Widening the basin of convergence for the bundle adjustment type of problems in computer vision

Je Hyeong Hong

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Christ's College

5 March 2018

Widening the basin of convergence for the bundle adjustment type of problems in computer vision

Je Hyeong Hong

Abstract

Bundle adjustment is the process of simultaneously optimizing camera poses and 3D structure given image point tracks. In structure-from-motion, it is typically used as the final refinement step due to the nonlinearity of the problem, meaning that it requires sufficiently good initialization. Contrary to this belief, recent literature showed that useful solutions can be obtained even from arbitrary initialization for fixed-rank matrix factorization problems, including bundle adjustment with affine cameras. This property of wide convergence basin of high quality optima is desirable for any nonlinear optimization algorithm since obtaining good initial values can often be non-trivial. The aim of this thesis is to find the key factor behind the success of these recent matrix factorization algorithms and explore the potential applicability of the findings to bundle adjustment, which is closely related to matrix factorization.

The thesis begins by unifying a handful of matrix factorization algorithms and comparing similarities and differences between them. The theoretical analysis shows that the set of successful algorithms actually stems from the same root of the optimization method called variable projection (VarPro). The investigation then extends to address why VarPro outperforms the joint optimization technique, which is widely used in computer vision. This algorithmic comparison of these methods yields a larger unification, leading to a conclusion that VarPro benefits from an unequal trust region assumption between two matrix factors.

The thesis then explores ways to incorporate VarPro to bundle adjustment problems using projective and perspective cameras. Unfortunately, the added nonlinearity causes a substantial decrease in the convergence basin of VarPro, and therefore a bootstrapping strategy is proposed to bypass this issue. Experimental results show that it is possible to yield feasible metric reconstructions and pose estimations from arbitrary initialization given relatively clean point tracks, taking one step towards initialization-free structure-from-motion.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

In addition, this disseration integrates and extends the candidate's first-authored conference papers referenced below:

1. Secrets of Matrix Factorization: Approximations, Numerics, Manifold Optimization and Random Restarts. **J.H. Hong** and A.W. Fitzgibbon. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4130–4138, 2015.

2. Projective Bundle Adjustment from Arbitrary Initialization using the Variable Projection Method. **J.H. Hong**, C. Zach, A.W. Fitzgibbon and R. Cipolla. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, pages 477–493, 2016.

3. Revisiting the Variable Projection Method for Separable Nonlinear Least Squares Problems. **J.H. Hong**, C. Zach and A.W. Fitzgibbon. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5939–5947, 2017.

4. pOSE: Pseudo Object Space Error for Initialization-Free Bundle Adjustment. **J.H. Hong** and C. Zach. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, to appear, 2018.

Je Hyeong Hong
5 March 2018

Acknowledgements

I would like to sincerely thank three great researchers/supervisors, in the order of alphabetical surnames, without whom this thesis would not have happened:

- **Prof. Roberto Cipolla** at University of Cambridge for setting up an excellent research environment, introducing great researchers (Andrew and Christopher) and providing funding opportunities for my PhD,
- **Dr Andrew Fitzgibbon** at Microsoft for teaching me the basics of optimization, providing essential tools for research, including but not limited to paper writing and MATLAB coding, and sharing great research insights, and
- **Dr Christopher Zach** at Toshiba Research Europe for providing and sharing lots of clever ideas and insights in structure-from-motion and optimization and guiding the overall story of this work.

It has been a real pleasure to have opportunities to work under their guidance.

Additionally, I would like to acknowledge Dr Alexander Fix at Oculus Research for helping me to improve my C++ skills and extend my knowledge of optimization. Also, thanks to Dr Helena Browne at the University of Cambridge for providing support as college tutor.

PhD fundings from **Microsoft** and **Toshiba Research Europe** are gratefully acknowledged. Additionally, conference travels were generously supported by Christ's College, the British Machine Vision Association (BMVA), the Cambridge University Engineering Department and the Cambridge Philosophical Society.

Lastly, acknowledgements to my family and friends. To my loving parents, who have been great supporters of me in both happy and difficult times. Same goes to my girlfriend Ranjoo Choi for her enormous moral support, and I hope you are with me for the rest of my life. Thanks to my friends from the Cambridge University Korean Society, namely Keehoon Kang, who has made my Cambridge life memorable and enjoyable. Finally, it has been great to meet and discuss with smart colleagues in the Computer Vision group, and I wish their very best in their careers.

Table of contents

List of figures

List of tables

Table 1 Common symbols used in this thesis. Some symbols may be redefined locally.

	Symbol	Dim.	Meaning
Linear algebra	$\mathbb{R}$		Real space
	$\mathbb{S}$		Symmetric real space
	$O(n)$		Orthogonal group of dimension n
	$SO(n)$		Special orthogonal group of dimension n
	$SE(n)$		Special Euclidean group of dimension n
	$x_{i,j}$	$\mathbb{R}$	(i,j)-th element of some matrix $\mathtt{X} \in \mathbb{R}^{m \times n}$
	x_i	$\mathbb{R}$	i-th element of some vector $\mathbf{x} \in \mathbb{R}^n$
	$\mathbf{x}_j$	$\mathbb{R}^m$	j-th column vector of some matrix $\mathtt{X} \in \mathbb{R}^{m \times n}$
	$[\mathbf{x}]_\times$	$\mathbb{R}^{3 \times 3}$	A skew-symmetric matrix representing $\mathbf{x} \times$. $([\mathbf{a}]_\times \mathbf{b} = \mathbf{a} \times \mathbf{b})$
Optimization	f	$\mathbb{R}$	Cost function
	$\mathbf{g}$	$\mathbb{R}^n$	Gradient
	$\mathtt{H}$	$\mathbb{S}^n$	Hessian
	ε	$\mathbb{R}^m$	Error residual such that $f := \frac{1}{2}\|\varepsilon(\mathbf{x})\|_2^2$
	$\mathtt{J}$	$\mathbb{R}^{m \times n}$	Jacobian of $\varepsilon(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^n$
	λ	$\mathbb{R}$	Damping factor
3D vision	$\tilde{\mathbf{u}}$	$\mathbb{R}^{n+1}$	Homogeneous representation of some vector $\mathbf{u} \in \mathbb{R}^n$
	$\hat{\mathbf{u}}$	$\mathbb{R}^n$	Normalized $\mathbf{u} \in \mathbb{R}^n$
	$\mathbf{x}$	$\mathbb{R}^3$	3D point
	$\mathbf{m}$	$\mathbb{R}^2$	Projection on the image plane
	$\mathtt{P}$	$\mathbb{R}^{3 \times 4}$	Camera projection matrix
	$\mathtt{K}$	$\mathbb{R}^{3 \times 3}$	Calibration matrix
	$\mathtt{R}$	$\mathbb{R}^{3 \times 3}$	Rotation matrix $\in SO(3)$
	$\mathbf{t}$	$\mathbb{R}^3$	Translation
	$\mathbf{c}$	$\mathbb{R}^3$	Camera centre
	$\mathtt{E}$	$\mathbb{R}^{3 \times 3}$	Essential matrix
	$\mathtt{F}$	$\mathbb{R}^{3 \times 3}$	Fundamental matrix
	$\mathtt{H}$	$\mathbb{R}^{3 \times 3}$	Homography matrix
Matrix factorization	$\mathtt{M}$	$\mathbb{R}^{m \times n}$	Measurement matrix
	$\mathtt{W}$	$\mathbb{R}^{m \times n}$	Weight matrix
	$\mathtt{U}$	$\mathbb{R}^{m \times r}$	First matrix factor
	$\mathtt{V}$	$\mathbb{R}^{n \times r}$	Second matrix factor
	$\mathbf{u}$	$\mathbb{R}^{mr}$	$\mathrm{vec}(\mathtt{U})$
	$\mathbf{v}$	$\mathbb{R}^{nr}$	$\mathrm{vec}(\mathtt{V}^\top)$
	$\tilde{\mathtt{W}}$	$\mathbb{R}^{p \times mn}$	A truncated form of $\mathrm{diag}(\mathrm{vec}(\mathtt{W}))$
	$\tilde{\mathbf{m}}$	$\mathbb{R}^p$	$\tilde{\mathtt{W}}\,\mathrm{vec}(\mathtt{M})$
	$\tilde{\mathtt{U}}$	$\mathbb{R}^{p \times nr}$	$\tilde{\mathtt{W}}(\mathtt{I}_n \otimes \mathtt{U})$
	$\hat{\mathtt{V}}$	$\mathbb{R}^{p \times mr}$	$\tilde{\mathtt{W}}(\mathtt{V} \otimes \mathtt{I}_m)$
	$\mathtt{R}$	$\mathbb{R}^{m \times n}$	$\mathtt{W} \odot (\mathtt{U}\mathtt{V}^\top - \mathtt{M})$
	$\mathtt{Z}$	$\mathbb{R}^{mr \times nr}$	$(\mathtt{W} \odot \mathtt{R}) \otimes \mathtt{I}_r$

Chapter 1

Introduction

Obtaining an accurate 3D reconstruction and camera poses from a set of overlapping images is a fundamental problem in computer vision. This task is typically accomplished through a multistage pipeline such as structure-from-motion (SfM) and simultaneous localization and mapping (SLAM). Almost all SfM pipelines and several SLAM pipelines including PTAM [Klein and Murray, 2009] and ORB-SLAM [Mur-Artal et al., 2015] are feature-based, meaning they first extract sparse features from all or a subset of images, make feature correspondences across the images to form tracks of 2D features and finally solve simultaneously for the camera poses and underlying 3D structure which best explain the observations. This final step is widely known as bundle adjustment, which plays a crucial role in greatly improving the accuracy of reconstruction.

The name *bundle adjustment* originates from that this process can be viewed as *adjusting* a *bundle* of light rays, in such a way that each ray starts from its 3D feature point and converges at its camera (or optic) centre. A schematic diagram is illustrated in Fig. 1.1. If no noise is present, each ray perfectly intersects its 2D feature observation on the image plane of the camera involved. Under the presence of noise, however, this is almost always not the case, and consequently bundle adjustment requires solving an optimization problem to adjust these rays such that they produce estimates which are as close as possible to observed data according to some criterion.

Whilst there exists several ways to define the “proximity” of a ray to its measurement, a widely accepted metric, also known as the gold standard error [Hartley and Zisserman, 2004], considers the distance between a 2D observation and its corresponding ray projection on the image plane. Often, the optimal solution is the one which minimizes the sum of squared distances between the observations and the respective ray projections. In terms of equations, the solution can be expressed as

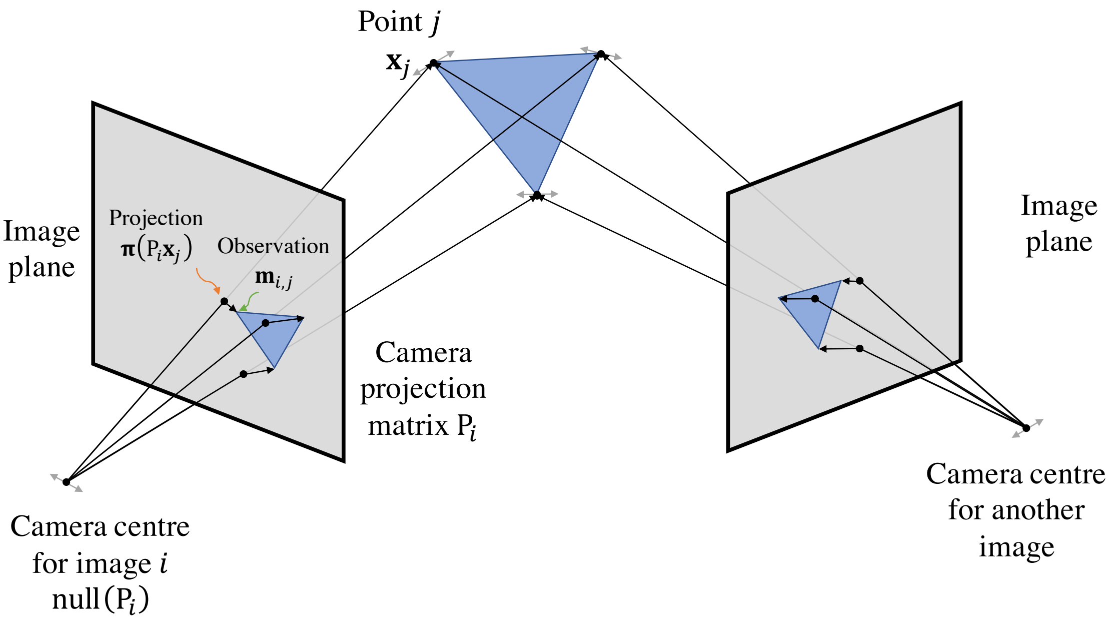

Point j
$\mathbf{x}_j$
Image plane
Projection
$\pi(\mathrm{P}_i\mathbf{x}_j)$
Observation
$\mathbf{m}_{i,j}$
Camera projection matrix P_i
Image plane
Camera centre for image i
null(P_i)
Camera centre for another image

Fig. 1.1 A schematic diagram of the bundle adjustment process. Both camera parameters $\mathtt{P}_i$ and point parameters $\mathbf{x}_j$ are simultaneously refined at each iteration to minimize the weighted sum of errors on the image planes.

$$\arg\min_{\{\mathtt{P}_i\},\{\tilde{\mathbf{x}}_j\}} \sum_{(i,j)\in\Omega} \rho\left(\|\pi(\mathtt{P}_i\tilde{\mathbf{x}}_j) - \mathbf{m}_{i,j}\|_2^2\right), \tag{1.1}$$

where $\mathtt{P}_i \in \mathbb{R}^{3\times 4}$ is the camera projection matrix for image i, $\tilde{\mathbf{x}}_j \in \mathbb{R}^4$ is the homogeneous representation of feature point j ($\mathbf{x}_j \in \mathbb{R}^3$), $\mathbf{m}_{i,j} \in \mathbb{R}^2$ is the 2D observation of feature point j in image i, $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ is the perspective division operator defined as $\pi([x,y,z]^\top) := [x/z, y/z]^\top$, Ω denotes a set of visible observations and $\rho : \mathbb{R} \to \mathbb{R}$ is an isotropic robust kernel to truncate the effect of outliers. If $\rho(s) = s$, then the solution can be interpreted as the maximum likelihood estimate (MLE) of camera and point parameters given data under isotropic Gaussian noise (see Section 2.2.4). Depending on the type of the camera model used, $\mathtt{P}_i$ for each i may be parameterized using a smaller number of parameters.

Solving (1.1) is known to be difficult mainly for the following reasons:

- **Nonlinearity** The problem is nonlinear due to the existence of the projection function $\pi(\cdot)$ mentioned above, which is is non-convex, the bilinear interaction between the camera and point parameters and the robust kernel $\rho(\cdot)$. If a calibrated camera model such as the pinhole camera model is used, then $\mathtt{P}_i$ has additional structural constraints, making the problem even more nonlinear. Since nonlinear least squares problems are solved using an iterative algorithm, good initial estimates are required to obtain a good local minimum, but there is no gold standard rule for obtaining the initial values.

- **Presence of outliers** There can be outliers in 2D point tracks due to feature mismatches, which are undetected at the track generation phase. This is more likely to occur when repetitive structures or reflective symmetry are present in the scene.

- **Gauge freedom** (1.1) does not have a unique solution since any invertible matrix $\mathtt{H} \in \mathbb{R}^{4\times 4}$ can be applied to $\mathtt{P}_i$ and $\tilde{\mathbf{x}}_j$ to produce the same objective value with $\mathtt{P}_i\mathtt{H}$ and $\mathtt{H}^{-1}\tilde{\mathbf{x}}_j$.

Due to the above reasons, bundle adjustment has typically been applied only at the end of a SfM pipeline, once good initial estimates of camera poses and 3D structure are obtained through a series of heuristics and algorithms.

In recent years, however, some surprising results were reported in the field of fixed-rank matrix factorization with missing data, which aims to estimate a sparse measurement matrix by a product of two fixed-rank matrices. Okatani et al. [2011] and Gotardo and Martinez [2011] showed that, on some small computer vision datasets, best experimentally observed minima can be obtained even from arbitrarily initialized matrices over 90% of runs, effectively widening the basin of convergence for matrix factorization. This rather astonishing success was achieved by employing the method called Wiberg [1976] or variable projection [Golub and Pereyra, 1973], equipped with the second-order nonlinear least squares optimization algorithm called Levenberg-Marquardt [Levenberg, 1944; Marquardt, 1963].

A natural question arising from this is whether the above results could be extended to bundle adjustment, which can be formulated as a rank-4 nonlinear matrix factorization problem with missing data, with the additional nonlinearity arising from the projection function $\pi(\cdot)$. Typically, well-known SfM pipelines such as VisualSfM [Wu, 2013] spend most processing time in incrementally registering camera views and performing bundle adjustment on a subset of total images in order to obtain good initialization for the final bundle adjustment process. However, such initialization is not only costly due to iterative running of bundle adjustment but also inevitably prioritizes on the initial pair of camera views, accumulating drift errors over the course of time and potentially failing to close loops. This motivates research towards the direction of pure global structure-from-motion, in which all views are considered equally and simultaneously. In this case, camera position errors are more evenly distributed thereby potentially providing a better, if not best, representation of the scene. In recent years, several global SfM pipelines [Moulon et al., 2013; Olsson and Enqvist, 2011; Sweeney et al., 2015] have been proposed but these still rely on custom averaging techniques to initialize rotations and translations of each camera, which can be sensitive to outlier correspondences. Hence, if it is possible to widen the basin of convergence for bundle adjustment by incorporating the success factor behind the variable projection (VarPro) method, then one could potentially get rid of the costly and biased initialization steps altogether, streamlining the state-of-the-art SfM pipeline without compromising its accuracy. This is the ultimate goal towards which this thesis strives but only partially achieves, largely due to the systematic generation of gross outliers from feature mismatches.

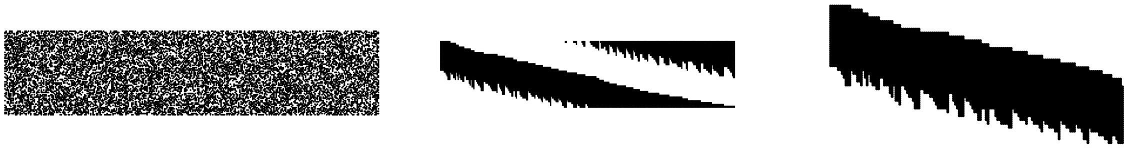
1.1 Thesis structure

To achieve the aforementioned goal, this dissertation first illustrates a range of essential background materials in Chapter 2 and reviews relevant literature. It then moves on to make a detailed analysis of some of the recent successful matrix factorization algorithms in order to unveil the secrets behind the widened basin of convergence. As a result of this analysis, Chapter 3 reveals that all of these successful algorithms can be unified and viewed as employing the method known as variable projection (VarPro) [Golub and Pereyra, 1973]. Chapter 4 then further unifies VarPro and other optimization approaches, intuitively illustrating why a standard joint optimization algorithm frequently fails on this type of problems and how VarPro can be scaled to support larger problem size.

In Chapter 5, it is investigated whether the findings from Chapter 3 and 4 can be applied to non-bilinear problems such as bundle adjustment with projective or pinhole cameras. As the empirical evidence implies that the benefits of VarPro is largely limited to bilinear problems, a two-stage strategy which uses affine optima (from arbitrary initialization) to warm start projective bundle adjustment is proposed. In Chapter 6, the affine stage is replaced by the pseudo object space error (pOSE) optimization, which yields solutions that better captures the projectiveness of scenes. It then provides a stratified strategy that can yield metric reconstructions from arbitrary initialization. The results show that this strategy can effectively enlarge the basins of convergence of accurate solutions, but like other global SfM pipelines, is only limited to cases when given sufficiently clean image point tracks. In the last chapter, conclusions and limitations of this work are addressed along with some potential avenues for future work.

1.2 What is a bundle adjustment type of problem?

In this work, a *bundle adjustment type* of problem is defined as a (potentially robustified) nonlinear least squares optimization problem which has the following properties:



(a) Random (Gaussian) (b) Banded, loop closed (c) Banded, no loop closure

Fig. 1.2 Types of missing data patterns encountered in matrix factorization and structure-from-motion (SfM). For SfM, each row represents a view and each column represent a feature point. Black means visible $(i, j) \in \Omega$ and white means missing. (a) is an ideal scenario of random sampling where some theoretical guarantees of convergence to a global optimum are proposed in the literature (e.g. Candès and Recht [2009]). Unfortunately, structured patterns like (b) and (c) are more frequently encountered in structure-from-motion due to occlusions.

1. **factorized bilinear interaction between two sets of variables:** (1.1) comprises a bilinear factorization between the first two rows of each camera matrix and each homogeneous feature point, which has innate gauge freedom and makes the problem rank-deficient.
2. **structured missing pattern:** missing data in structure-from-motion typically arises from occlusions and tracking failures, which form a structured pattern and therefore does not benefit from theoretical guarantees (e.g. Candès and Recht [2009]) of minimum required number of samples or convergence to a global optimum.
3. **presence of outliers:** structure-from-motion usually incurs significant numbers of outliers due to feature mismatches and incorrect point track generation.

By the above criteria, the rank-3 matrix factorization problem with a mean vector can also be viewed as a bundle adjustment type of problem since this arises when using the affine camera model [Okatani and Deguchi, 2007; Tomasi and Kanade, 1992].

1.3 Contributions and limitations

Some key novel contributions of this thesis are as follows:

+ **Unification of fixed-rank matrix factorization literature:** In Chapter 3, several matrix factorization algorithms published in computer vision and numerical analysis are unified, revealing that use of the variable projection (VarPro) method is key to success for this type of problem.

+ **Unification of bivariate least squares optimization strategies:** In Chapter 4, VarPro and the joint optimization (with the Schur complement trick) algorithms briefly reviewed in Section 2.1 are fully unified, showing the exact similarities and differences between them. This allows one to intuitively answer why joint optimization frequently fails on bilinear factorization problems. Also, contrary to some belief in the literature [Cabral et al., 2013; Chen, 2008], it is shown in Chapter 4 that variable projection (or Wiberg) has similar iteration complexity to joint optimization and therefore can be implemented efficiently. This enables current optimization libraries such as Ceres Solver [Agarwal et al., 2014] to correctly support VarPro with minor modifications.

+ **An initialization-free strategy for metric bundle adjustment given relatively clean point tracks:** In Chapter 6, a stratified strategy for initialization-free metric bundle adjustment is proposed for datasets with sufficiently clean point tracks. The experimental results show that the convergence basins of accurate solutions can be widened for small to medium-sized datasets comprising up to around 300 images.

On the other hand, the **major limitation** of this dissertation is that the aforementioned initialization-free strategy devised in Chapter 6 is still limited to the case where 2D point tracks are relatively clean. This means that two or three-view geometric verification using RANSAC (illustrated in Section 2.2.1) is still required as part of the global structure-from-motion pipeline, significantly reducing the runtime benefit of the proposed initialization-free strategy. Also, the proposed method has only been tested for small to medium-sized datasets, and its application to larger sequences may require additional steps such as first decomposing the dataset to smaller pieces with overlapping projections, optimizing each of them followed by stitching the overlapping camera locations back together. Other limitations are summarized in Chapter 7.

Chapter 2

Background

This chapter illustrates essential background materials, namely bivariate optimization methods, structure-from-motion pipelines and low-rank matrix-factorization, followed by relevant literature reviews.

2.1 Bivariate optimization strategies

This section summarizes some of the well-known optimization strategies for bivariate nonlinear least squares problems. The derivations specific to matrix factorization, separable nonlinear least squares and non-separable nonlinear least squares problems are provided in Chapter 3, Chapter 4 and Section 5.1 respectively.

Bivariate least squares is a type of problem where optimization variables can be partitioned into two blocks. The aim is to solve

$$\underset{\mathbf{u}\in\mathbb{R}^p,\ \mathbf{v}\in\mathbb{R}^q}{\arg\min}\ f(\mathbf{u},\mathbf{v}) := \underset{\mathbf{u}\in\mathbb{R}^p,\ \mathbf{v}\in\mathbb{R}^q}{\arg\min}\ \frac{1}{2}\|\varepsilon(\mathbf{u},\mathbf{v})\|_2^2. \tag{2.1}$$

Note that bivariate does not necessarily mean bilinear—e.g. projective bundle adjustment is bivariate in cameras and points but not bilinear.

2.1.1 Block coordinate descent (ALS)

Block coordinate descent (also known as alternation) implicitly assumes that $\mathbf{u}$ and $\mathbf{v}$ are independent, and updates each set of variables by taking the other as constant. This is illustrated in Algorithm 1, in which a set of initial values are required for $\mathbf{u}$ and $\mathbf{v}$.

For a bilinear problem, treating the two blocks of variables independently can yield a set of closed-form update equations which can be obtained from the first optimality condition.

Algorithm 1 Block coordinate-descent for bivariate least squares

```
**Inputs:** $\mathbf{u} \in \mathbb{R}^p$, $\mathbf{v} \in \mathbb{R}^q$
**repeat**
    $\mathbf{u} \leftarrow \arg\min_{\mathbf{u}} f(\mathbf{u},\ \mathbf{v})$
    $\mathbf{v} = \arg\min_{\mathbf{v}} f(\mathbf{u},\ \mathbf{v})$
**until** convergence
**Outputs:** $\mathbf{u} \in \mathbb{R}^p$, $\mathbf{v} \in \mathbb{R}^q$
```

Algorithm 2 Joint optimization for bivariate least squares

```
**Inputs:** $\mathbf{u} \in \mathbb{R}^p$, $\mathbf{v} \in \mathbb{R}^q$
$\mathbf{z} \leftarrow [\mathbf{u};\mathbf{v}]$
**repeat**
    **repeat**
        Compute $\Delta\mathbf{z}$ by using the linearized joint submodel around $(\mathbf{u},\mathbf{v})$.
        $\Delta\mathbf{u} \leftarrow [\Delta\mathbf{z}]_{1:p}$
        $\Delta\mathbf{v} \leftarrow [\Delta\mathbf{z}]_{(p+1):(p+q)}$
    **until** $f(\mathbf{u}+\Delta\mathbf{u},\mathbf{v}+\Delta\mathbf{v}) < f(\mathbf{u},\mathbf{v})$
    $\mathbf{z} \leftarrow \mathbf{z}+\Delta\mathbf{z}$
    $\mathbf{u} \leftarrow [\mathbf{z}]_{1:p}$
    $\mathbf{v} \leftarrow [\mathbf{z}]_{(p+1):(p+q)}$
**until** convergence
**Output:** $\mathbf{u} \in \mathbb{R}^p$, $\mathbf{v} \in \mathbb{R}^q$
```

Otherwise, either a first or second-order optimization algorithm would be used to iteratively solve for **u** given **v** and **v** given **u** respectively.

This approach is prone to slow convergence in cases where **u** and **v** are strongly coupled in the objective function.

2.1.2 Joint optimization

Joint optimization minimizes over $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{v} \in \mathbb{R}^q$ simultaneously by stacking the two parameters (e.g. $\mathbf{z} := [\mathbf{u};\mathbf{v}]$), and using a first or second-order optimization algorithm to solve with respect to the concatenated vector of variables. This is illustrated in Algorithm 2.

Using the Schur complement trick Depending on the problem structure, the linearized subproblem solved here may have a structured sparse pattern like Fig. 2.1a. This is typically the case for bundle adjustment [Triggs et al., 2000]. In such case, it is more beneficial to apply the Schur complement trick and solve the reduced system matrix [Lourakis and Argyros, 2009].

The Schur complement trick can be derived as follows. First, note that the Hessian (or its approximation) matrix shown in Fig. 2.1a can be block-decomposed as

$$\begin{bmatrix} \mathtt{A} & \mathtt{B} \\ \mathtt{B}^\top & \mathtt{D} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u} \\ \Delta\mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} \tag{2.2}$$

where

$$\mathtt{A} = \frac{\partial^2 f(\mathbf{u},\mathbf{v})}{\partial \mathbf{u}^2} + \lambda \mathtt{I} \qquad \mathtt{B} = \frac{\partial^2 f(\mathbf{u},\mathbf{v})}{\partial \mathbf{u} \partial \mathbf{v}} \qquad \mathtt{D} = \frac{\partial^2 f(\mathbf{u},\mathbf{v})}{\partial \mathbf{v}^2} + \lambda \mathtt{I} \tag{2.3}$$

$$\mathbf{a} = -\frac{\partial f(\mathbf{u},\mathbf{v})}{\partial \mathbf{u}} \qquad \mathbf{b} = -\frac{\partial f(\mathbf{u},\mathbf{v})}{\partial \mathbf{v}}. \tag{2.4}$$

(2.2) can be rewritten as two simultaneous equations

$$\mathtt{A}\Delta\mathbf{u} + \mathtt{B}\Delta\mathbf{v} = \mathbf{a} \tag{2.5}$$

$$\mathtt{B}^\top\Delta\mathbf{u} + \mathtt{C}\Delta\mathbf{v} = \mathbf{b}. \tag{2.6}$$

Eliminating $\Delta\mathbf{v}$ from (2.5) yields

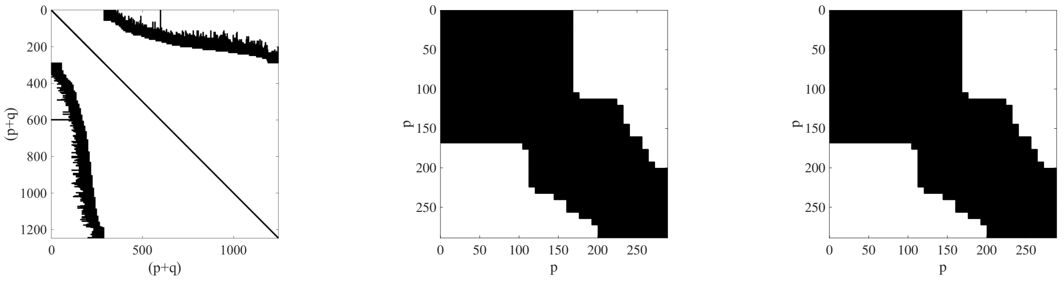
$$\Delta\mathbf{u} = (\mathtt{A} - \mathtt{B}\mathtt{D}^{-1}\mathtt{B}^\top)^{-1}(\mathbf{a} - \mathtt{B}\mathtt{D}^{-1}\mathbf{b}) \tag{2.7}$$

where $\mathtt{A} - \mathtt{B}\mathtt{D}^{-1}\mathtt{B}^\top$ is known as the Schur complement. Now, substituting (2.7) back into (2.6) yields

$$\Delta\mathbf{v} = \mathtt{D}^{-1}(\mathbf{b} - \mathtt{B}^\top\Delta\mathbf{u}). \tag{2.8}$$

There are two advantages from applying this trick. First, since $\mathtt{D}$ is block-diagonal, inverting $\mathtt{D}^{-1}$ is relatively cheap when forming the Schur complement. Second, the reduced system matrix $(\mathtt{A} - \mathtt{B}\mathtt{D}^{-1}\mathtt{B}^\top)$ is always equally or better conditioned than the original system matrix. A simple proof for this is provided by Agarwal et al. [2010].

Embedded point iteration (EPI) After each simultaneous update of $\mathbf{u}$ and $\mathbf{v}$, one may optionally further minimize over one set of the parameters (e.g. $\mathbf{v}$). This was proposed by Jeong et al. [2010] in the context of bundle adjustment, in which this can be viewed as performing joint optimization of cameras and points followed by additional triangulation. As mentioned in the related work section (Section 2.4), some believe this is equivalent to variable projection, but this will be proved incorrect in Chapter 4.



(a) Joint optimization (b) Joint, Schur complement (c) Variable projection

Fig. 2.1 The sparsity patterns of linear systems solved by different optimization methods on the trimmed dinosaur dataset. Note the sparsity pattern of the Schur complement-reduced system is the same as that of the variable projection, triggering some to believe that VarPro is simply joint optimization with the Schur complement trick applied (which is not true). Their exact similiaries and differences are analyzed in Section 4.2.3.

2.1.3 Variable projection (VarPro)

The variable projection (VarPro) method minimizes $f(\mathbf{u},\mathbf{v})$ by first eliminating $\mathbf{v}$ optimally from the problem and then solving a reduced problem only in $\mathbf{u}$. In terms of equations, VarPro solves

$$f^*(\mathbf{u}) := f(\mathbf{u},\mathbf{v}^*(\mathbf{u})) \quad \text{where} \quad \mathbf{v}^*(\mathbf{u}) := \arg\min_{\mathbf{v}} f(\mathbf{u},\mathbf{v}). \tag{2.9}$$

Algorithm 3 presents a high level implementation of this method.

Separable nonlinear least squares VarPro was first proposed by Golub and Pereyra [1973] for solving a class of problems known as *separable* nonlinear least squares (SNLS), which can be formulated as

$$f(\mathbf{u},\mathbf{v}) = \frac{1}{2}\|\mathtt{G}(\mathbf{u})\mathbf{v} - \mathbf{z}(\mathbf{u})\|_2^2 \tag{2.10}$$

where $\mathtt{G} : \mathbb{R}^p \to \mathbb{R}^{m\times q}$ is a potentially nonlinear matrix function of $\mathbf{u}$ and $\mathbf{z} : \mathbb{R}^p \to \mathbb{R}^m$ is vector function of $\mathbf{u}$. For these SNLS problems, the optimal $\mathbf{v}$ given $\mathbf{u}$ can be obtained simply via linear least squares from Section A.2.2, yielding

$$\mathbf{v}^*(\mathbf{u}) = \mathtt{G}(\mathbf{u})^\dagger \mathbf{z}(\mathbf{u}). \tag{2.11}$$

Algorithm 3 Variable projection for bivariate least squares

```
Inputs: $\mathbf{u} \in \mathbb{R}^p$
$\mathbf{v} \leftarrow \arg\min_{\mathbf{v}} f(\mathbf{u},\mathbf{v})$
repeat
    repeat
        Compute $\Delta\mathbf{u}$ by using the linearized VarPro submodel around $(\mathbf{u},\mathbf{v})$.
        $\Delta\mathbf{v} \leftarrow \arg\min_{\Delta\mathbf{v}} f(\mathbf{u}+\Delta\mathbf{u},\mathbf{v}+\Delta\mathbf{v})$
    until $f(\mathbf{u}+\Delta\mathbf{u},\mathbf{v}+\Delta\mathbf{v}) < f(\mathbf{u},\mathbf{v})$
    $\mathbf{u} \leftarrow \mathbf{u}+\Delta\mathbf{u}$
    $\mathbf{v} \leftarrow \mathbf{v}+\Delta\mathbf{v}$
until convergence
Output: $\mathbf{u} \in \mathbb{R}^p$, $\mathbf{v} \in \mathbb{R}^q$
```

Hence, the new reduced cost function $f^*(\mathbf{u}) \in \mathbb{R}$ is

$$f^*(\mathbf{u}) = \frac{1}{2}\|\mathtt{G}(\mathbf{u})\mathbf{v}^*(\mathbf{u}) - \mathbf{z}(\mathbf{u})\|_2^2 = \frac{1}{2}\|\varepsilon(\mathbf{u},\mathbf{v}^*(\mathbf{u}))\|_2^2 =: \frac{1}{2}\|\varepsilon^*(\mathbf{u})\|_2^2, \tag{2.12}$$

and the corresponding Jacobian $\mathtt{J}^*(\mathbf{u}) := d\varepsilon^*(\mathbf{u})/d\mathbf{u}$ can be obtained via chain rule

$$\begin{aligned}\mathtt{J}^*(\mathbf{u}) = \mathtt{J}_{\mathbf{u}}(\mathbf{u},\mathbf{v}^*(\mathbf{u})) &= \frac{\partial\varepsilon(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{\partial\mathbf{u}} + \frac{\partial\varepsilon(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{\partial\mathbf{v}}\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} \\ &=: \mathtt{J}_{\mathbf{u}}(\mathbf{u},\mathbf{v}^*(\mathbf{u})) + \mathtt{J}_{\mathbf{v}}(\mathbf{u},\mathbf{v}^*(\mathbf{u}))\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}}.\end{aligned} \tag{2.13}$$

(2.13) shows exactly the difference between block coordinate descent and VarPro. The first algorithm assumes $\mathbf{u}$ and $\mathbf{v}$ are independent and therefore $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u} = 0$, whereas VarPro computes derivatives by considering how the eliminated parameters $\mathbf{v}$ will change. $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u}$ can be expressed as

$$\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} = [\mathtt{G}(\mathbf{u})]^\dagger\frac{d\mathbf{z}(\mathbf{u})}{d\mathbf{u}} + \frac{d\left[\mathtt{G}(\mathbf{u})^\dagger\right]}{d\mathbf{u}}\mathbf{z}(\mathbf{u}) \tag{2.14}$$

which requires either the pseudoinverse differentiation rule (A.25) or applying the implicit function theorem (illustrated below) for further expansion.

Applying (A.25) to the second term of (2.14) yields

$$\begin{aligned}\partial\left[\mathtt{G}(\mathbf{u})^\dagger\right]\mathbf{z}(\mathbf{u}) &= -\mathtt{G}(\mathbf{u})^\dagger\partial[\mathtt{G}(\mathbf{u})]^\dagger\mathtt{G}(\mathbf{u})\mathbf{z}(\mathbf{u}) +^{-1}(\mathtt{G}(\mathbf{u})^\top\mathtt{G}(\mathbf{u}))\partial[\mathtt{G}(\mathbf{u})]^\top(\mathtt{I}-\mathtt{G}(\mathbf{u})\mathtt{G}(\mathbf{u})^\dagger)\mathbf{z}(\mathbf{u}) \\ &= -\mathtt{G}(\mathbf{u})^\dagger\partial[\mathtt{G}(\mathbf{u})]\mathbf{v}^*(\mathbf{u}) -^{-1}(\mathtt{G}(\mathbf{u})^\top\mathtt{G}(\mathbf{u}))\partial[\mathtt{G}(\mathbf{u})]^\top\varepsilon^*(\mathbf{u}),\end{aligned} \tag{2.15}$$

leading to

$$\begin{aligned}\partial \mathbf{v}^*(\mathbf{u}) &= \partial\left[\mathtt{G}(\mathbf{u})^\dagger\right]\mathbf{z}(\mathbf{u}) + \mathtt{G}(\mathbf{u})^\dagger \partial\mathbf{z}(\mathbf{u}) \\ &= -\mathtt{G}(\mathbf{u})^\dagger \partial[\mathtt{G}(\mathbf{u})]\mathbf{v}^*(\mathbf{u}) + \mathtt{G}(\mathbf{u})^\dagger \partial\mathbf{z}(\mathbf{u}) - (\mathtt{G}(\mathbf{u})^\top \mathtt{G}(\mathbf{u}))^{-1}\partial[\mathtt{G}(\mathbf{u})]^\top \varepsilon^*(\mathbf{u}). \end{aligned} \tag{2.16}$$

By noting that

$$\mathtt{J}_{\mathbf{u}}(\mathbf{u},\mathbf{v}) = \frac{\partial\varepsilon(\mathbf{u},\mathbf{v})}{\partial\mathbf{u}} = \frac{\partial[\mathtt{G}(\mathbf{u})]}{\partial\mathbf{u}}\mathbf{v}^*(\mathbf{u}) - \frac{\partial\mathbf{z}(\mathbf{u})}{\partial\mathbf{u}}, \tag{2.17}$$

(2.16) becomes

$$\partial\mathbf{v}^*(\mathbf{u}) = -\mathtt{G}(\mathbf{u})^\dagger \partial\varepsilon(\mathbf{u},\mathbf{v}^*(\mathbf{u})) - (\mathtt{G}(\mathbf{u})^\top \mathtt{G}(\mathbf{u}))^{-1}\partial[\mathtt{G}(\mathbf{u})]^\top \varepsilon^*(\mathbf{u}). \tag{2.18}$$

Finally, using (2.17) and the equality $\mathtt{G}(\mathbf{u}) = \mathtt{J}_{\mathbf{v}}(\mathbf{u})$ yields

$$\begin{aligned}\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} &= -\mathtt{G}(\mathbf{u})^\dagger \frac{\partial\varepsilon(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{\partial\mathbf{u}} - (\mathtt{G}(\mathbf{u})^\top \mathtt{G}(\mathbf{u}))^{-1}\frac{d[\mathtt{G}(\mathbf{u})]^\top \varepsilon^*(\mathbf{u})}{d\mathbf{u}} \\ &= -\mathtt{J}_{\mathbf{v}}(\mathbf{u})^\dagger \mathtt{J}_{\mathbf{u}}(\mathbf{u},\mathbf{v}^*(\mathbf{u})) - (\mathtt{J}_{\mathbf{v}}(\mathbf{u})^\top \mathtt{J}_{\mathbf{v}}(\mathbf{u}))^{-1}\frac{d[\mathtt{J}_{\mathbf{v}}(\mathbf{u})]^\top \varepsilon^*(\mathbf{u})}{d\mathbf{u}}. \end{aligned} \tag{2.19}$$

View as a constrained problem Ruhe and Wedin [1980] and Okatani and Deguchi [2007] used the implicit differentiation to derive the gradients. Since VarPro always has optimal $\mathbf{v}$ for given $\mathbf{u}$, they used the fact that the gradient with respect to $\mathbf{v}$, $\nabla_{\mathbf{v}} f(\mathbf{u},\mathbf{v}^*(\mathbf{u})) = \mathtt{J}_{\mathbf{v}}(\mathbf{u})^\top \varepsilon(\mathbf{u},\mathbf{v}^*(\mathbf{u}))$, is 0 for any $\mathbf{u}$. They then implicitly differentiated this expression with respect to $\mathbf{u}$ to yield an expression for $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u}$. This can be seen as solving

$$\arg\min_{\mathbf{u}\in\mathbb{R}^p,\mathbf{v}\in\mathbb{R}^q} f(\mathbf{u},\mathbf{v}) \quad \text{s.t.} \quad \nabla_{\mathbf{v}} f(\mathbf{u},\mathbf{v}) = 0 \tag{2.20}$$

$$\Rightarrow \arg\min_{\mathbf{u}\in\mathbb{R}^p,\mathbf{v}\in\mathbb{R}^q} \frac{1}{2}\|\mathtt{G}(\mathbf{u})\mathbf{v} - \mathbf{z}(\mathbf{u})\|_2^2 \quad \text{s.t.} \quad \mathtt{J}_{\mathbf{v}}(\mathbf{u})^\top \varepsilon(\mathbf{u},\mathbf{v}) = 0 \tag{2.21}$$

By applying the implicit function theorem from Section A.2.6, one yields that, around the local neighbourhood of $\mathbf{v} = \mathbf{v}^*(\mathbf{u})$,

$$\frac{d\mathbf{v}}{d\mathbf{u}} = -(\mathtt{J}_{\mathbf{v}}(\mathbf{u})^\top \mathtt{J}_{\mathbf{v}}(\mathbf{u}))^{-1}\frac{\partial}{\partial\mathbf{u}}\left[\mathtt{J}_{\mathbf{v}}(\mathbf{u})^\top \varepsilon(\mathbf{u},\mathbf{v})\right]. \tag{2.22}$$

Since $\mathbf{v} = \mathbf{v}^*(\mathbf{u})$ for the constraint to be satisfied, above will yield (2.19) without requiring any complex pseudoinverse differentiation rule for computing the derivatives.

