatives of the original objective (1). These Jacobians are used by both Joint optimization and VarPro but are evaluated at different points in the parameter space — at each iteration, Joint optimization evaluates the Jacobians at $(\tilde{\mathbf{p}}, \tilde{\mathbf{x}})$ whereas Linear and Nonlinear VarPro evaluate them at $(\tilde{\mathbf{p}}, \tilde{\mathbf{x}}^*(\tilde{\mathbf{p}}))$, where $\tilde{\mathbf{x}}^*(\tilde{\mathbf{p}})$ denotes a set of feature points which locally minimizes (1) given the camera parameters $\tilde{\mathbf{p}}$.

The third derivative, which involves a second-order derivative of the objective, is only required by Linear and Nonlinear RW1.

Table 3. A list of derivatives required for implementing affine and projective bundle adjustment algorithms based on the methods illustrated in §2. The camera parameters ($\tilde{\mathbf{p}}$) consist of $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{s}$, and the point parameters ($\tilde{\mathbf{x}}$) consist of $\mathbf{x}$ and $\mathbf{t}$. Note that the effective column size of these quantities will vary depending on the parameterization of the camera model used. The Jacobians are the first-order derivatives of $\boldsymbol{\varepsilon}(\mathbf{p}, \mathbf{q}, \mathbf{s}, \mathbf{x}, \mathbf{t}, \boldsymbol{\mu})$ in (22).

Required derivatives	Affine		Projective	
	Hom.	Inhom.	Hom.	Inhom.
Jacobian w.r.t. cameras ($\mathrm{J}_{\tilde{\mathbf{p}}}$)	$\left[\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{p}} \frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{s}}\right]$	$\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{p}}$	$\left[\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{p}} \frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{q}} \frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{s}}\right]$	$\left[\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{p}} \frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{q}}\right]$
Jacobian w.r.t. points ($\mathrm{J}_{\tilde{\mathbf{x}}}$)	$\left[\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{x}} \frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{t}}\right]$	$\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{x}}$	$\left[\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{x}} \frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{t}}\right]$	$\frac{\partial \boldsymbol{\varepsilon}}{\partial \mathbf{x}}$
$\frac{\partial [\mathrm{J}_{\tilde{\mathbf{x}}}^\dagger] \boldsymbol{\varepsilon}}{\partial \tilde{\mathbf{p}}}$ (required by RW1)	$\left[\frac{\partial [\mathrm{J}_{\tilde{\mathbf{x}}}^\dagger]}{\partial \mathbf{p}} \frac{\partial [\mathrm{J}_{\tilde{\mathbf{x}}}^\dagger]}{\partial \mathbf{s}}\right] \boldsymbol{\varepsilon}$	$\frac{\partial [\mathrm{J}_{\tilde{\mathbf{x}}}^\dagger] \boldsymbol{\varepsilon}}{\partial \mathbf{p}}$	$\left[\frac{\partial [\mathrm{J}_{\tilde{\mathbf{x}}}^\dagger]}{\partial \mathbf{p}} \frac{\partial [\mathrm{J}_{\tilde{\mathbf{x}}}^\dagger]}{\partial \mathbf{q}} \frac{\partial [\mathrm{J}_{\tilde{\mathbf{x}}}^\dagger]}{\partial \mathbf{s}}\right] \boldsymbol{\varepsilon}$	$\left[\frac{\partial [\mathrm{J}_{\tilde{\mathbf{x}}}^\dagger]}{\partial \mathbf{p}} \frac{\partial [\mathrm{J}_{\tilde{\mathbf{x}}}^\dagger]}{\partial \mathbf{q}}\right] \boldsymbol{\varepsilon}$

5.2 Constraining local scale freedoms in homogeneous camera models

Homogeneous camera models have local scale freedoms for each camera and point (see Table 2). We need to constrain these scales appropriately for the second-order update to be numerically stable — manually fixing an entry in each camera and point (as in the inhomogeneous coordinate system) may lead to numerical instability if some points or cameras are located in radical positions.

To do this, we apply a Riemannian manifold optimization framework [1, 14]. The intuition behind this is that scaling each point and each camera arbitrarily does not change the objective, and therefore, each point and each camera can be viewed as lying on the Grassmann manifold (which is a subset of the Riemannian manifold).