Ruhe and Wedin's algorithms Ruhe and Wedin [1980] proposed 3 Gauss Newton-based algorithms solving (2.12). Although they are formally termed Algorithms 1, 2 and 3, here these will be referred to as RW1, RW2 and RW3 respectively for convenience.

RW1 applies the Gauss-Newton approximation of the Hessian in solving (2.12). RW2 makes a first order approximation in computing $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u}$, discarding the rightmost second order term in (2.14). RW3 is the same as the block coordinate descent in Section 3.1.4 since it assumes $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u} = 0$. Detailed derivations can be found in the original paper by Ruhe and Wedin [1980].

Connection to joint optimization with the Schur complement trick Due to the fact that both VarPro and the Schur complement trick in joint optimization eliminate a set of variables, some speculated [Agarwal et al., 2014; Strelow, 2012b] that they are equal but this is proven not to be the case in Section 4.2.3.

Application to nonseparable problems It is often the case that the problem is nonseparable, meaning that it cannot eliminate either of **u** and **v** in closed form. Strelow [2012b] extended this to apply to nonseparable problems of the form $\arg\min_{\mathbf{u},\,\mathbf{v}} \|\varepsilon(\mathbf{u},\mathbf{v})\|_2^2$. Section 5.1 illustrates the derivation process.

2.1.4 Behaviours on the Rosenbrock function

The Rosenbrock function is one of the widely used cost functions for testing the performance of optimization algorithms. It is defined as

$$f(u,v) = (a-u)^2 + b(v-u^2)^2 \tag{2.23}$$

or equivalently

$$\|\varepsilon(u,v)\|_2^2 := \left\| \begin{matrix} a-u \\ \sqrt{b}(v-u^2) \end{matrix} \right\|_2^2 . \tag{2.24}$$

As shown in Fig. 2.2, the function has a banana-shaped valley, which is usually difficult to overcome without using its curvature information. The figure shows the performance of different algorithms when starting inside the valley. The block coordinate descent shows the worst performance, not converging after 100 iterations. On the other hand, VarPro converges in 4 iterations. The joint optimization-based algorithms perform in between. The test empirically

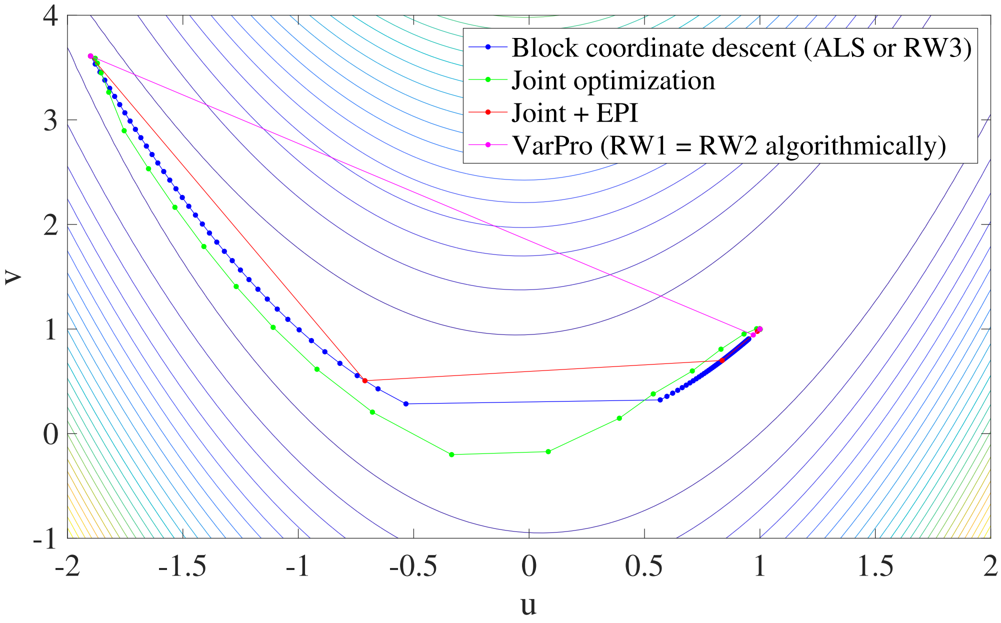


Fig. 2.2 Behaviours of different optimization methods on the Rosenbrock function from Section 2.1.4 ($a = 1$, $b = 10$, $u_0 = -1.9$, $v_0 = u_0^2$,). Block coordinate descent does not converge after 100 iterations, while joint optimization, joint optimization with embedded point iterations (EPI) and variable projection (VarPro) converge in 61, 7 and 4 iterations respectively.

shows VarPro is different from applying the Schur complement trick in joint optimization, and this point will be further investigated in Section 4.2.3.

2.2 Structure-from-motion (SfM) pipelines

Structure-from-motion (SfM) is a multistage pipeline that reconstructs a scene from a set of overlapping images. The first half of the pipeline is well-established, extracting sparse features in each image and building accurate correspondences between them based on geometric cues. The second half comprises several optimization steps including bundle adjustment that lead to valid 3D reconstruction and camera poses. The second stage is termed an *incremental pipeline* if images are registered incrementally [Agarwal et al., 2010; Schönberger and Frahm, 2016; Snavely et al., 2006], and called a *global pipeline* [Moulon et al., 2013; Olsson and Enqvist, 2011; Sweeney et al., 2015] if they are considered simultaneously. This section illustrates the first stage and summarizes each pipeline. A graphical illustration of incremental and global approaches can be found in Fig. 2.3 and Fig. 2.4 respectively.

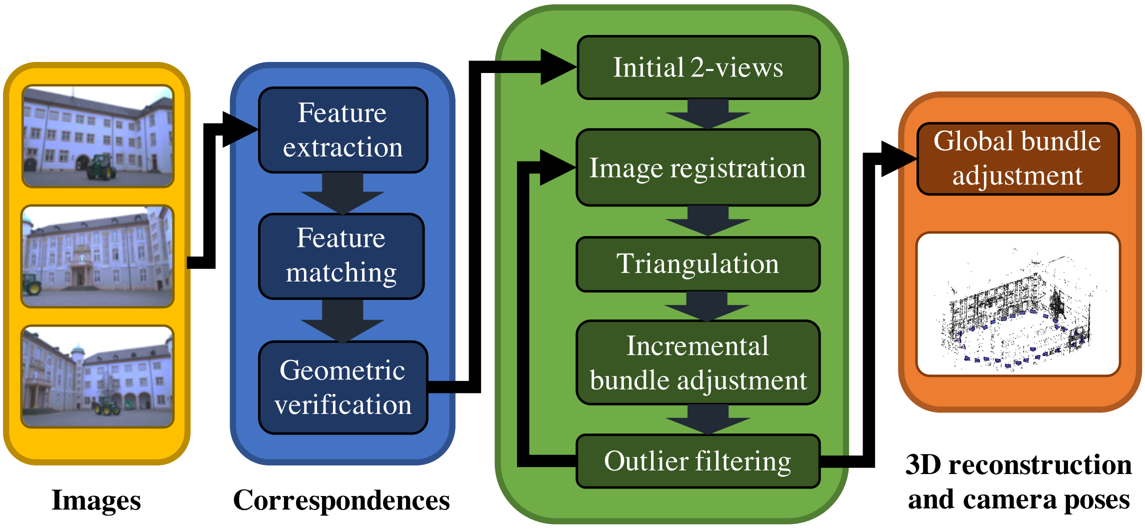


Fig. 2.3 An overview of a typical incremental structure-from-motion pipeline.

2.2.1 Finding correspondences

SfM pipelines use a sparse representation of images to form correspondences between them. This stage itself comprises multiple steps (see Fig. 2.3), starting from feature detection, description, score-based matching and finally geometric verification of matches.

Feature extraction and matching

Generating a sparse representation is typically achieved by using a hand-crafted feature detector that uses local pixelwise information to detect blobs. There is a handful of methods available—Harris corner detector [Harris and Stephens, 1988] was one of the earlier detectors that gained popularity for its rotational invariance. The turning point was Lowe's scale invariance feature transform (SIFT) [Lowe, 2004], which significantly improved the accuracy of feature description and matching. Since then, numerous feature detection and/or description methods have been proposed such as speeded-up robust features (SURF) [Bay et al., 2006] and affine invariant-SIFT (ASIFT) [Yu and Morel, 2011], which can be viewed as variants of SIFT, and other faster detectors aimed for realtime applications such as FAST [Rosten and Drummond, 2005, 2006] and Oriented FAST and BRIEF (ORB) [Rublee et al., 2011]. In case the images are ordered, for instance come from a video stream, then one may apply an optical flow-based method such as the KLT tracker [Lucas and Kanade, 1981]. Recently, a neural net based approach has also been proposed [Yi et al., 2016] that learns to detect and describe features. This section will summarize the SIFT framework below, which is still a default feature detector

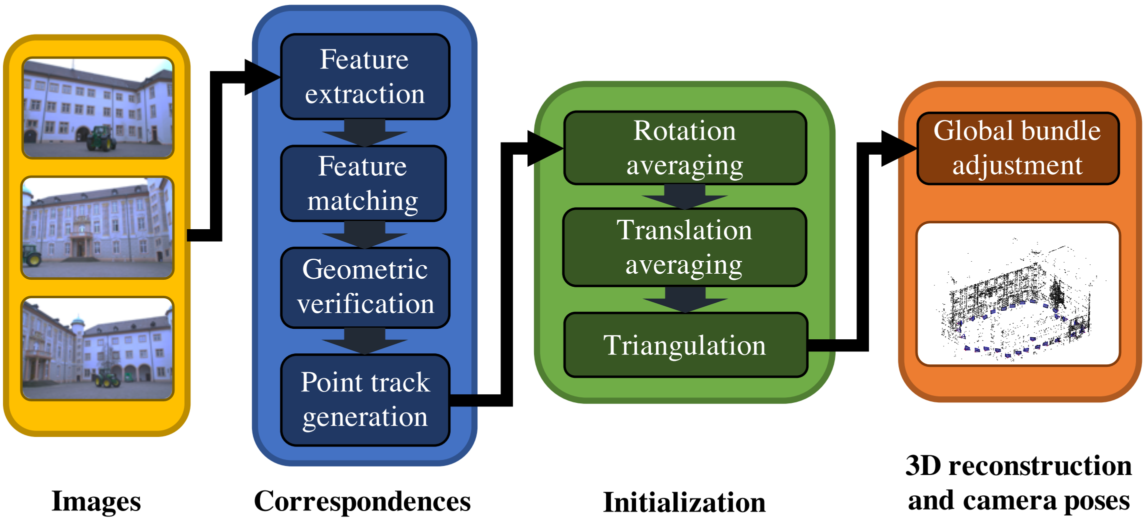


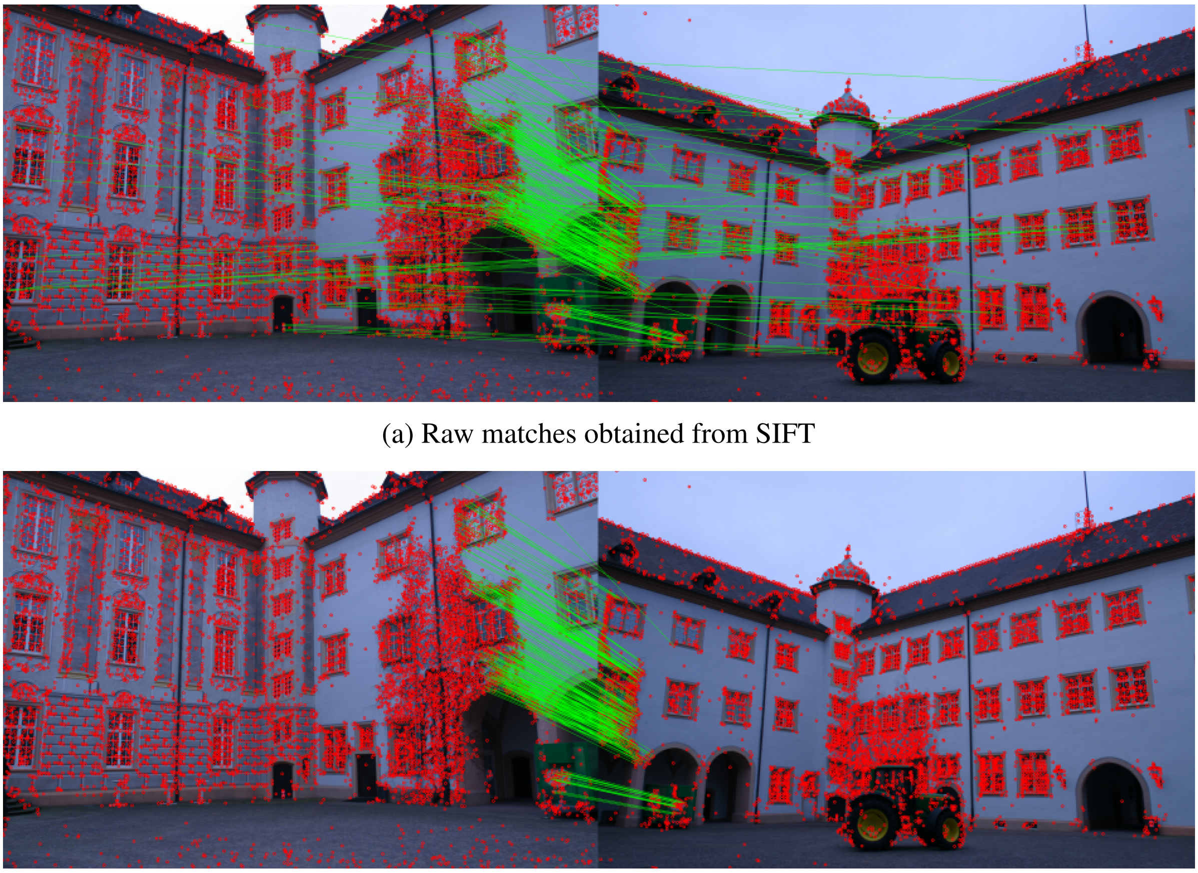
Fig. 2.4 An overview of a typical global structure-from-motion pipeline.

and descriptor for several state-of-the-art structure-from-motion pipelines [Schönberger and Frahm, 2016; Sweeney et al., 2015].

SIFT feature detection SIFT's feature detection works by first blurring an image using Gaussian filters of different scales, thereby forming a pyramid of blurred images with different scales. Extrema points in each scale can be detected using the Laplacian of Gaussian (LoG) filter, but this is approximated by computing the difference of neighbouring Gaussian (DoG) blurred images for practical implementation. Then extrema (key) points are found by searching across the pixels and scales (i.e. considering a $3 \times 3 \times 3$ block) after which some are discarded based on the image intensity threshold and the eigenvalue test (to get rid of edges).

SIFT feature description Once keypoints are detected, the orientation of each keypoint is first computed to be used for establishing rotational invariance later on. For this purpose, the gradients (comprising scale and direction) around each keypoint at the detected scale are computed and binned into a histogram of 36 bins and covering 360°. For each keypoint, the highest peak is used to compute the overall orientation of the keypoint. Additionally, any other peak which is at least 80% the height of the highest peak is used to create a new keypoint at the same spot but with different orientation (in order to try many potential choices).

Finally, each keypoint is "fingerprinted" by generating histograms of gradients around the key point. Typically, this is achieved by segmenting 16×16 neighbouring pixel gradients into 4×4 blocks each of which produces an 8-bin histogram of gradients. One thing to note is that



(a) Raw matches obtained from SIFT

(b) Two-view verified matches

Fig. 2.5 Example showing SIFT matches before and after 2-view verified matches. Each red point denotes a SIFT feature. Gross outliers are discarded by enforcing epipolar constraints as illustrated in Section 2.2.1). This image was drawn using COLMAP [Schönberger and Frahm, 2016].

gradients are weighted depending on its distance from the keypoint. Storing information from 16 histograms requires a $16 \times 8 = 128$ real vector.

Now, the above 16 gradient histograms are adjusted by the keypoint's orientation to gain rotational invariance. Also, the values of the 128 feature vector are thresholded and renormalized to remove contrast and saturation, thereby obtaining invariance to light illumination.

SIFT feature matching Once each keypoint is assigned a feature vector, matching between two features is carried out by searching for the nearest vector in 128-dimensional Euclidean distance. To gain some robustness, two features are matched only if the following two conditions are met:

1. two keypoints are closest to each other, and

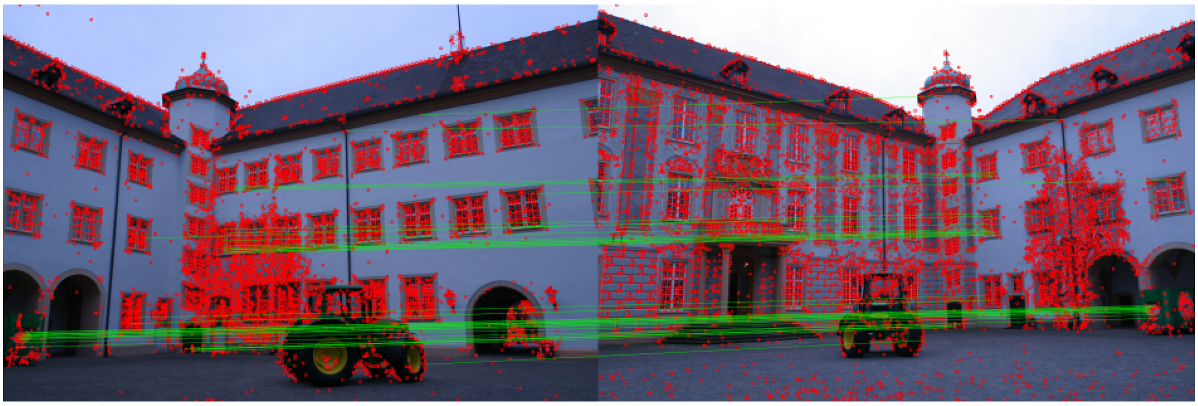


Fig. 2.6 An example of false inlier matches after 2-view geometric verification due to repetitive structures. Each red point denotes a SIFT feature. This image was drawn using COLMAP [Schönberger and Frahm, 2016].

2. for each of the two keypoints, the distance to its second closest keypoint is at least 30-40% longer than that between the two keypoints considered here.

Both conditions encourage only unique matches to survive. Yet since this matching technique is based on some algebraic distance, it should be verified using geometric constraints, which will be discussed in the next paragraph.

Geometric verification of matches

The pairwise matches obtained from the previous section are likely to contain gross outliers as they have not incorporated geometric constraints (see Fig. 2.5a for an example). This section summarizes a way to verify them and discard outliers.

Two-view verification For each pair of views, the inlier correspondences are identified using the RANSAC-based estimation of two-view geometry using the method illustrated in Section A.4.5. Once detected outlier matches are discarded, the obtained geometric quantity (the essential, fundamental or homography matrix) is refined by minimizing the Sampson error illustrated in Section A.4.3 (which can be viewed as a first order approximation of the reprojection error). The result is shown in Fig. 2.5.

Triplet filtering Although the two-view verification stage gets rid of many gross outliers, it cannot remove all the false matches that arise from symmetries in the scene (e.g. right and left sides of a car) and/or repetitive structures. Fig. 2.5b illustrates an example. These false positives can significantly deteriorate the reconstruction accuracy [Enqvist et al., 2011; Wilson and Snavely, 2013; Zach et al., 2010]. In triplet filtering, a set of pairwise matches between

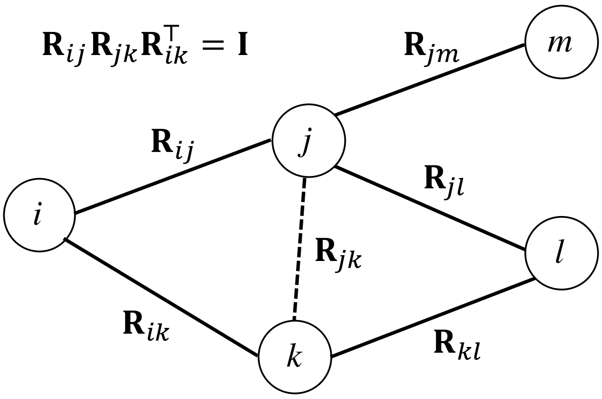


Fig. 2.7 An illustration of a scene graph and triplet filtering. Each vertex represents an image and each edge represents the pairwise feature matches between the connected vertices. Each triplet loop formed should form an identity rotation matrix and any edge (e.g. $\mathtt{R}_{jk}$) which forms erroneous triplets should be discarded. Solving this correctly and efficiently is an active research problem [Moulon et al., 2013; Olsson and Enqvist, 2011; Sweeney et al., 2015; Wilson and Snavely, 2013].

a pair of images is defined as an edge in an undirected graph, where each vertex represents an image. Then, all the triple loops formed by these edges are considered to find potentially false edges. This is carried out by considering the relative rotation in each of the participating edges and verifying that the product of three rotations yields the identity rotation up to some predefined threshold value. (Fig. 2.7 provides a graphical illustration.) This requires camera instrinsics to be known. One may be generous [Moulon et al., 2013; Sweeney et al., 2015] and discard edges only if they do not participate in any of the "good" triplets, or can be more strict and discard minimum number of edges to make all the triplets satisfy the constraints (used in Section 6.4). This removes a substantial number of false positive edges, contributing to a cleaner scene graph.

Zach et al. [2010] generalized this approach to incorporate longer loops, although the maximum loop size was limited to 6 for practical implementation.

2.2.2 Incremental pipeline

A schematic diagram of an incremental pipeline is provided in Fig. 2.4. Given a set of pairwise matches, An incremental pipeline first selects two views which have large number of pairwise matches as well as wide baseline such that points can be triangulated stably.

It then proceeds by registering a camera view which shares the most number of matches with the previous two views. This can be tackled by using a perspective-n-point (PnP) algorithm [Lepetit et al., 2008; Zheng et al., 2013], which estimates a projection matrix that best represents the relationship between the triangulated 3D points and their observations. Once a new camera is registered, the pairwise matches between them are used to triangulate more points. (Some matches may be false so they need to be filtered out during this phase.) This approach can be seen as an alternating scheme of optimizing camera poses given points followed by optimizing points given cameras, which is known to be unstable on its own [Schönberger and Frahm, 2016]. Therefore it is crucial that joint optimization of cameras and points via bundle adjustment takes place every time n images are added, where a typical value for n is between 1 and 6. Sometimes, bundle adjustment may be applied locally around the newly added camera view for efficiency. Nevertheless, due to the incremental nature of the approach, the quality of solution depends substantially on the choice of initial pair of views and is prone to accumulated camera position errors known as *drifts*.

2.2.3 Global pipeline

As illustrated in Fig. 2.4, a global pipeline typically generates a set of consistent point tracks from pairwise matches, makes an initial estimate of all camera poses given the point tracks, triangulates points given the camera pose estimates and then performs bundle adjustment. These steps are reviewed below.

The approach illustrated in Chapter 6 can be viewed as a global approach but does not require initial estimates of camera poses or points (see Fig. 6.4).

Generating point tracks from pairwise matches

For global pipelines, pairwise matches need to be converted to consistent point tracks before proceeding to the rotation and translation averaging stages. This is because it solves global bundle adjustment right after camera parameters are estimated, and therefore matches cannot be robustly filtered out during the optimization phase (which is the case for the incremental approach). Also, false matches lead to inconsistent point tracks in which the track comprises more than one point from one image. (Fig. 6.5 shows an illustration.) By not stitching the matches correctly, the solution quality may substantially disimprove.

Sweeney et al. [2015]'s Theia and Moulon et al. [2013]'s OpenMVG use a simple approach—if a feature point in one image is already registered in a point track, then any other feature points from the same image are prohibited from being added (which can happen from inconsistent pairwise matches). However, this is just a temporary measure since it is uncertain which of the feature points from that image is actually a true inlier. Generating true inlier point tracks in presence of false pairwise matches is an active field of research [Maset et al., 2017].

Algorithm 4 Olsson and Enqvist [2011]'s algorithm for creating point tracks from pairwise matches

```
Inputs: images (vertices) and pairwise image connections (edges) forming $(\mathscr{V}, \mathscr{E})$.
For each feature point ($k$) in each image ($i$), initialize a point track $T(i_k) = \{i_k\}$.
Select one image ($i$) arbitrarily and add it to the set of visited images $I$. i.e. $I = \{i\}$.
repeat
    Select an image ($j$) adjacent to one of the previously visited images (i.e. $i \in I$) that has the maximum edge weight (given by the # of feature matches between images $i$ and $j$).
    for each feature match $i_m$ and $j_n$ do
        if $T(i_m) \cup T(j_n)$ form a consistent track (i.e. no $\geq 2$ points in any image) then
            Merge point tracks $T(i_m)$ and $T(j_n)$
        end if
    end for
until all edges are tried. i.e. $\mathscr{E} = \emptyset$.
Output: consistent point tracks
```

This section concludes by reviewing a simple algorithm by Olsson and Enqvist [2011], which tries to alleviate the above problem by incorporating edge weights derived from the number of pairwise matches. In this algorithm, pairs of images with higher number of matches are processed earlier, gaining priority in the point track generation process. The mathematical proofs are provided by Olsson and Enqvist [2011].

Rotation and translation averaging

The next step comprises finding initial camera poses which best explain the relative transformations observed in each edge of the scene graph (e.g. Fig. 2.7). For an edge connecting image i and j, the following constraint holds:

$$\mathtt{R}_{i,j} = \mathtt{R}_i^\top \mathtt{R}_j \tag{2.25}$$

$$\mathbf{t}_{i,j} \simeq \mathtt{R}_i^\top (\mathbf{t}_j - \mathbf{t}_i) \tag{2.26}$$

where $[\mathtt{R}_{i,j},\ \mathbf{t}_{i,j}] \in \mathrm{SE}(3)$ represents an up-to-scale relative transformation, and $[\mathtt{R}_i,\ \mathbf{t}_i] \in \mathrm{SE}(3)$ represents the camera matrix for image i in world coordinates. Solving above is typically carried out in two stages—first estimating global rotations using (2.25) followed by estimating global translations using (2.26).

Rotation averaging Hartley et al. [2013] gives a detailed illustration of various rotation averaging techniques in the literature. In brief, the problem can be formulated as

$$\arg\min_{\{\mathtt{R}_i\}} \sum_{(i,j)\in\Omega} d(\mathtt{R}_{i,j}, \mathtt{R}_i^\top \mathtt{R}_j)^2 \tag{2.27}$$

where $\mathtt{R}_i \in \mathrm{SO}(3)$ is the rotation matrix of camera i in world coordinates, $\mathtt{R}_i j \in \mathrm{SO}(3)$ is the relative rotation from camera i's coordinates to camera j's, and $d : \mathrm{SO}(3) \to \mathbb{R}$ is some distance measure. (One can minimize the L^1-norm instead.)

Since (2.27) is a nonlinear problem, it is usually initialized by taking one camera as a reference frame and propagating estimates of global rotations along the graph [Olsson and Enqvist, 2011]. Such initialization can be sensitive to outliers. Nevertheless, recent work has successfully implemented robust rotation averaging on large-scale datasets [Chatterjee and Govindu, 2013].

Translation averaging Once global rotations are estimated, the translation averaging tries to find best translations by applying the constraint (2.26). This is usually the trickest part, since incorrect rotations can lead to incorrect translation parameters. The one which has enjoyed most success is Wilson and Snavely [2014]'s 1DSfM.

2.2.4 Bundle adjustment

The bundle adjustment problem is a nonlinear joint minimization of the reprojection errors across all point tracks over the participating camera poses and 3D points. The problem is formally defined in (1.1). In case the perspective camera model is used, the minimization would involve either SO(3) or homogeneous quantities, which can be solved using the manifold optimization framework illustrated in Section A.2.5.

Initialization Since (1.1) is a nonlinear problem, an incremental approach uses previously bundle adjusted camera poses and 3D points along with newly registered cameras and triangulated points for initialization. A global approach uses the estimates of camera parameters yielded from rotation and translation averaging to first triangulate 3D points, and these estimates are used for starting bundle adjustment.

On the other hand, the strategies described in Chapter 6 do not require these initial estimates but need sufficiently clean tracks for it to succeed.

Probabilisitic interpretation

It is also possible to represent bundle adjustment as finding the maximum likelihood estimate of data (2D image points) given model parameters (cameras and 3D points) assuming i.i.d. Gaussian noise on the image plane. This yields

$$
\begin{aligned}
& p(\{\mathbf{m}_{i,j}\}|\{\mathtt{P}_i\},\{\mathbf{x}_j\},\sigma) && (2.28)\\
&= \Pi_{(i,j)\in\Omega} p(\mathbf{m}_{i,j}|\mathtt{P}_i,\mathbf{x}_j) && (2.29)\\
&= \Pi_{(i,j)\in\Omega} \frac{1}{\det(2\pi\Sigma)^{-1/2}} \exp\left(-\frac{1}{2}\left(\mathbf{m}_{i,j}-\pi(\mathtt{P}_i\mathbf{x}_j)\right)^\top \Sigma^{-1}\left(\mathbf{m}_{i,j}-\pi(\mathtt{P}_i\mathbf{x}_j)\right)\right)\\
&= \Pi_{(i,j)\in\Omega} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\|\mathbf{m}_{i,j}-\pi(\mathtt{P}_i\mathbf{x}_j)\|_2^2}{2\sigma^2}\right). && (2.30)
\end{aligned}
$$

Since $\log : \mathbb{R} \to \mathbb{R}$ is a monotonically increasing function, maximizing (2.28) is equivalent to maximizing its log quantity. Hence,

$$
\begin{aligned}
& \underset{\{\mathtt{P}_i\},\{\mathbf{x}_j\},\sigma}{\arg\max} \quad \log p(\{\mathbf{m}_{i,j}\}|\{\mathtt{P}_i\},\{\mathbf{x}_j\},\sigma)\\
= \; & \underset{\{\mathtt{P}_i\},\{\mathbf{x}_j\},\sigma}{\arg\min} \sum_{(i,j)\in\Omega} \frac{1}{2\sigma^2}\|\mathbf{m}_{i,j}-\pi(\mathtt{P}_i\mathbf{x}_j)\|_2^2 + \frac{1}{2}\log(2\pi\sigma^2). \qquad (2.31)
\end{aligned}
$$

In bundle adjustment, σ (the variance of the sensor noise) is usually assumed to be a nonzero constant, although in theory it could also be minimized jointly. Setting σ as constant yields (1.1) without the robust loss function, which may be added by assuming a different noise distribution such as Cauchy or other t-distributions.

In reality, the isotropic assumption may not always be valid, for instance if there is light saturation or motion blur in the image. (To avoid this, the exposure time of the camera needs to be shortened which will compensate the overall brightness of the image.)

Connection to fixed-rank matrix factorization

Bundle adjustment has a somewhat close connection to rank-3 matrix factorization with a mean vector and potential missing data. This is illustrated in Section 2.3.1.

2.2.5 Metric upgrade

As briefly mentioned in Section A.3.3, a reconstruction obtained in the projection frame has gauge freedom that needs to be resolved for the reconstruction to be valid in the metric frame. Resolving this ambiguity and enforcing the structural constraint of the perspective camera model ($\mathtt{P}_i = \mathtt{K}_i[\mathtt{R}_i\mathbf{t}_i]$) is known as the *metric upgrade*. The approach can be extended to include a step of estimating unknown camera intrinsic parameters in which it is more frequently called self or *autocalibration*. This process is a well-studied topic in 3D vision [Armstrong et al., 1996; Chandraker et al., 2010; Gherardi and Fusiello, 2010; Hartley, 1994; Heyden and Astrom, 1996, 1997; Pollefeys and van Gool, 1999; Pollefeys et al., 1999], but this dissertation will assume that camera intrinsics are known a priori from some standard calibration [Zhang, 2000].

The first part of the metric upgrade requires finding an ambiguity matrix $\mathtt{H}$ which transforms a stack of camera matrices to a solution closest the $\mathrm{SE}(3)$ manifold. The aim is to find a ambiguity-resolving matrix $\mathtt{H} \in \mathbb{R}^{4\times 4}$ such that

$$\mathtt{P}_i\mathtt{H} = \mathtt{P}_i \begin{bmatrix} \mathtt{A} & \mathbf{0} \\ \mathbf{c}^\top & 1 \end{bmatrix} \approx \mathtt{K}_i \begin{bmatrix} \mathtt{R}_i & \mathbf{t}_i \end{bmatrix} \quad \forall \quad 1 \leq i \leq n \tag{2.32}$$

where n is the total number of images, $\mathtt{K}_i \in \mathbb{R}^{3\times 3}$ is the upper-triangular calibration matrix of camera i, $[\mathtt{R}_i,\ \mathbf{t}_i] \in SE(3)$ represents camera i's pose and $[\mathbf{c};1] \in \mathbb{R}^4$ represents the plane at infinity in homogeneous form. The last column of $\mathtt{H}$ can be set to $[\mathbf{0}^\top 1]^\top$ as it only accounts for global translation and scaling. Since this thesis assumes camera intrinsics are known a priori, one can utilize the normalized quantity $\tilde{\mathtt{P}}_i := \mathtt{K}_i^{-1}\mathtt{P}_i$. By defining $\tilde{\mathtt{H}} \in \mathbb{R}^{3\times 4}$ to comprise the three left-most columns of $\mathtt{H}$, it yields

$$\tilde{\mathtt{P}}_i\tilde{\mathtt{H}}\tilde{\mathtt{H}}^\top\tilde{\mathtt{P}}_i^\top = \tilde{\mathtt{P}}_i \begin{bmatrix} \mathtt{I} & \mathbf{c} \\ \mathbf{c}^\top & \|\mathbf{c}\|^2 \end{bmatrix} \tilde{\mathtt{P}_i}^\top \approx \beta_i\mathtt{R}_i\mathtt{R}_i^\top = \beta_i\mathtt{I} \tag{2.33}$$

where $\beta_i \in \mathbb{R}$ is some scale factor. Since $\tilde{\mathtt{H}}$ is just an identity matrix stacked with $\mathbf{c}^\top \in \mathbb{R}^3$, we only need to minimize over 3 parameters. This optimzation procedure is often referred to as finding the plane at infinity. For this problem, previous studies have used either the modulus constraint [Pollefeys and van Gool, 1999], which promotes the three eigenvalues of each normalized camera $\tilde{\mathtt{P}}_i$ to be equal in magnitude, or the Frobenius norm constraint [Pollefeys et al., 1999], which minimizes the L^2-distance between the normalized camera matrix $\tilde{\mathtt{P}}_i/\beta_i$ and the identity matrix $\mathtt{I}$ for each camera. i.e.

$$\min_{\mathbf{c},\{alpha_i\}} \sum_{i=2}^{n} \|\alpha_i\tilde{\mathtt{P}}_i\tilde{\mathtt{H}}(\mathbf{c})\tilde{\mathtt{H}}(\mathbf{c})^\top\tilde{\mathtt{P}}_i - \mathtt{I}\|_F^2. \tag{2.34}$$

where $\alpha_i := 1/\beta_i$. Pollefeys et al. [1999] removed α_i by solving instead

$$\min_{\mathbf{c}} \sum_{i=2}^{n} \left\| \frac{\tilde{\mathtt{P}}_i\tilde{\mathtt{H}}(\mathbf{c})\tilde{\mathtt{H}}(\mathbf{c})^\top\tilde{\mathtt{P}}_i}{\|\tilde{\mathtt{P}}_i\tilde{\mathtt{H}}(\mathbf{c})\tilde{\mathtt{H}}(\mathbf{c})^\top\tilde{\mathtt{P}}_i\|_F} - \frac{\mathtt{I}}{\|\mathtt{I}\|_F} \right\|_F^2. \tag{2.35}$$

Initialization Solving (2.34) or (2.35) requires nonlinear least squares optimization, which can be sensitive to initialization. Pollefeys et al. [1999] proposed to first solve a linearized form of (2.35) in which the rank-3 constraint of $\tilde{\mathtt{H}}\tilde{\mathtt{H}}^\top \in \mathbb{R}^{4\times4}$ is relaxed by replacing $\|\mathbf{c}\|^2$ with a new independent variable. The scale factors $\{\alpha_i\}$ are effectively eliminated by normalizing the scale of each camera prior to optimization and penalizing the differences between the diagonal entries of $\tilde{\mathtt{P}}_i\tilde{\mathtt{H}}(\mathbf{c})\tilde{\mathtt{H}}(\mathbf{c})^\top\tilde{\mathtt{P}}_i$. After the linear solution is computed, the rank-3 constraint is imposed by taking the closest rank-3 approximation via SVD from Section A.1.5, which is then used as an initializer.

Finding closest rotation matrix

Enforcing the metric constraint on the upgraded solution can be achieved by projecting the rotation part of each camera to $\mathrm{SO}(3)$ using the method proposed by Arun et al. [1987] illustrated in Section A.1.9.

Cheirality

Gherardi and Fusiello [2010] proposes to flip all signs of camera translations and 3D points depending on the global cheirality of the reconstructed scene. Finally, any 3D point behind any observing camera may be optionally discarded.

2.3 Low-rank matrix factorization

This section briefly addresses various low (and possibly fixed)-rank matrix factorization problems with missing data. Some examples in computer vision and machine learning are introduced followed by the formal definitions of the problem.

2.3.1 Applications

Below are some of the real-world problems that can be solved using matrix factorization. Note that each model requires some strong assumptions that needs to be paid attention.

Rigid structure-from-motion

This rigid SfM model assumes a static object scene with multiple camera views. As mentioned in Section 2.4, the factorization approach was first proposed by Tomasi and Kanade [1992] for the weak perspective and affine camera models. This was later extended by others to include para perspective [Poelman and Kanade, 1997] and projective cameras [Sturm and Triggs, 1996].

(a) Rigid SfM (b) Non-rigid SfM (c) Photometric stereo (d) Recommender sys.

Fig. 2.8 Some applications of low-rank matrix factorization models. Applying a factorization scheme to physical systems require some strong assumptions or have constraints lifted.

Here, the illustration goes in the reverse order, starting from projective bundle adjustment, down to affine factorization. The internal constraints are dropped for simplicity.

The L^2 norm equation for bundle adjustment (1.1) can be rewritten as as

$$\arg\min_{\mathtt{P}_i,\tilde{\mathbf{x}}_j} \|\pi(\mathtt{P}_i\tilde{\mathbf{x}}_j) - \mathbf{m}_{i,j}\|_2^2 = \arg\min_{\mathtt{P},\tilde{\mathtt{X}}} \|\mathtt{W} \odot (\mathtt{M} - \Pi(\mathtt{P}\tilde{\mathtt{X}}))\|_F^2 \tag{2.36}$$

where $\mathtt{P} \in \mathbb{R}^{3F\times 4}$ is a row stack of cameras $\{\mathtt{P}_i\}$, $\mathtt{X} \in \mathbb{R}^{4\times N}$ is a column stack of homogeneous 3D point $\{\tilde{\mathbf{x}}_j\}$ and $\Pi : \mathbb{R}^{3F\times N}$ is a matrix representation of $\{\pi : \mathbb{R}^3 \to \mathbb{R}^2\}$. (2.36) can be viewed as nonlinear matrix factorization. This minimizes the sum of errors on the image planes, and the measurement (sensor) error distribution widens as depth increases (it has a cone shape).

Now since

$$\pi(\mathtt{P}_i, \tilde{\mathbf{x}}_j) := \frac{\mathtt{P}_{i,1:2}, \tilde{\mathbf{x}}_j}{\mathbf{p}_{i,3}^\top \tilde{\mathbf{x}}_j} \tag{2.37}$$

where $\mathtt{P}_i, 1:2$ is the first two rows of the camera matrix $\mathtt{P}_i$ and $\mathbf{p}_{i,3}^\top$ is the last row of the same matrix, one may try and instead minimize

$$\arg\min_{\mathtt{P}_i,\tilde{\mathbf{x}}_j} \|\mathtt{P}_{i,1:2}\tilde{\mathbf{x}}_j - \mathbf{m}_{i,j}(\mathbf{p}_{i,3}^\top, \tilde{\mathbf{x}}_j)\|_2^2 \tag{2.38}$$

which is known as *projective factorization*. Here, errors in the actual 3D space is minimized (rather than on the image planes). The measurement error distribution is constant with respect to depth (i.e. a cylindrical shape along depth). Solving (2.38) leads to degenerate solutions with cameras and points going to 0, and therefore it is typically solved with some constraint that the depths $\{\mathtt{P}_i\tilde{\mathbf{x}}_j\}$ have to be positive. Some recent theoretical study on suitable constraints is provided by Nasihatkon et al. [2015]. These are briefly revisited in Chapter 6.

Since (2.38) is nonlinear in both cameras and points, an affine projection-assumption is required. Given N tracked points of a rigid object in F frames or images, the affine transformation of the j-th point $\tilde{\mathbf{x}}_j = [x_j,\ y_j,\ z_j,\ 1]^\top$ into the the i-th frame image pixel $\mathbf{m}_{i,j} = [u_{i,j},\ v_{i,j}]^\top$ can be described by the equation

$$\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} = \begin{bmatrix} p_{i1,1} & p_{i1,2} & p_{i1,3} & p_{i1,4} \\ p_{i2,1} & p_{i2,2} & p_{i2,3} & p_{i2,4} \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ z_j \\ 1 \end{bmatrix} \Rightarrow \mathbf{m}_{i,j} = \mathtt{P}_i \tilde{\mathbf{x}}_j \tag{2.39}$$

Now adding all N points in the i-th frame yields

$$\begin{bmatrix} \mathbf{m}_{i,1} & \cdots & \mathbf{m}_{i,N} \end{bmatrix} = \mathtt{P}_i \begin{bmatrix} \tilde{\mathbf{x}}_1 & \cdots & \tilde{\mathbf{x}}_N \end{bmatrix}, \tag{2.40}$$

and adding all camera matrices for F frames produces

$$\begin{bmatrix} \mathbf{m}_{1,1} & \cdots & \mathbf{m}_{1,N} \\ \vdots & \ddots & \vdots \\ \mathbf{m}_{F,1} & \cdots & \mathbf{m}_{F,N} \end{bmatrix} = \begin{bmatrix} \mathtt{P}_1 \\ \mathtt{P}_2 \\ \vdots \\ \mathtt{P}_F \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_1 & \tilde{\mathbf{x}}_2 & \cdots & \tilde{\mathbf{x}}_N \end{bmatrix} \tag{2.41}$$

$$\Rightarrow \mathtt{M} =: \mathtt{P} \begin{bmatrix} \mathtt{X} \\ \mathbf{1}^\top \end{bmatrix} =: \mathtt{P}\tilde{\mathtt{X}} \tag{2.42}$$

where $\mathtt{M} \in \mathbb{R}^{2F \times N}$, $\mathtt{P} \in \mathbb{R}^{2F \times 4}$ and $\tilde{\mathtt{X}} \in \mathbb{R}^{4 \times N}$. Hence this is a structured rank-4 matrix factorization problem with the bottom row of $\tilde{\mathtt{X}}$ set to $\mathtt{1}^\top$.

It has also be derived by Zheng et al. [2012] that the weak perspective projection, which is an instance of affine camera in the calibrated setting, has a model linear in 3D points but is nonlinear in camera parameters (quaternions).

For SfM problems, the major sources of missing data are occlusions and failures in the tracking process. Hence, the aim is to recover these missing elements whilst maintaining a good match with existing observations.

Non-rigid structure-from-motion (SfM)

The non-rigid SfM model by Bregler et al. [2000] assumes that non-rigid objects can be depicted as a weighted sum of B basis shapes. The weights determine the proportion of each basis shape for each frame. This basis shape method was later discovered to be a dual representation of the smooth point trajectory method [Akhter et al., 2011], in which each non-rigid point is modelled as a weighted sum of discrete cosine basis terms.

Given N tracking points and F frames, the j-th point at the i-th frame $\mathbf{x}_{i,j} = [x_{i,j},\ y_{i,j},\ z_{i,j}]^\top$ is modelled as

$$\mathbf{x}_{i,j} = \sum_{k=1}^{B} \lambda_{i,k}\mathbf{s}_{k,j} = \begin{bmatrix} \lambda_{i,1}\mathtt{I} & \cdots & \lambda_{i,B}\mathtt{I} \end{bmatrix} \begin{bmatrix} \mathbf{s}_{1,j} \\ \vdots \\ \mathbf{s}_{B,j} \end{bmatrix} \tag{2.43}$$

where $\mathbf{s}_{k,j} \in \mathbb{R}^3$ is the position of the j-th point of basis shape k and $\lambda_{i,k} \in \mathbb{R}$ is the weighting factor for the k-th basis shape at frame i. Similar to Section 2.4.3, adding all N points yield

$$\begin{bmatrix} \mathbf{x}_{i,1} & \cdots & \mathbf{x}_{i,N} \end{bmatrix} = \begin{bmatrix} \lambda_{i,1}\mathtt{I} & \cdots & \lambda_{i,B}\mathtt{I} \end{bmatrix} \begin{bmatrix} \mathbf{s}_{1,1} & \cdots & \mathbf{s}_{1,N} \\ \vdots & \ddots & \vdots \\ \mathbf{s}_{B,1} & \cdots & \mathbf{s}_{B,N} \end{bmatrix}, \tag{2.44}$$

and again adding all F frames generates

$$\begin{bmatrix} \mathbf{x}_{1,1} & \cdots & \mathbf{x}_{1,N} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{F,1} & \cdots & \mathbf{x}_{F,N} \end{bmatrix} = \begin{bmatrix} \lambda_{1,1}\mathtt{I} & \cdots & \lambda_{1,B}\mathtt{I} \\ \vdots & \ddots & \vdots \\ \lambda_{F,1}\mathtt{I} & \cdots & \lambda_{F,B}\mathtt{I} \end{bmatrix} \begin{bmatrix} \mathbf{s}_{1,1} & \cdots & \mathbf{s}_{1,N} \\ \vdots & \ddots & \vdots \\ \mathbf{s}_{B,1} & \cdots & \mathbf{s}_{B,N} \end{bmatrix}. \tag{2.45}$$

The points in each image is projected using an affine transformation matrix, and this can be extended to all images using a concatenated block-diagonal matrix of affine camera matrices. This is similar to the approach used in Section 2.4.3 but also takes account of changes due to the non-rigidity of the object.

Defining $\tilde{\Lambda} := \operatorname{diag}([\lambda,\ \lambda,\ \lambda,\ 1])$ and $\tilde{\mathbf{s}} := [\mathbf{s};\ 1]$, projecting the point tracks to the corresponding image planes yields

$$\begin{aligned} \begin{bmatrix} \mathbf{m}_{1,1} & \cdots & \mathbf{m}_{1,N} \\ \vdots & \ddots & \vdots \\ \mathbf{m}_{F,1} & \cdots & \mathbf{m}_{F,N} \end{bmatrix} &= \begin{bmatrix} \mathtt{P}_1 & & \\ & \ddots & \\ & & \mathtt{P}_F \end{bmatrix} \begin{bmatrix} \tilde{\Lambda}_{1,1}\mathtt{I} & \cdots & \tilde{\Lambda}_{1,B}\mathtt{I} \\ \vdots & \ddots & \vdots \\ \tilde{\Lambda}_{F,1}\mathtt{I} & \cdots & \tilde{\Lambda}_{F,B}\mathtt{I} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{s}}_{1,1} & \cdots & \tilde{\mathbf{s}}_{1,N} \\ \vdots & \ddots & \vdots \\ \tilde{\mathbf{s}}_{B,1} & \cdots & \tilde{\mathbf{s}}_{B,N} \end{bmatrix} \\ &= \begin{bmatrix} \tilde{\Lambda}_{1,1}\mathtt{P}_1 & \cdots & \tilde{\Lambda}_1^B\mathtt{P}_1 \\ \vdots & \ddots & \vdots \\ \tilde{\Lambda}_{F,1}\mathtt{P}_F & \cdots & \tilde{\Lambda}_{F,B}\mathtt{P}_F \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{s}}_{1,1} & \cdots & \tilde{\mathbf{s}}_{1,N} & \\ \vdots & \ddots & \vdots & \\ \tilde{\mathbf{s}}_{B,1} & \tilde{\mathbf{s}}_{B,2} & \cdots & \tilde{\mathbf{s}}_{B,N} \end{bmatrix}. \end{aligned} \tag{2.46}$$

Above is a structured rank-$4B$ problem. However, if the measurements are re-aligned so that the translational components of cameras are removed, the rank of the problem becomes $3B$. Like in Section 2.4.3, missing data may arise from occlusions and tracking failures.

Surface recovery from light illumination

Below is an extension of the problem formulation by Buchanan [2004] based on the work of Julia et al. [2008].

The assumptions are that the surface is Lambertian and that the light source is infinite distance away from the surface in some direction, which are strong assumptions. In this case, the light intensity from diffuse illumination is the dot product of the flat surface normal $\mathbf{n} := [n_x,\ n_y,\ n_z]^\top \in \mathbb{R}^3$ and the light source direction $\mathbf{d} := [d_x,\ d_y,\ d_z]^\top \in \mathbb{R}^3$. This is added with the ambiance illumination factor a, which can vary across different pixels but does not depend on the light source direction. Hence, the light intensity of the j-th pixel at frame i, $m_{i,j}$, can be modelled as

$$m_{i,j} = \begin{bmatrix} d_{x,i} \\ d_{y,i} \\ d_{z,i} \end{bmatrix}^\top \begin{bmatrix} n_{x,j} \\ n_{y,j} \\ n_{z,j} \end{bmatrix} + a_j = \mathbf{d}_i{}^\top \mathbf{n}_j + a_j = \begin{bmatrix} \mathbf{d}_i \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{n}_j \\ a_j \end{bmatrix} \tag{2.47}$$

where $\mathbf{d}_i$ is the light direction at frame i, $\mathbf{n}_j$ is the surface normal of pixel j and a_j is the ambiance illumination at pixel j.

The model discussed here is a static diffuse object under varied illumination conditions. The direction of the light source differs for each of F frames. Given that each frame consists of N pixels, defining $\tilde{\mathbf{d}}_i := [\mathbf{d}_i;\ 1]$ and $\tilde{\mathbf{n}}^j := [\mathbf{n}_j;\ a_j]$ and extending to all N pixels yield

$$\begin{bmatrix} m_{i,1} & \cdots & m_{i,N} \end{bmatrix} = \tilde{\mathbf{d}}_i{}^\top \begin{bmatrix} \tilde{\mathbf{n}}_1 & \cdots & \tilde{\mathbf{n}}_N \end{bmatrix}, \tag{2.48}$$

and including all F frames generate

$$\begin{bmatrix} m_{1,1} & \cdots & m_{1,N} \\ \vdots & \ddots & \vdots \\ m_{F,1} & \cdots & m_{F,N} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{d}}_1{}^\top \\ \vdots \\ \tilde{\mathbf{d}}^{F\top} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{n}}_1 & \cdots & \tilde{\mathbf{n}}_N \end{bmatrix}. \tag{2.49}$$

This is a rank-4 matrix factorization problem, and a rank-3 problem if the effect of ambient illumination is neglected. Pixels which are shadows or heavily saturated with light are regarded as missing elements.

Recommender system

A recommender system usually comprises an incomplete matrix of user ratings. Taking the Netflix prize dataset [Bennett and Lanning, 2007] as an example, the rows represent the films, the columns represent the users and the values represent the film ratings.

The matrix may be modelled as a matrix factorization problem where one matrix depicts the film traits and the other matrix describes the user traits.

Missing data arises from limited number of user ratings. The missing proportion is comparatively high when compared with other type of datasets because of lack of user participation.

2.3.2 Analytic solution for the fully visible case

Assume there exists a full noisy measurement matrix $\mathtt{M} \in \mathbb{R}^{m \times n}$ with $m \leq n$, and the rank of the system r is known a priori with r being smaller than m and n. Then, the optimal solution $\mathtt{M}_r$ can be obtained by first taking the reduced landscape SVD of $\mathtt{M}$ to yield $\mathtt{U}\Sigma\mathtt{V}^\top$, setting all the singular values except the first r to zeros and then finally multiplying back the SVD products with updated singular values. In terms of equation,

$$\mathtt{M}_r = \mathtt{U}\mathrm{diag}(\sigma_1, \cdots, \sigma_r, 0, \cdots, 0)\mathtt{V}^\top, \tag{2.50}$$

where σ_k represents the k-th largest singular value of $\mathtt{M}$. Above result can be proved by using the Eckart–Young–Mirsky theorem [Markovsky, 2008].

Since SVD is a rank-revealing decomposition, it is also possible to determine the de-facto rank of the system by thresholding the singular values directly.

2.3.3 Problem statement in presence of missing data

In this section, the problem statement for low-rank matrix factorization with missing data is derived from a more fundamental problem of low-rank matrix completion.

As purportedly stated in the literature, for a partially-filled measurement matrix $\mathtt{M} \in \mathbb{R}^{m \times n}$ without noise, the problem is to find

$$\min \ \mathrm{rank}(\hat{\mathtt{M}}) \qquad \text{subject to} \qquad \hat{m}_i^j = \mathtt{m}_i^j \quad \forall\, (i, j) \in \Omega \tag{2.51}$$

where Ω denotes a set of visible entries. This is also known as the exact matrix completion [Candès and Recht, 2009].

When measurements are corrupted by noise, the equality constraint is replaced by an inequality constraint with the tolerance Δ such that (2.51) is transformed to

$$\min\ \operatorname{rank}(\hat{\mathtt{M}}) \qquad \text{subject to} \qquad \|\mathtt{W} \odot (\hat{\mathtt{M}} - \mathtt{M})\|_F^2 \leq \Delta^2 \tag{2.52}$$

where the (i, j)-th element of $\mathtt{W}$ is $0\ \forall\ (i, j) \notin \Omega$. Since this is a NP-hard problem, a convex relaxation alternative, also known as the nuclear-norm minimization, is widely used [Candès and Plan, 2010; Mazumder et al., 2010]. This formulation solves

$$\min_{\mathtt{X} \in \mathbb{R}^{m \times n}} \|\mathtt{X}\|_* \qquad \text{subject to} \qquad \|\mathtt{W} \odot (\mathtt{X} - \mathtt{M})\|_F^2 \leq \Delta^2 \tag{2.53}$$

where the nuclear norm $\|\cdot\|_*$ is given in Section A.1.6.

The Lagrangian relaxation of (2.53) solves

$$\min_{\mathtt{X}} \|\mathtt{X}\|_* + \lambda(\|\mathtt{W} \odot (\mathtt{X} - \mathtt{M})\|_F^2 - \Delta^2) \tag{2.54}$$

$$\equiv \min_{\mathtt{X}} \|\mathtt{X}\|_* + \lambda \|\mathtt{W} \odot (\mathtt{X} - \mathtt{M})\|_F^2 \tag{2.55}$$

where λ is a dual variable for the relaxed constraint. As noted by Candès and Plan [2010], one may apply a trust-region strategy mentioned in Section A.2.3 and obtain $\lambda(\Delta)$ such that $\|\mathtt{W} \odot (\mathtt{X} - \mathtt{M})\|_F^2 = \Delta^2$. In this thesis, the work is confined to a particular case where the dual variable is set, and therefore any strategy such as solving for the dual variable λ is beyond the scope of this dissertation.

A widely-used problem formulation in the fields of applied mathematics [Candès and Plan, 2010; Mazumder et al., 2010; Mishra et al., 2013] and computer vision [Cabral et al., 2013] is obtained by re-formulating (2.55) with $\mu := 1/\lambda$ such that

$$\min_{\mathtt{X}} \|\mathtt{W} \odot (\mathtt{X} - \mathtt{M})\|_F^2 + \frac{1}{\lambda}\|\mathtt{X}\|_* = \min_{\mathtt{X}} \|\mathtt{W} \odot (\mathtt{X} - \mathtt{M})\|_F^2 + \mu\|\mathtt{X}\|_*. \tag{2.56}$$

When the rank of the problem is known to be r, $\mathtt{X}$ is decomposed into two matrices $\mathtt{U} \in \mathbb{R}^{m \times r}$ and $\mathtt{V} \in \mathbb{R}^{n \times r}$ such that $\mathtt{X} = \mathtt{U}\mathtt{V}^\top$. Since the nuclear norm formulation does not directly enforce the rank of the system, which is against the model of rigid structure-from-motion that has rank 4, this work uses the alternative formulation mentioned in the work of Mazumder et al. [2010] and Cabral et al. [2013] which is

$$\|\mathtt{X}\|_* = \min_{\mathtt{X} = \mathtt{U}\mathtt{V}^\top} \frac{1}{2}(\|\mathtt{U}\|_F^2 + \|\mathtt{V}\|_F^2) \tag{2.57}$$

where the Frobenius norm $||\cdot||_F$ is given in Section A.1.6 (the full proof is illustrated by Mazumder et al. [2010]). However, as noted by Ma et al. [2011], the advantage of being able to differentiate the constraint comes at the cost of introducing non-convexity in which only a local minimizer is guaranteed. Also, r needs to be chosen appropriately, and that itself is another open question.

Summing up all the results, the problem Chapter 3 solves is

$$\min_{\mathtt{U},\mathtt{V}} ||\mathtt{W}\odot(\mathtt{U}\mathtt{V}^\top-\mathtt{M})||_F^2+\mu(||\mathtt{U}||_F^2+||\mathtt{V}||_F^2) \tag{2.58}$$

where the rank r and the hyperparameter μ is known.

Other variants

This subsection briefly summarizes some variants to (2.58) in the literature.

Changes to the regularizer Buchanan and Fitzgibbon [2005]'s Damped Newton algorithm uses the regularizer $\mu||\mathtt{U}||_F^2+\nu||\mathtt{V}||_F^2$ where μ and ν are both hyperparameters. This imposes an additional degree of freedom but whether this improves the algorithm performance has not been tested.

The algorithms proposed by Chen [2008, 2011]; Gotardo and Martinez [2011]; Okatani and Deguchi [2007]; Okatani et al. [2011] applies variable projection to the unregularized version of (2.58) by setting $\mu=0$.

Boumal and Absil [2011]'s RTRMC uses a custom regularizer $\mu||\mathtt{U}\mathtt{V}^\top||_{\not{\Omega}}^2$ where $\not{\Omega}$ denotes a set of missing entries. In other words, only the square norms of missing elements are penalized. By default, the algorithm uses zero regularization.

Some algorithms mentioned in the work of Mazumder et al. [2010], Ma et al. [2011], Candès and Plan [2010] and Mishra et al. [2013] use the nuclear norm penalty in (2.56) since it preserves the problem convexity and does not require the rank to be explicitly stated.

Changes to the norm The L1 norm-equivalent version of (2.56) is

$$\min_{\mathtt{X}} ||\mathtt{W}\odot(\mathtt{X}-\mathtt{M})||_1+\mu||\mathtt{X}||_*, \tag{2.59}$$

and thus the L1 norm-equivalent version of (2.58) is

$$\min_{\mathtt{U},\mathtt{V}} ||\mathtt{W}\odot(\mathtt{U}\mathtt{V}^\top-\mathtt{M})||_1+\mu(||\mathtt{U}||_F^2+||\mathtt{V}||_F^2). \tag{2.60}$$

Cabral et al. [2013]'s ALM solves (2.60) using the augmented Lagrangian method. [Shen et al., 2014]'s LMaFit solves the unregularized version of (2.59) using the augmented Lagrangian method. Eriksson and van den Hengel [2010] applies variable projection to the unregularized version of (2.59).

2.4 Related work

This section provides an overall review of the literature mostly relevant to this dissertation, namely matrix factorization, the variable projection (VarPro) method and factorization-based structure-from-motion approaches. Other related materials are reviewed in the forthcoming background section.

2.4.1 Fixed-rank matrix factorization with missing data

A branch of problems involving low-rank matrix factorization and missing data is visible in various computer vision and machine learning applications; bundle adjustment using affine cameras [Tomasi and Kanade, 1992], non-rigid structure-from-motion using basis shapes or point trajectory basis functions [Bregler et al., 2000], photometric stereo assuming ambient light and Lambertian surfaces [Belhumeur and Kriegman, 1996] and recommender systems [Bennett and Lanning, 2007] just to name a few.

Over the last two decades, the computer vision and machine learning community have seen a plethora of low-rank matrix factorization algorithms [Boumal and Absil, 2011; Buchanan and Fitzgibbon, 2005; Cabral et al., 2013; Chen, 2008; Del Bue et al., 2012; Kennedy et al., 2016; Okatani and Deguchi, 2007; Okatani et al., 2011; Vidal and Hartley, 2004] . Many of those algorithms were based on the space-efficient alternating least squares algorithm, with extremely poor convergence properties. Buchanan and Fitzgibbon [2005] introduced damping with a damped Newton algorithm, but again ignored Wiberg. Okatani and Deguchi [2007] reconsidered Wiberg, showing its strong convergence properties, and then Okatani et al. [2011] combined damping and Wiberg to boost convergence rates to near 100% on some previously-difficult problems. At the same time Gotardo and Martinez [2011]'s column space fitting (CSF) algorithm showed similar improvements.

Although the derivations of individual successful algorithms [Chen, 2008; Gotardo and Martinez, 2011; Okatani et al., 2011] may have been inspired by different sources, they are all related to the approach dubbed variable projection (VarPro) [Golub and Pereyra, 1973], in which the problem is solved by eliminating one matrix of larger dimension from the original objective function and using a second order optimizer such as Levenberg-Marquardt [Levenberg, 1944; Marquardt, 1963]. This observation forms the basis of the unifications made in Chapter 3.

Ruhe and Wedin [1980] considered VarPro in the Gauss-Newton scenario and presented three numbered algorithms abbreviated here as RW1, RW2, RW3 respectively. RW1 is the Gauss-Newton solver on the reduced objective (see Section 2.1). RW2, like Wiberg, approximates this Jacobian by eliminating a term. RW3 is alternation (ALS). Whether using the full Gauss-Newton matrix or its approximation is better is still debated [O'Leary and Rust, 2013], and has been explored previously in the context of matrix factorization [Chen, 2008; Gotardo and Martinez, 2011]. Chapter 3 extends these comparisons.

2.4.2 Variable projection (VarPro)

Variable projection (VarPro) was first proposed by Golub and Pereyra [1973] for separable nonlinear least squares problems, and was applied to principal components analysis (i.e. matrix factorization) by Wiberg [1976]. In short, it applies a second order optimizer such as Levenberg-Marquardt [Levenberg, 1944; Marquardt, 1963] on a reduced objective, which is obtained by optimally eliminating (or projecting out) one set of the unknowns. It is especially applicable to factorization problems, since in these problem instances one of the involved unknown factors can be eliminated in closed form.

Several work [Gotardo and Martinez, 2011; Okatani et al., 2011; O'Leary and Rust, 2013]) (and Chapter 3 and 4 of this dissertation), have experimentally demonstrated that VarPro applied on matrix factorization problems has much higher probabilities to reach a better optimum than using a second order method on the full (joint) problem (*joint optimization*, i.e. without eliminating one set of unknowns). Golub and Pereyra [2002] also noted that VarPro for the aforementioned type of problems not only leads to a reduction in the number of parameters but also decreases the number of iterations required for convergence. Yet the reasons for such difference in the algorithmic behaviours has not been carefully analyzed in the literature, with VarPro just being mostly used as a black box tool. O'Leary and Rust [2013] stated that the reduction in the problem dimension could be a reason for an improved efficiency and potential reduction in the number of local minima, allowing a better chance for global convergence [O'Leary and Rust, 2013].

On the contrary, Zollhöfer et al. [2014] argued that employing joint optimization may yield better results than using variable projection for lifted robust optimization problems (see Section A.2.4), which have model parameters (θ) and relaxed robust kernel weights ($\mathbf{w}$) as optimization variables [Zach, 2014]. The conclusion drawn in this work is based on a robust 2D line fitting example incorporating a truncated quadratic robust kernel, which assigns a fixed cost for a residual when its norm is greater than the inlier radius. In their VarPro implementation, the robust weights $(\mathbf{w})$ were optimally eliminated over the model parameters (θ), and thus VarPro's inferior performance arises from the optimal $\mathbf{w}$ given θ $(\mathbf{w}^*(\theta))$ having zero gradients $(d\mathbf{w}^*/d\theta \approx \mathbf{0})$ when the residual is large. Since bad initial model parameters lead to large residuals and therefore zero weights and gradients, VarPro falls into a local minimum from which it cannot recover. On the other hand, when jointly optimizing over θ and $\mathbf{w}$, Zollhöfer et al. [2014] set each robust weight to 1, allowing each residual an initial "opportunity" to participate as inliers. Instead, if one optimally eliminates the model parameters (i.e. obtain $\theta^*(\mathbf{w})$) when using VarPro and set the initial weights $\mathbf{w} = \mathbf{1}$, it can be shown that joint optimization and variable projection converge to the global optimum with similar probabilities. Alternatively, one may choose a robust kernel with non-zero gradients such as the Huber [Huber, 1964] kernel that may allow VarPro to escape initial bad local minimum. Designing a gold standard algorithm for solving robust optimization problems remains a challenge to this date.

Several papers pointed out some structural similarity between Joint optimization and VarPro. Ruhe and Wedin [1980] and Okatani et al. [2011] 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 [2012b] pointed out that VarPro performs additional minimization over the eliminated parameters. The Ceres solver [Agarwal et al., 2014], which is a widely-used nonlinear optimization library, also assumes the same. Chapter 4 shows 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.

VarPro/Wiberg was believed to be comparatively slow and memory consuming [Cabral et al., 2013; Chen, 2008], which is incorrect as will be shown in Chapter 4. With regards to scalable implementation of VarPro, RTRMC [Boumal and Absil, 2011] is in principle indirectly solving the VarPro-reduced problem, which is also what this work essentially proposes. However, their algorithm is implemented based on the assumption that the problem is regularized, which may be suitable for machine learning recommender systems and other random matrices but suffers from numerical instability when performed on SfM problems (as will be demonstrated in Chapter 3), where the regularizer is not a good idea because it essentially puts unrealistic priors on camera and point parameters (see Section 3). Chapter 4 provides a numerically stable and scalable VarPro algorithm which is tested and works well on matrix factorization problems of various sizes and densities.

2.4.3 Structure-from-motion

Since Snavely et al. [2006]'s work on Photo Tourism and [Agarwal et al., 2010]'s work on "Building Rome in a day", the community has seen a tremendous advancement in structure-from-motion [Schönberger and Frahm, 2016; Wu et al., 2011], with some pipelines now being able to handle a scene with nearly a hundered thousand images. To make the pipeline robust to outliers, these successful pipelines generally go through an iterative scheme of image registration (camera resectioning) using a PnP algorithm [Lepetit et al., 2008; Zheng et al., 2013], triangulation of newly added image points and finally bundle adjustment [Triggs et al., 2000] over the registered cameras and points at each iteration (see Section 2.2 for an illustration). Performing bundle adjustment regularly is crucial in improving the accuracy of the solution [Schönberger and Frahm, 2016] but this significantly increases runtime as the number of registered images and 3D points increases. Also, a fundamental shortcoming of an incremental (non-global) approach is that the scheme is biased towards the initial pair of views, possessing an intrinsic vulnerability to successive accumulation of drift and therefore potentially failing to close loops.

Global structure-from-motion

Above issues with incremental approaches inspired some research effort towards global structure-from-motion [Moulon et al., 2013; Olsson and Enqvist, 2011; Sweeney et al., 2015] in which all cameras and image points are considered simultaneously from the beginning using bundle adjustment [Triggs et al., 2000]. Unlike incremental approaches which merge point tracks as new images are registered, global approaches require relatively clean global point tracks (generated from pairwise feature matches or a tracker) prior to bundle adjustment. Since bundle adjustment is a non-trivial nonlinear problem, much of research attention has been focused on developing good initialization techniques.

One of the widely used initialization methods is called *motion averaging*, in which initial camera poses are estimated by "averaging" all relative rotations between adjacent images extracted from the epipolar geometry [Hartley and Zisserman, 2004]. This is typically solved in two stages, first solving for global rotations of cameras [Hartley et al., 2013] followed by translations [Wilson and Snavely, 2014]. Although the state-of-the-art rotation averaging techniques are becoming more robust and scalable [Chatterjee and Govindu, 2013], there is no gold standard rule for translation averaging, which has always been a more difficult problem due to potential errors arising from incorrect initialization of rotations.

Factorization-based structure-from-motion

Factorization-based methods refers to the problem of estimating camera poses and 3D structure by solving a factorization objective given the measurement matrix, where the columns representing individual point tracks. They are generally global methods working in a non-incremental fashion simultaneously integrating all image observations. It differs from the global SfM pipeline mentioned in the previous section that the minimized objective is not a reprojection error but a type of object space error, making the problem bilinear or trilinear depending on the problem formulation. One major benefit of employing a factorization-based method is that closed form solutions for camera poses and 3D points can be obtained if all projections are visible.

It was Tomasi and Kanade [1992] who first introduced the factorization approach in structure-from-motion. Their very first work showed that, given fully visible and outlier free image observations under the orthographic camera model, it is possible to recover both camera poses and 3D structure using the singular value decomposition (SVD). Their work was later generalized to other affine camera models such as the weak perspective and para perspective models. Sturm and Triggs [1996] proposed projective factorization methods in which projective depths are added as new variables and estimated in an alternating fashion, and Oliensis and Hartley [2007] presents a variant of the Sturm and Triggs' method guaranteed to converge. All of these algorithms require all observations to be visible, but they can be generalized to problems with missing data by replacing the SVD step in joint estimation of poses and 3D points with iterative matrix factorization with rank a priori given [Buchanan and Fitzgibbon, 2005; Gotardo and Martinez, 2011; Okatani et al., 2011].

Note that projective factorization by itself is an ill-posed problem with degenerate solutions, comprising a non-useful global optimum at which camera matrices and/or 3D points (in homogeneous parameterization) collapse to zero. Hence, the problem requires extra constraints to yield physically meaningful reconstructions, and these are usually added as penalty terms on the projective depths of visible projections. Some of the widely used constraints include constraining the sum of projective depths (or their squares) across each 3D point or each camera to a certain value (e.g. 1).

Recently, Nasihatkon et al. [2015] reviewed various options on how to choose these constraints in order to properly avoid degenerate solutions. In this work, the authors showed that some conditions proposed in the literature including the aforementioned constraints only lead to necessary but not sufficient conditions for guaranteeing a valid reconstruction. Furthermore, they proposed sufficient and necessary conditions on the projective depths to ensure non-degenerate solutions given fully visible point tracks without noise. These conditions are named "generalized projective reconstruction theorem" (GPRT), and one sufficient condition is to fix a set of projective depths for projections which form a step-like matrix along the measurement matrix (e.g. Fig. 6.9). They showed, on noise-free fully visible datasets, that a simple projective algorithm alternatingly minimizing between different sets of variables can yield suitable projective reconstructions by applying the aforementioned GPRT constraint. Instead of hard-fixing these selected depths, Magerand and Del Bue [2017] introduced softer penalty terms on each of them and demonstrated good experimental performance in presence of noise. The “GPRT constraint” applied in Chapter 6 of this dissertation follows this direction and attempts to extend to cases with missing observations by ensuring that the depth constraints on the steplike matrix only comprise visible observations. (This is partially because 3D points which are not visible are not guaranteed to be in front of the scene and therefore could have negative depths.) As will be demonstrated in Chapter 6, the seemingly strong theoretical insight of GPRT seems limited to ideal cases with fully visible image observations, which does not directly apply to practical sequences.

Several incremental methods for projective reconstruction do exist, which are designed to handle outliers in the image points as well as missing observations [Avidan and Shashua, 2001; Fitzgibbon and Zisserman, 1998; Heyden et al., 1999; Magerand and Del Bue, 2017; Mahamud and Hebert, 2000; Martinec and Pajdla, 2002]. For these methods, strong geometric constraints (reviewed by Hartley and Zisserman [2004] and in Chapter 2) can be used to determine sensible initial cameras and 3D structure. This initialization is subsequently used as a starting point for nonlinear least squares optimization (termed bundle adjustment) over all unknowns (see [Triggs et al., 2000] for a review).

Initialization-free structure-from-motion

An initialization-free approach refers to a method which does not require a careful initialization of camera poses and 3D points to yield a feasible reconstruction of the scene. These approaches attempt to find solution that most closely resembles an image observation matrix.

When there is no missing data, the L_2-norm solution of affine bundle adjustment (factorization) can be solved directly by applying the SVD from Section A.1.5 [Tomasi and Kanade, 1992] on the measurement matrix. For projective factorization, Hartley [1997] and Nasihatkon et al. [2015] present iterative alternation algorithms which minimize over projective depths given camera poses and 3D points and then over camera poses and 3D points given projective depths (via SVD). These can be broadly classified as initialization-free approaches but at least require full visibility pattern.

When missing data arises, which is almost always the case in practical situations, camera poses and 3D points can no longer be jointly estimated in closed form. For the case of affine bundle adjustment (or factorization), it is demonstrated in Chapter 3 that various matrix algorithms employing the variable projection (VarPro) method [Gotardo and Martinez, 2011; Okatani et al., 2011] benefit from wide basins of convergence of useful optima. Zheng et al. [2012] incorporated metric constraints to extend the scheme to weak perspective cameras, maintaining linearity in 3D points but making the problem nonlinear in quaternion parameters. Despite solving a separable nonlinear least squares problem, this approach on its own showed a dramatically decreased basin of convergence, and therefore Zheng et al. [2012] bootstrapped this method to affine factorization. Their work was tested only on a toy dinosaur example with inlier point tracks.

In the projective and metric camera settings, variable projection cannot be directly applied since the problem cannot eliminate either cameras or points in closed form. Strelow [2012b] proposed a nonlinear extension of VarPro, but used noisy ground truth (i.e. not arbitrary) camera matrices and points as initialization rather than arbitrary ones.

This dissertation is believed to be the first work to attempt bundle adjustment with perspective camera models from arbitrary initialization using the variable projection method. Unfortunately, the strategy devised here is still limited to the case where observed point tracks are sufficiently clean with a small number of feature mismatches, still requiring a robust point track generation step that can be computationally expensive. It is an open research question whether devising a fully initialization-free approach starting from raw matches is possible and practically feasible.

Improving initialization vs widening the convergence basins of useful optima A majority of research in structure-from-motion has focused on finding better ways for retrieving good initial estimates of camera poses and 3D points to solve bundle adjustment more easily. The main advantage of improving initialization is that solving the (nonlinear) bundle adjustment problem is greatly simplified. Also, some initialization techniques such as motion averaging [Chatterjee and Govindu, 2013; Olsson and Enqvist, 2011; Sweeney et al., 2015] are fast and have small runtime cost. However, these methods are usually deterministic, meaning that if initialization fails, it is difficult to improve the solution. Incremental structure-from-motion, which initializes each bundle adjustment from the previous bundle adjustment solution and is therefore less likely to fail, can take a long time as bundle adjustment has to be repeatedly carried out whenever a new image is registered.

On the other hand, initialization-free approaches potentially have the benefit of converging to a good minimum without requiring good initialization, which is the main reason for investigation in this thesis. If a run fails, it can be re-run from a different starting point. Unfortunately, the initialization-free approaches discussed above [Hartley, 1997; Nasihatkon et al., 2015; Tomasi and Kanade, 1992] and developed in this dissertation still require image observations to be provided in the form of point tracks and to comprise mostly inlier correspondences, meaning that feature matches still have to be geometrically verified. Also, these approaches are demonstrated to be slower than the state-of-the-art global structure-from-motion pipeline via motion averaging (e.g. Theia [Sweeney, 2017]) as they solve a large nonlinear least squares problem for a longer duration due to bad initialization.

Large-scale structure-from-motion

Large SfM sequences are usually sparse as only a fraction of total 3D points are usually visible in each image. Solving for the whole sequence can lead to optimization and numerical issues arising from the difficulty of solving data with banded and sparse observation patterns [Kennedy et al., 2016].

To overcome this problem, such datasets are usually divided into an overlapping set of smaller dense (or full) sequences each of which can then be solved by global, incremental, factorization-based or initialization-free approaches. (Note that using a SVD-based algorithm [Nasihatkon et al., 2015; Tomasi and Kanade, 1992] requires full visibility in each subsequence.) Global or factorization-based methods can segment these measurement blocks at the beginning since they already have the image observation matrix available from match or track filtering [Larsson et al., 2014]. For incremental approaches, the scene model can be separated into smaller models as the pipeline iterates if not enough geometrically verified matches exist between the previous camera views and the newly registered image [Schönberger and Frahm, 2016; Wu, 2013].

Once the local reconstructions of the scene are retrieved, the reconstructions can be stitched together to yield a global reconstruction. For example, if there are two subsequences, this can be achieved by fusing the overlapping camera views and resolving the scale ambiguity by considering the overlapping 3D points between those views. The stitched model can then be further refined using global bundle adjustment. Although stitching can parallelize runtime and decrease the problem difficulty, it can lead to model bias errors due to a particular way the measurements are segmented.