In essence, optimization on the Grassmann manifold can be achieved [14] by projecting each Jacobian to its tangent space, computing the second-order update of parameters on the tangent space then retracting back to the manifold by normalizing each

camera and/or point. This is numerically stable since the parameters are always updated orthogonal to the current solution. Details of our implementation can be found in [11].

5.3 Constraining gauge freedoms for the VarPro-based algorithms

Unlike scale freedoms, gauge freedoms are present in all camera models listed in §4.

Our VarPro-based algorithms eliminate points such that the matrix of whole camera parameters lie on the Grassmann manifold. This means that any set of cameras which share the same column space as the current set does not change the objective. With the inhomogeneous affine camera model, the matrix of whole cameras lie on a more structured variant of the Grassmann manifold as the scales are fixed to 1.

Since homogeneous camera models require both scale and gauge freedoms to be removed simultaneously (and the Jacobians are already projected to get rid of the scale freedoms), we incorporate a technique introduced in [17] to penalize the matrix of whole cameras updating along the column space of the current matrix, and this constrains all 16 gauge freedoms. (This approach can be viewed [10] as a relaxed form of the manifold optimization framework described in §5.2.) With the inhomogeneous affine model, we manipulate this technique to prevent from overconstraining the problem, and this removes 9 out of 12 gauge freedoms. More details are included in [11].

We have not implemented a gauge-constraining technique for Joint optimization but [16] could be applied.

5.4 Remarks

Combining all the aforementioned techniques yield 16 algorithms which are listed in [11]. We use 4 of them (see Table 4) to synthesize two-stage meta algorithms in §6.

As mentioned in §2.3, Nonlinear VarPro requires iterative inner minimization over points given cameras at each iteration. Our algorithms initialize points from the closest algebraic solution obtained using the Direct Linear Transformation (DLT) method [9].

Table 4. A list of affine and projective bundle adjustment algorithms used for our two-stage meta-algorithms in §6. We compile these algorithms using the building blocks from §5.

ID	Camera	Form	Strategy	Algorithm	Constrained gauge
AHRW2P	Affine	Homogeneous	Nonlinear VarPro	RW2	16 / 16
AIRW2P	Affine	Inhomogeneous	Linear VarPro	RW2	9 / 12
PHRW1P	Projective	Homogeneous	Nonlinear VarPro	RW1	16 / 16
PHJP	Projective	Homogeneous	Joint optimization	LM	None

6 Two-stage meta-algorithms for projective bundle adjustment

Initially, we attempted to use the projective bundle adjustment algorithms compiled in §5 directly on the datasets listed in Table 6 and Table 7. However, our preliminary

investigation showed that none of these work out of the box as the Linear VarPro-based algorithms do for the inhomogeneous affine case.

To resolve this, we propose the following strategy: perform affine bundle adjustment first and then use the outputs to initialize projective bundle adjustment. This is inspired by the fact that some projective algorithms such as projective matrix factorization [9, 21] and trilinear projective bundle adjustment [20] initialize all camera depths to 1, which is equivalent to employing the affine camera model. The key difference between our strategy and the aforementioned methods is that our approach enforces the affine model throughout the first stage whereas other methods can switch to the projective model straight after initialization. Since our strategy essentially places a prior on the affine model, it is important to check how this performs on strong perspective scenes.

For the first (affine) stage, we choose AIRW2P (Affine Inhomogeneous RW2 with manifold Projection) and AHRW2P (Affine Homogeneous RW2 with manifold Projection). We opt for the VarPro-based algorithms, which have large convergence basins for the affine case. We drop the RW1 series as they perform substantially slower than the RW2 series with comparable success rates. (Similar phenomenon is reported in [8, 10].)

For the second (projective) stage, we choose PHRW1P (Projective Homogeneous RW1 with manifold Projection) and PHJP (Projective Homogeneous Joint optimization with manifold Projection). We drop PHRW2P after observing its poor performance on some of the datasets used. None of the inhomogeneous projective algorithms are selected due to numerical stability issues (see §5.2).

Table 5. A list of two-stage meta-algorithms used in our experiments.