Chapter 3

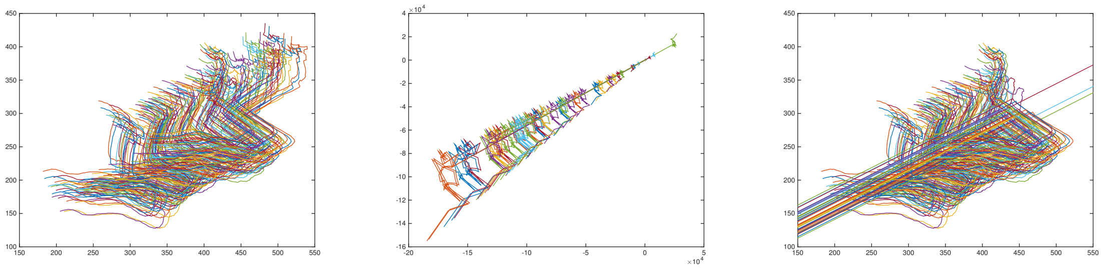
Secrets of fixed-rank matrix factorization with missing data

Many problems in computer vision and machine learning involve finding low-rank factors of a given $m \times n$ matrix $\mathtt{M}$, where some of the values are unobserved or weighted by another $m \times n$ matrix $\mathtt{W}$. Some of the basic applications are addressed in Section 2.3.1, one of which is bundle adjustment with affine or weak perspective camera models [Tomasi and Kanade, 1992]. As mentioned in Section 2.3.3, this type of problems typically involves minimization of the error function

$$f(\mathtt{U},\mathtt{V}) = \|\mathtt{W} \odot (\mathtt{U}\mathtt{V}^\top - \mathtt{M})\|_F^2 + \mu(\|\mathtt{U}\|_F^2 + \|\mathtt{V}\|_F^2) \tag{3.1}$$

over unknown matrices $\mathtt{U},\mathtt{V}$ each with rank r. The operator $\odot$ is Hadamard or elementwise product, and the norms are Frobenius. For simplicity, this chapter will investigate the the L^2 problem as stated above, but (3.1) can be robustified as shown in Section A.2.4. Additionally, although the question of choosing hyperparameters μ and r is of great interest, the focus here is on finding the best known optimum of (3.1) on a given dataset, assuming μ and r have been chosen. This is partially because this thesis concentrates on the application of matrix factorization for rigid structure-from-motion, which is modelled as a rank-4 system with the last column of $\mathtt{V}$ fixed.

When the number of non-zero entries of $\mathtt{W}$ is small, and particularly when entries are not missing uniformly, this is a difficult problem. In recent years, the revival of algorithms based on the variable projection (VarPro) method, also widely known as Wiberg, has yielded great advances in success rates, which is defined as the percentage of runs from arbitrary starting points that reach the global or the best observed optimum. Many benchmarks once thought to be difficult are now solved by almost all successful algorithms.



(a) Best solution (0.3228) (b) 2^{nd} best solution (0.3230) (c) 2^{nd} best, zoomed to image

Fig. 3.1 Reconstructions of point trajectories in the standard *Giraffe* (GIR in Table 3.6) sequence. These illustrate that a solution with function value just 0.12% above the optimum can have significantly worse extrapolation properties. Even when zooming in to eliminate gross outliers, it is clear that numerous tracks have been incorrectly reconstructed.

For many other benchmarks, however, even the best current algorithms succeed only occasionally, with only a small fraction of runs successful. However, success rates for any algorithm X are easily improved through the use of a meta-algorithm, called “RUSSO-X”, which simply runs algorithm X from different random starting points until the same best-so-far optimum value is seen twice. (RUSSO-X stands for “Restart X Until Seen Same Optimum”.) It will be later shown that this procedure dramatically increases success rate. On five of 13 benchmarks, it yields the best known optimum nearly 100% of the time. On other benchmarks, where there are other local optima with large basins of convergence, one might nevertheless hope to choose the best from several runs of RUSSO, and indeed simply repeating it five times brings the number of successes up to 11. The final two benchmarks are not solved by any algorithm tested.

The natural question then is how to select algorithm X. This must depend on both its success rate and its runtime. For example if X has a 5% success rate and runs in one second, RUSSO-X has an expected runtime of about 44 seconds. If Y has a 95% success rate, and takes an hour, RUSSO-Y will take two hours, so algorithm X is clearly preferable. Thus, the criterion for comparison of candidates must ultimately be mean runtime, and must be *wall-clock time*. Any proxy, such as iteration counts, may not reflect real-world practice. Even floating point operation counts (FLOPs) may not reflect real-world performance due to cache and memory effects, CPU pipelining, etc. Of course, having decided to measure time, each algorithm should be implemented as efficiently as possible and take advantage of all the speedup tricks used by the others. In order to achieve this, this chapter develops a unified derivation of several recent algorithms. The unification process has the benefit that their similarities and differences are clarified, but also the practical benefit that each algorithm is assembled from common components, for each of which the best performing implementation can be chosen. By doing so, it is possible to present new implementations of algorithms which are generally faster and more reliable than the publicly available codes. This work also proposes some hybrid algorithms which are better again.

This chapter's contributions therefore are:

+ A **unified theoretical derivation and analysis** of several existing matrix factorization algorithms.
+ **New re-implementations** of those algorithms which are faster and more reliable.
+ **New implementations** of standard **variable projection (VarPro)** algorithms which offer an order of magnitude performance improvement over the best previously available algorithms.
+ Assessment of the **meta-algorithm** RUSSO which improves success rates over all existing algorithms.
+ Some **new datasets**, including more challenging instances of existing datasets, and new problem classes.

Conversely, there are limitations of the current work: this work focuses on small to medium scale problems, under a million visible entries and tens of thousands of unknowns. Some of the conclusions drawn here could change for larger problems, but it is hoped that the unified derivations in this chapter will accelerate future research in the large-scale domain.

Also, the unification process is largely limited to the case with no regularization ($\mu = 0$), since many of the algorithms presented in the computer vision literature are limited to this case. This issue is discussed right before the start of the next section.

Not all existing algorithms are unified, although they are included in the experiments on e.g. augmented Lagrangian methods [Cabral et al., 2013; Del Bue et al., 2012]. Also, these experiments are run single threaded but it should be straightforward to write multithreaded code with the findings from this chapter.

Strelow's generalization of the Wiberg algorithm to nonseparable problems [Strelow, 2012a,b] is not explicitly treated here but will be addressed in Section 5.1.

Issues with the "global" optimum criterion

This section discusses some issues associated with the "global" (or best observed) optimum criterion, which is employed in this chapter to compare different algorithms.

Do we know the global optimum? None of the algorithms considered attempt to find global optima, yet it is of interest to determine the extent to which they do in fact do so. Of all the experiments run in the factorization literature on the same standard datasets, the best known optima have been reached many times independently on perhaps hundreds of millions of runs. Furthermore, in some datasets, these best-known optima are among the only small set of solutions that are reached multiple times.

Of course, even though no lower value is known, these might still not be global optima for these standard datasets. The global optimum for a given problem might have a tiny basin of convergence which no random initialization can ever hope to hit. But conversely, if some of the generally-agreed optima are just local optima with a very large basin of convergence, then it is impossible to do better with any currently known methods, so it remains valuable to know how to find such optima reliably and efficiently. Fig. 3.1 shows that it is certainly worth finding the best optimum; it is an example where a local optimum 0.12% above the best known objective provides a solution with qualitatively poor extrapolation.

Some datasets do have more than one strong local optimum, and for some applications it may be valuable to find more than one, on which there is considerable research [Martí, 2003], but this is beyond the scope here. In these experiments, a non-100% success rate for RUSSO gives a measure of the volume of the convergence basins of secondary optima. In these benchmarks, this was no larger than 70%, so about 3 to 5 runs of RUSSO on average does yield the best known optimum.

Quality of non-global optima If an optimization problem is formulated properly including necessary regularization, the global (or best) optimum should correspond to the best practical solution. Yet depending on the dataset, other non-global optima may also produce practically suitable solutions. This behaviour is sometimes observed in the stratified metric bundle adjustment strategy proposed in Chapter 6, in which 2nd or 3rd best optima from the first stage can yield equally good metric reconstruction in the final stage.

The issue is that this phenomenon is not consistently observed across all datasets. Some datasets have second minima that correspond to physically infeasible solutions. In order to maintain a consistent success criterion, a conservative compromise has been made here (same as in other literature [Chen, 2008; Gotardo and Martinez, 2011; Okatani et al., 2011]) to concentrate only on the best optimum of each dataset, potentially downweighting each algorithm's true performance in Fig. 3.7 and limiting the scope of the analysis.

Regularization issues

Regularization is typically applied to prevent overfitting of model parameters to given data. For instance, the term added in (3.1) is an instance of regularization to promote lower-dimensional models [Cabral et al., 2013; Mazumder et al., 2010]. For rigid and non-rigid structure-from-motion with ordered set of images (e.g. from a video), one can add a term to promote temporal smoothness of camera poses or image point trajectories [Dai et al., 2017]. This can be interpreted as specifying the prior distribution of estimated model parameters.

Unfortunately, structure-from-motion can often come across unordered set of images in which case the aforementioned temporal smoothness term cannot be applied. Moreover, Larsson et al. [2014] has demonstrated that applying the nuclear norm (the convex form of the regularizer in (3.1)) performs poorly when applied to solving the affine factorization problem. To intuitively see the potential cause of this regularization's failure, consider the L^2-norm affine bundle adjustment example formulated as a matrix factorization instance from Section 2.3.3. By noting the probabilistic interpretation of bundle adjustment in Section 2.2.4 (with known sensor noise σ), this problem can be formulated as

$$\arg\min_{\mathtt{P},\mathtt{X}} \left\| \mathtt{W} \odot \left(\mathtt{P} \begin{bmatrix} \mathtt{X} \\ \mathbf{1}^\top \end{bmatrix} - \mathtt{M} \right) \right\|_F^2 = \arg\max_{\mathtt{P},\mathtt{X}} \log p(\mathtt{M} | \mathtt{P}, \mathtt{X}, \sigma) \tag{3.2}$$

where $\mathtt{P} \in \mathbb{R}^{2F \times 4}$ is a row-stack of inhomogeneous affine camera matrices and $\mathtt{X} \in \mathbb{R}^{3 \times N}$ is a column-stack of 3D points. Now, adding the regularizer in (3.1) can be represented as

$$\begin{aligned} &\arg\min_{\mathtt{P},\mathtt{X}} \left\| \mathtt{W} \odot \left(\mathtt{P} \begin{bmatrix} \mathtt{X} \\ \mathbf{1}^\top \end{bmatrix} - \mathtt{M} \right) \right\|_F^2 + \frac{\mu}{2} \|\mathtt{P}\|_F^2 + \frac{\mu}{2} \|\mathtt{X}\|_F^2 \\ &= \arg\max_{\mathtt{P},\mathtt{X}} \log p(\mathtt{M} | \mathtt{P}, \mathtt{X}, \sigma) + \log p(\mathtt{P} | \mu) + \log p(\mathtt{X} | \mu) \\ &\propto \arg\max_{\mathtt{P},\mathtt{X}} \log p(\mathtt{P}, \mathtt{X} | \mathtt{M}, \sigma, \mu), \end{aligned} \tag{3.3}$$

meaning that it can be viewed as a joint maximum a posteriori (MAP) estimate of $\mathtt{P}$ and $\mathtt{X}$ with isotropic zero-mean Gaussian priors on cameras and points. First, it is evident that the means of $\mathtt{P}$ and $\mathtt{X}$ will depend on the mean of $\mathtt{M}$. In addition, even if $\mathtt{M}$ is normalized to have zero mean and unit variance, it is unlikely that the cameras and 3D points follow this distribution as these parameters are not randomly sampled from Gaussian. Since it is difficult to find suitable priors for camera and point parameters, bundle adjustment is almost always carried out without additional regularization. (The rank constraints on $\mathtt{P}$ and $\mathtt{X}$ still exist.)

Although it is possible to apply different regularization for each dataset depending on the nature of each problem, the newly added term can deteriorate the convergence basin of the best optimum if its residual is not linear in $\mathtt{U}$ and $\mathtt{V}$ (also described in Chapter 5). Since this dissertation is mostly interested in fixed-rank matrix factorization as a tool for affine factorization, this chapter mainly considers the un-regularized cases (except the ARU family of the algorithms in Table 3.4). Note that this can lead to overfitted solutions for some type of problems such as non-rigid structure-from-motion, and therefore the output should be refined by solving a properly formulated regularized problem.

3.1 Synthesis of current approaches

This section derives several existing approaches in a unified way, including analytic forms of derivatives and other relevant quantities useful for optimization.

3.1.1 Plug and play notation

This section presents a "plug and play" summary of these options, where text colouring and bracketing is used to indicate which components are active in each of several variants. For example, an unregularized Gauss-Newton algorithm is given by including only the terms in black. Levenberg-Marquardt (LM) is obtained by adding black and ⟨angle-bracketed pink⟩ terms. Damped Newton [Buchanan and Fitzgibbon, 2005] combines black with $[\text{bracketed orange}]_{FN}$ and ⟨angle-bracketed pink⟩. While this may appear unnecessarily garish, it allows us quickly to observe that LM is not just the same as adjusting the regularizer μ, because the latter has additional terms in the gradient. It also shows clearly the differences between Gauss-Newton and full Newton.

3.1.2 Common symbols

Some notations that will be used (also listed in Table 1): the space of symmetric $n \times n$ matrices is denoted $\mathbb{S}^n$. The total number of unknowns is $N = mr + nr$. This section depends on several vec and Kronecker product ($\otimes$) identities from [Magnus and Neudecker, 2007; Minka, 2000], which are repeated in Section A.1.2. Given the plethora of naming conventions in the literature, it was felt it did not reduce complexity to choose one of them, and instead mnemonic names are used: $\mathtt{M}$ is the Measurements; $\mathtt{W}$ is Weights; and $\mathtt{U}$ and $\mathtt{V}$ are a pair of unknowns. For the case of bundle adjustment with affine cameras, $\mathtt{U}$ would correspond to a stacked camera matrix (i.e. $\mathtt{P}$) and $\mathtt{V}$ would correspond to a column-stacked matrix of 3D points (i.e. $\mathtt{X}$). Finally, some quantities that are used throughout are the identity matrix $\mathtt{I}_r$, and the constant matrix $\mathtt{K}_{mr}$ defined as

$$\mathtt{K}_{mr}\,\mathrm{vec}(\mathtt{U}) = \mathrm{vec}(\mathtt{U}^\top), \tag{3.4}$$

and the residue matrix $\mathtt{R}$ and its transformation $\mathtt{Z}$ which are

$$\mathtt{R}(\mathtt{U},\mathtt{V}) := \mathtt{W}\odot(\mathtt{U}\mathtt{V}^\top - \mathtt{M}) \tag{3.5}$$

$$\mathtt{Z}(\mathtt{U},\mathtt{V}) := (\mathtt{W}\odot\mathtt{R})\otimes\mathtt{I}_r. \tag{3.6}$$

When the above occur "starred", e.g. $\mathtt{R}^*$, they are a function of $\mathtt{U}$ only i.e. $\mathtt{R}^*(\mathtt{U}) := \mathtt{R}(\mathtt{U},\mathtt{V}^*(\mathtt{U}))$, and when applied to vector arguments, e.g. $\mathtt{R}^*(\mathbf{u})$, they reshape the arguments. A tilde over a variable, such as $\tilde{\mathtt{U}},\tilde{\mathtt{W}}$ is mnemonic: the variable's definition is a sparse matrix whose non-zero entries are (weighted) copies of the entries of the tilded matrix.

3.1.3 Vectorization of the cost function

(3.1) is first written in a vectorized form. Noting that $\|\mathtt{X}\|_F^2 = \operatorname{trace}(\mathtt{X}^\top\mathtt{X}) = \|\operatorname{vec}(\mathtt{X})\|_2^2$, minimizing (3.1) is equivalent to minimizing

$$\|\varepsilon(\mathtt{U},\mathtt{V})\|_2^2 := \left\|\begin{matrix}\varepsilon_1(\mathtt{U},\mathtt{V})\\ \varepsilon_2(\mathtt{U})\\ \varepsilon_3(\mathtt{V})\end{matrix}\right\|_2^2 := \left\|\begin{matrix}\Pi\operatorname{vec}(\mathtt{W}\odot(\mathtt{U}\mathtt{V}^\top-\mathtt{M}))\\ \sqrt{\mu}\operatorname{vec}(\mathtt{U})\\ \sqrt{\mu}\operatorname{vec}(\mathtt{V}^\top)\end{matrix}\right\|_2^2$$

where Π is a $p\times mn$ projector matrix with p being the number of visible elements which eliminates known-zero entries from ε_1. By using some identities from Section A.1.2, define

$$\varepsilon_1(\mathtt{U},\mathtt{V}) = \Pi\operatorname{diag}(\operatorname{vec}\mathtt{W})\operatorname{vec}(\mathtt{U}\mathtt{V}^\top - \mathtt{M}) \tag{3.7}$$

$$:= \tilde{\mathtt{W}}\operatorname{vec}(\mathtt{U}\mathtt{V}^\top) - \tilde{\mathtt{W}}\mathbf{m} \tag{3.8}$$

where $\tilde{\mathtt{W}}$ is $\Pi\operatorname{diag}(\operatorname{vec}(\mathtt{W}))$ and $\mathbf{m}$ is $\operatorname{vec}(\mathtt{M})$. Defining $\mathbf{u} := \operatorname{vec}(\mathtt{U})$, $\mathbf{v} := \operatorname{vec}(\mathtt{V}^\top)$ and $\tilde{\mathbf{m}} := \tilde{\mathtt{W}}\mathbf{m}$ yields

$$\varepsilon_1(\mathtt{U},\mathtt{V}) = \tilde{\mathtt{W}}(\mathtt{I}_n\otimes\mathtt{U})\operatorname{vec}(\mathtt{V}^\top) - \tilde{\mathbf{m}} = \tilde{\mathtt{U}}\mathbf{v} - \tilde{\mathbf{m}}. \qquad // \ \tilde{\mathtt{U}} := \tilde{\mathtt{W}}(\mathtt{I}\otimes\mathtt{U}) \tag{3.9}$$

$$(= \tilde{\mathtt{V}}\mathbf{u} - \tilde{\mathbf{m}}.) \qquad // \ \tilde{\mathtt{V}} := \tilde{\mathtt{W}}(\mathtt{V}\otimes\mathtt{I}) \tag{3.10}$$

Again, recall that $\tilde{\mathtt{U}}$ is a rearrangement of the entries of $\mathtt{U}$, and hence of $\mathbf{u}$. The resulting vectorized cost function is

$$\left\|\begin{matrix}\varepsilon_1(\mathbf{u},\mathbf{v})\\ \varepsilon_2(\mathbf{u})\\ \varepsilon_3(\mathbf{v})\end{matrix}\right\|_2^2 = \left\|\begin{matrix}\tilde{\mathtt{U}}\mathbf{v}-\tilde{\mathbf{m}}\\ \sqrt{\mu}\mathbf{u}\\ \sqrt{\mu}\mathbf{v}\end{matrix}\right\|_2^2 = \left\|\begin{matrix}\tilde{\mathtt{V}}\mathbf{u}-\tilde{\mathbf{m}}\\ \sqrt{\mu}\mathbf{u}\\ \sqrt{\mu}\mathbf{v}\end{matrix}\right\|_2^2. \tag{3.11}$$

Table 3.1 Computations for joint optimization. Best viewed in colour or noting brackets.

Quantity	Gauss-Newton + [Full Newton]$_{FN}$ w/o {Regularization} w/o ⟨Damping⟩
Cost vector $\varepsilon \in \mathbb{R}^{p+\{N\}}$	$\varepsilon = \begin{bmatrix} \tilde{\mathtt{U}}\mathbf{v} - \mathbf{m} \\ \left\{ \begin{matrix} \mathbf{u} \\ \mathbf{v} \end{matrix} \right\} \end{bmatrix} = \begin{bmatrix} \hat{\mathtt{V}}\mathbf{u} - \mathbf{m} \\ \left\{ \begin{matrix} \mathbf{u} \\ \mathbf{v} \end{matrix} \right\} \end{bmatrix}$
Jacobian $\mathtt{J} \in \mathbb{R}^{p+\{N\} \times N}$	$\mathtt{J} = \begin{bmatrix} \hat{\mathtt{V}} & \tilde{\mathtt{U}} \\ \{\sqrt{\mu}\mathtt{I}_{mr}\} & \\ & \{\sqrt{\mu}\mathtt{I}_{nr}\} \end{bmatrix}$
Gradient $\mathbf{g} \in \mathbb{R}^{N}$	$\mathbf{g} = 2\begin{bmatrix} \hat{\mathtt{V}}^{\top}\varepsilon_1 + \{\mu\mathbf{u}\} \\ \tilde{\mathtt{U}}^{\top}\varepsilon_1 + \{\mu\mathbf{v}\} \end{bmatrix}$
Hessian $\mathtt{H} \in \mathbb{S}^{N}$	$\mathtt{H} = 2\begin{bmatrix} \hat{\mathtt{V}}^{\top}\hat{\mathtt{V}} + \{\mu\mathtt{I}_{mr}\} + \langle\lambda\mathtt{I}_{mr}\rangle & \hat{\mathtt{V}}^{\top}\tilde{\mathtt{U}} + [\mathtt{K}_{mr}^{\top}\mathtt{Z}]_{FN} \\ \tilde{\mathtt{U}}^{\top}\hat{\mathtt{V}} + [\mathtt{Z}^{\top}\mathtt{K}_{mr}]_{FN} & \tilde{\mathtt{U}}^{\top}\tilde{\mathtt{U}} + \{\mu\mathtt{I}_{nr}\} + \langle\lambda\mathtt{I}_{nr}\rangle \end{bmatrix}$

3.1.4 Block coordinate descent (ALS)

The approach first reviewed is block coordinate descent, partly in support of the VarPro derivation in Section 3.1.6. This approach was reviewed in the context of bivariate optimization strategy in Section 2.1. Since the cost function is bilinear, one can eliminate either $\mathbf{u}$ or $\mathbf{v}$, by taking the partial derivative with respect to $\mathtt{U}$ and $\mathtt{V}$ and setting them to 0. i.e.

$$\mathbf{v}^*(\mathbf{u}) = \arg\min_{\mathbf{v}} \left\| \begin{matrix} \tilde{\mathtt{U}}\mathbf{v} - \tilde{\mathbf{m}} \\ \sqrt{\mu}\mathbf{v} \end{matrix} \right\|_2^2 = (\tilde{\mathtt{U}}^{\top}\tilde{\mathtt{U}} + \mu\mathtt{I})^{-1}\tilde{\mathtt{U}}^{\top}\tilde{\mathbf{m}} = \tilde{\mathtt{U}}^{-\mu}\tilde{\mathbf{m}}, \tag{3.12}$$

where $\mathtt{X}^{-\mu} := (\mathtt{X}^{\top}\mathtt{X} + \mu\mathtt{I})^{-1}\mathtt{X}^{\top}$. A similar calculation produces $\mathbf{u}^*(\mathbf{v}) = (\tilde{\mathtt{V}}^{\top}\tilde{\mathtt{V}} + \mu\mathtt{I})^{-1}\tilde{\mathtt{V}}^{\top}\tilde{\mathbf{m}}$. Alternation updates one matrix at a time, so iteration k is:

$$\mathbf{v}_{k+1} = \tilde{\mathtt{U}}_k^{-\mu}\tilde{\mathbf{m}} \tag{3.13}$$

$$\mathbf{u}_{k+1} = \tilde{\mathtt{V}}_{k+1}^{-\mu}\tilde{\mathbf{m}}. \tag{3.14}$$

As previously reported in the literature [Buchanan and Fitzgibbon, 2005], this approach is known to be vulnerable to flatlining. Similar behaviour was observed on the Rosenbrock function in Section 2.1.4.

3.1.5 Joint optimization

As illustrated in Section 2.1, joint optimization updates all $(m+n)r$ parameters simultaneously by stacking into the vector $\mathbf{x} = [\mathbf{u};\mathbf{v}]$ and using second-order optimization. The iteration k

Table 3.2 Computations for variable projection. Best viewed in colour or noting brackets.

Quantity	Definition	Gauss-Newton w/o {Regularization} w/o ⟨Damping⟩
$\mathbf{v}^*(\mathbf{u})$ $\mathbf{v}^* \in \mathbb{R}^{nr}$	$\mathbf{v}^* := \arg\min_{\mathbf{v}} f(\mathbf{u},\mathbf{v})$	$\mathbf{v}^* = (\tilde{\mathtt{U}}^\top \tilde{\mathtt{U}} + \{\mu \mathtt{I}_{nr}\})^{-1} \tilde{\mathtt{U}}^\top \tilde{\mathbf{m}}$ ($\tilde{\mathbf{m}} \in \mathbb{R}^p$ consists of non-zero elements of $\tilde{\mathtt{W}}\mathbf{m}$.)
Derivative of $\mathbf{v}^*$ $\frac{d\mathbf{v}^*}{d\mathbf{u}} \in \mathbb{R}^{mr \times nr}$	$\frac{d\mathbf{v}^*}{d\mathbf{u}} := \frac{d\,\mathrm{vec}(\mathtt{V}^{*\top})}{d\,\mathrm{vec}(\mathtt{U})}$	$\frac{d\mathbf{v}^*}{d\mathbf{u}} = -(\tilde{\mathtt{U}}^\top \tilde{\mathtt{U}} + \{\mu \mathtt{I}_{nr}\})^{-1} (\tilde{\mathtt{U}}^\top \hat{\mathtt{V}}^* + \mathtt{Z}^{*\top} \mathtt{K}_{mr})$ $(\mathtt{Z}^* := (\mathtt{W} \odot \mathtt{R}^*) \otimes \mathtt{I}_r)$
Cost vector $\varepsilon^* \in \mathbb{R}^{p+\{N\}}$	$\varepsilon^* := \begin{bmatrix} \varepsilon_1^* \\ \left\{ \begin{matrix} \varepsilon_2^* \\ \varepsilon_3^* \end{matrix} \right\} \end{bmatrix}$	$\varepsilon^* = \begin{bmatrix} \tilde{\mathtt{U}}\mathbf{v}^* - \mathbf{m} \\ \left\{ \begin{matrix} \mathbf{u} \\ \mathbf{v}^* \end{matrix} \right\} \end{bmatrix}$
Jacobian $\mathtt{J}_{\mathbf{u}} \in \mathbb{R}^{p+\{N\} \times mr}$	$\mathtt{J}_{\mathbf{u}} := \begin{bmatrix} \frac{d\varepsilon_1^*}{d\mathbf{u}} \\ \left\{ \begin{matrix} \frac{d\varepsilon_2^*}{d\mathbf{u}} \\ \frac{d\varepsilon_3^*}{d\mathbf{u}} \end{matrix} \right\} \end{bmatrix}$	$\mathtt{J}_{\mathbf{u}} = \begin{bmatrix} \hat{\mathtt{V}}^* + \tilde{\mathtt{U}} \frac{d\mathbf{v}^*}{d\mathbf{u}} \\ \left\{ \begin{matrix} \sqrt{\mu}\mathtt{I}_{mr} \\ \sqrt{\mu}\frac{d\mathbf{v}^*}{d\mathbf{u}} \end{matrix} \right\} \end{bmatrix}$
Cost function $f^* \in \mathbb{R}$	$f^* := \lVert \varepsilon^* \rVert_2^2$	$f^* = \lVert \mathtt{W} \odot (\mathtt{U}\mathtt{V}^{*\top} - \mathtt{M}) \rVert_F + \{\mu \lVert \mathtt{U} \rVert_F + \mu \lVert \mathtt{V}^* \rVert_F\}$
Gradient $\mathbf{g}^* \in \mathbb{R}^{mr}$	$\mathbf{g}^* := \frac{df^*}{d\mathbf{u}}$	$\frac{1}{2}\mathbf{g}^* = (\hat{\mathtt{V}}^* + \tilde{\mathtt{U}}\frac{d\mathbf{v}^*}{d\mathbf{u}})^\top \varepsilon_1^* + \left\{ \mu \left(\frac{d\mathbf{v}^*}{d\mathbf{u}}\right)^\top \mathbf{v}^* + \mu \mathbf{u} \right\}$
Gauss-Newton $\mathtt{H}_{GN}^* \in \mathbb{S}^{mr}$	$\mathtt{H}_{GN}^* := \frac{d\mathbf{g}^*}{d\mathbf{u}} + \langle \lambda \mathtt{I} \rangle$	$\frac{1}{2}\mathtt{H}_{GN}^* = (\hat{\mathtt{V}}^* + \tilde{\mathtt{U}}\frac{d\mathbf{v}^*}{d\mathbf{u}})^\top (\hat{\mathtt{V}}^* + \tilde{\mathtt{U}}\frac{d\mathbf{v}^*}{d\mathbf{u}})$ $+ \left\{ \mu \left(\frac{d\mathbf{v}^*}{d\mathbf{u}}\right)^\top \left(\frac{d\mathbf{v}^*}{d\mathbf{u}}\right) + \mu \mathtt{I}_{mr} \right\} + \langle \lambda \mathtt{I}_{mr} \rangle$

update is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathtt{H}_k)^{-1}\mathbf{g} \tag{3.15}$$

where the matrix $\mathtt{H} \in \mathbb{S}^N$ is the Hessian $\nabla\nabla^\top f(\mathbf{x}_k)$ or its approximation, and $\mathbf{g}$ is the gradient $\nabla f(\mathbf{x}_k)$. Different choices lead to different, but closely related, algorithms.

All the derivations are included in Chapter B but a summary of derivation results can be found in Table 3.1.

Table 3.3 Unified analysis of algorithms in the literature and new proposals. The bottom row is the Hessian approximation whose terms are switched by the choice of algorithm.

Algorithm	Framework	Retraction
ALS	RW3 (ALS)	None
PF	RW3 (ALS)	`orth`
LM-S	Newton + ⟨Damping⟩	`orth`
LM-S$_{GN}$	RW1 (GN) + ⟨Damping⟩ (DRW1 equiv.)	`orth`
LM-M	Reduced$_r$ Newton + ⟨Damping⟩	`orth`
LM-M$_{GN}$	Reduced$_r$ RW1 (GN) + ⟨Damping⟩	`orth`
Wiberg	RW2 (Approx. GN)	None
DW	RW2 (Approx. GN) + ⟨Projection const.⟩$_P$ + ⟨Damping⟩	None
CSF	RW2 (Approx. GN) + ⟨Damping⟩ (DRW2 equiv.)	q-factor
RTRMC	Projected$_p$ Newton + {Regularization} + ⟨Trust Region⟩	q-factor
LM-S$_{RW2}$	RW2 (Approx. GN) + ⟨Damping⟩ (DRW2 equiv.)	q-factor
LM-M$_{RW2}$	Reduced$_r$ RW2 (Approx. GN) + ⟨Damping⟩	q-factor
DRW1	RW1 (GN) + ⟨Damping⟩	q-factor
DRW1P	RW1 (GN) + ⟨Projection const.⟩$_P$ + ⟨Damping⟩	q-factor
DRW2	RW2 (Approx. GN) + ⟨Damping⟩	q-factor
DRW2P	RW2 (Approx. GN) + ⟨Projection const.⟩$_P$ + ⟨Damping⟩	q-factor
$\frac{1}{2}\mathtt{H}^* = \mathtt{P}_r{}^\top \big(\hat{\mathtt{V}}^{*\top}(\mathtt{I}_p - [\tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger]_{RW2})\hat{\mathtt{V}}^* + [\mathtt{K}_{mr}^\top \mathtt{Z}^*(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}]_{RW1} \times [-1]_{FN}$ $+[\mathtt{K}_{mr}^\top \mathtt{Z}^*\tilde{\mathtt{U}}^\dagger\hat{\mathtt{V}}^*\mathtt{P}_p + \mathtt{P}_p\hat{\mathtt{V}}^{*\top}\tilde{\mathtt{U}}^{\dagger\top}\mathtt{Z}^{*\top}\mathtt{K}_{mr}]_{FN} + \langle\alpha\mathtt{I}_r \otimes \mathtt{U}\mathtt{U}^\top\rangle_P + \langle\lambda\mathtt{I}_{mr}\rangle\big)\mathtt{P}_r$		

3.1.6 Variable projection (VarPro)

The key derivation in applying VarPro is to compute the derivative of $\mathbf{v}^*(\mathbf{u})$, from (3.12). Substituting (3.12) to the original objective yields

$$\varepsilon_1^*(\mathbf{u}, \mathbf{v}^*(\mathbf{u})) = \tilde{\mathtt{U}}\mathbf{v}^* - \tilde{\mathbf{m}} = -(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{-\mu})\tilde{\mathbf{m}}. \tag{3.16}$$

Taking $\partial[\mathbf{v}^*]$ and applying (A.25) yields (with the full derivation in Section B.2.1)

$$\frac{d\mathbf{v}^*}{d\mathbf{u}} = -(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}} + \mu\mathtt{I}_{nr})^{-1}(\tilde{\mathtt{U}}^\top\tilde{\mathtt{V}}^* + \mathtt{Z}^{*\top}\mathtt{K}_{mr}) \tag{3.17}$$

where $\mathtt{Z}^*$ is from (3.6). A summary of derivations is provided in Table 3.2.

Since we are mostly interested in the unregularized case ($\mu = 0$) as regularization hurts the rigid structure-from-motion (SfM) performance (see (3.3)), the rest of this section will treat only the unregularized case. With the exception of the RTRMC algorithm of Boumal and Absil [2011], which applies regularized VarPro using the exact Hessian matrix, the rest of the considered algorithms use $\mu = 0$ and still obtain good optima with sound extrapolation. In this

ALS	VH_PF [8]	TW_WB [11]	TO_DW [9]	DRW1	DRW1P	DRW2	DRW2P	PG_CSF [7]	CH_LM_S [5]	CH_LM_S_GN [5]	CH_LM_S_RW2	CH_LM_M [5]	CH_LM_M_GN [5]	CH_LM_M_RW2	NB_RTRMC [2]
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		2	2	2	2	2	2	2	2	2	2	2	2	2	2
				3	3					3			3		
									4			4			4
					(5)		(5)								(5)
												6	6	6	
					(7)		(7)					7	7	7	(7)
					(8)		(8)					8	8	8	(8)
			9		9		9								
		10	10	10	10	10	10	10	10	10	10	10	10	10	10
11	11	11													
			12	12	12	12	12	12	12	12	12	12	12	12	12
			13	13	13	13	13	13	13	13	13	13	13	13	13
		14	14	14	14	14	14	14	14	14	14	14	14	14	14
15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
	16				16										
17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17
			18	18	18	18	18	18	18	18	18	18	18	18	18

```
inputs: $\mathtt{M}, \mathtt{W}, \mathtt{r}, \mathtt{U}, \lambda_0$
$\lambda \leftarrow \lambda_0$
$\widetilde{\mathtt{W}} \leftarrow \mathrm{diag}\big(\mathrm{vec}(\mathtt{W})\big)$ with zero-rows removed.
$\tilde{\mathbf{m}} \leftarrow \widetilde{\mathtt{W}}\,\mathrm{vec}(\mathtt{M})$
repeat
      $\mathbf{g} \leftarrow \mathtt{V}^*(\mathtt{U})^\top \mathrm{vec}(\mathtt{U}\mathtt{V}^*(\mathtt{U})^\top - \mathtt{M})$
  1:  $\mathtt{H} \leftarrow \mathtt{V}^{*\top}\mathtt{V}^*$
  2:  $\mathtt{H} \leftarrow \mathtt{H} - \widetilde{\mathtt{V}}^{*\top}\widetilde{\mathtt{U}}\widetilde{\mathtt{U}}^\dagger\widetilde{\mathtt{V}}^*$
  3:  $\mathtt{H} \leftarrow \mathtt{H} + \mathtt{K}_{mr}^\top \mathtt{Z}^*\big(\widetilde{\mathtt{U}}^\top\widetilde{\mathtt{U}}\big)^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}$
  4:  $\mathtt{H} \leftarrow \mathtt{H} - \mathtt{K}_{mr}^\top \mathtt{Z}^*\big(\widetilde{\mathtt{U}}^\top\widetilde{\mathtt{U}}\big)^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr} + \mathtt{K}_{mr}^\top\mathtt{Z}^*\widetilde{\mathtt{U}}^\dagger\widetilde{\mathtt{V}}^* + \widetilde{\mathtt{V}}^{*\top}\widetilde{\mathtt{U}}^{\dagger\top}\mathtt{Z}^{*\top}\mathtt{K}_{mr}$
  5:  $\mathtt{P} \leftarrow \mathtt{I} \otimes (\mathtt{I} - \mathtt{U}\mathtt{U}^\top) \in \mathbb{R}^{mr\times mr}$ // (5) is also a no-operation if 7 and 8 are.
  6:  $\mathtt{P} \leftarrow \mathtt{I} \otimes \mathtt{U}_\perp^\top \in \mathbb{R}^{(m-r)r\times mr}$
  7:  $\mathbf{g} \leftarrow \mathtt{P}\mathbf{g}$      // (7) is a no-operation since $\mathbf{g} = \mathtt{P}\mathbf{g}$.
  8:  $\mathtt{H} \leftarrow \mathtt{P}\mathtt{H}\mathtt{P}^\top$  // (8) is a no-operation since $\mathtt{H} = \mathtt{P}\mathtt{H}$.
  9:  $\mathtt{H} \leftarrow \mathtt{H} + \mathtt{I} \otimes \mathtt{U}\mathtt{U}^\top$ // relaxed constraint to promote $\mathtt{U}^\top\Delta\mathtt{U} = 0$.
 10:  repeat
 11:    $\Delta\mathtt{U} \leftarrow \mathrm{unvec}(\mathtt{H}^{-1}\mathbf{g})$
 12:    $\Delta\mathtt{U} \leftarrow \mathrm{unvec}\big((\mathtt{H} + \lambda\mathtt{I})^{-1}\mathbf{g}\big)$
 13:    $\lambda \leftarrow \lambda * 10$
 14:  until $f\big(\mathtt{U} + \Delta\mathtt{U}, \mathtt{V}^*(\mathtt{U} + \Delta\mathtt{U})\big) < f(\mathtt{U}, \mathtt{V}^*(\mathtt{U}))$
 15:  $\mathtt{U} \leftarrow \mathtt{U} + \Delta\mathtt{U}$
 16:  $\mathtt{U} \leftarrow qf(\mathtt{U})$ // [U,~]=qr(U,0) in MATLAB.
 17:  $\mathtt{V} \leftarrow \mathrm{unvec}\big(\widetilde{\mathtt{U}}^\dagger\tilde{\mathbf{m}}\big)$
 18:  $\lambda \leftarrow \lambda/100$
until convergence
outputs: $\mathtt{U}, \mathtt{V}$
```

- $\mathtt{K}_{mr}\mathrm{vec}(\mathtt{U}) = \mathrm{vec}(\mathtt{U}^\top)$
- $\widetilde{\mathtt{U}} \coloneqq \widetilde{\mathtt{W}}(\mathtt{I} \otimes \mathtt{U})$
- $\mathtt{V}^*(\mathtt{U}) = \mathrm{unvec}\big(\widetilde{\mathtt{U}}^\dagger\tilde{\mathbf{m}}\big)$
- $\widetilde{\mathtt{V}}^*(\mathtt{U}) \coloneqq \widetilde{\mathtt{W}}(\mathtt{V}^*(\mathtt{U}) \otimes \mathtt{I})$
- $\mathtt{Z}^*(\mathtt{U}) \coloneqq \Big(\mathtt{W} \odot \mathtt{W} \odot (\mathtt{M} - \mathtt{U}\mathtt{V}^{*\top}(\mathtt{U})\Big) \otimes \mathtt{I}$

Fig. 3.2 Pseudocodes for matrix factorization algorithms with high success rates. These algorithms are unified and shown to employ the variable projection method from Section 2.1.

case, expanding (3.17) and colourizing yields

$$\frac{d\mathbf{v}^*}{d\mathbf{u}} = -[\widetilde{\mathtt{U}}^\dagger\widetilde{\mathtt{V}}^*]_{RW2} - [(\widetilde{\mathtt{U}}^\top\widetilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}]_{RW1} \tag{3.18}$$

where the $[\text{green}]_{RW2}$ term is the approximation used in RW2 and Damped Wiberg, while $[\text{blue}]_{RW1}$ is added for RW1 or Chen's LM-S$_{GN}$. This adds corresponding switches to the Hessian approximation as shown in Table 3.3. Note that for this work's modification of Chen's algorithms the `orth` operator has been replaced by retraction using the q-factor (i.e. $\mathtt{U} = \mathtt{qorth}(\mathtt{U})$ is $[\mathtt{U}, \sim] = \mathtt{qr}(\mathtt{U}, 0)$ in MATLAB).

3.1.7 Manifold optimization

It is clear that the objective function for variable projection has gauge freedom [Boumal and Absil, 2011; Chen, 2008] which means that $f^*(\mathtt{U}) = f^*(\mathtt{UA})$ for any orthonormal $r \times r$ matrix $\mathtt{A}$. If $\mu = 0$ then this is true for any invertible $\mathtt{A}$. This is equivalent to solutions lying on the Grassmann manifold [Boumal and Absil, 2011; Chen, 2008], which was briefly illustrated in Section A.2.5. It is natural to ask if poor convergence on the matrix factorization problem could be caused by this gauge freedom, so methods to address it have been proposed. The book on

```
CE_LM  AB_DN
              inputs: $\mathtt{M}, \mathtt{W}, \mathtt{r}, \mathtt{U}, \mathtt{V}, \mu, \lambda_0$
              $\lambda \leftarrow \lambda_0$
              $\widetilde{\mathtt{W}} \leftarrow \mathrm{diag}(\mathrm{vec}(\mathtt{W}))$ with zero-rows removed.
              $\widetilde{\mathbf{m}} \leftarrow \widetilde{\mathtt{W}}\,\mathrm{vec}(\mathtt{M})$
              repeat
                  $\mathbf{g} \leftarrow \begin{bmatrix} \widetilde{\mathtt{V}}^\top \mathrm{vec}(\mathtt{U}\mathtt{V}^\top - \mathtt{M}) + \mu \mathrm{vec}(\mathtt{U}) \\ \widetilde{\mathtt{U}}^\top \mathrm{vec}(\mathtt{U}\mathtt{V}^\top - \mathtt{M}) + \mu \mathrm{vec}(\mathtt{V}^\top) \end{bmatrix}$
  1      1    1:  $\mathtt{H} \leftarrow \begin{bmatrix} \widetilde{\mathtt{V}}^\top \widetilde{\mathtt{V}} + \mu \mathtt{I} & \widetilde{\mathtt{V}}^\top \widetilde{\mathtt{U}} \\ \widetilde{\mathtt{U}}^\top \widetilde{\mathtt{V}} & \widetilde{\mathtt{U}}^\top \widetilde{\mathtt{U}} + \mu \mathtt{I} \end{bmatrix}$
         2    2:  $\mathtt{H} \leftarrow \mathtt{H} + \begin{bmatrix} 0 & \mathtt{K}_{mr}^\top Z \\ \mathtt{Z}^\top \mathtt{K}_{mr} & 0 \end{bmatrix}$
                  repeat
                      $\Delta\mathbf{z} \leftarrow (\mathtt{H} + \lambda \mathtt{I})^{-1}\mathbf{g}$
                      Retrieve $\Delta\mathtt{U}$ and $\Delta\mathtt{V}$ from $\Delta\mathbf{z}$.
                      $\lambda \leftarrow \lambda * 10$
                  until $f(\mathtt{U} + \Delta\mathtt{U}, \mathtt{V} + \Delta\mathtt{V}) < f(\mathtt{U}, \mathtt{V})$
                  $\mathtt{U} \leftarrow \mathtt{U} + \Delta\mathtt{U}$
                  $\mathtt{V} \leftarrow \mathtt{V} + \Delta\mathtt{V}$
                  $\lambda \leftarrow \lambda / 100$
              until convergence
              outputs: $\mathtt{U}, \mathtt{V}$
```

- $\mathtt{K}_{mr}\mathrm{vec}(\mathtt{U}) = \mathrm{vec}(\mathtt{U}^\top)$
- $\widetilde{\mathtt{U}} \coloneqq \widetilde{\mathtt{W}}(\mathtt{I} \otimes \mathtt{U})$
- $\widetilde{\mathtt{V}} \coloneqq \widetilde{\mathtt{W}}(\mathtt{V} \otimes \mathtt{I})$
- $\mathtt{Z} \coloneqq (\mathtt{W} \odot \mathtt{W} \odot (\mathtt{M} - \mathtt{U}\mathtt{V}^\top) \otimes \mathtt{I}$

Fig. 3.3 Pseudocodes for matrix factorization algorithms employing joint optimization.

Riemannian manifold optimization by Absil et al. [2008] suggests that instead of a standard second-order VarPro update

$$\Delta\mathbf{u} = -\mathtt{H}^{*-1}\mathbf{g}^* = \arg\min_{\delta} \mathbf{g}^{*\top}\delta + \frac{1}{2}\delta^\top \mathtt{H}^*\delta, \tag{3.19}$$

the update should be the solution to a projected subproblem with projected gradient $\mathbf{g}_p^*$ and Hessian $\mathtt{H}_p^*$

$$\Delta\mathbf{u} = \arg\min_{\delta \perp \mathbf{u}} \mathbf{g}_p^{*\top}\delta + \frac{1}{2}\delta^\top \mathtt{H}_p^*\delta \tag{3.20}$$

where $\delta \perp \mathbf{u}$ is the linear constraint $\mathtt{U}^\top \mathrm{unvec}(\delta) = 0$. This constrains the update to be made on the subspace tangential to the current $\mathtt{U}$. It turns out that the projected gradient and the Gauss-Newton matrix are the same as the originals. When $\Delta\mathbf{u}$ is applied, there is also retraction onto the manifold, so $\mathtt{U} = \mathtt{qorth}(\mathrm{unvec}(\mathbf{u} + \Delta\mathbf{u}))$.

Chen [2008] introduced the algorithm LM_M along with LM_S, which introduces Grassmann manifold projection using the approach of Manton et al. [2003]. This involves solving a reduced subproblem, in which the dimension of the update is reduced from $\mathbb{R}^{mr}$ to $\mathbb{R}^{(m-r)r}$

Table 3.4 Algorithms and their corresponding programming extensions in MATLAB.

Extension	Algorithm	Code implementation
ALS	Block coordinate descent	New MATLAB code
DW	Damped Wiberg	
DRW1	DRW1	
DRW1P	DRW1P	
DRW2	DRW2	
DRW2P	DRW2P	
PG_CSF	CSF	Author's original MATLAB code
TO_DW	Damped Wiberg	Author's original MATLAB ocde
NB_RTRMC	RTRMC	Author's original MATLAB code
CH_LM_S	LM-S	Modified MATLAB code
CH_LM_S_GN	LM-S_{GN}	Modified MATLAB code
CH_LM_S_RW2	LM-S_{RW2}	Modified MATLAB code
CH_LM_M	LM-M	Modified MATLAB code
CH_LM_M_GN	LM-M_{GN}	Modified MATLAB code
CH_LM_M_RW2	LM-M_{RW2}	Modified MATLAB code
CE_LM	CE_LM	New code using the Ceres solver
CE_LMI	CE_LMI	
CE_ALM	CE_ALM	
CE_ALMI	CE_ALMI	
CE_ARULM	CE_ARULM	
CE_ARULMI	CE_ARULMI	

which is orthogonal to current $\mathtt{U}$, minimizing

$$\Delta \mathbf{u} = \mathtt{P}_r{}^\top \left(\arg\min_{\delta} \mathbf{g}_r^{*\top} \delta + \delta^\top \mathtt{H}_r^* \delta \right) \tag{3.21}$$

where $\mathtt{P}_r \in \mathbb{R}^{(m-r)r \times mr}$ is the tangent space projection matrix and $\mathtt{H}_r^* \in \mathbb{S}^{(m-r)r}$ is the reduced Hessian projected to the tangent space of the manifold. These expressions are written in Section B.3. A similar connection was also recently made in the field of control theory [Usevich and Markovsky, 2014].

The hard constraint $\mathtt{U}^\top \text{unvec}(\delta) = 0$ may also be relaxed by adding a penalty term as follows:

$$\mathbf{g}_p^{*\top} \delta + \delta^\top \mathtt{H}_p^* \delta + \langle \alpha \| \mathtt{U}^\top \text{unvec}(\delta) \|_2^2 \rangle_P. \tag{3.22}$$

This in fact introduces the same term as the one introduced by Okatani et al. [2011] when $\alpha = 1$.

Table 3.5 Comparison on one available codebase [Chen, 2008] before and after profile-guided optimization on the trimmed dinosaur (Din) sequence

Algorithm	Successes / 20		Time (ms / iter)	
	Orig.	Mod.	Orig.	Mod.
LM-S	4	4	380	140
LM-M	1	6	369	143
LM-M_{GN}	15	19	205	109

3.1.8 Summary

Table 3.3 summarizes several existing algorithms in terms of the above components, showing how each comprises some subset. It also includes some "new" algorithms such as DRW2, which is just Ruhe and Wedin's original Algorithm II, but with damping, which has not previously been proposed for matrix factorization. (See Section 2.4.2 for information on Ruhe and Wedin algorithms.) Fig. 3.2 and Fig. 3.3 show the unified pseudocode for algorithms shown to employ variable projection and those performing joint optimization.

3.2 Implementational improvements

Having isolated key components of the algorithm, it is now straightforward to re-implement existing methods, and to explore issues of numerical stability and speed. In doing so, some "secrets" have been uncovered: three sources of speed improvement, and one important numerical stability enhancement. Of course it does not mean these have been kept deliberately secret, but that they are apparently small elements which can have a large impact on performance.

Numerical issues emerge in the implementation of these algorithms. Foremost is in the inversion of the Hessian approximation, and in the projection onto the manifold. It was found that the use of QR factorization as proposed by Okatani et al. [2011] in both of these (rather than MATLAB's `chol` or `orth` respectively) significantly improves accuracy and performance. Note that speed improvements might also adversely affect numerical stability, but in fact in all cases investigated, it *improves* it.

Profile-guided optimization amounts to exploiting standard MATLAB tricks for code speed-up, including Mex file implementations of some Kronecker products. This offered a factor of 2-3 improvement on the set of Chen's algorithms as shown in Table 3.5.

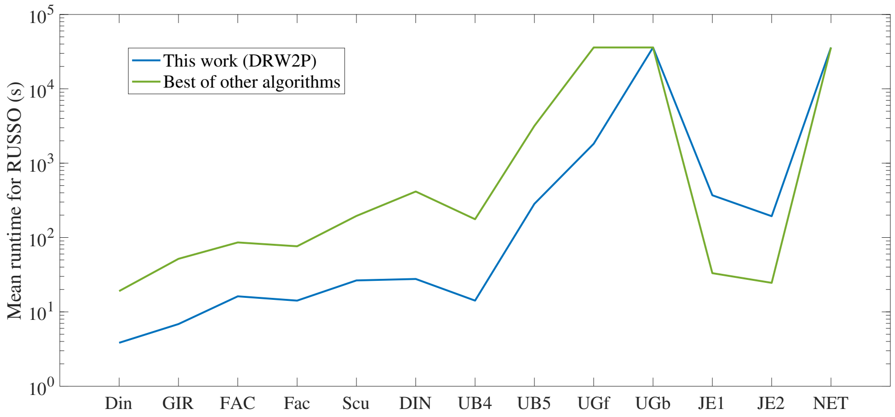


Fig. 3.4 Performance improvement by dataset. On the small and medium scale problems, our new implementations of VarPro (blue) provide a speedup over existing algorithms of a factor of 5.0 to 16.7. On the UGb dataset, no algorithm converged to the same solution twice.

Removal of redundant computations Not all existing implementations took advantage of the symmetry in the Hessian, and in fact there is another internal symmetry (see Section B.6 which yields a 4-fold speed-up).

UW-block coalescing A more subtle speed-up is obtained in the QR decomposition of $\tilde{\mathtt{U}}$. For the unregularized case, $\mathbf{v}^* = \tilde{\mathtt{U}}^\dagger \tilde{\mathbf{m}}$. It is noted that the use of QR decomposition by Okatani et al. [2011] is very useful since the decomposed outputs can be re-used for the computation of the Gauss-Newton matrix. Given that $\tilde{\mathtt{U}} = \tilde{\mathtt{U}}_Q \tilde{\mathtt{U}}_R$, the above equation becomes

$$\mathbf{v}^* = \tilde{\mathtt{U}}_R^{-1} \tilde{\mathtt{U}}_Q^\top \tilde{\mathbf{m}}. \tag{3.23}$$

Since $\tilde{\mathtt{U}} = \tilde{\mathtt{W}}(\mathtt{I}_n \otimes \mathtt{U})$ where $\tilde{\mathtt{W}}$ is the truncated version of diag vec$(\mathtt{W})$, one can observe that $\tilde{\mathtt{U}}$ is block-diagonal with the i-th block being $\tilde{\mathtt{W}}_i \mathtt{U}$ where $\tilde{\mathtt{W}}_i$ is the truncated version of diag vec$(\mathtt{W}_i)$ and $\mathtt{W}_i$ is the i-th column of the weight matrix $\mathtt{W}$. Noting that $\tilde{\mathtt{U}}_Q^\top \tilde{\mathtt{U}}_Q = \mathtt{I}$ by definition, $\tilde{\mathtt{U}}_Q$ is also going to have the same block-diagonal structure with the i-th block being the q-factor of the i-th block of $\tilde{\mathtt{U}}$. In other words, if there exists a set of same columns in the weight matrix, one only has to compute the q-factor of the corresponding blocks once. Since all the real datasets tested here consist of indicator-type weight matrix which has values either 0 or 1, such repetition is more likely to occur. The speed-ups obtained from this approach were a factor of 2.3 on average, ranging from 1 (no speed-up, on random-like data) to 5 (full dino sequence). Some timings are reported in Fig. 3.4.

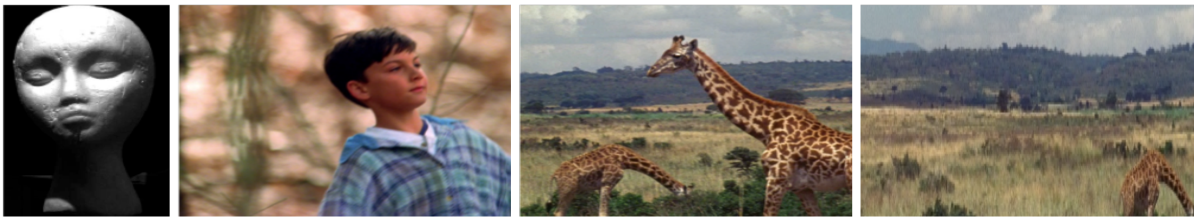
Table 3.6 Datasets used for the experiments. [*] For UGb, this minimum has not been found by any other run.

ID	Dataset	Dimension	r	Nonzeros	Fill (%)	Best known opt.
Din	Dinosaur trimmed	72×319	4	5,302	23.1	1.084673
GIR	Giraffe	166×240	6	27,794	69.8	0.322795
FAC	Face	$20 \times 2,944$	4	34,318	58.3	0.022259
Fac	Face trimed	$20 \times 2,596$	4	33,702	64.9	0.022461
SCU	Sculpture	$46 \times 16,301$	3	498,422	66.5	0.089680
DIN	Dinosaur	$72 \times 4,983$	4	32,684	9.2	1.134558
UB4	UM boy	$110 \times 1,760$	4	27,902	14.4	1.266484
UB5	UM boy	$110 \times 1,760$	5	27,902	14.4	0.795494
UGf	UM giraffe foreground	$380 \times 4,885$	6	168,286	9.1	0.774258
UGb	UM giraffe background	$380 \times 6,310$	4	164,650	6.9	0.603904*
JE1	Jester 1	$100 \times 24,983$	7	1,810,455	72.5	3.678013
JE2	Jester 2	$100 \times 23,500$	7	1,708,993	72.7	3.703549
NET	Netflix 2k	$2,000 \times 50,000$	4	2,606,298	2.7	0.806635

3.3 Datasets

Table 3.6 lists all the datasets used in the experiments. The problem classes are: rigid SfM (dinosaur [Buchanan and Fitzgibbon, 2005]), non-rigid SfM (giraffe [Buchanan and Fitzgibbon, 2005], UM boy and UM giraffe foreground and background [Rav-Acha et al., 2008]), photometric stereo (face [Buchanan and Fitzgibbon, 2005] and sculpture [Cabral et al., 2013]) and recommender systems (Jester [Goldberg et al., 2001] and Netflix). Some of the sample images are illustrated in Fig. 2.8 and Fig. 3.5. The missing data patterns of all rigid and non-rigid SfM datasets are structured without loop closures, resembling that of Fig. 1.2c. On the other hand, the recommender system sequences have more random structure like Fig. 1.2a. Photometric stereo examples are somewhere in between.

As some datasets were trimmed in some literature, this work also includes the original sets in the evaluation, indicated by the use of all caps for the originals, and mixed case for the trimmed sets. While the top 6 and bottom 3 sequences in Table 3.6 are widely used standard datasets in computer vision and machine learning, the generality of newly introduced non-rigid datasets of Rav-Acha et al. [2008] has not been previously tested in the scope of matrix factorization. This is briefly discussed in Section 3.5.



(a) Sculpture (b) UM boy (c) UM giraffe foreground (d) UM giraffe background

Fig. 3.5 New datasets used in matrix factorization. Sculpture is from Cabral et al. [2013], and the others are non-rigid "unwrap mosaic" (UM) sequences from Rav-Acha et al. [2008].

3.4 Experimental results

All experiments were carried out on a Macbook Pro (Late 2013) with 2.3 GHz Intel Core i7 Haswell processor and 16 GB 1600 MHz DDR3 memory. This work used the Ceres solver [Agarwal et al., 2014] for joint optimization w/o inner iterations. All other algorithms used MATLAB R2014b. All experiments were run in single-threaded mode to compare the speed of the algorithms. All algorithms are essentially equally parallelizable.

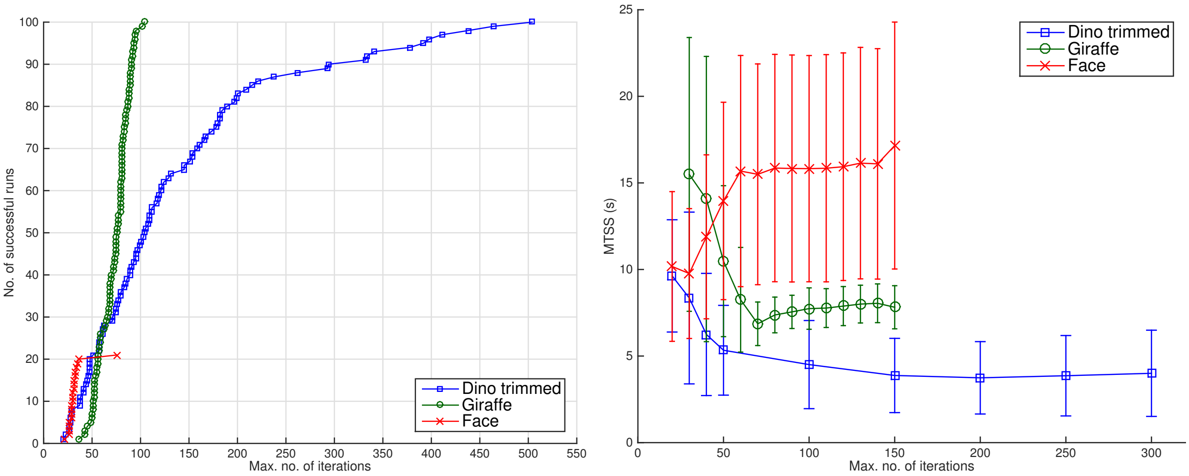
3.4.1 Experimental conditions

On each dataset, each algorithm was run multiple times, usually between 20 and 100, from random starting points. This was done by drawing initial entries of $\mathtt{U}$ from a multivariate Normal distribution $\mathscr{N}(\mathbf{0}, \mathtt{I})$. i.e. $\mathtt{U} = \mathtt{randn}(\mathtt{m}, \mathtt{r})$.

In order to set up a fair competing environment on each dataset, all algorithms at the same run number were initialized with the same starting points. For those algorithms requiring initial $\mathtt{V}$ as well (e.g. CE_LM and DB_BALM) were given $\mathtt{V}^*(\mathtt{U})$ computed from (3.12) so that they would start from the same random points as other VarPro-based algorithms, which require only initial $\mathtt{U}$.

Each run was continued until either when the maximum number of iterations (set to be 300) was reached or the cost function improvement dropped below the pre-defined tolerance value of 10^{-10}. The corresponding results are shown in Fig. 3.7.

RUSSO timing The RUSSO and RUS3O success rates were computed by measuring the time taken to reach the same so-far-best optimum twice and thrice respectively. Note that a "so-far best" optimum observed by an algorithm on a dataset does not necessarily refer to the best known (global) optimum for that dataset. For instance, if an algorithm runs RUSSO on a dataset with the global minimum value of 0.8 and reports 1.1, 1.2, 1.0 1.0 consecutively, it will terminate and report failure ($1.0 > 0.8$) along with total runtime across 4 runs. On the next



(a) Success rate.

(b) Mean time to 2$^{\text{nd}}$ success.

Fig. 3.6 Performance as a function of MaxIter (algorithm DRW2P). While the success rate metric increases monotonically with MaxIter, it is better in practice to fail early and try another starting point. Runtime to second success allows selection of a value that improves real-world performance for a given class of problem.

RUSSO run, if the same algorithm reports 1.1, 1.0, 0.80 and 1.0, it will continue running since the so-far best minimum of 0.8 has only been observed once.

3.4.2 Limitations

Some results are incomplete because of the following reasons:

1. The original Damped Wiberg code (TO_DW) [Okatani et al., 2011] and RTRMC [Boumal and Absil, 2011] failed to run on FAC and SCU datasets. This is because these algorithms are not designed to handle underdetermined cases (i.e. does not compute the pseudo-inverse) when optimally solving for each row of $\mathbf{v}^*(\mathbf{u})$. Such cases arise when one or more columns of the measurement matrix $\mathtt{M}$ each observes elements less than the predefined rank r of the model.

2. CSF [Gotardo and Martinez, 2011] runs extremely slowly on UGb dataset.

3. The algorithms implemented using the Ceres solver were incapable of handling sparse matrices as inputs, which is required to handle large datasets such as NET.

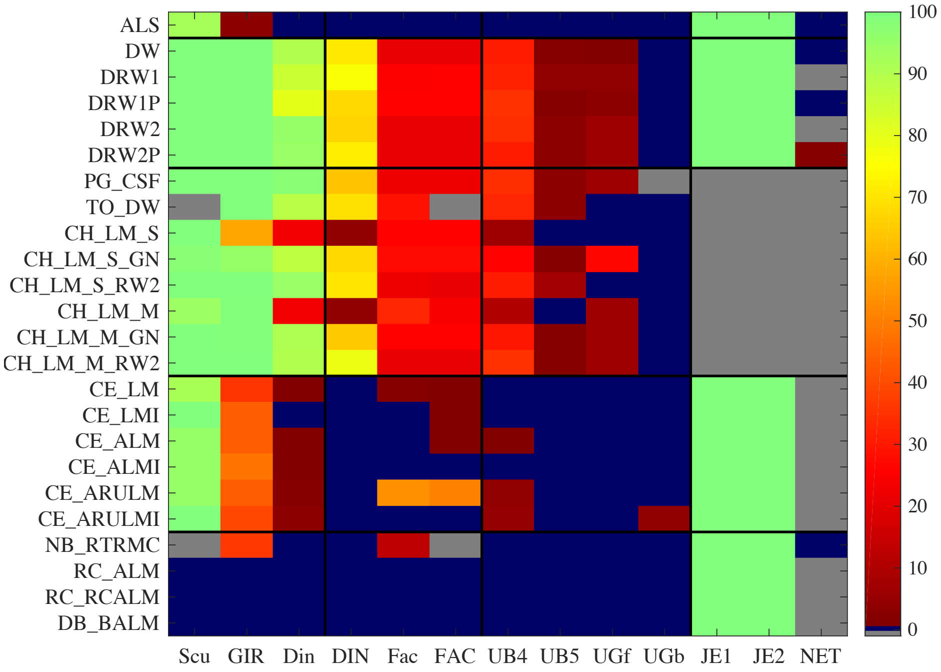


Fig. 3.7 Nominal success rates in reaching the best known optimum. Grey cells crashed or timed out. The “Face” (Fac) and “Unwrap BG” (UGb) datasets have large secondary minima so 30-50% of runs terminate at those. The variable projection-based algorithms (2nd and 3rd blocks of rows) succeed on the largest number of benchmarks.

3.5 Discussions

By discussing runtime, other algorithm parameters may be more sensibly set. For example, the maximum number of iterations taken is an algorithm parameter which affects runtime and accuracy. Fig. 3.6 shows how, while accuracy always increases with MaxIter, the safety net of RUSSO allows smaller values to be chosen, improving overall performance.

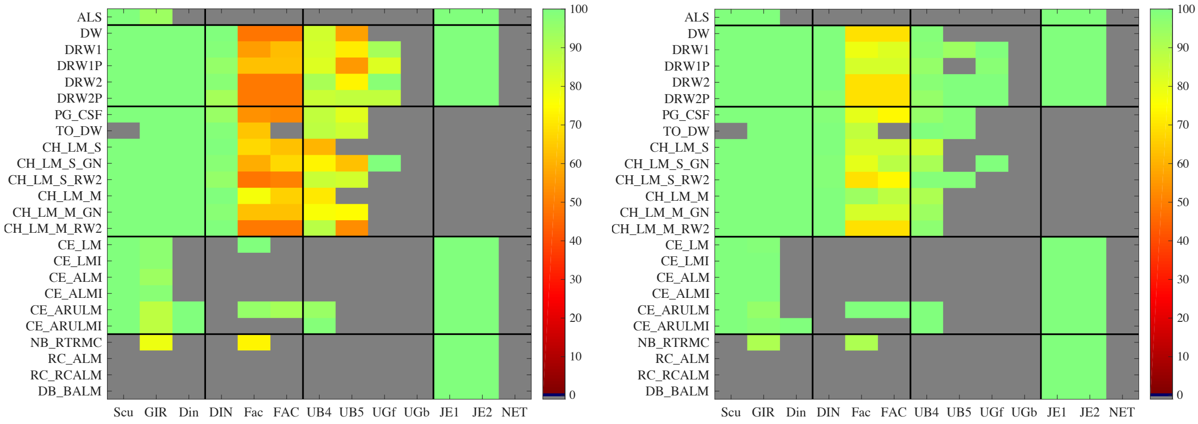
Re-implementing standard variable projection algorithms (the “DRW” set), increased their performance on the small and medium-scale problems over the state of the art by an order of magnitude (factors range from 5 to 15). New datasets are introduced, which seem more tricky to optimize than those previously in use, and open-source implementations of all code, data and evaluations are available at https://github.com/jhh37/matrix-factorization.

In order to quantify the speed-up, several existing publicly available codes were hand-optimized. This work showed that the speedup obtained by optimization, while potentially significant, is not the main reason for the new performance. Conversely, the re-implementation of Damped Wiberg algorithm [Okatani et al., 2011] is comparable to the state-of-the-art, but the new contribution is to cast it as an instance of variable projection without manifold projection.

It remains the case that alternation wrapped in RUSSO (RUSSO-ALS) is still competitive for some specialized datasets, for example the "easy" sets with high fill rates (SCU, Gir), but also the large-scale datasets (JE1, JE2). However, even then, it is beaten by RUSSO-DRW2P. However, out-of-core implementation of the latter does not have an easy off-the-shelf implementation to date. It is worth noting that the nominal and RUSSO success rates in Fig. 3.7 and Fig. 3.8 are also related to missing data patterns—sequences with comparatively large proportion of visible elements and/or random missing patterns are successfully reconstructed by most of the algorithms presented in Table 3.4. On the other hand, sequences with diagonally structured missing pattern (rigid and non-rigid SfM datasets) are mostly only successful with VarPro-inspired algorithms. (Similar phenomenon was reported by Kennedy et al. [2016].) For more random missing pattern, it could be suspected that the measurement matrix fully or approximately satisfies the restricted isometry property [Recht et al., 2010] , meaning even block coordinate descent-based approaches [Jain et al., 2013] may be able to converge to global minimum.

In Fig. 3.7, the nominal success rates of tested algorithms are relatively poor on Rav-Acha et al.'s non-rigid SfM datasets (the third column-group from the left), especially on the UGb dataset. One reason could be that these datasets are relatively sparse and have banded missing data patterns both of which increase the difficulty of each sequence. Another reason is that no regularizer is added to promote temporal smoothness, leading to multiple overfitted solutions. Unfortunately, adding a temporal smoothness may narrow the basin of convergence of the best optimum if it is not linear in $\mathtt{U}$ and $\mathtt{V}$, and so enforcing a relaxed nuclear norm constraint in (3.1) could be a better alternative for computing practical solutions. Finally, UGf comprises two sets of points tracks with different motions due to the existence of two giraffes. Similarly, UGb comprises mostly rigid background point tracks undergoing translation, but also contains some frontal non-rigid tracks that do not share the same motion as others. Consequently, UGf and UGb sequences do not generalize well in their current forms, and thus filtering either one set of tracks to unify the nature of motion would be required to produce more general sequences with better convergence behaviours.

One thing to note is that the DRW1 family, which applies the full Gauss Newton approximation on the reduced problem, has no real benefit over the DRW2 family, which makes some approximations on the reduced Jacobian. This behaviour is also observed in several papers [Gotardo and Martinez, 2011; Kaufman, 1975]. It could potentially be explained in the following way. First, note that $\mathbf{v}^*(\mathbf{u})$ can essentially be thought as taking an optimal Gauss-Newton step $\Delta\mathbf{v}^*(\mathbf{u}) = -\mathtt{J}_\mathbf{v}(\mathbf{u})^\top \varepsilon(\mathbf{u},\mathbf{v})$ given $\mathbf{u}$. This means that the reduced problem can be thought as a function of $\mathbf{u}$ and $\Delta\mathbf{v}^*(\mathbf{u})$. Now, to obtain the VarPro Jacobian, one needs to differentiate through $\Delta\mathbf{v}^*(\mathbf{u})$, and this yields $-\mathtt{J}_\mathbf{v}(\mathbf{u})^\top \mathtt{J}_\mathbf{u}(\mathbf{v}) + h.o.t.$. Approximating this with the first term



(a) RUSSO-X success rates

(b) RUS3O-X success rates

Fig. 3.8 RUSSO and RUS3O success rates in reaching the best known optimum. Gray cells now also include timeouts where the optimum was not seen twice (e.g. a success rate of 1% would require thousands of runs, so timeout is more likely).

yields RW2, and therefore RW1 can be thought as containing some unnecessary higher order term due to the variable elimination step.

It is interesting to observe that DRW1 and DRW2, which do not fill the rank ambiguity arising from the nature of the Grassmann manifold, still perform comparably to their rank-filled counterparts DRW1P and DRW2P respectively. This is because the rank is filled by the damping factors from the Levenberg-Marquardt algoritm [Levenberg, 1944; Marquardt, 1963]. Although it is mentioned correctly in [Dellaert, 2014] that damping is in theory not supposed to be used for this purpose, it does provide good rank-filling behaviour, which cannot be achieved by the Gauss-Newton algorithm with line search.

In the context of bundle adjustment, there are two points that require further investigation. First, it is surprising that joint optimization (CE_LM), which is widely used in performing bundle adjustment citetriggs00 and other joint estimation of parameters in computer vision [Taylor et al., 2016], shows extremely poor performance for matrix factorization. Second, the algorithm CE_LMI employs what is known as “inner iterations” in the Ceres solver, which is defined as VarPro on the official website [Agarwal et al., 2014]. Yet, the results shown in Fig. 3.7 and Fig. 3.8 show inferior performance to other variable projection-based algorithms. These two issues are jointly investigated in the first half of Chapter 4.

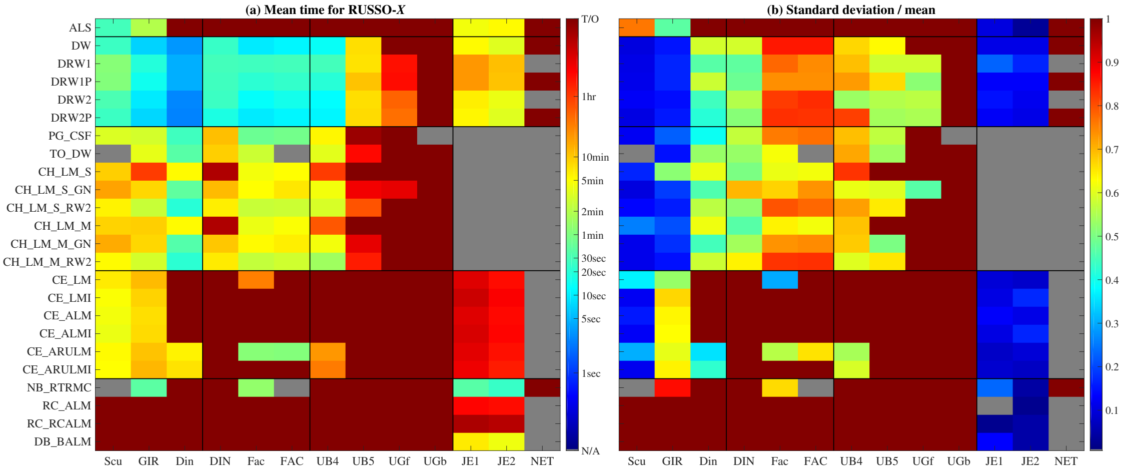


Fig. 3.9 Runtime on all algorithms, all datasets. (a) Mean runtime of RUSSO-X. (b) Standard deviation divided by mean. Where $\sigma = T/O$, too few runs converged to estimate RUSSO's performance. On the UGb dataset, no algorithm converged to the same solution twice.

3.6 Conclusions

This chapter addressed the problem of matrix factorization with missing data and known rank. Most importantly, it was shown that many recent successful algorithms with wide basin of convergence can be unified as employing the variable projection (VarPro) method, which is empirically verified as key to improving the convergence basin on the majority of datasets. Other factors such as incorporating manifold optimization show limited improvements to algorithm performance.

It was also shown that joint optimization performs poorly on matrix factorization problems with banded occlusion patterns. The forthcoming chapters are to address this problem.

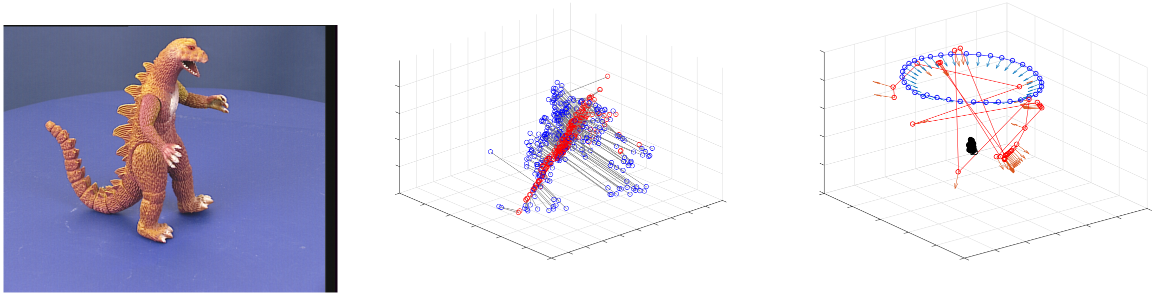
Chapter 4

Revisiting the variable projection method for affine bundle adjustment

In Chapter 3, it was experimentally observed that use of the variable projection (VarPro) method is key to widening the convergence basins of useful optima for (bilinear) matrix factorization problems, of which affine bundle adjustment is an instance. This chapter explores to find the cause of VarPro's success and search for ways to make VarPro more scalable, with aims to apply the secrets of VarPro to nonseparable bundle adjustment problems. The investigation leads to further unification of some well-known optimization methods and a scalable algorithm for VarPro, which are not limited to matrix factorization problems and therefore are illustrated in a separate chapter.

4.1 Motivation and contributions

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. [Chan and Esedoglu, 2004]), and lifting methods for MAP inference (e.g. [Ishikawa, 2003; Weiss et al., 2007]), 3D model fitting [Cashman and Fitzgibbon, 2013], and robust costs [Zach, 2014]. 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) Dinosaur (36 images) (b) Estimated 3D points (c) Estimated cameras

Fig. 4.1 An example where different reconstructions are obtained between the variable projection and joint optimization methods. For affine bundle adjustment, 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 work 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). As illustrated in Section 4.2.3 and Chapter 3, VarPro optimally eliminates some of the unknowns in an optimization problem, and is therefore especially applicable to separable nonlinear least squares problems described below. Low-rank matrix factorization is a problem class appearing in signal processing (e.g. blind source separation) [Golub and Pereyra, 2002], in machine learning (e.g. factor analysis), but also in 3D computer vision to obtain e.g. affine and non-rigid reconstructions as illustrated in Section 2.3.1. The success of VarPro methods is often reported in the literature (especially for geometric reconstruction problems), and similar results were also obtained in Chapter 3. But to current knowledge, there is a lack of understanding why variable projection is so beneficial in this case.

This 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 chapter that joint optimization suffers from an intrinsic numerical ill-conditioning for matrix factorization problems, and therefore is prone to "stalling".

Although this work focuses on the matrix factorization tasks, the presented algorithmic analysis also holds for the more general class of Separable nonlinear Least Squares problems [Golub and Pereyra, 1973]. A Separable nonlinear Least Squares (SNLS) problem is defined as minimizing

$$\|\mathtt{G}(\mathbf{u})\mathbf{v} - \mathbf{z}(\mathbf{u})\|_2^2 \tag{4.1}$$

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 $\mathtt{G} : \mathbb{R}^p \to \mathbb{R}^{m \times q}$ and $\mathbf{z} : \mathbb{R}^p \to \mathbb{R}^m$ 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 [Golub and Pereyra, 2002], ranging from exponential fitting [O'Leary and Rust, 2013] to chemistry, mechanical engineering and medical imaging.

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

$$\min_{\mathtt{U},\mathtt{V}} \|\mathtt{W} \odot (\mathtt{U}\mathtt{V}^\top - \mathtt{M})\|_F^2 \tag{4.2}$$

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 (4.2) into (4.1).

This work presents the following contributions:

+ Section 4.2 provides an extended algorithmic review of the known methods for solving separable nonlinear least squares (SNLS) problems, namely joint optimization with and without embedded point iterations (EPI) and variable projection (VarPro), all of which were briefly reviewed in Section 2.1. Unlike previous work, this chapter explicitly considers the effect of Levenberg-style damping, without which none of the alternatives perform well.

+ Section 4.3 unifies the aforementioned methods and shows that the joint methods and VarPro effectively share the same algorithmic structure but differ in details.

+ Section 4.5 provides an empirical analysis of how the joint methods fail while VarPro succeeds despite the small algorithmic difference between the two branches of methods.

+ In Section 4.4, a simple scalable strategy is proposed 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: its scope 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.

$$\begin{aligned}
&\big(\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-\lambda_k})\mathtt{J}_{\mathbf{u}_k}+\lambda_k\mathtt{I}\big)\Delta\mathbf{u}_k=-\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-\lambda_k})\varepsilon_k && \text{Joint} && (4.10)\\
&\big(\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-\lambda_k})\mathtt{J}_{\mathbf{u}_k}+\lambda_k\mathtt{I}\big)\Delta\mathbf{u}_k=-\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-0})\varepsilon_k=-\mathtt{J}_{\mathbf{u}_k}^\top\varepsilon_k && \text{Joint+EPI} && (4.15)\\
&\big(\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-0}\ )\mathtt{J}_{\mathbf{u}_k}+\lambda_k\mathtt{I}\big)\Delta\mathbf{u}_k=-\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I}-\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-0})\varepsilon_k=-\mathtt{J}_{\mathbf{u}_k}^\top\varepsilon_k && \text{VarPro} && (4.25)
\end{aligned}$$

Fig. 4.2 The key equations for $\Delta\mathbf{u}_k$ according to the three separable nonlinear least squares optimization approaches. The apparently small differences in where damping is applied give rise to very different convergence properties.

4.2 Algorithmic review of known methods

This section makes an extended algorithmic review of the joint optimization and variable projection (VarPro) methods, which were first illustrated in Section 4.2. 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.