ID	First-stage (affine) algorithm	Second-stage (projective) algorithm
TSMA1	AHRW2P	PHRW1P
TSMA2	AIRW2P	PHRW1P
TSMA3	AHRW2P	PHJP
TSMA4	AIRW2P	PHJP

7 Experiments

All experiments were carried out on a workstation with 2.2 GHz Intel Xeon E5-2660 processor and 32 GB 1600 MHz DDR3 memory. We used MATLAB R2015b in single-threaded mode to run all the experiments.

We tested on various small synthetic (Table 6) and real SfM datasets (Table 7) derived from circular motion (Din, Dio, Di2, Hou), non-circular motion (Btb), forward movement (Cor, R47, Sth, Wil) and small number of frames (Lib, Me1, Me2, Wad).

On each dataset, we ran all four two-stage meta-algorithms listed in Table 5 for 100 runs. On each run, initial camera poses and points were drawn from $\mathcal{N}(\mathbf{0}, \mathtt{I})$. The first stage of each meta-algorithm minimized the affine version of (1), and the second stage minimized the projective version of the same problem. We set the maximum number of iterations in each stage to 1000 and the function value tolerance to 10^{-9}.

Table 6. Synthetic tracks of 319 points randomly generated on a sphere of radius 10.0 viewed from 36 cameras. The cameras are equidistantly positioned and form a ring of radius d, looking down the sphere at 60° from the vertical axis. We employ structured visibility patterns with high missing rates to depict real tracks with occlusions and tracking failures. $\mathcal{N}(\mathbf{0}, \mathtt{I})$ noise is added.

Dataset	d	Loop closed	Missing (%)	Best affine cost	Best projective cost
S30L	30.0	Yes	77.56	2.993821	0.861392
S30	30.0	No	76.92	2.947244	0.842539
S20L	20.0	Yes	77.61	7.261506	0.862520
S20	20.0	No	76.92	7.157783	0.842511
S13L	13.0	Yes	77.61	25.100919	0.863871
S13	13.0	No	76.92	24.831476	0.844125
S12L	12.0	Yes	77.61	34.871149	0.863867
S12	12.0	No	76.92	34.730023	0.844817
S11L	11.0	Yes	77.61	55.782547	0.863274
S11	11.0	No	76.92	56.946272	0.845271
S10.5L	10.5	Yes	77.61	80.700734	0.862545
S10.5	10.5	No	76.92	85.970046	0.844771

We then compared how many fractions of runs each meta-algorithm converged to the best observed minimum on each dataset, defining this quantity as the success rate. The success rates of different meta-algorithms are compared in Fig. 2a and Fig. 2b.

Throughout this paper, we report the normalized cost values which can be computed as follows: $\sqrt{\text{Equation (1)} \;/\; (2 \times \text{Total number of visible frames over all cameras})}$.

8 Discussions

Fig. 2a and Fig. 2b show that TSMA1 and TSMA2 return global optimum in a large fraction of runs on most datasets. Considering that each run is initialized from arbitrary cameras and points, we believe that these are novel and valuable results.

On the synthetic datasets (Fig. 2a), all our meta-algorithms yield high success rates (74–100%) until the ground truth cameras are moved radically close to the sphere (e.g. S10.5/L). As discussed in §6, this is somewhat expected since our strategy is inevitably biased towards affine reconstruction. However, one should bear in mind that these are extreme cases where the cameras are located only 0.5 unit away from the surface of the sphere of radius 10.0, and our strategy still succeeds with high probability on strong perspective datasets such as S11/L S12/L and S13/L. The presence of loop closure does not seem to influence success rates massively.

On the real sequences (Fig. 2b), each of TSMA1 and TSMA2 achieves 88–100% on all datasets but one (Lib for TSMA1 and Cor for TSMA2). This demonstrates that these methods work well in practice as they provide consistent performances across different kinds of camera motions.

Regarding the first (affine) stage algorithms, we do not observe a clear boost in success rates from employing AHRW2P instead of AIRW2P. (We only observe this on the Cor dataset (see Fig. 2b), which comprises forward camera movements.) This is

Table 7. Real datasets used for the experiments. f denotes the number of frames and n denotes the number of feature points. *Di2 was generated by projecting real points from synthetic camera poses made deliberately close to the 3D structure thereby inducing strong perspective effects.