The following terms (which are specific to separable nonlinear least squares problems) are defined below:

$$\varepsilon(\mathbf{u},\mathbf{v}) := \mathtt{G}(\mathbf{u})\mathbf{v}-\mathbf{z}(\mathbf{u}) \tag{4.3}$$

$$\mathtt{J}_{\mathbf{u}}(\mathbf{u},\mathbf{v}) := \frac{\partial\varepsilon(\mathbf{u},\mathbf{v})}{\partial\mathbf{u}} = \frac{d[\mathtt{G}(\mathbf{u})]\mathbf{v}}{d\mathbf{u}} - \frac{d\mathbf{z}(\mathbf{u})}{d\mathbf{u}} \tag{4.4}$$

$$\mathtt{J}_{\mathbf{v}}(\mathbf{u}) := \frac{\partial\varepsilon(\mathbf{u},\mathbf{v})}{\partial\mathbf{v}} = \mathtt{G}(\mathbf{u}) \tag{4.5}$$

$$\mathtt{Q}_{\mathbf{v}}(\mathbf{u}) := \mathtt{I}-\mathtt{J}_{\mathbf{v}}(\mathbf{u})^{\dagger}\mathtt{J}_{\mathbf{v}}(\mathbf{u}). \tag{4.6}$$

4.2.1 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 Section A.2.3) is applied to solve

$$\min_{\mathbf{x}}\|\varepsilon(\mathbf{x})\|_2^2 := \min_{\mathbf{x}=[\mathbf{u};\mathbf{v}]}\|\varepsilon(\mathbf{u},\mathbf{v})\|_2^2. \tag{4.7}$$

Hence, the update equations for joint optimization follow (A.81) with $\Delta\mathbf{x}_k := [\Delta\mathbf{u}_k;\Delta\mathbf{v}_k]$, $\varepsilon_k := \varepsilon(\mathbf{u}_k,\mathbf{v}_k)$ 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.

$$\begin{bmatrix} \mathtt{J}_{\mathbf{u}_k}^\top \mathtt{J}_{\mathbf{u}_k} + \lambda_k \mathtt{I} & \mathtt{J}_{\mathbf{u}_k}^\top \mathtt{J}_{\mathbf{v}_k} \\ \mathtt{J}_{\mathbf{v}_k}^\top \mathtt{J}_{\mathbf{u}_k} & \mathtt{J}_{\mathbf{v}_k}^\top \mathtt{J}_{\mathbf{v}_k} + \lambda_k \mathtt{I} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}_k \\ \Delta\mathbf{v}_k \end{bmatrix} = - \begin{bmatrix} \mathtt{J}_{\mathbf{u}_k}^\top \varepsilon_k \\ \mathtt{J}_{\mathbf{v}_k}^\top \varepsilon_k \end{bmatrix}. \tag{4.8}$$

Schur complement reduced system As mentioned in Section 2.1.2, the Schur complement is often suggested to solve (4.8) efficiently (e.g. [Triggs et al., 2000]): instead of solving for a $(p+q)\times(p+q)$ system matrix, the Schur complement reduces the problem to two subproblems of size $p\times p$ and $q\times q$, respectively,

$$\mathtt{S}_{\lambda_k} := \mathtt{I} - \mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-\lambda_k} \tag{4.9}$$

$$\Delta\mathbf{u}_k = -\left(\mathtt{J}_{\mathbf{u}_k}^\top \mathtt{S}_{\lambda_k}\mathtt{J}_{\mathbf{u}_k} + \lambda_k\mathtt{I}\right)^{-1}\mathtt{J}_{\mathbf{u}_k}^\top \mathtt{S}_{\lambda_k}\varepsilon_k \tag{4.10}$$

$$\Delta\mathbf{v}_k = -\mathtt{J}_{\mathbf{v}_k}^{-\lambda_k}\left(\varepsilon_k + \mathtt{J}_{\mathbf{u}_k}\Delta\mathbf{u}_k\right). \tag{4.11}$$

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 $\Delta\mathbf{u}$ and will play the central role in Section 4.3.

4.2.2 Embedded point iterations (EPI)

Embedded point iterations (EPI) is a method proposed to accelerate classical bundle adjustment [Jeong et al., 2010] by using a nested optimization approach: after computing the standard Gauss-Newton or LM updates, one set of unknowns ($\mathbf{v}$) 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 [Agarwal et al., 2014] but the difference to VarPro is that the joint optimization model is used to update $\mathbf{u}$ (i.e. using (4.10) instead of (4.25)).

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

$$\begin{aligned} \mathbf{v}_{k+1} &= \arg\min_{\mathbf{v}} \|\varepsilon(\mathbf{u}_{k+1},\mathbf{v})\|_2^2 \\ &= \arg\min_{\mathbf{v}} \|\mathtt{G}(\mathbf{u}_{k+1})\mathbf{v} - \mathbf{z}(\mathbf{u}_{k+1})\|_2^2 \\ &= \mathtt{G}(\mathbf{u}_{k+1})^\dagger \mathbf{z}(\mathbf{u}_{k+1}). \end{aligned} \tag{4.12}$$

Note that $\mathbf{v}_{k+1}$ is independent of the previous value of $\mathbf{v}$, and therefore (4.11) can be bypassed altogether in this case. The fact that the previous iterate $\mathbf{v}_k$ is optimal for $\|\varepsilon(\mathbf{u}_k,\mathbf{v})\|_2^2$ implies that

$$0 = \mathtt{J}_{\mathbf{v}}(\mathbf{u}_k)^\top \varepsilon(\mathbf{u}_k,\mathbf{v}_k) = \mathtt{J}_{\mathbf{v}_k}^\top \varepsilon_k. \tag{4.13}$$

Hence, (4.8) simplifies to

$$\begin{bmatrix} \mathtt{J}_{\mathbf{u}_k}^\top \mathtt{J}_{\mathbf{u}_k} + \lambda_k \mathtt{I} & \mathtt{J}_{\mathbf{u}_k}^\top \mathtt{J}_{\mathbf{v}_k} \\ \mathtt{J}_{\mathbf{v}_k}^\top \mathtt{J}_{\mathbf{u}_k} & \mathtt{J}_{\mathbf{v}_k}^\top \mathtt{J}_{\mathbf{v}_k} + \lambda_k \mathtt{I} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}_k \\ \Delta\mathbf{v}_k \end{bmatrix} = - \begin{bmatrix} \mathtt{J}_{\mathbf{u}_k}^\top \varepsilon_k \\ 0 \end{bmatrix} \tag{4.14}$$

and using the Schur complement yields

$$\Delta\mathbf{u}_k = -\left(\mathtt{J}_{\mathbf{u}_k}^\top \mathtt{S}_{\lambda_k} \mathtt{J}_{\mathbf{u}_k} + \lambda_k \mathtt{I}\right)^{-1} \mathtt{J}_{\mathbf{u}_k}^\top \varepsilon_k. \tag{4.15}$$

4.2.3 Variable projection (VarPro)

VarPro reduces the problem of minimizing (4.1) 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

$$\mathbf{v}^*(\mathbf{u}) := \arg\min_{\mathbf{v}} \|\mathtt{G}(\mathbf{u})\mathbf{v} - \mathbf{z}(\mathbf{u})\|_2^2 = \mathtt{G}(\mathbf{u})^\dagger \mathbf{z}(\mathbf{u}). \tag{4.16}$$

Inserting (4.16) into (4.1) yields the reduced problem,

$$\min_{\mathbf{u}} \|\varepsilon^*(\mathbf{u})\|_2^2 := \min_{\mathbf{u}} \|\varepsilon(\mathbf{u},\mathbf{v}^*(\mathbf{u}))\|_2^2 \tag{4.17}$$

$$= \min_{\mathbf{u}} \|\left(\mathtt{G}(\mathbf{u})\mathtt{G}(\mathbf{u})^\dagger - \mathtt{I}\right)\mathbf{z}(\mathbf{u})\|_2^2. \tag{4.18}$$

(4.18) can also be viewed as problem defined in a reduced subspace since one can reformulate the residual in (4.18) as

$$\begin{aligned} \varepsilon^*(\mathbf{u}) &= \left(\mathtt{I} - \mathtt{G}(\mathbf{u})\mathtt{G}(\mathbf{u})^\dagger\right)(\mathtt{G}(\mathbf{u})\mathbf{v} - \mathbf{z}(\mathbf{u})) \\ &= \left(\mathtt{I} - \mathtt{J}_{\mathbf{v}}(\mathbf{u})\mathtt{J}_{\mathbf{v}}(\mathbf{u})^\dagger\right)\varepsilon(\mathbf{u},\mathbf{v}) = \mathtt{Q}_{\mathbf{v}}(\mathbf{u})\varepsilon(\mathbf{u},\mathbf{v}) \end{aligned} \tag{4.19}$$

for any value of $\mathbf{v}$, where $\mathtt{Q}_{\mathbf{v}}(\mathbf{u})$ is the orthogonal projector defined in (4.6).

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 Section A.2.3) to solve (4.18) and therefore requires the Jacobian of the reduced residual $\varepsilon^*(\mathbf{u})$. From Section 2.1.3, the total derivative of (4.17) reads as

$$\mathtt{J}^*(\mathbf{u}) = \mathtt{J}_\mathbf{v}(\mathbf{u}, \mathbf{v}^*(\mathbf{u}))\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} + \mathtt{J}_\mathbf{u}(\mathbf{u}, \mathbf{v}^*(\mathbf{u})), \tag{4.20}$$

where $\mathtt{J}_\mathbf{u}$ and $\mathtt{J}_\mathbf{v}$ are the Jacobians of the original residual (4.3).

Computing $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u}$ and its approximations The derivative of $\mathbf{v}^*(\mathbf{u})$ is derived in Section 2.1.3, which yields the following expression

$$\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} = -\mathtt{J}_\mathbf{v}(\mathbf{u})^\dagger \mathtt{J}_\mathbf{u}(\mathbf{u}, \mathbf{v}^*(\mathbf{u})) - (\mathtt{J}_\mathbf{v}(\mathbf{u})^\top \mathtt{J}_\mathbf{v}(\mathbf{u}))^{-1}\frac{d[\mathtt{J}_\mathbf{v}(\mathbf{u})]^\top \varepsilon^*(\mathbf{u})}{d\mathbf{u}}. \tag{4.21}$$

Inserting (4.21) into (4.20) yields

$$\mathtt{J}^*(\mathbf{u}) = \mathtt{Q}_\mathbf{v}(\mathbf{u})\mathtt{J}_\mathbf{u}(\mathbf{u}, \mathbf{v}^*(\mathbf{u})) - \mathtt{J}_\mathbf{v}(\mathbf{u})^{\dagger\top}\frac{d[\mathtt{J}_\mathbf{v}(\mathbf{u})]^\top \varepsilon^*(\mathbf{u})}{d\mathbf{u}}. \tag{4.22}$$

Note that (4.21) (and therefore (4.22)) 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 [1980]). 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 [Buchanan and Fitzgibbon, 2005; Gotardo and Martinez, 2011; Okatani et al., 2011].

Another approximation, called RW2 (Ruhe and Wedin Algorithm 2), discards the second term in (4.21), leading to

$$\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} \approx -\mathtt{J}_\mathbf{v}(\mathbf{u})^\dagger \mathtt{J}_\mathbf{u}(\mathbf{u}, \mathbf{v}^*(\mathbf{u})) \tag{4.23}$$

$$\Rightarrow \mathtt{J}^*(\mathbf{u}) \approx \mathtt{Q}_\mathbf{v}(\mathbf{u})\mathtt{J}_\mathbf{u}(\mathbf{u}, \mathbf{v}^*(\mathbf{u})). \tag{4.24}$$

Despite the naming convention, RW2 was first proposed by Kaufman [1975] as an efficient way to implement VarPro. There is significant empirical evidence [Golub and Pereyra, 2002; Gotardo and Martinez, 2011; Kaufman, 1975; Ruhe and Wedin, 1980] over the past 40 years that RW2-VarPro has similar convergence property to the fully-derived VarPro while benefiting from reduced computational complexity. Consequently, this work 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 (4.24) into (A.81), the update equation for VarPro at iteration k is derived:

$$\Delta\mathbf{u}_k = -(\mathtt{J}_{\mathbf{u}_k}^\top(\mathtt{I} - \mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^\dagger)\mathtt{J}_{\mathbf{u}_k} + \lambda_k\mathtt{I})^{-1}\mathtt{J}_k^\top\varepsilon_k \tag{4.25}$$

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

Improving numerical stability In (4.25), computing $\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^\dagger = \mathtt{J}_{\mathbf{v}_k}(\mathtt{J}_{\mathbf{v}_k}^\top\mathtt{J}_{\mathbf{v}_k})^{-1}\mathtt{J}_{\mathbf{v}_k}^\top$ 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 $\mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^\dagger = \mathtt{J}_{\mathbf{v}_{\mathtt{Q},k}}\mathtt{J}_{\mathbf{v}_{\mathtt{Q},k}}^\top$. 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.

4.3 Unifying methods

This section shows how the joint optimization and variable projection (VarPro) methods, which were separately reviewed in Section 4.2, are exactly related. The work specifically compares between joint optimization (Joint), joint optimization with embedded point iterations (Joint+EPI) and variable projection with RW2 approximation (VarPro).

4.3.1 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 $\varepsilon^*(\mathbf{u}_0) = \varepsilon(\mathbf{u}_0, \mathbf{v}^*(\mathbf{u}_0))$ 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, it will be assumed 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.

4.3.2 Comparing update equations

In light of (4.10), (4.15), and (4.25) one can directly compare the updates for $\Delta\mathbf{u}_k$ induced by the different methods in Fig. 4.2. The following relations were used to emphasize the connection between the various update rules: $\mathtt{J}_{\mathbf{v}_k}^\dagger = \mathtt{J}_{\mathbf{v}_k}^{-0}$, and $\varepsilon_k = (\mathtt{I} - \mathtt{J}_{\mathbf{v}_k}\mathtt{J}_{\mathbf{v}_k}^{-0})\varepsilon_k$ when $\mathbf{v}_k = \mathbf{v}^*(\mathbf{u}_k)$.

using (4.19). It is apparent in Fig. 4.2 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 $\Delta\mathbf{v}_k$ 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 $\Delta\mathbf{v}_k$ in the reduced residual, and VarPro disables damping of $\Delta\mathbf{v}_k$ entirely.

In addition to how $\Delta\mathbf{u}$ is determined in each iteration, the three algorithms also differ in the update for $\Delta\mathbf{v}$: 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 $\mathbf{u}_{k+1} = \mathbf{u}_k + \Delta\mathbf{u}_k$.

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. This topic will be discussed this in Section 4.4.

2. Second, it allows us to reason about the differences between joint optimization and VarPro. Section 4.5 analyzes the impact of damping of $\Delta\mathbf{v}$ in matrix factorization problems, and how it distorts the update directions unfavourably 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 parameter for $\Delta\mathbf{v}$, which is denoted 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 4.2). The steps in these algorithms are presented in Table 4.1.

4.4 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 (reviewed in Section A.2.2) to indirectly solve (4.25) would be preferred.

However, this alone may not replicate VarPro's large convergence basin. For matrix factorization problems, even though Boumal and Absil [2011]'s RTRMC indirectly solves the VarPro submodel using a preconditioned conjugate gradient solver, it shows poor performance

Table 4.1 A unified pseudocode for the methods reviewed in this section. joint = joint optimization, Joint+EPI = joint optimization with embedded point iterations, VarPro = variable projection with RW2 approximation. All methods are based on the Levenberg-Marquardt algorithm from Section A.2.3. Note that ε, $\mathtt{J}_{\mathbf{u}}$, $\mathtt{J}_{\mathbf{v}}$ used on their own refer to $\varepsilon(\mathbf{u},\mathbf{v})$, $\mathtt{J}_{\mathbf{u}}(\mathbf{u},\mathbf{v})$ and $\mathtt{J}_{\mathbf{v}}(\mathbf{u},\mathbf{v})$ respectively. For simplicity, the initial $\mathbf{v}$ is set to be optimal for the initial value of $\mathbf{u}$.

Joint	**Joint+EPI**	**VarPro**	**inputs**: $\mathbf{u} \in \mathbb{R}^p, \mathbf{v} = \arg\min_{\mathbf{v}} \|\varepsilon(\mathbf{u},\mathbf{v})\|_2^2 \in \mathbb{R}^q$
●	●	●	1: $[\lambda_{\mathbf{u}} \,;\, \lambda_{\mathbf{v}}] \leftarrow [10^{-4} \,;\, 10^{-4}]$
		●	2: $\lambda_{\mathbf{v}} \leftarrow 0$
●	●	●	3: **repeat**
●	●	●	4: **repeat**
●	●	●	5: $\mathbf{g} \leftarrow \mathtt{J}_{\mathbf{u}}{}^{\top}(\mathtt{I} - \mathtt{J}_{\mathbf{v}}\mathtt{J}_{\mathbf{v}}{}^{-\lambda_{\mathbf{v}}})\varepsilon$
●	●	●	6: $\Delta\mathbf{u} \leftarrow -\left(\mathtt{J}_{\mathbf{u}}{}^{\top}(\mathtt{I} - \mathtt{J}_{\mathbf{v}}\mathtt{J}_{\mathbf{v}}^{-\lambda_{\mathbf{v}}})\mathtt{J}_{\mathbf{u}} + \lambda_{\mathbf{u}}\mathtt{I}\right)^{-1}\mathbf{g}$
●			7: $\Delta\mathbf{v} \leftarrow \arg\min_{\Delta\mathbf{v}} \|\varepsilon + \mathtt{J}_{\mathbf{u}}\Delta\mathbf{u} + \mathtt{J}_{\mathbf{v}}\Delta\mathbf{v}\|_2^2 + \lambda_{\mathbf{v}}\|\Delta\mathbf{v}\|_2^2$
	●	●	8: $\Delta\mathbf{v} \leftarrow \arg\min_{\Delta\mathbf{v}} \|\varepsilon(\mathbf{u}+\Delta\mathbf{u},\mathbf{v}) + \mathtt{J}_{\mathbf{v}}(\mathbf{u}+\Delta\mathbf{u})\Delta\mathbf{v}\|_2^2$
●	●	●	9: $[\lambda_{\mathbf{u}} \,;\, \lambda_{\mathbf{v}}] \leftarrow 10\,[\lambda_{\mathbf{u}} \,;\, \lambda_{\mathbf{v}}]$
●	●	●	10: **until** $\|\varepsilon(\mathbf{u}+\Delta\mathbf{u},\mathbf{v}+\Delta\mathbf{v})\|_2^2 < \|\varepsilon\|_2^2$
●	●	●	11: $[\mathbf{u} \,;\, \mathbf{v}] \leftarrow [\mathbf{u} \,;\, \mathbf{v}] + [\Delta\mathbf{u} \,;\, \Delta\mathbf{v}]$
●	●	●	12: $[\lambda_{\mathbf{u}} \,;\, \lambda_{\mathbf{v}}] \leftarrow 0.01\,[\lambda_{\mathbf{u}} \,;\, \lambda_{\mathbf{v}}]$
●	●	●	13: **until** convergence
			output: $\mathbf{u} \in \mathbb{R}^p$, $\mathbf{v} \in \mathbb{R}^q$

on several SfM dataset in Chapter 3. It is believed 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.

The proposed strategy comes down to solving a numerically more-stable QR-factorized reduced system in Section 4.2.3 with the MINRES solver [Paige and Saunders, 1975], which is one of the Krylov subspace methods described in Section A.2.2 for solving inhomogeneous linear least squares problems of the form

$$\min_{\mathbf{x}} \|\mathtt{A}\mathbf{x} - \mathbf{b}\|_2^2 \tag{4.26}$$

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

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

Table 4.2 A taxonomy of methods based on the findings of Section 4.3 and the corresponding average probabilities of reaching the best optimum on the *trimmed dinosaur* sequence in Table 4.3.

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

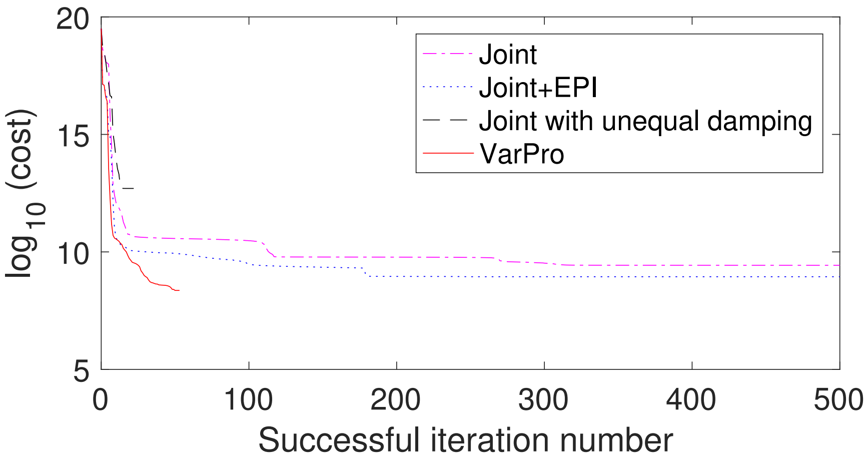


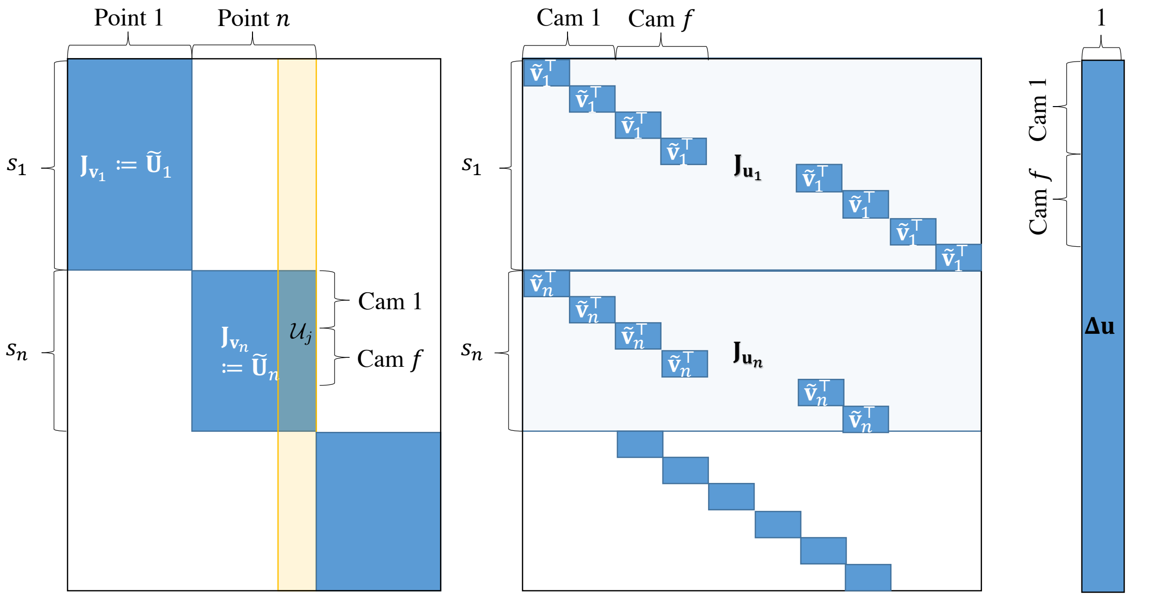
Fig. 4.3 A convergence plot of the variable projection and joint optimization-based algorithms on *trimmed dinosaur*. For this example, VarPro converges to the best known optimum (4.23×10^3) in less than 100 iterations whereas joint and Joint+EPI both exhibit flat-lining behaviours. joint with unequal damping terminates quickly at a bad minimum.

4.5 Early stopping of joint optimization

This section outlines why joint optimization is prone to early stopping (or stalling) for bilinear factorization problems (for example, see Fig. 4.3). 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. 4.2, the update equation for $\Delta\mathbf{u}$ can be written as follows,

$$\left(\mathtt{J}_\mathbf{u}{}^\top(\mathtt{I} - \mathtt{J}_\mathbf{v}\mathtt{J}_\mathbf{v}^{-\lambda_\mathbf{v}})\mathtt{J}_\mathbf{u} + \lambda_\mathbf{u}\mathtt{I}\right)\Delta\mathbf{u} = \mathbf{b}, \tag{4.27}$$


Point 1
Point n
s_1
s_n
$\mathbf{J}_{\mathbf{v}_1} := \tilde{\mathtt{U}}_1$
$\mathbf{J}_{\mathbf{v}_n} := \tilde{\mathtt{U}}_n$
$\mathcal{U}_j$
Cam 1
Cam f
Cam 1
Cam f
s_1
s_n
$\tilde{\mathbf{v}}_1^\top$
$\mathbf{J}_{\mathbf{u}_1}$
$\tilde{\mathbf{v}}_n^\top$
$\mathbf{J}_{\mathbf{u}_n}$
1
Cam 1
Cam f
$\mathbf{\Delta u}$

Block-structure of $\mathbf{J}_\mathbf{v}$

Block-structure of $\mathbf{J}_\mathbf{u}$

Structure of$\mathbf{\Delta u}$

Fig. 4.4 Typical jacobian block structure for bilinear problems.

where $\lambda_\mathbf{u} > 0$, $\lambda_\mathbf{v} \geq 0$ and $\mathbf{b}$ is one of the r.h.s. in Fig. 4.2. Let the singular value decomposition of $\mathtt{J}_\mathbf{v}$ be given by

$$\mathtt{J}_\mathbf{v} = \begin{bmatrix} \mathscr{U} & \tilde{\mathscr{U}} \end{bmatrix} \begin{bmatrix} \Sigma \\ \mathtt{0} \end{bmatrix} \mathscr{V}^\top \tag{4.28}$$

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

$$\begin{aligned} \mathtt{J}_\mathbf{v}^{-\lambda_\mathbf{v}} &= \left(\mathtt{J}_\mathbf{v}^\top \mathtt{J}_\mathbf{v} + \lambda_\mathbf{v} \mathtt{I}\right)^{-1} \mathtt{J}_\mathbf{v}^\top = \mathscr{V}(\Sigma^2 + \lambda_\mathbf{v}\mathtt{I})^{-1} \mathscr{V}^\top \mathtt{J}_\mathbf{v}^\top \\ &= \mathscr{V}(\Sigma^2 + \lambda_\mathbf{v}\mathtt{I})^{-1} \Sigma \mathscr{U}^\top. \end{aligned} \tag{4.29}$$

Consequently,

$$\begin{aligned} \mathtt{I} - \mathtt{J}_\mathbf{v} \mathtt{J}_\mathbf{v}^{-\lambda_\mathbf{v}} &= \left[\mathscr{U}, \tilde{\mathscr{U}}\right] \left[\mathscr{U}, \tilde{\mathscr{U}}\right]^\top - \mathscr{U} \Sigma^2 (\Sigma^2 + \lambda_\mathbf{v}\mathtt{I})^{-1} \mathscr{U}^\top \\ &= \tilde{\mathscr{U}} \tilde{\mathscr{U}}^\top + \mathscr{U} \left(\mathtt{I} - \Sigma^2 (\Sigma^2 + \lambda_\mathbf{v}\mathtt{I})^{-1}\right) \mathscr{U}^\top \\ &= \tilde{\mathscr{U}} \tilde{\mathscr{U}}^\top + \mathscr{U} \tilde{\Sigma}_{\lambda_\mathbf{v}}^2 \mathscr{U}^\top, \end{aligned} \tag{4.30}$$

where $\tilde{\Sigma}_{\lambda_{\mathbf{v}}}$ is defined as $\operatorname{diag}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_q)$, in which $\tilde{\sigma}_i := \sqrt{\lambda_{\mathbf{v}}/(\sigma_i^2 + \lambda_{\mathbf{v}})}$ for $i = 1, \ldots, q$. Observe that (4.27) is also the first order optimality condition for

$$\begin{aligned} &\min_{\Delta\mathbf{u}} \left\| \left(\tilde{\mathscr{U}} + \mathscr{U}\tilde{\Sigma}_{\lambda_{\mathbf{v}}}\right)^\top \mathtt{J}_{\mathbf{u}}\Delta\mathbf{u} \right\|_2^2 + \lambda_{\mathbf{u}}\|\Delta\mathbf{u}\|^2 - 2\mathbf{b}^\top\Delta\mathbf{u} \\ &= \min_{\Delta\mathbf{u}} \left\| \begin{bmatrix} \tilde{\mathscr{U}}^\top \\ \tilde{\Sigma}_{\lambda_{\mathbf{v}}}\mathscr{U}^\top \end{bmatrix} \mathtt{J}_{\mathbf{u}}\Delta\mathbf{u} \right\|_2^2 + \lambda_{\mathbf{u}}\|\Delta\mathbf{u}\|^2 - 2\mathbf{b}^\top\Delta\mathbf{u} \end{aligned} \tag{4.31}$$

since $\tilde{\mathscr{U}}^\top\mathscr{U} = \mathtt{0}$. Equation (4.31) reveals the local quadratic model of the least squares objective w.r.t. $\Delta\mathbf{u}$ 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 $\mathtt{J}_{\mathbf{v}}^\top$.

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 $\Delta\mathbf{u}$) 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 $\lambda \to \infty$, $\tilde{\Sigma}_{\lambda_{\mathbf{v}}} = \mathtt{I}$, and due to $[\tilde{\mathscr{U}}, \mathscr{U}]$ being an orthogonal matrix, (4.31) degenerates to a block-coordinate method for $\mathbf{u}$, which is known to perform poorly on matrix factorization problems [Okatani et al., 2011]).

One 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 (4.31) corresponds essentially to a block-coordinate approach. Now, suppose that $\mathtt{J}_{\mathbf{v}}$ is rank deficient. For simplicity, let us make an even stronger assumption that $\mathtt{J}_{\mathbf{v}} \approx \mathtt{0}$ (and therefore $\sigma_i \ll \lambda_{\mathbf{v}}$ and $\tilde{\Sigma}_{\lambda_{\mathbf{v}}} \approx \mathtt{I}$).

To illustrate an intuitive idea, let us focus on the updates $\Delta\mathbf{v}$ 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 $\Delta\mathbf{u}$ since $\mathbf{u}$ and $\mathbf{v}$ are correlated in SNLS problems. As written in (4.11), the joint optimization family of algorithms compute

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

and therefore

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

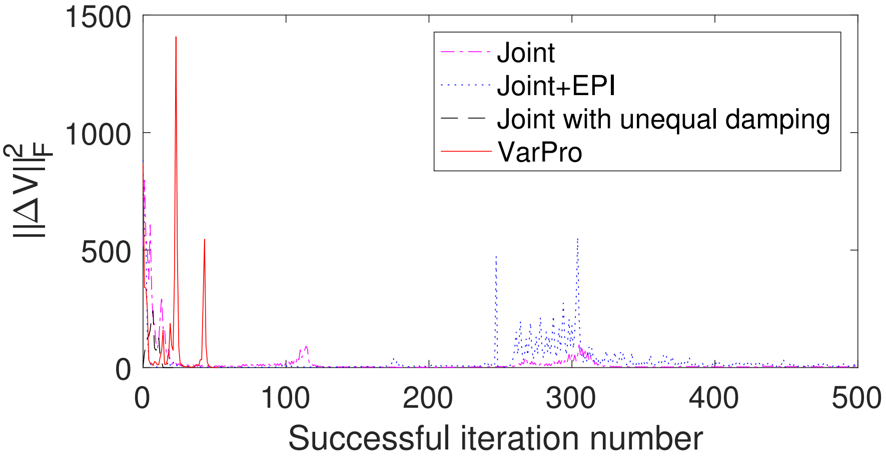


Fig. 4.5 Iteration plot of $\|\Delta\mathbf{v}\|_2$ for different algorithms. Note that variable projection allows significantly large updates on the eliminated parameters (e.g. 3D points) during optimization, whereas joint optimization algorithms make infinitesimally small updates during the flatlining phase (>300 iterations).

using the assumption $\sigma_i \ll \lambda_{\mathbf{v}}$ for all i. On the other hand, the analysis shown in Section 4.3 shows that VarPro has no damping on $\mathbf{v}$, and therefore its corresponding update $\Delta\mathbf{v}$ is

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

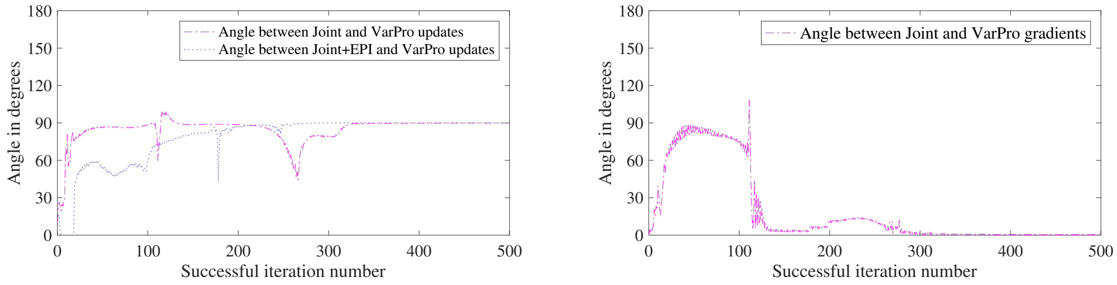
leading to

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

where $\bar{\sigma} = \max_i \sigma_i$. Consequently, this yields $\left\|\Delta\mathbf{v}_{\text{joint}}\right\| / \left\|\Delta\mathbf{v}_{\text{varpro}}\right\| \approx \bar{\sigma}^2/\lambda_{\mathbf{v}} \ll 1$ under the aforementioned assumptions. Hence, the update $\Delta\mathbf{v}_{\text{joint}}$ will be much smaller than the update $\Delta\mathbf{v}_{\text{varpro}}$. In the more general setting with $\mathrm{J}_{\mathbf{v}}$ being near singular instead of close to the zero matrix, one may observe that $\Delta\mathbf{v}_{\text{joint}}$ will be much shorter than $\Delta\mathbf{v}_{\text{varpro}}$ in certain directions. The lack of update $\Delta\mathbf{v}_{\text{joint}}$ (in certain directions) is reflected in the local quadratic model (4.31) for $\Delta\mathbf{u}$: reducing residuals is entirely the responsibility of $\Delta\mathbf{u}$.

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

To see how this affects the algorithm performance, this work resorts 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 $\mathrm{J}_{\mathbf{v}}$ can arise when a bundle of rays corresponding to a 3D point is almost collinear. In such a case, the joint optimization submodel fixes $\Delta\mathbf{v}$ (the point update)



(a) Update direction

(b) Gradient direction

Fig. 4.6 Angular differences of update and gradient directions between variable projection and joint optimization. When the joint optimization-based algorithms start to flatline, the gradients are similar to VarPro's gradient but the additional damping present in the joint optimization algorithms effectively augments the Gauss-Newton matrix such that the resultant updates are orthogonal.

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 $\Delta\mathbf{v}$, allowing camera updates $\Delta\mathbf{u}$ to make more adventurous moves.

4.6 Experimental results

To verify the algorithmic analysis in Section 4.3 and in Section 4.5, this chapter conducted two experiments solving affine bundle adjustment, which can be formulated as a matrix factorization problem [Tomasi and Kanade, 1992]. It will be shown in Section 5.1 that the obtained affine solutions can be used to bootstrap projective bundle adjustment.

In the first experiment, the VarPro-MR strategy was tested against joint optimization (Joint), joint optimization with embedded point iterations (Joint+EPI) and variable projection (VarPro) on a variety of SfM datasets. VarPro-MR was less efficiently implemented in MATLAB while the other methods were implemented within the Ceres Solver framework [Agarwal et al., 2014]. As the previous code analysis showed that the current Ceres solver version implements Joint+EPI rather than VarPro, the Ceres solver was patched to properly support VarPro (without MINRES) based on the unification work from Section 4.3.

For each run on each algorithm, each element of $\mathbf{u}_0$ was sampled 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, each algorithm was run 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 the best objective values observed were used in all runs across all implemented algorithms. The function tolerance value

Table 4.3 Experimental results for affine bundle adjustment on various datasets. For each algorithm on each dataset, the percentage of runs that converged to the best known optimum of that dataset is reported with corresponding median runtime in seconds inside the parentheses. VarPro-MR stands for VarPro applying the MINRES solver from Section A.2.2. This work used a comparatively less efficient implementation of VarPro-MR while other algorithms used a patched version of the Ceres Solver [Agarwal et al., 2014] that correctly implements VarPro.

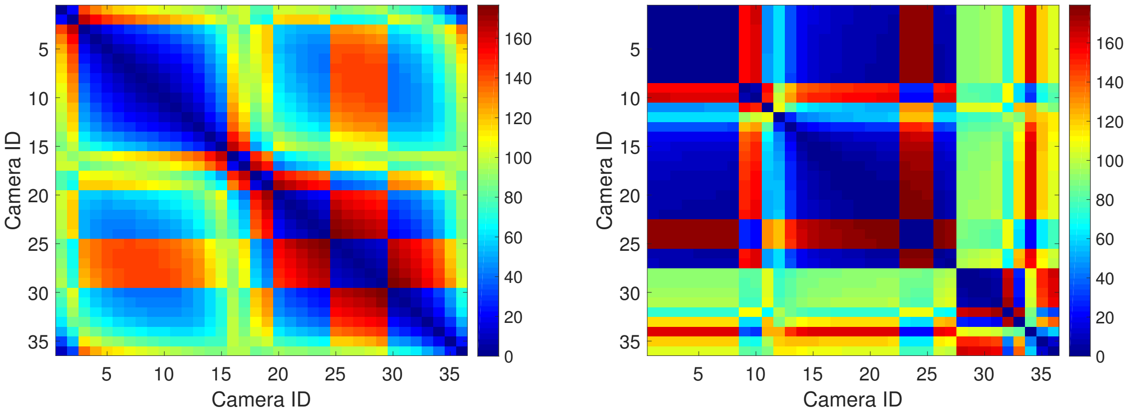
Sequence	# img	# pts	Joint	Joint+EPI	VarPro	VarPro-MR
Blue bear (trim.)	196	827	10 (238)	20 (155)	**88 (22.3)**	76 (21.9)
Corridor	11	737	40 (8.71)	4 (14.8)	**100 (1.07)**	**100 (0.78)**
Dinosaur (trim.)	36	319	4 (5.95)	24 (9.38)	**94 (1.55)**	**99 (3.96)**
Dinosaur	36	4983	0 (28.6)	0 (62.1)	**100 (13.9)**	36 (38.9)
House	10	672	44 (4.90)	8 (9.71)	**100 (0.30)**	**100 (0.41)**
Road scene #47	11	150	44 (1.88)	32 (3.00)	**100 (0.16)**	**100 (0.17)**
Stockholm guild.	43	1000	92 (45.1)	48 (35.7)	100 (22.8)	**100 (3.12)**
Wilshire	190	411	38 (409)	94 (9.90)	100 (7.64)	**100 (1.96)**
Ladybug	49	7776	0 (77.3)	0 (155)	**50 (49.7)**	0 (155)
Trafalgar Square	21	11315	0 (76.2)	0 (160)	**100 (14.7)**	100 (56.4)
Dubrovnik	16	22106	38 (159)	0 (346)	**100 (23.6)**	100 (32.9)
Venice	52	64053	0 (913)	0 (1495)	**80 (123)**	60 (329)

was set to 10^{-9} and the maximum number of successful iterations was set to 300. For VarPro-MR, the relative tolerance value was to 10^{-6} and the maximum number of inner iterations was set to 300. The reported objective values in Fig. 4.3 and Fig. 4.7 are half of the values computed from (4.2).

Table 4.3 shows that VarPro-MR mostly retains the large basin of convergence observed for standard VarPro. Note that its slower speed for larger sparse dataset may be due to its comparatively inefficient implementation.

In the second experiment, the behaviours of the four algorithms described in Section 4.3 were closely observed on the *trimmed dinosaur* dataset [Buchanan and Fitzgibbon, 2005] 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 4.2 shows that the use of EPI improves the success rate on its own but must be accompanied by the removal of the damping factor λ_v (i.e. switch to VarPro) to dramatically boost the algorithm performance.

Fig. 4.3 illustrates the typical "stalling" behaviour shared by joint and Joint+EPI. Section 4.5 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. 4.7 and Fig. 4.1, which shows that the angles between some affine camera directions (e.g. a set of cameras from ID 1 to ID 8) are



(a) VarPro (best optimum)

(b) Joint+EPI (failure)

Fig. 4.7 Angles between the directions of output affine cameras on the *trimmed dinosaur* dataset (comprising circular motion) in the projective frame. (a) shows smoothly varying angles between consecutive cameras except for cameras 25 to 29. This is simply caused by an ambiguity in the direction of affine camera rays (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.

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.

4.7 Conclusions

This chapter 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. Section 4.2.3 also tackled the question why VarPro has much higher success rates than joint optimization for certain problem classes important in computer vision.

Chapter 5

Stratified projective reconstruction without careful initialization

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 which have strong geometric constraints [Hartley and Zisserman, 2004], the step of finding a good initializer incrementally for the full-scale bundle adjustment also lacks in theoretical understanding. Hence, it could be beneficial to bypass this stage altogether and investigate initialization-free methods for bundle adjustment. One step towards the application of VarPro in bundle adjustment is the nonlinear extension of VarPro explored by Strelow [2012b], which this work takes as a starting point in Section 5.1.

If an affine or weak perspective camera model is given (or assumed), determining pose and 3D structure amounts to solving a matrix factorization problem as observed in previous chapters. This is an easy task if all points are visible in every image. However, if the visibility pattern is sparse and structured as induced by feature tracking (e.g. Fig. 1.2b or Fig. 1.2c), matrix factorization algorithms employing the variable projection (VarPro) method are highly successful even when the poses and the 3D structure are initialized arbitrarily [Okatani et al., 2011]. This is also observed in Chapter 3 and Section 4.2.3. Thus, the initial structure-from-motion (SfM) computation can be entirely bypassed in the affine case. One obvious question is whether this is also true when using a projective camera model. This the main motivation of the work presented in this section.

Formally, this work is interested in finding global minimizers of the following L^2 projective bundle adjustment problem

$$\min_{\{\mathtt{P}_i\}\{\tilde{\mathbf{x}}_j\}} \sum_{\{i,j\}\in\Omega} \|\pi\left(\mathtt{P}_i\tilde{\mathbf{x}}_j\right) - \tilde{\mathbf{m}}_{i,j}\|_2^2 \tag{5.1}$$

without requiring good initial values for the unknowns. In (5.1) 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. $\tilde{\mathbf{m}}_{i,j} \in \mathbb{R}^2$ is the observed projection of point j in frame i. Ω denotes a set of visible observations and $\pi(\cdot)$ is the perspective division such that $\pi([x,y,z]^\top) := [x/z, y/z]^\top$. This division introduces an additional nonlinearity to the objective, and thus (5.1) can be interpreted as a nonlinear matrix factorization problem.

This work's quest to solve (5.1) directly without the help of an initial structure and motion estimation step leads to the following contributions:

+ **Reinterpretation of nonseparble VarPro:** this section provides a simple rederivation of Strelow's application of VarPro to nonseparable cases.

+ **Extension of Ruhe and Wedin algorithms:** this section extends the separable nonlinear least squares algorithms in [Ruhe and Wedin, 1980] to apply to nonseparable problems such as (5.1). This allows nonseparable variable projection to be applicable to illustrated in Section 4.2.3 to be applicable almost to the same extent.

+ **A unified camera model incorporating affine and projective cases:** this chapter unifies affine and projective bundle adjustment as special cases of a more general problem class. As a byproduct this work obtains numerically better conditioned formulations for projective bundle adjustment.

Conversely, there are limitations of this work: the scope is confined to the L^2-norm projective formulation, and consequently one 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.

5.1 Variable projection for nonseparable problems

Similar to Section 4.2.3, one needs to find $\mathbf{v}^*(\mathbf{u}) := \arg\min_{\mathbf{v}} \|\varepsilon(\mathbf{u},\mathbf{v})\|_2^2$ and solve

$$\arg\min_{\mathbf{u}} \|\varepsilon^*(\mathbf{u})\|_2^2 := \arg\min_{\mathbf{u}} \|\varepsilon(\mathbf{u},\mathbf{v}^*(\mathbf{u}))\|_2^2. \tag{5.2}$$

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, a second-order iterative solver such as Levenberg-Marquardt [Levenberg, 1944; Marquardt, 1963] is applied to approximately solve $\arg\min_{\mathbf{v}} \|\varepsilon(\mathbf{u},\mathbf{v})\|_2^2$ and store the final solution in $\hat{\mathbf{v}}^*$.

Now, assuming that $\hat{\mathbf{v}}^*$ has converged, (5.2) can be linearized with respect to $\mathbf{v}$ as follows

$$\arg\min_{\Delta\mathbf{v}} \|\varepsilon(\mathbf{u},\hat{\mathbf{v}}^*) + \mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}^*)\Delta\mathbf{v}\|_2^2 \tag{5.3}$$

Notice that (5.3) is linear in $\Delta\mathbf{v}$, meaning that standard VarPro from Section 2.4.2 can be applied to optimally eliminate $\Delta\mathbf{v}$ (call this $\Delta\mathbf{v}^*$) and solve for $\mathbf{u}$. This approximation allows us to estimate $d\mathbf{v}^*(\mathbf{u})/d\mathbf{u}$ by computing $d\Delta\mathbf{v}^*/d\mathbf{u}$:

$$\frac{d\mathbf{v}^*(\mathbf{u})}{d\mathbf{u}} \approx \frac{d\Delta\mathbf{v}^*}{d\mathbf{u}} = -\frac{\partial}{\partial\mathbf{u}}[\mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}^*)^{\dagger}\varepsilon(\mathbf{u},\hat{\mathbf{v}}^*)]. \tag{5.4}$$

Hence, the Jacobian of the $\Delta\mathbf{v}$-reduced problem (5.3) $\mathtt{J}^*$ can be obtained (by the chain rule, refer to Section 2.4.2) as

$$\mathtt{J}^*(\mathbf{u}) := \mathtt{J}_{\mathbf{u}}(\mathbf{u},\hat{\mathbf{v}}^*) - \mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}^*)\frac{\partial}{\partial\mathbf{u}}\left[\mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}^*)^{\dagger}\varepsilon(\mathbf{u},\hat{\mathbf{v}}^*)\right] \tag{5.5}$$

This expression can be further expanded (using the differentiation rule for matrix pseudo-inverses in (A.25)) in the exact same way as the standard VarPro method does.

One thing to bear in mind is that, before each update of $\mathbf{u}$, $\mathbf{v}$ must be locally optimal. In standard VarPro, this is achieved by taking a simple linear least squares, but here it will have to be done through an iterative minimization over $\mathbf{v}$ given $\mathbf{u}$, and the accuracy of obtained solution will depend on the tolerance level.

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

5.1.1 Ruhe and Wedin algorithms for nonseparable problems

The nonseparable extensions of the original Ruhe and Wedin [1980] algorithms are acquired as follows: since original RW1 applies the Gauss-Newton algorithm on the $\mathbf{v}$-reduced problem, this work proposes that nonlinear RW1 employs the Gauss-Newton algorithm using the Jacobian derived in (5.5). (This is essentially the same as [Strelow, 2012b]'s General Wiberg.) Original RW2 makes an approximation of the RW1 Jacobian by discarding a second order derivative of the residual vector $\varepsilon(\mathbf{u},\hat{\mathbf{v}}^*)$. For nonlinear RW2, the same approximation is made. Lastly, original RW3 assumes $\mathbf{u}$ and $\mathbf{v}$ to be independent. For the nonseparable case, this translates to $d\Delta\mathbf{v}^*/d\mathbf{u} = 0$, yielding $\mathtt{J}_{\mathbf{v}}(\mathbf{u},\hat{\mathbf{v}}^*)$ as the approximate Jacobian of nonlinear RW3.

Table 5.1 A list of approximate Jacobians used by the proposed nonlinear extension of the Ruhe and Wedin algorithms. Nonlinear RW1 applies the Gauss-Newton (GN) algorithm on the reduced problem, nonlinear RW2 makes an approximation to the Jacobian as described in Section 5.1.1 and nonlinear RW3 makes a further approximation which turns it into alternation. $\hat{\mathbf{v}}^*$ is obtained using a second-order iterative solver (see Section 5.1).

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

5.1.2 Efficient implementation

In Section 5.1, it is shown that algorithmically, one iteration of nonlinear variable projection amounts to performing one VarPro-reduced update over **u** followed by one minimization over **v** (or vice versa). This means that the unification work in Section 4.2.3 between variable projection and joint optimization for separable nonlinear least squares can be extended to nonseparable cases. The **only change** which is required is to make VarPro compatiable with nonseparable problems is to replace the linear least squares for obtaining $\mathbf{v}^*(\mathbf{u})$ by an iterative solver such as LM. As Section 4.2.3 covered how to efficiently implement standard VarPro with the building blocks of joint optimization, one can now do the same with nonlinear VarPro just by using an iterative minimizer for computing $\mathbf{v}^*(\mathbf{u})$. Ceres solver [Agarwal et al., 2014] was patched to correctly implement VarPro in both separable and nonseparable cases.

5.2 A unified notation for uncalibrated camera models

This section presents a unified notation for uncalibrated (affine and projective) cameras which allows a modularized compilation of bundle adjustment algorithms for a unified implementation.

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

$$\mathtt{P}_i := \mathtt{P}(\mathbf{p}_i, \mathbf{q}_i, s_i, \mu_i) := \begin{bmatrix} p_{i,1} & p_{i,2} & p_{i,3} & p_{i,4} \\ p_{i,5} & p_{i,6} & p_{i,7} & p_{i,8} \\ \mu_i q_{i,1} & \mu_i q_{i,2} & \mu_i q_{i,3} & s_i \end{bmatrix} =: \begin{bmatrix} \mathbf{p}_{i,1:}^{\top} \\ \mathbf{p}_{i,2:}^{\top} \\ \begin{bmatrix} \mu_i \mathbf{q}_i^{\top} & s_i \end{bmatrix} \end{bmatrix} \tag{5.6}$$

Table 5.2 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	$\pi(\mathtt{P}_i, \tilde{\mathbf{x}}_j) = \pi(\alpha_i \mathtt{P}_i, \beta_j \tilde{\mathbf{x}}_j)$	
Inhom.	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	

where $\mathbf{p}_i = [\mathbf{p}_{i,1:}^\top, \mathbf{p}_{i,2:}^\top]^\top = [p_{i,1}, \cdots, p_{i,8}]^\top$ and $\mathbf{q}_i = [q_{i,1}, q_{i,2}, q_{i,3}]^\top$ 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

$$\tilde{\mathbf{x}}_j := \tilde{\mathbf{x}}(\mathbf{x}_j, t_j) := \begin{bmatrix} \mathbf{x}_j^\top & t_j \end{bmatrix} := \begin{bmatrix} x_{j1} & x_{j2} & x_{j3} & t_j \end{bmatrix}^\top \tag{5.7}$$

where $\mathbf{x}_j = \begin{bmatrix} x_{j,1}, x_{j,2}, x_{j,3} \end{bmatrix}^\top$ 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. (5.6) and (5.7) lead to a unified projection function

$$\pi_{i,j} := \pi(\mathtt{P}_i, \tilde{\mathbf{x}}_j) = \pi(\mathtt{P}(\mathbf{p}_i, \mathbf{q}_i, s_i, \mu_i), \tilde{\mathbf{x}}(\mathbf{x}_j, t_j)) = \frac{1}{\mu_i \mathbf{q}_i^\top \mathbf{x}_j + s_i t_j} \begin{bmatrix} \mathbf{p}_{i,1:}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2:}^\top \tilde{\mathbf{x}}_j \end{bmatrix}. \tag{5.8}$$

Table 5.2 shows that the affine and the projective models (in both homogeneous and inhomogeneous forms) are specific instances of the unified model described above.

5.3 Compilation of affine and projective bundle adjustment algorithms

This section presents the building blocks of the proposed bundle adjustment algorithms for uncalibrated cameras which stem from Section 5.1.1 and Section 5.2. To simplify notations, the variables introduced in Section 5.2 are stacked 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. Additionally, $\tilde{\mathbf{p}}$ is defined as the collection of the camera parameters $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{s}$. (Note that μ is not included.) Equation (5.1) can now be rewritten as

$$\min_{\mathbf{p},\mathbf{q},\mathbf{s},\mathbf{x},\mathbf{t}} \|\varepsilon(\mathbf{p},\mathbf{q},\mathbf{s},\mathbf{x},\mathbf{t},\mu)\|_2^2. \tag{5.9}$$

This work assumes that μ (the projectiveness vector) is fixed during optimization. (Finding an optimal way to adjust μ at each iteration could be future work and it does perform poorly when looking in the first instance.) The proposed algorithms first eliminate points ($\tilde{\mathbf{x}}$), generating a reduced problem over camera poses ($\tilde{\mathbf{p}}$), but this order could be reversed to eliminate poses first as described in Section 6.1 of [Triggs et al., 2000].

By considering all the options available in Table 5.2, a sum of 16 algorithms are compiled and listed in Table 5.4.

5.3.1 Required derivatives

From the unification made in Section 5.1, Only three types of derivatives are required to implement all the algorithms mentioned in Table 5.4 irrespective of the camera model used.

The first two derivatives are the Jacobian with respect to camera poses ($\mathtt{J}_{\tilde{\mathbf{p}}}$) and the Jacobian with respect to feature points ($\mathtt{J}_{\tilde{\mathbf{x}}}$) which are the first order derivatives of the original objective (5.1). 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 (5.1) 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 5.3 A list of derivatives required for implementing affine and projective bundle adjustment algorithms based on the methods illustrated in Section 2.1. 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 $\varepsilon(\mathbf{p},\mathbf{q},\mathbf{s},\mathbf{x},\mathbf{t},\mu)$ in (5.9).

Derivatives	Affine		Projective	
	Homogeneous	Inhom.	Homogeneous	Inhom.
$\mathtt{J}_{\tilde{\mathbf{p}}}$ (over cameras)	$\left[\frac{\partial\varepsilon}{\partial\mathbf{p}}\frac{\partial\varepsilon}{\partial\mathbf{s}}\right]$	$\frac{\partial\varepsilon}{\partial\mathbf{p}}$	$\left[\frac{\partial\varepsilon}{\partial\mathbf{p}}\frac{\partial\varepsilon}{\partial\mathbf{q}}\frac{\partial\varepsilon}{\partial\mathbf{s}}\right]$	$\left[\frac{\partial\varepsilon}{\partial\mathbf{p}}\frac{\partial\varepsilon}{\partial\mathbf{q}}\right]$
$\mathtt{J}_{\tilde{\mathbf{x}}}$ (over points)	$\left[\frac{\partial\varepsilon}{\partial\mathbf{x}}\frac{\partial\varepsilon}{\partial\mathbf{t}}\right]$	$\frac{\partial\varepsilon}{\partial\mathbf{x}}$	$\left[\frac{\partial\varepsilon}{\partial\mathbf{x}}\frac{\partial\varepsilon}{\partial\mathbf{t}}\right]$	$\frac{\partial\varepsilon}{\partial\mathbf{x}}$
$\frac{\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]\varepsilon}{\partial\tilde{\mathbf{p}}}$ (RW1)	$\left[\frac{\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]}{\partial\mathbf{p}}\frac{\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]}{\partial\mathbf{s}}\right]\varepsilon$	$\frac{\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]\varepsilon}{\partial\mathbf{p}}$	$\left[\frac{\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]}{\partial\mathbf{p}}\frac{\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]}{\partial\mathbf{q}}\frac{\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]}{\partial\mathbf{s}}\right]\varepsilon$	$\left[\frac{\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]}{\partial\mathbf{p}}\frac{\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]}{\partial\mathbf{q}}\right]\varepsilon$

5.3.2 Constraining scale freedom in homogeneous projection models

Homogeneous camera models have local scale freedoms for each camera and point (see Table 5.2). These scales need to be constrained 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, the Riemannian manifold optimization framework [Absil et al., 2008; Manton et al., 2003] illustrated in Section A.2.5 is applied. 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 [Manton et al., 2003] 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 the implementation can be found in Chapter B.

5.3.3 Constraining gauge freedom for the VarPro-based algorithms

Unlike scale freedoms, gauge freedoms are present in all camera models listed in Section 5.2.

The proposed VarPro-based algorithms eliminate points such that the vertical stack of camera matrices lie on the Grassmann manifold. This means that any set of cameras which

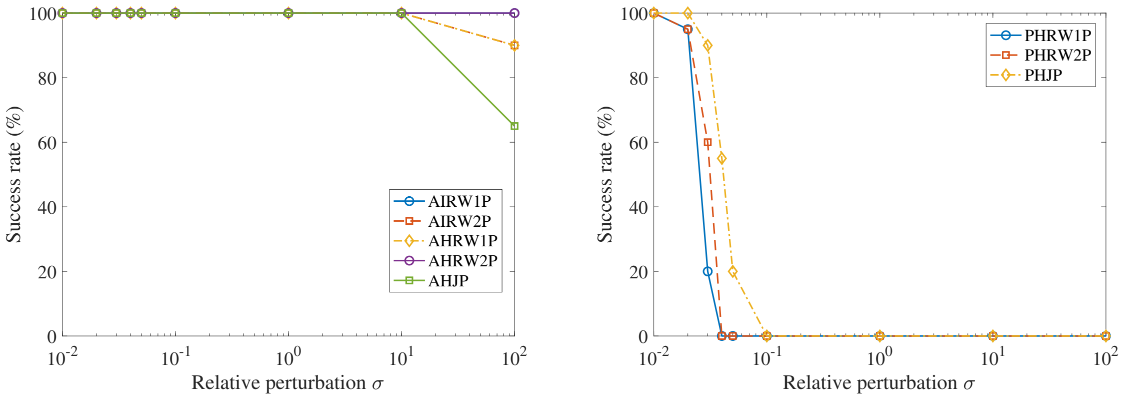
Table 5.4 A list of affine and projective bundle adjustment algorithms used for the proposed two-stage meta-algorithms in Section 5.5. These algorithms are compiled using the building blocks from Section 5.3.

ID	Camera	Form	Method	Algorithm	Gauge fix
AHRW1P	Affine	Homogeneous	Nonlinear VarPro	RW1	16 / 16
AIRW1P		Inhom.	VarPro	RW1	9 / 12
AHRW2P		Homogeneous	Nonlinear VarPro	RW2	16 / 16
AIRW2P		Inhom.	VarPro	RW2	9 / 12
AHRW3P		Homogeneous	Nonlinear VarPro	RW3 (ALS)	0 / 0
AIRW3P		Inhom.	VarPro	RW3 (ALS)	0 / 0
AHJP		Homogeneous	Joint	LM	None
AIJP		Inhom.	Joint	LM	None
PHRW1P	Projective	Homogeneous	Nonlinear VarPro	RW1	16 / 16
PIRW1P		Inhom.	Nonlinear VarPro	RW1	16 / 16
PHRW2P		Homogeneous	Nonlinear VarPro	RW2	16 / 16
PIRW2P		Inhom.	Nonlinear VarPro	RW2	16 / 16
PHRW3P		Homogeneous	Nonlinear VarPro	RW3 (ALS)	0 / 0
PIRW3P		Inhom.	Nonlinear VarPro	RW3 (ALS)	0 / 0
PHJP		Homogeneous	Joint	LM	None
PIJP		Inhom.	Joint	LM	None

share the same column space as the current set does not change the objective. With the inhomogeneous affine camera model, the vertical stack matrix of all 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), this work incorporates a technique introduced by Okatani et al. [2011], which penalizes the stacked matrix of cameras updating along the column space of the current stacked matrix. Applying it constrains all 16 gauge freedoms. (This approach can be viewed as a relaxed form of the manifold optimization framework described in Section 5.3.2.) For the inhomogeneous affine model, this technique is manipulated to prevent from overconstraining the problem, and this removes 9 out of 12 gauge freedoms. More details are included in Chapter C.

This work has not implemented a gauge-constraining technique for joint optimization, but a technique proposed by Mishra et al. [2013] could be applied in the future.



(a) Affine algorithms

(b) Projective algorithms

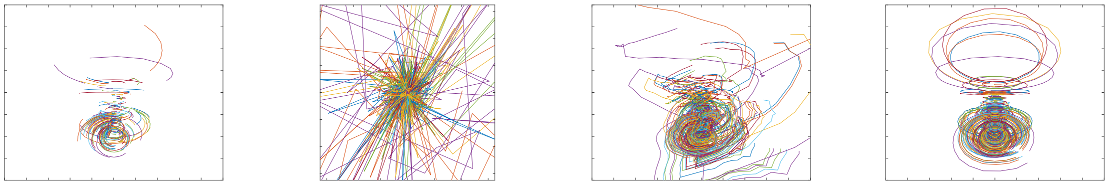
Fig. 5.1 Success rates of affine and projective VarPro algorithms on the *trimmed dinosaur* sequence. Each algorithm is initialized from a perturbed best solution. Relative perturbation is defined as the standard deviation of added white Gaussian noise divided by the average norm of the best solution's camera parameters. (a) confirms that affine VarPro algorithms have very large convergence basin of the best optimum, while (b) shows narrowed convergence basin even for the projective algorithms employing Strelow [2012b]'s nonlinear VarPro. Although the joint optimization-based projective algorithm (PHJP) shows a slightly wider convergence basin than the other nonlinear VarPro-based algorithms, this varies per dataset.

5.4 Narrow convergence basins of the projective bundle adjustment algorithms

Initially, it was attempted to apply the projective bundle adjustment algorithms compiled in Section 5.3 on the datasets listed in Table 5.6 and Table 5.7.

For each of the projective algorithms, the first two rows of each camera matrix and each 3D point were all sampled arbitrarily from $\mathscr{N}(\mathbf{0}, \mathtt{I})$ as in the previous sections. The last row of each camera matrix was initialized as $[0001]$ in order to keep VarPro's initialization condition consistent between affine and projective bundle adjutment problems. Each algorithm was allowed 50 attempts (from different initial points) on each dataset. The maximum number of iterations was set to 300 and the function tolerance value was set to 10^{-9}.

Unfortunately, none of the projective algorithms converged to the best optimum of each dataset across 100 different runs when initialized arbitrarily as mentioned above. In order to verify this disappointing result, the more numerically-stable set of projective algorithms (AHRW2P, AHRW1P and PHJP) were instead tested on the *trimmied dinosaur* sequence initialized from a perturbed best solution, which can be an "easier" starting point depending on the degree of perturbation. This time, each algorithm was allowed 20 runs, and the tested range of relative perturbations (relative to the average norm of the best solution's camera parameters)



(a) Observed tracks (b) Initial (377.29) (c) Best affine (d) Best projective

Fig. 5.2 Visualization of the trimmed dinosaur tracks with closer cameras (Di2 in Table 5.7) recovered using an affine boostrapping strategy tracks recovered using the proposed two-stage meta-algorithms. In each run, each meta-algorithm is initialized from random camera poses and points (Fig. 5.2b). In the first stage, it performs affine bundle adjustment using either a linear or nonlinear VarPro-based algorithms, reaching the best affine optimum (Fig. 5.2c) in 91% of all runs. The outputs are then used to initialize projective bundle adjustment. Although Di2 has strong perspective effects, the recommended meta-algorithms (TSMA1 and TSMA2) both reach the best projective optimum (Fig. 5.2d) in 90–98% of all runs.

was between 10^{-2} and 10^{2}. The numbers of runs reaching the best optimum are reported in Figure 5.1. Note that this perturbation experiment is a concise variant of the experiments carried out by Strelow [2012b]. In this work, the afffine results are also plotted to give an idea about the size of reduction in the convergence basin experienced by the nonlinear VarPro projective bundle adjustment algorithm.

The result of narrowed convergence basin is not totally surprising given that Zheng et al. [2012]'s weak perspective reconstruction algorithm using VarPro also failed to converge when initialized randomly, despite their problem looking supposedly "easier" by being linear in 3D points (and nonlinear in quaternions). Consequently, the next section proposes a simple two-stage strategy that can bypass this convergence issue.

5.5 Projective bundle adjustment from affine factorization

Section 5.4 observed that the nonlinear (nonseparable) extension of VarPro has substantially reduced basins of convergence of useful optima for projective bundle adjustment. To circumvent this issue, this section proposes the following strategy: perform affine bundle adjustment first and then use the outputs to initialize projective bundle adjustment. It has been noted that Zheng et al. [2012] uses a similar two-stage meta algorithm for bundle adjustment with weak perspective cameras (i.e. affine factorization followed by weak perspective bundle adjustment). It is also inspired by the fact that some projective algorithms such as projective matrix factorization [Hartley and Zisserman, 2004; Sturm and Triggs, 1996] and trilinear projective bundle adjustment [Strelow, 2012b] initialize all camera depths to 1, which is

Table 5.5 A list of two-stage meta-algorithms used in the experiments.

ID	1st stage algorithm	2nd stage algorithm
TSMA1	AHRW2P	PHRW1P
TSMA2	AIRW2P	PHRW1P
TSMA3	AHRW2P	PHJP
TSMA4	AIRW2P	PHJP

equivalent to employing the affine camera model. The key difference between the proposed strategy and the aforementioned projective methods is that the former enforces the affine model throughout the first stage whereas other methods can switch to the projective model straight after initialization. Since the proposed strategy essentially places a prior on the affine model, it is important to check how this would perform on strong perspective scenes.

This section provides numerical experiments to identify the method yielding the highest success rate overall on real and synthetic datasets. The chapter concludes 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.

The first (affine) stage uses either AIRW2P (Affine inhomogeneous RW2 with manifold Projection) and AHRW2P (affine homogeneous RW2 with manifold projection), since these VarPro-based algorithms have wide convergence basins. The RW1 series are dropped as they perform substantially slower than the RW2 counterparts with no additional benefits in the success rates. (Similar phenomenon is reported by Gotardo and Martinez [2011] and in Chapter 3.)

For the second (projective) stage, PHRW1P (projective homogeneous RW1 with manifold projection) and PHJP (projective homogeneous joint optimization with manifold projection) are selected. PHRW2P is dropped due to its poor performance on some of the datasets used. None of the inhomogeneous projective algorithms are selected due to numerical instability (see Section 5.3.2).

5.6 Experiments

The proposed two-stage meta-algorithms (TSMAs) were tested on various small synthetic (Table 5.6) and real SfM datasets (Table 5.7) 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).

Table 5.6 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. This dataset has synthetic structured visibility patterns with high missing rates in order to depict real tracks with occlusions and tracking failures. $\mathscr{N}(\mathbf{0}, \mathtt{I})$ noise is added.

Sequence	d	Loop closed	Missing (%)	Best affine	Best projective
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

On each dataset, each TSMA listed in Table 5.5 were run 100 times. On each run, initial camera poses and points were sampled from $\mathscr{N}(\mathbf{0}, \mathtt{I})$. The first stage of each meta-algorithm minimized the affine version of (5.1), and the second stage minimized the projective version of the same problem. The maximum number of iterations in each stage was set to 1000 and the function tolerance value was set to 10^{-9}.

It was 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. 5.3a and Fig. 5.3b.

This section reports the normalized cost values which can be computed as follows:

$$\sqrt{\text{Equation (5.1)} \;/\; (2 \times \text{Total number of visible frames over all cameras})}.$$

5.7 Discussions

Fig. 5.3a and Fig. 5.3b 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, this is believed to be novel and valuable results.

On the synthetic datasets (Fig. 5.3a), all the proposed meta-algorithms yield high success rates (74–100%) until the ground truth cameras are moved radically close to the sphere (e.g.

Table 5.7 Small real datasets used for the experiments. Datasets are listed by the decreasing "affineness" of the scenes (computed by taking the ratio of the best affine minimum value to the best projective minimum value). *Di2 was generated by projecting 3D points from synthetic camera poses made deliberately close to the 3D structure thereby inducing strong perspective effects.

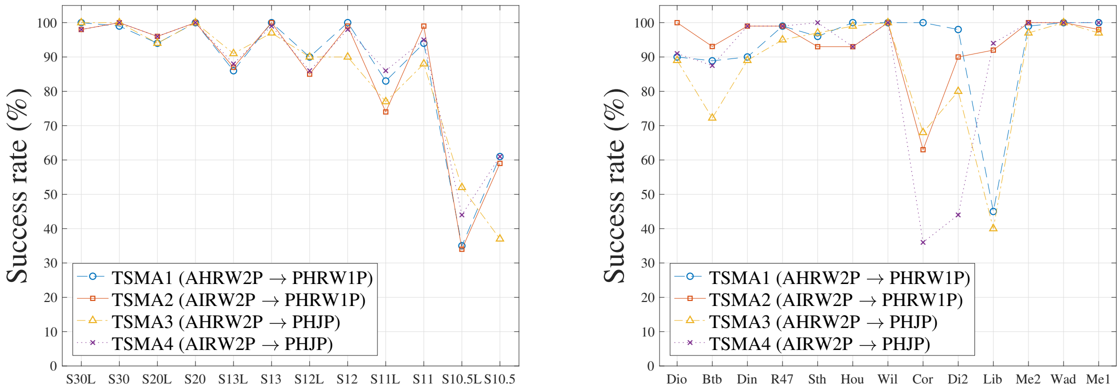
ID	Sequence	# img	# pts	Missing (%)	Best affine	Best proj
Dio	Dinosaur	36	4983	90.84	1.217574	1.166165
Btb	Blue bear (trim.)	196	827	80.71	0.530633	0.489283
Din	Dinosaur (trim.)	36	319	76.92	1.270153	1.114493
R47	Road scene #47	11	150	47.09	4.402777	3.344768
Sth	Stockholm guildhall	43	1000	18.01	8.833195	5.619975
Hou	House	10	672	57.65	2.750877	0.441660
Wil	Wilshire	190	411	60.73	2.703663	0.423892
Cor	Corridor	11	737	50.23	2.237213	0.272462
Di2*	Dinosaur (trim.) closer	36	319	76.92	9.380473	0.838414
Lib	Univ. of Oxford library	3	667	29.24	4.180297	0.172830
Me2	Merton college 2	3	475	21.61	3.995869	0.158851
Wad	Wadham college	5	1331	54.64	3.424812	0.135711
Me1	Merton college 1	3	717	22.13	3.176176	0.118450

S10.5/L). This is somewhat expected since the proposed two-stage 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 the proposed 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.

It is worth noting that datasets with loop closures (the 'L' family) seems to have lower success rates overall for which no clear answer has been found.

On the real sequences (Fig. 5.3b), 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, there is no clear advantage in success rates of employing AHRW2P instead of AIRW2P. The difrerence is only observed on the Cor dataset (see Fig. 5.3b), which comprises forward camera movements. This is against the original 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



(a) Synthetic datasets

(b) Real datasets

Fig. 5.3 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.) This work concludes that TSMA1 and TSMA2 are winners by narrow margins.

instability caused by the use of inhomogeneous coordinates is not a major issue for the affine case. (It is still an issue for the projective model.)

In addition to the main experiments, the work investigated to see if the proposed meta-algorithms could serve as an initializer for the full bundle adjustment process. A projective bundle adjustment algorithm such as PHRW1P was run 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 remark, it is believed that the proposed meta-algorithms could be applied to a segment of large datasets to trigger incremental or full bundle adjustment.

An additional investigation tried incrementing μ (the projectiveness parameter) gradually to make the affine-projective transition smoother, but this strategy performed as poor as running projective bundle adjustment without affine initialization. Implementing a fully unified algorithm still remains a challenge to this date.

Finally, Strelow's nonlinear VarPro uses an iterative solver to optimize over the eliminated set of variables $\mathbf{v}$, and then assumes that the residual is locally linear in $\mathbf{v}$. This can be viewed as approximating the update $\Delta\mathbf{v}$ as a single Gauss-Newton step in the final iteration. It would be interesting to check if formulating $\Delta\mathbf{v}$ exactly as the sum of all iterative updates taken by $\mathbf{v}$ would improve the convergence behaviour of the algorithm (despite potentially slower speed). One could also approximate it as a sum of two or three previous updates.

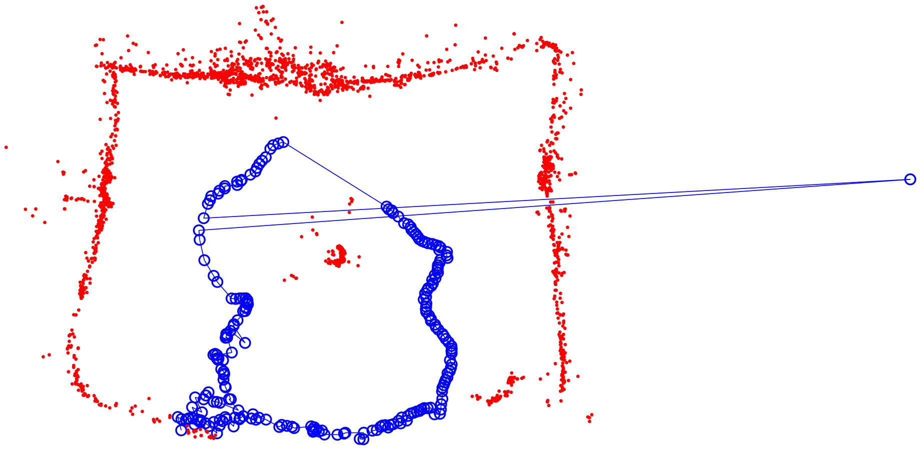


Fig. 5.4 A case where bootstrapping bundle adjustment from affine factorization fails. Points are in reds, and camera centres are in blues. It is suspected that, since affine camera centres are at infinity with undefined sign of direction, bootstrapped projective bundle adjustment can choose the wrong side to converge in case the observed scene has small depth variation.

5.7.1 Issue with affine factorization for metric reconstruction

Since the 2-stage meta-algorithms (TSMAs) yield projective reconstructions from the affine bundle adjustment solutions. It was attempted to upgrade these projective solutions to metric using the method proposed by Pollefeys et al. [1999] (illustrated in Section 2.2.5), assuming the camera intrinsics are approximately known.

Although using affine factorization yielded suitable reconstructions for many of the sequences, it failed to reconstruct correctly in the metric frame when a scene is largely planar with small depth variation, with one or more cameras being “reflected” off the scene. Fig. 5.4 shows an illustration. Since an affine camera has its centre at infinity, it can be better modelled as a ray passing through an image plane (as discussed in Section A.3.3). The problem is that the camera centre may lie on either side of the ray since the two ends $+\infty$ and $-\infty$ meet at infinity. It is believed that projective bundle adjustment is somewhat doing the job of bringing in these affine camera centres closer to the scene during the course of optimization. Now, when the scene is largely planar, the projective algorithm’s update may induce the centre to converge from the wrong side (due to a lack of geometric cues). This seems to be an inevitable problem with affine factorization, and therefore an alternate problem formulation, pOSE, is proposed here which is able to better account for the projectiveness of scenes as well as maintain bilinear problem structure for efficient VarPro implementation.

5.8 Conclusions

This work 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. As a byproduct, it presented further derivations, which bring the nonseparable VarPro into the unification proposed in Section 4.2.3. Unfortunately, the answer is that the success rate of VarPro-based algorithms cannot be directly replicated in the projective setting.

This work also demonstrated that the convergence basin of best seen projective optimum can be greatly enhanced using the right combination of methods. By unifying affine and projective factorization problems, numerically better conditioned formulations were also derived to solve these instances. 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. It was also briefly discussed that initially applying affine factorization can potentially lead to incorrect metric reconstructions with “flipped” cameras in planar scenes.

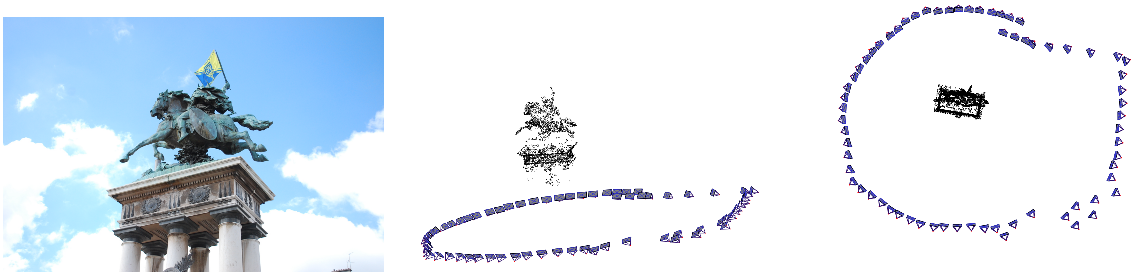
Chapter 6

Initialization-free stratified metric reconstruction using the pseudo object space error

It was observed in Chapter 5 that the property of large convergence basin for VarPro is limited to bilinear factorization problems, requiring a bootstrapping strategy in which an affine factorization problem is solved first, with its solution being used to trigger projective bundle adjustment. In this chapter, the aforementioned stratified approach is modified and extended to stably reconstruct scenes in the metric frame given relatively clean image point tracks. To achieve this, the initial affine factorization stage is replaced by an optimization step minimizing the *pseudo object space error* (pOSE), which is similar to projective factorization and therefore more closely resembles the actual bundle adjustment solution.

6.1 Motivation and contributions

Structure-from-motion (SfM, or visual SLAM or multi-view 3D reconstruction) aims to generate 3D models and camera poses from multiple overlapping images. A complete SfM framework usually consists of several stages, starting from feature extracting and matching, and ranging to a final bundle adjustment step aiming to explain all image observations and correspondences between them by finding the most probable configuration of 3D structure and camera poses and parameters. The first stages of a typical SfM pipeline (feature extraction, robust matching and relative pose verification) are relatively well understood, and different SfM toolkits and frameworks vary surprisingly little in their respective implementations of these initial steps. Likewise, the final bundle adjustment is generally understood as instance of a non-



(a) Sample image (b) Side view (c) Top view

Fig. 6.1 Reconstruction of Vercingetorix from arbitrary initial camera views and points using the pseudo object space error (pOSE).

linear least squares problem and implemented accordingly using e.g. the Levenberg-Marquardt algorithm.

It is also well understood that bundle adjustment requires a fairly good initialization for the 3D structure and camera matrices in order to determine a good (local) minimum. The various proposals in the literature for structure-from-motion computation (e.g. [Enqvist et al., 2011; Moulon et al., 2013; Schönberger and Frahm, 2016; Snavely et al., 2006; Sweeney et al., 2015; Wu, 2013] among many others) are very diverse in how this initial estimate is obtained. Since no gold-standard method for SfM computation has emerged over several decades of research in this field, this work conjectures that a) SfM is a hard problem and b) the sources of its difficulty are not well understood. By looking at a typical bundle adjustment objective,

$$\min_{\substack{\{\mathtt{R}_i\},\{\mathbf{t}_i\},\{\mathbf{x}_j\} \\ \mathtt{R}_i \in SO(3)}} \sum_{(i,j)\in\Omega} \rho\left(\|\pi\left(\mathtt{K}_i[\mathtt{R}_i \mid \mathbf{t}_i]\tilde{\mathbf{x}}_j\right) - \mathbf{m}_{i,j}\|_2^2\right), \tag{6.1}$$

(where ρ is a robust cost function, $\pi(x,y,z) = (x/z, y/z)^\top$ is the perspective projection function, $\{\mathtt{K}_i[\mathtt{R}_i \mid \mathbf{t}_i] = \mathtt{P}_i\}$ for the camera matrices, $\{\mathbf{x}_j\}/\{\tilde{\mathbf{x}}_j\}$ are the inhomogeneous and homogenous 3D points, and $\mathbf{m}_{i,j} \in \mathbb{R}^2$ is the 2D observation of point j in image i respectively) one may notice that this objective function has several sources of nonlinearities as mentioned in Chapter 1.

It is known from Section 5.1 that applying bundle adjustment for perspective (pinhole) camera models directly from arbitrary starting points is futile. At the same time it is known that the affine camera model is non-problematic for bundle adjustment (provided the right optimization approach is used), even without a sensible initialization of poses and 3D structure. This observation is leveraged in Section 5.5, where it is proposed to solve SfM by a sequence of bundle adjustment tasks with increasing difficulty: the first bundle adjustment round solves SfM for the affine camera model, which is followed by a projective bundle adjustment stage utilizing the pinhole camera model.

This work proposes to replace the affine bundle objective by a variant of the object-space error—which is defined here as the “pseudo object-space error” (or *pOSE*, see Section 6.2)—better suited for perspective camera models. It is shown empirically that, so long as sufficiently clean image point tracks are provided, this pseudo object-space error (pOSE) retains wide convergence basins of useful optima and can therefore be used as the initial stage of the proposed bundle adjustment strategy from arbitrary initialization (see Section 6.3). This also implies that the major reason for SfM being a hard problem is getting the data association step right in order to remove false positive correspondences (further discussed in Section 6.4). The initial projective reconstruction is upgraded to the metric frame by employing a simplified self-calibration step (which assumes that at least approximate camera intrinsics are given).