ID	Dataset	f	n	Missing (%)	Best affine	Best projective
Btb	Blue teddy bear (trimmed) (Ponce)	196	827	80.71	0.530633	0.489283
Cor	Corridor (VGG)	11	737	50.23	2.237213	0.272462
Din	Dinosaur (trimmed) [5]	36	319	76.92	1.270153	1.114493
Dio	Dinosaur (VGG)	36	4983	90.84	1.217574	1.166165
Di2*	Dinosaur (trimmed) closer cameras	36	319	76.92	9.380473	0.838414
Hou	House (VGG)	10	672	57.65	2.750877	0.441660
Lib	Oxford University Library (VGG)	3	667	29.24	4.180297	0.172830
Me1	Merton College 1 (VGG)	3	717	22.13	3.176176	0.118450
Me2	Merton College 2 (VGG)	3	475	21.61	3.995869	0.158851
R47	Road scene point tracks #47	11	150	47.09	4.402777	3.344768
Sth	Stockholm Guildhall (trimmed) [18]	43	1000	18.01	8.833195	5.619975
Wad	Wadham College (VGG)	5	1331	54.64	3.424812	0.135711
Wil	Wilshire (Ponce)	190	411	60.73	2.703663	0.423892

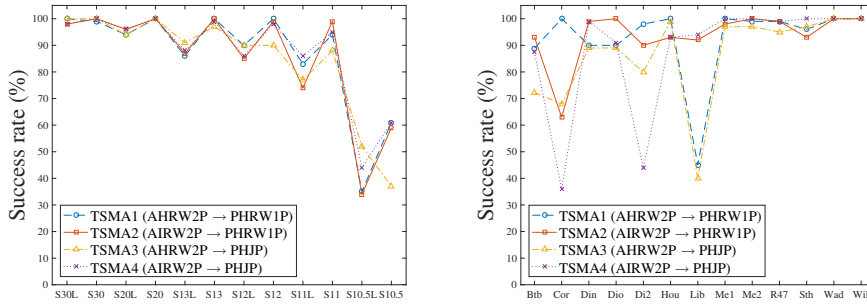
against our hypothesis that AHRW2P, which is a numerically-stable reformulation of AIRW2P, should perform better on strong perspective sequences. The results imply that the potential numerical instability caused by the use of inhomogeneous coordinates is not a major issue in the affine case. (It is still an issue for the projective model.)

In addition to the main experiments, we investigated to see if our meta-algorithms could serve as an initializer for the full bundle adjustment process. We ran a projective bundle adjustment algorithm, namely PHRW1P, on the full dinosaur dataset (Dio) with the initial camera values set to those of the global optimum of the trimmed dataset (Din). This allowed PHRW1P to reach the global optimum of the full sequence within 10 iterations. Based on this observation, we believe that our meta-algorithms could be applied to a segment of large datasets to trigger incremental or full bundle adjustment.

We also tried incrementing μ (the projectiveness parameter) gradually to make the affine-projective transition smoother, but this strategy performed comparable to projective bundle adjustment without affine initialization. Implementing a fully unified algorithm still remains a challenge.

9 Conclusions

In this work we analysed if the Variable Projection (VarPro) method, which is highly successful in finding global minima in affine factorization problems without careful initialization, is equally effective in the projective scenario. Unfortunately, the answer is that the success rate of VarPro-based algorithms cannot be directly replicated in the projective setting. Thus, we proposed and evaluated several meta-algorithms to overcome this shortcoming, and each of the winning methods (TSMA1 and TSMA2) obtained success rates between 88 and 100% on all real datasets but one. Experimentally it turns out that using an affine factorization based on VarPro to warm-start projective bundle adjustment is essential to boost the success rate.



(a) Synthetic datasets (b) Real datasets

Fig. 2. The figures show the success rates of each meta-algorithm on each dataset. (A run is counted as *successful* if and only if it reaches the best known optimum of the dataset used.) We conclude that TSMA1 and TSMA2 are winners by narrow margins.

We demonstrated that the convergence basin can be greatly enhanced using the right combination of methods. By unifying affine and projective factorization problems we also derived numerically better conditioned formulations to solve these instances.

Future work includes the followings: addressing outliers in the measurements and therefore robustness in the cost function (e.g. by incorporating robust kernel reformulation [25]) and to operate in metric instead of projective space by restricting the unknowns to the respective Lie group. A highly ambitious goal is to solve large datasets as introduced in [3] via an initialization-free approach.

Acknowledgement

The work was supported by Microsoft, Toshiba Research Europe and JNE Systech.

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press (2008)
2. Agarwal, S., Mierle, K., Others: Ceres solver. http://ceres-solver.org (2014)
3. Agarwal, S., Snavely, N., Seitz, S.M., Szeliski, R.: Bundle adjustment in the large. In: Proceedings of the 11th European Conference on Computer Vision (ECCV): Part II. pp. 29–42 (2010)
4. Boumal, N., Absil, P.A.: RTRMC: A Riemannian trust-region method for low-rank matrix completion. In: Advances in Neural Information Processing Systems 24 (NIPS 2011), pp. 406–414 (2011)
5. Buchanan, A.M., Fitzgibbon, A.W.: Damped Newton algorithms for matrix factorization with missing data. In: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). vol. 2, pp. 316–322 (2005)

6. Golub, G.H., Pereyra, V.: The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. SIAM Journal on Numerical Analysis (SINUM) 10(2), 413–432 (1973)
7. Golub, G.H., Pereyra, V.: Separable nonlinear least squares: the variable projection method and its applications. In: Proceedings of Inverse Problems. pp. 1–26 (2002)
8. Gotardo, P.F., Martinez, A.M.: Computing smooth time trajectories for camera and deformable shape in structure from motion with occlusion. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 33(10), 2051–2065 (Oct 2011)
9. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, second edn. (2004)
10. Hong, J.H., Fitzgibbon, A.W.: Secrets of matrix factorization: Approximations, numerics, manifold optimization and random restarts. In: Proceedings of the 2015 IEEE Internatonal Conference on Computer Vision (ICCV). pp. 4130–4138 (2015)
11. Hong, J.H., Zach, C., Fitzgibbon, A.W., Cipolla, R.: Projective bundle adjustment from arbitrary initialization using the variable projection method: Supplementary document (2016), https://github.com/jhh37/projective-ba
12. Jeong, Y., Nister, D., Steedly, D., Szeliski, R., Kweon, I.S.: Pushing the envelope of modern methods for bundle adjustment. In: Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1474–1481 (2010)
13. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Quarterly of Applied Mathmatics 2(2), 164–168 (1944)
14. Manton, J.H., Mahony, R., Hua, Y.: The geometry of weighted low-rank approximations. IEEE Transactions on Signal Processing 51(2), 500–514 (Feb 2003)
15. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial and Applied Mathematics 11(2), 431–441 (1963)
16. Mishra, B., Meyer, G., Bach, F., Sepulchre, R.: Low-rank optimization with trace norm penalty. SIAM Journal on Optimization (SIOPT) 23(4), 2124–2149 (2013)
17. Okatani, T., Yoshida, T., Deguchi, K.: Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms. In: Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV). pp. 842–849 (2011)
18. Olsson, C., Enqvist, O.: Stable structure from motion for unordered image collections. In: Proceedings of 17th Scandinavian Conference on Image Analysis (SCIA). pp. 524–535 (2011)
19. Ruhe, A., Wedin, P.Å.: Algorithms for separable nonlinear least squares problems. SIAM Review (SIREV) 22(3), 318–337 (1980)
20. Strelow, D.: General and nested Wiberg minimization: L2 and maximum likelihood. In: Proceedings of the 12th European Conference on Computer Vision (ECCV), pp. 195–207 (2012)
21. Sturm, P., Triggs, B.: A factorization based algorithm for multi-image projective structure and motion. In: Proceedings of the 4th European Conference on Computer Vision (ECCV), pp. 709–720 (1996)
22. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. International Journal of Computer Vision (IJCV) 9(2), 137–154 (1992)
23. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment - A modern synthesis. In: Proceedings of the International Workshop on Vision Algorithms: Theory and Practice. pp. 298–372. ICCV '99 (2000)
24. Wiberg, T.: Computation of principal components when data are missing. In: Proceedings of the 2nd Symposium of Computational Statistics. pp. 229–326 (1976)
25. Zach, C.: Robust bundle adjustment revisited. In: Proceedings of the 13th European Conference on Computer Vision (ECCV). pp. 772–787 (2014)