Fig. 6.1 shows an illustrative outcome of the proposed method (i.e. the 3D point cloud and respective camera poses), which was obtained from random initialization. Fig. 6.2 demonstrates that the pseudo object-space error (which is parametrized by a blending weight $\eta \in [0,1]$) is able to capture the perspective structure of the pinhole camera model ($\eta \approx 0$) significantly better than the affine camera model ($\eta \gg 0$).

6.2 Pseudo Object Space Error (pOSE)

This section proposes a surrogate objective for bundle adjustment cost which keeps the bilinear matrix factorization structure in its residual. In short, the proposed *“pseudo object-space error”* cost (or *“pOSE”*) is a convex combination of the object space error and the affine factorization error,

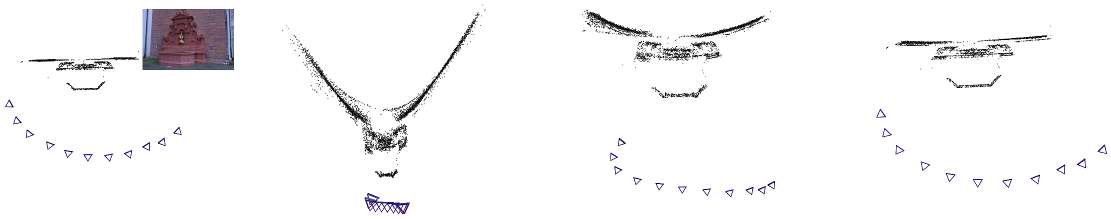
$$\ell_{\mathsf{OSE}} := \sum_{(i,j)\in\Omega} \|\mathsf{P}_{i,1:2}\mathbf{x}_j - (\mathbf{p}_{i,3}^\top \mathbf{x}_j)\mathbf{m}_{i,j}\|_2^2 \tag{6.2}$$

$$\ell_{\mathsf{Affine}} := \sum_{(i,j)\in\Omega} \|\mathsf{P}_{i,1:2}\mathbf{x}_j - \mathbf{m}_{i,j}\|_2^2 \tag{6.3}$$

$$\ell_{\mathsf{pOSE}} := (1-\eta)\ell_{\mathsf{OSE}} + \eta\,\ell_{\mathsf{Affine}} \tag{6.4}$$

for an $\eta \in [0,1]$. The notations $\mathsf{P}_{i,1:2} \in \mathbb{R}^{2\times 4}$ and $\mathbf{p}_{i,3} \in \mathbb{R}^4$ are used for the first two rows and the last row of the camera matrix $\mathsf{P}_i \in \mathbb{R}^{3\times 4}$, respectively.

Calling ℓ_{OSE} an object space error is a slight misnomer, since it penalizes squared point-line distances parallel to the image plane (instead of the shortest point-line distances perpendicular to the line). Nevertheless the terminology “object space error” is kept based on the fact, that the error is induced by distances in 3D object space. Further, the proper object space error is more suited to model spherical projections rather than projection onto the image plane.



(a) Ground truth (b) $\eta = 0.5$ (c) $\eta = 0.1$ (d) $\eta = 0.01$

Fig. 6.2 Metric reconstructions of Fountain-P11 from solving pOSE (see Section 6.2).

Note that ℓ_{pOSE} can be written as

$$\ell_{\mathsf{pOSE}} = \sum_{(i,j)\in\Omega} \left\| \begin{matrix} \sqrt{1-\eta}\left(\mathtt{P}_{i,1:2}\tilde{\mathbf{x}}_j - (\mathbf{p}_{i,3}^\top \tilde{\mathbf{x}}_j)\mathbf{m}_{i,j}\right) \\ \sqrt{\eta}\left(\mathtt{P}_{i,1:2}\tilde{\mathbf{x}}_j - \mathbf{m}_{i,j}\right) \end{matrix} \right\|_2^2, \tag{6.5}$$

which immediately reveals, that ℓ_{pOSE} is an instance of matrix factorization problems. It is also evident that the non-zero pattern of the Hessian (or Gauss-Newton approximated Hessian) is the same as for ℓ_{OSE}.

ℓ_{pOSE} avoids degenerate solutions of projective factorization

It can be shown that ℓ_{pOSE} (likely) avoids degenerate solutions. First, note that the degenerate projective optimum at which all cameras or homogeneous points collapse to zero is now penalized by the newly introduced affine term in (6.5). This means that, as long as there is sufficient contribution of the affine projection term, the previous degenerate optimum no longer exists. To verify this, each individual pOSE objective is first defined as $f_{i,j}$ such that $\ell_{\mathsf{pOSE}} = \sum_{(i,j)\in\Omega} f_{i,j}$. By defining $\mathbf{y} := \mathtt{P}_i\tilde{\mathbf{x}}_j$, $f_{i,j}$ can be written as

$$\begin{aligned} f_{i,j} &= (1-\eta)\|\mathbf{y}_{1:2} - y_3\mathbf{m}_{i,j}\|_2^2 + \eta\|\mathbf{y}_{1:2} - \mathbf{m}_{i,j}\|_2^2 \\ &= (1-\eta)\left(\|\mathbf{y}_{1:2}\|_2^2 - 2y_3\mathbf{m}_{i,j}^\top\mathbf{y}_{1:2} + \|y_3\mathbf{m}_{i,j}\|_2^2\right) + \eta\left(\|\mathbf{y}_{1:2}\|_2^2 - 2\mathbf{m}_{i,j}^\top\mathbf{y}_{1:2} + \|\mathbf{m}_{i,j}\|_2^2\right) \\ &= \|\mathbf{y}_{1:2}\|_2^2 - 2\mathbf{m}_{i,j}^\top\big((1-\eta)y_3+\eta\big)\mathbf{y}_{1:2} + (1-\eta)\|y_3\mathbf{m}_{i,j}\|_2^2 + \eta\|\mathbf{m}_{i,j}\|_2^2 \\ &= \|\mathbf{y}_{1:2} - \big((1-\eta)y_3+\eta\big)\mathbf{m}_{i,j}\|_2^2 - \big((1-\eta)y_3+\eta\big)^2\|\mathbf{m}_{i,j}\|_2^2 \\ &\qquad + \left((1-\eta)y_3^2+\eta\right)\|\mathbf{m}_{i,j}\|_2^2. \end{aligned} \tag{6.6}$$

Expanding the coefficient c of $\|\mathbf{m}_{i,j}\|_2^2$ from (6.6) yields

$$\begin{aligned} c &:= (1-\eta)y_3^2 + \eta - \big((1-\eta)y_3 + \eta\big)^2 \\ &= (1-\eta)y_3^2 + \eta - (1-\eta)^2 y_3^2 - 2(1-\eta)\eta y_3 - \eta^2 \\ &= (1-\eta)(1-(1-\eta))y_3^2 - 2(1-\eta)\eta y_3 + \eta(1-\eta) \\ &= (1-\eta)\left(\eta y_3^2 - 2\eta y_3 + \eta\right) \\ &= \eta(1-\eta)(y_3-1)^2. \end{aligned} \tag{6.7}$$

Hence, $f_{i,j}$ simplifies to

$$f_{i,j} = \|\mathbf{y}_{1:2} - \big((1-\eta)y_3 + \eta\big)\mathbf{m}_{i,j}\|_2^2 + \eta(1-\eta)(y_3-1)^2\|\mathbf{m}_{i,j}\|_2^2. \tag{6.8}$$

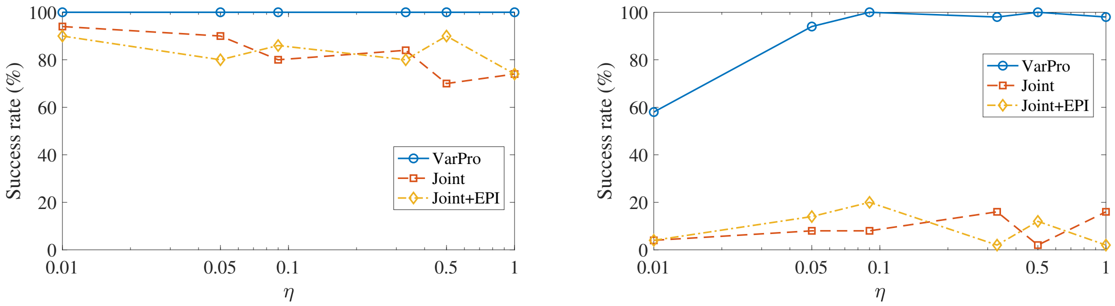
Inserting the expression for $\mathbf{y} = \mathtt{P}_i\tilde{\mathbf{x}}_j$ and summing over all image observations yields

$$\begin{aligned} \ell_{\mathsf{pOSE}} = &\sum_{(i,j)\in\Omega} \|\mathtt{P}_{i,1:2}\tilde{\mathbf{x}}_j - \left((1-\eta)\mathbf{p}_{i,3}^\top\tilde{\mathbf{x}}_j + \eta\right)\mathbf{m}_{i,j}\|^2 \\ &+ \eta(1-\eta)\sum_{(i,j)\in\Omega}\|\mathbf{m}_{i,j}\|^2\left(\mathbf{p}_{i,3}^\top\tilde{\mathbf{x}}_j - 1\right)^2. \end{aligned} \tag{6.9}$$

The first term is essentially an object space error (measured parallel to the image plane) between the 3D point and a distorted line-of-sight, which is a convex combination of pinhole and affine camera rays. The second term favours solutions (as long as $\|\mathbf{m}_{i,j}\|$ is non-zero) that have visible projective depths close to 1. Consequently, ℓ_{pOSE} can be understood as regularized and modified object space error.

Since ℓ_{pOSE} intrinsically favours solutions with positive projective depths for observed image points, it is not expected to produce degenerate solutions for general input data (although it is not guaranteed to avoid degeneracies for all possible inputs, e.g.$\mathbf{m}_{i,j} = 0$ for all i and j).

Alternative regularizations Degenerate solutions for $\{\mathtt{P}_i\}$ and $\{\tilde{\mathbf{x}}_j\}$ can be avoided by enforcing $\mathtt{P}_i\tilde{\mathbf{x}}_j \geq \delta$ (for all $(i,j) \in \Omega$) for some $\delta > 0$. Adding these constraint to ℓ_{OSE} makes the problem non-smooth and much more difficult to solve. Adding a barrier function, such as $-\alpha\sum_{(i,j)\in\Omega}\log\left(\mathtt{P}_i\tilde{\mathbf{x}}_j - \delta\right)$ for an $\alpha > 0$, would require either using joint optimization or nonlinear VarPro. As mentioned above, joint optimization frequently leads to stalling behaviour, and nonlinear VarPro has been demonstrated to have a significantly smaller basin of convergence (see Section 5.1). Consequently, the work rules out adding inequality constraints on the projective depths and respective penalizers or barrier functions.



(a) Fountain-P11

(b) Small dinosaur

Fig. 6.3 Some demonstration of success rates of VarPro and joint optimization algorithms for different η values. The success rate counts how many times out of some fixed number of runs each algorithm yields the best observed optimum for each setting of η from arbitrary initialization. On Fountain-P11, which is a relatively "easy" dataset with dense and mostly unique correspondences, the success rates are less sensitive to the value of η across all tested algorithms. However, on the small dinosaur sequence, which is a more difficult dataset due to its banded missing data pattern, the success rates of all algorithms decrease as η decreases, and it can only be solved efficiently by VarPro above certain value of η.

Instead of defining the target objective as a convex combination of an object-space error and an affine factorization error, one can consider directly a regularized object-space error ℓ_{rOSE} by combining ℓ_{OSE} with a term penalizing visible projective depths,

$$\begin{aligned}\ell_{\mathsf{rOSE}} = (1-\eta) \sum_{(i,j)\in\Omega} \|\mathsf{P}_{i,1:2}^{\top}\tilde{\mathbf{x}}_j - (\mathbf{p}_{i,3}^{\top}\tilde{\mathbf{x}}_j)\mathbf{m}_{i,j}\|^2 \\ + \eta \sum_{(i,j)\in\Omega} (\mathbf{p}_{i,3}^{\top}\tilde{\mathbf{x}}_j - 1)^2. \end{aligned} \tag{6.10}$$

Note that ℓ_{rOSE} essentially constrains the projective depths of all observed image points, not only for a subset as required by the GPRT reviewed in Section 6.5. If there exists a perfect solution with zero object-space error (and therefore $\mathsf{P}_{i,1:2}^{\top}\tilde{\mathbf{x}}_j = (\mathbf{p}_{i,3}^{\top}\tilde{\mathbf{x}}_j)\mathbf{m}_{i,j}$ for all $(i,j) \in \Omega$), then ℓ_{pOSE} and ℓ_{rOSE} reduce to

$$\begin{aligned}\ell_{\mathsf{pOSE}} &= \eta \sum_{(i,j)\in\Omega} \|\mathsf{P}_{i,1:2}\tilde{\mathbf{x}}_j - \mathbf{m}_{i,j}\|_2^2 \\ &= \eta \sum_{(i,j)\in\Omega} \|\mathbf{m}_{i,j}\|_2^2 \left(\mathbf{p}_{i,3}^{\top}\tilde{\mathbf{x}}_j - 1\right)^2 \end{aligned} \tag{6.11}$$

$$\ell_{\mathsf{rOSE}} = \eta \sum_{(i,j)\in\Omega} (\mathbf{p}_{i,3}^{\top}\tilde{\mathbf{x}}_j - 1)^2. \tag{6.12}$$

Since observations $\mathbf{m}_{i,j}$ are on the image plane (with depth 1), for regular but not extremely wide field-of-view cameras, $\|\mathbf{m}_{i,j}\| \leq 1$, and ℓ_{pOSE} perturbs the object-space error ℓ_{OSE} less than ℓ_{rOSE} in this case. Assessing the pros and cons of ℓ_{rOSE} over ℓ_{pOSE} is a subject of future work. Inspired by one set of sufficient conditions for projective reconstruction [Nasihatkon et al., 2015], this work introduces ℓ_{GPRT}, which penalizes projective depths in analogy to ℓ_{rOSE}, but only for indices (i, j) being on a step-like matrix. This choice essentially fixed the projective frame of the reconstruction.

Alternative objectives The choice of using a perturbed object-space error is largely motivated by the bilinear nature of the cost function and the availability of efficient (VarPro) algorithms. As shown in Section 5.1, the classical bundle adjustment objective has a very narrow basin of convergence to reach a good solution. For similar reasons, this work ruled out employing convex objectives such as ℓ_1 and Huber cost functions (e.g. [Dalalyan and Keriven, 2009; Seo et al., 2009; Zach and Pollefeys, 2010]).

VarPro vs. joint optimization In Section 4.2.3 it was argued, that in affine factorization for structure-from-motion the VarPro method and joint optimization (i.e. using the Levenberg-Marquardt method jointly w.r.t P_i and $\tilde{\mathbf{x}}_j$) behave very differently: joint optimization suffers from "stalling" behaviour whenever affine camera rays are close to being parallel (since a small update of such camera parameters yield large updates for the 3D points, which is prohibited by Levenberg damping). VarPro avoids this shortcoming by allowing 3D points to freely follow the updates of camera parameters in all cases.

The situation for ℓ_{pOSE} is similar to the affine setting, but one has stronger conditions for camera rays to be parallel: in the affine setting, the parallelity of optical axis is sufficient, whereas in the projective scenario the 3×3 submatrices need to (approximately) satisfy $\mathsf{P}_i(1\!:\!3, 1\!:\!3) \propto \mathsf{P}_j(1\!:\!3, 1\!:\!3)$. Hence, one might expect that joint optimization shows stalling behaviour less often in the projective setting than in the affine setting, but empirically joint optimization is still inferior to VarPro for the tested choices of η (see Fig. 6.3). Since the success rate of VarPro increases for larger values of η, the selected value for η represents a tradeoff between success rate and the amount of geometric distortion.

6.3 Stratified bundle adjustment

This section now illustrates a multistage pipeline for initialization-free bundle adjustment. The key idea is simple—given a set of point tracks and randomly sampled camera and point parameters, seek for an initial solution (camera poses and 3D structure) by solving the pOSE

Algorithm 5 The proposed initialization-free BA pipeline

Input: a set of geometrically-verified point tracks

1. Solve the L2-norm pOSE problem (see Section 6.2) from arbitrarily sampled cameras and points using VarPro.
2. Refine the solution using a robustified projective BA algorithm incorporating nonlinear VarPro, and discard points with large reprojection errors.
3. Upgrade the above solution to metric and throw away points which do not satisfy cheirality constraints.
4. Refine the solution in the metric space.

Output: metric camera poses and 3D reconstruction

optimization problem (see Section 6.2), refine it in the projective frame, upgrade the solution to metric followed by a final metric refinement step. These steps are also summarized in Algorithm 5.

pOSE optimization As shown in Section 6.2, pOSE is designed to closely resemble a projective factorization cost as well as keeping the objective bilinear. This is because bilinear problems have been empirically shown to have wide basin of convergence for the variable projection (VarPro) family of algorithms [Chen, 2008; Gotardo and Martinez, 2011; Okatani et al., 2011]. In this work, Ruhe and Wedin Algorithm 2 [Kaufman, 1975; Ruhe and Wedin, 1980] is used, which is a variant of the VarPro algorithm that uses an approximated Jacobian, and is therefore easier to implement as shown in Section 4.2.3 (and also slightly more efficient). Recall that incorporating robustness at this stage would a) introduce many local minima and b) destroy the bilinear structure convenient for VarPro. This work attempts to mitigate this issue by generating mostly-inlier tracks using the approach illustrated in Section 6.4.

In this stage, each 3D point $\tilde{\mathbf{x}}_j$ is parameterized in Euclidean coordinates (i.e. $\tilde{\mathbf{x}}_j = [\mathbf{x}_j^\top\, 1]^\top$) as incorporating homogeneous parameterization requires projecting the local Jacobian of each 3D point to its tangent space, which is computationally costly and has no noticeable reduction in the number of iterations.)

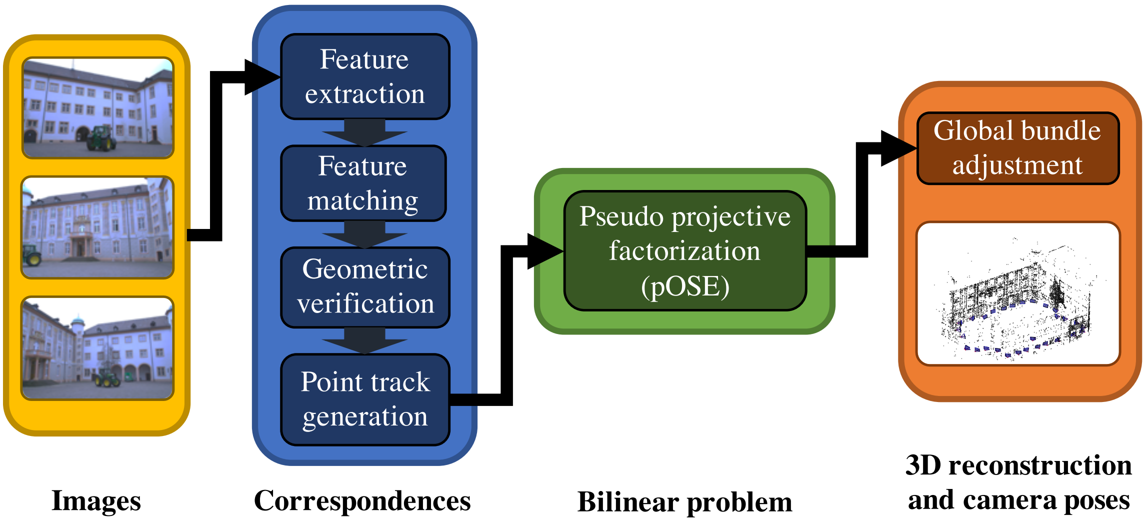


Fig. 6.4 A potential application of pOSE for widening the basin of convergence.

Projective refinement Once cameras and points are obtained by optimizing ℓ_{pOSE}, they are refined by minimizing the gold standard [Hartley and Zisserman, 2004] objective

$$\sum_{(i,j)\in\Omega} \rho \left(\left\| \frac{\mathrm{P}_{i,1:2}\tilde{\tilde{\mathbf{x}}}_j}{\mathbf{p}_{i,3}^\top \tilde{\tilde{\mathbf{x}}}_j} - \mathbf{m}_{i,j} \right\|^2 \right), \tag{6.13}$$

where $\rho : \mathbb{R} \to \mathbb{R}$ represents an isotropic robust kernel. Although (5.1) can be solved by jointly optimizing over the cameras and points, Strelow [2012b]'s nonlinear VarPro is implemented, which is an extension of standard VarPro to nonseparable problems (i.e. nonlinear in both sets of variables) and is demonstrated to have a slightly wider basin of convergence than joint optimization (see Section 5.5). Further, it has been shown in Section 5.1 that the iteration complexity of VarPro is approximately equal to that of standard joint optimization with embedded point iterations [Jeong et al., 2010], which simply amounts to performing additional triangulations after each joint update of camera and point parameters.

The optimization is carried out in homogeneous coordinates incorporating the Riemannian manifold optimization described in Section A.2.5 (similar to local parameterization [Agarwal et al., 2014; Hartley and Zisserman, 2004]). After the refinement, points that have a maximum reprojection error of larger than 2 pixels are discarded.

Metric upgrade implementation The resulting refined projective camera matrices need to be upgraded to a metric frame to satisfy the $SE(3)$ manifold constraints after taking out the calibration matrices. Since it is assumed that camera intrinsics are known a priori, the metric

Algorithm 6 Generating point tracks from images

Input: images and camera intrinsics

1. Obtain pairwise feature matches using SIFT.
2. Verify matches using two view geometric constraints.
3. Verify matches using triplet filtering.
4. Convert pairwise matches to optimal point tracks by using Olsson and Enqvist [2011]'s algorithm.
5. Mask out track segments that are not consistent with the estimated epipolar geometries.

Output: point tracks

upgrade method reviewed in Section 2.2.5 is applied with some modifications to allow VarPro to be used once more.

This work uses the variable projection [Golub and Pereyra, 1973] method described in Section 2.1 to solve this efficiently. In such case, eliminating α_i optimally in (2.34) yields a similar problem to (2.35), which is solved by Pollefeys et al. [1999], but they are not the same.

Metric refinement The aforementioned metric upgrade procedure usually increases the total reprojection error as metric cameras have lower degrees of freedom than projective ones. Hence, an additional step is required to refine camera poses and 3D structure. For this purpose, the refinement step minimizes the gold standard [Hartley and Zisserman, 2004] metric reprojection error

$$\sum_{(i,j)\in\Omega} \rho\left(\|\pi\left(\mathtt{K}_i[\mathtt{R}_i \mid \mathbf{t}_i]\tilde{\tilde{\mathbf{x}}}_j - \mathbf{m}_{i,j}\right)\|_2^2\right), \tag{6.14}$$

where $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ is the projection function to bring 3D points onto the image plane. Cameras and points are jointly optimized since there is no visible advantage in employing nonlinear VarPro.

6.4 Generating point tracks from matches

The stratified BA pipeline illustrated in Section 6.3 requires point tracks consisting mostly of inliers. This is due to the limitation that the pOSE objective (6.4) (or any reasonable objective) cannot be robustified without sacrificing the convergence basin of VarPro. Hence, the goal of the method described in this section is to generate as clean tracks as possible before feeding them into the proposed BA pipeline. Unfortunately, this is a non-trivial problem on its own for two reasons. First, the estimated epipolar geometries may be grossly incorrect due to perceptual aliasing (e.g. [Zach et al., 2010]). Second, naively connecting matches across multiple images may form an inconsistent track in which more than one feature from the same image could participate (c.f. Fig. 6.5). Solving these issues is still an active research problem in computer vision (e.g. [Maset et al., 2017] for a recent development). The employed approach focuses on scalability and is summarized in Algorithm 6.

Two view geometric verification The first step is a standard 2-view geometric verification step using Nister [2004]'s 5-point algorithm (also reviewed in Section A.4.4) to remove the "easy" outliers, and the outputted essential matrices are further refined as illustrated in A.4.5 using the surviving inlier matches. This threshold is set to be 1.5 px for 2MP images and scale this threshold linearly with the image size.

Triplet filtering Outliers arising from repetitive structures such as windows may remain and may lead to grossly incorrect essential matrices. Preliminary runs showed that the stratified pipeline in Algorithm 5 can fail if only pairwise geometric verification is applied on scenes with high repetitive structures (e.g. Castle-P19/30). Many of these false positive relative poses can be detected by checking the self-consistency of relative rotations for image triplets [Moulon et al., 2013; Sweeney et al., 2015; Zach et al., 2010]. The scheme requires the chained relative rotations to have at most a 5° residual angle. This work refrains from including larger loops [Enqvist et al., 2011; Zach et al., 2010] due to its high computation cost.

The method used in this work iteratively discards the currently most violating image pair (i.e. the one participating in the largest number of inconsistent triplets), until all triplets are consistent. Thus, the utilized triplet filtering method is more aggressive in removing image pairs than those utilized by Moulon et al. [2013] and Sweeney et al. [2015], which only require an image pair to participate in at least one consistent triplet.

Point tracking algorithm After matches are geometrically verified using two-view and three-view constraints, Olsson and Enqvist [2011]'s algorithm illustrated in Algorithm 4 is employed to generate algebraically-optimal consistent tracks. In summary, when connecting pairwise matches leads to two or more features from the same image being joined (e.g. Fig. 6.5), this algorithm selects one based on a track reliability criterion measured by the minimum number of feature matches.

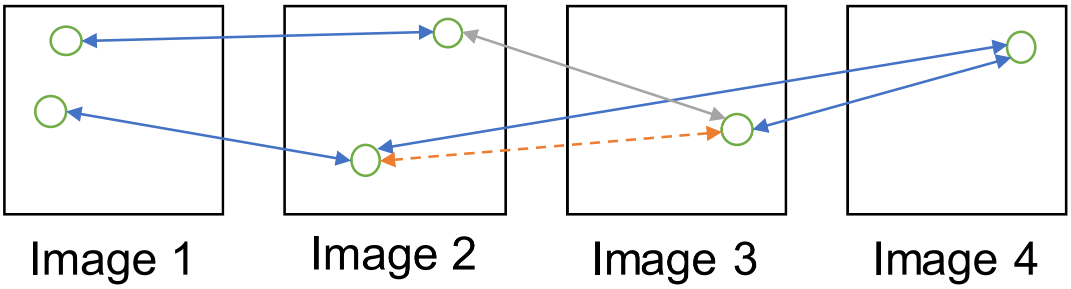
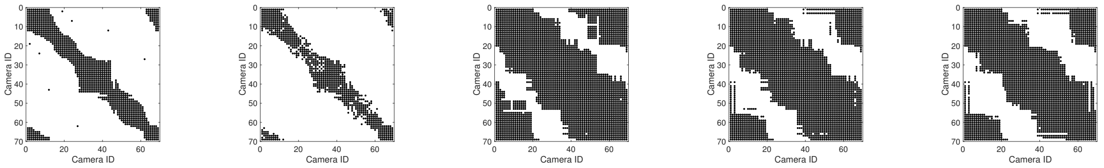


Fig. 6.5 A simple illustration of a potential issue arising in generating point tracks. The solid blue and grey lines represent 2-view geometrically-verified matches. Naively joining these matches leads to two features participating in image 2. Hence, a match (solid grey) is discarded according to some predefined rule or algorithm. Consequently, this induces a new feature match between image 2 and 3 (orange). This work argues that these implicitly arising matches should be verified to satisfy existing 2-view geometric constraints.



(a) 2-views (b) Triplets (c) Initial track (d) Revisit 2-views (e) Final tracks

Fig. 6.6 Evolution of the camera adjacency matrix for Vercingetorix during the track generation stage illustrated in 2.2.1. (d) is used for the experiment and (e) is outputted from the best solution obtained. Note that connecting pairwise matches to form point tracks usually **densifies** adjacency matrix.

Revisiting the geometric constraints As shown in Fig. 6.5, generated point tracks may induce new matches indirectly, depending on which matches pass the local verification steps. It is possible that the induced feature matches do not satisfy the respective two-view epipolar and cheirality constraints. This work proposes to revisit each of these newly created matches and verify that they satisfy all desired constraints. The matches that fail this test are treated as missing data. This is only a partial solution as multiple real point tracks may still be incorrectly merged into a single point track. Ultimately, one should incorporate geometric constraints in measuring the point track reliability.

Fig. 6.6 shows the evolution of the camera adjacency matrix over the track generation stage illustrated in 6.4. This shows that converting triplet-verified pairwise matches to point tracks inevitably **densifies** connections. Fig. 6.6 (d) also shows that revisiting two-view matches removes many geometrically inconsistent connections, taking one step closer to the baseline adjacency matrix shown in 6.6 (e).

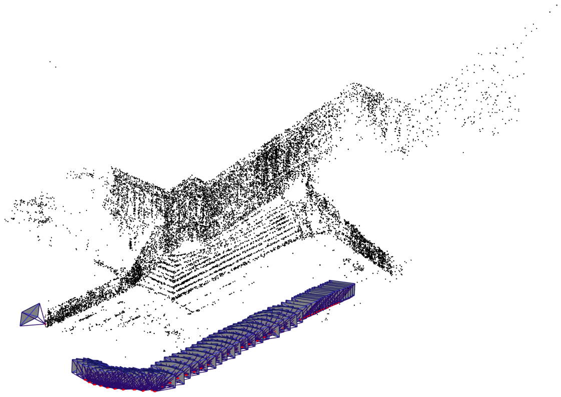


Fig. 6.7 An almost-successful reconstruction of Alcatraz Courtyard from arbitrary initialization. As the leftmost camera centre is far off from its baseline position, this is considered a failure run.

Table 6.1 Optimization settings used for the experiments

Property	Value
Relative function tolerance	10^{-9}
Relative gradient tolerance	10^{-6}
Relative parameter tolerance	10^{-9}
Initial damping factor	10^{-1}
Max. num. of iterations for pOSE	400
Max. num. of iterations for projective BA	400
Max. num. of iterations for metric upgrade	50
Max. num. of iterations for metric BA	200
Min. num of inliers threshold for 2-view	30
Min. length of each point track	4

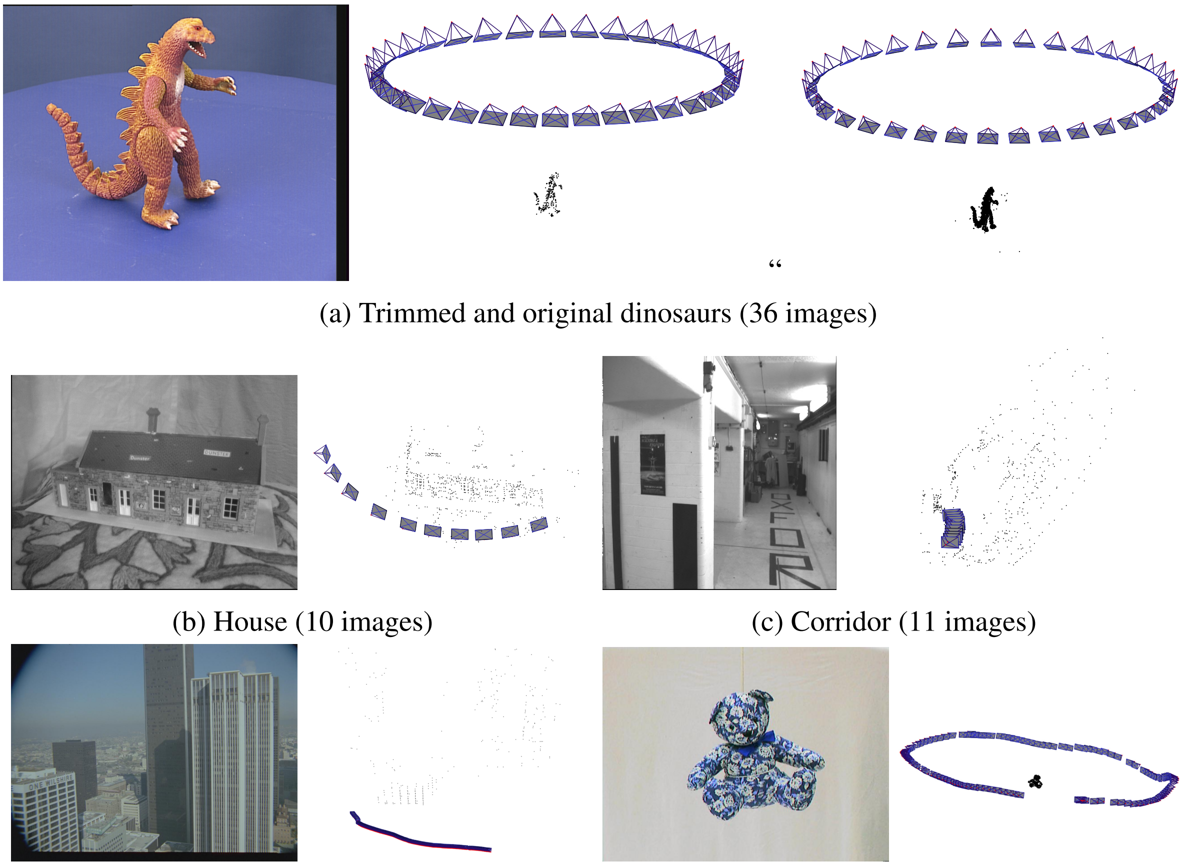
6.5 Experimental procedures

The experiments were designed to

1. empirically observe the size of the convergence basin of the pOSE-based stratified BA strategy (proposed in Section 6.3) on point tracks with mostly inliers,

2. check whether the GPRT constraints [Nasihatkon et al., 2015] are still sufficient conditions for solving the object space error on sequences with noise and missing data, and

3. whether the pOSE-based BA pipeline equipped with the point track generation algorithm from Section 6.4 can accurately solve real SfM problems.



(a) Trimmed and original dinosaurs (36 images)

(b) House (10 images)

(c) Corridor (11 images)

(d) Wilshire (190 images)

(e) Blue bear (196 images)

Fig. 6.8 Successful reconstructions of some of the classic sequences from arbitrary initialization of cameras and 3D points. Tracks are generated using the strategy described in Section 6.4.

To answer these questions, two experiments were carried out on small to medium-sized real SfM sequences. For convenience, pOSE will refer to the pOSE-based stratified BA pipeline, and GPRT will refer to the same pipeline but with the first stage objective replaced by the object space error (6.2) with the GPRT constraints (using a step-like matrix, c.f. Section 6.2).

Implementation Experiments were conducted on a machine with Intel Core i7-7800X CPU (6 cores) and 32GB RAM. For the feature detection and matching stage, this work used a modified version of COLMAP [Schönberger and Frahm, 2016] to generate two-view verified pairwise matches and output corresponding essential matrices employing an exhaustive matching technique. All other stages were either implemented in MATLAB or using the Google Ceres Solver [Agarwal et al., 2014] library patched to enable the VarPro method [Golub and Pereyra, 1973] according to the guidelines in Section 4.2.3.

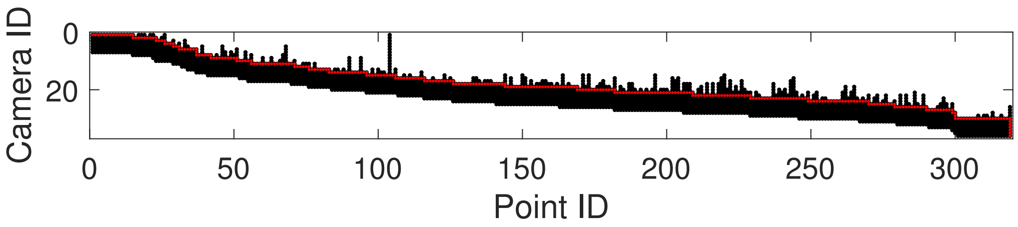


Fig. 6.9 Visibility (black) of the trimmed dinosaur dataset and our steplike mask (red) used to attempt GPRT [Nasihatkon et al., 2015] in presence of missing data. Note that the above steplike mask is only constraining depths which are visible (i.e. overlays on the black region).

Determining the value of η In order to determine an appropriate value of η, a non-uniform grid search was carried out on Strecha et al.'s benchmark datasets (listed in Table 6.3), which have ground truth camera positions, the small dinosaur and vercingetorix datasets, which are circular motion sequences with more-affine views, and Olsson and Enqvist's strongly-perspective Lund cathedral dataset (small trimmed version). The considered values of η were 0.01, 0.05, 0.09, 0.33, 0.5 and 1. For each value of η on each dataset, the number of runs (out of 20 random initialization points) yielding accurate 3D reconstructions (through the stratified bundle adjustment strategy in Section 6.3) was recorded. While "easier" datasets such as Fountain-P11 and Herz-Jesu-P8 retrieved accurate 3D reconstructions across all the considered values of η, more-affine circular motion sequences had convergence issues with pOSE for $\eta = 0.01$ (e.g. Fig. 6.3b). On the other hand, the perspective Lund cathedral sequence did not have convergence issues during the pOSE stage but the pOSE solution with high η values led to perturbed 3D reconstructions. By considering the worst success rate performance for each η value, it was decided to set η to 0.05.

Creating a steplike mask for GPRT The general projective reconstruction theorem (GPRT) proposed by Nasihatkon et al. [2015] is proven for the case of noise-free and fully-visible measurements. Although further investigation is needed to properly generalize GPRT for handling noise and missing data, this work attempted to extend this simply by generating a steplike mask that overlays on top of the visible region of data (e.g. Fig. 6.9). Such implementation is from the empirical finding that constraining depths of visible projections performs better than constraining depths of invisible projections. (Such phenomenon could be due to invisible points actually lying behind the camera, meaning that the depth constraints on missing data could lead to incorrect solutions.)

Optmization settings Table 6.1 shows the optimization settings used in our experiments.

Number of experimental runs The results were obtained for each dataset from 50 runs of pOSE-based stratified bundle adjustment (BA) and 20 runs of GPRT-based stratified BA. It was verified that an increase in the number of GPRT-based BA runs does not change the overall trend observed in Tables 6.2, 6.4 and 6.5.

Baseline For each dataset in Tables 6.4 and 6.5, the baseline data used was either a set of ground truth measurements (Strecha et al.'s sequences: Fountain-P11, Entry-P10, Herz-Jesu-P*s and Castle-P*s) or is obtained from COLMAP [Schönberger and Frahm, 2016], which is a robust and reliable incremental SfM pipeline. (For datasets in Table 6.2, point tracks were converted to custom inlier feature matches before going through an incremental pipeline.)

Classifying success Each run is deemed successful if its mean camera position deviation from the baseline is less than two times that of the best minimum (observed in total of N runs). In terms of equations, if the mean camera position error from the baseline in run k is denoted by $\bar{\varepsilon}_k$, the total number of runs made by N ($\approx$ 100) and the *success threshold* by $\varepsilon_{\text{success}}$, then

$$\begin{aligned} \bar{\varepsilon}_{\min} &:= \min(\bar{\varepsilon}_1, \ldots, \bar{\varepsilon}_N), \qquad \text{and} \\ \varepsilon_{\text{success}} &= 2\bar{\varepsilon}_{\min}. \end{aligned}$$

Each reconstruction algebraically classified *successful* were double checked visually. It is believed that the choice of factor 2 is strict (see Fig. 6.7 for a failure example) but somewhat arbitrary, and future work should introduce more rigorous statistical analysis such as median absolute deviation (MAD) of reached optima. Nevertheless, for most datasets (except for Castle-P*s), there are usually finite number of distinct observable basins which are very close in terms of camera position error such that the corresponding success rates are not very sensitive to changes in this factor.

Comparison to other SfM pipelines As Strecha et al. [2008] provides datasets with ground truth camera poses, these are frequently used as benchmarks for comparing the accuracy of 3D reconstructions. To verify the feasibility of reconstructions obtained from the strategy described in Section 6.3, the mean camera position errors (with scale ambiguity) were compared against the results obtained by other structure-from-motion pipelines, namely VisualSfM [Wu et al., 2011], Theia [Sweeney, 2017], COLMAP [Schönberger and Frahm, 2016], Olsson and Enqvist's method and OpenMVG [Moulon et al., 2013]. VisualSfM and COLMAP are incremental pipelines, starting from a pair of images (feature matches) and incrementally

Table 6.2 A list of small classic inlier point tracks used for the experiment and corresponding results. $\bar{t}$ represents mean runtime. pOSE-based BA pipeline has large success rates for inlier tracks whereas the GPRT-based pipeline fails on many of these datasets.

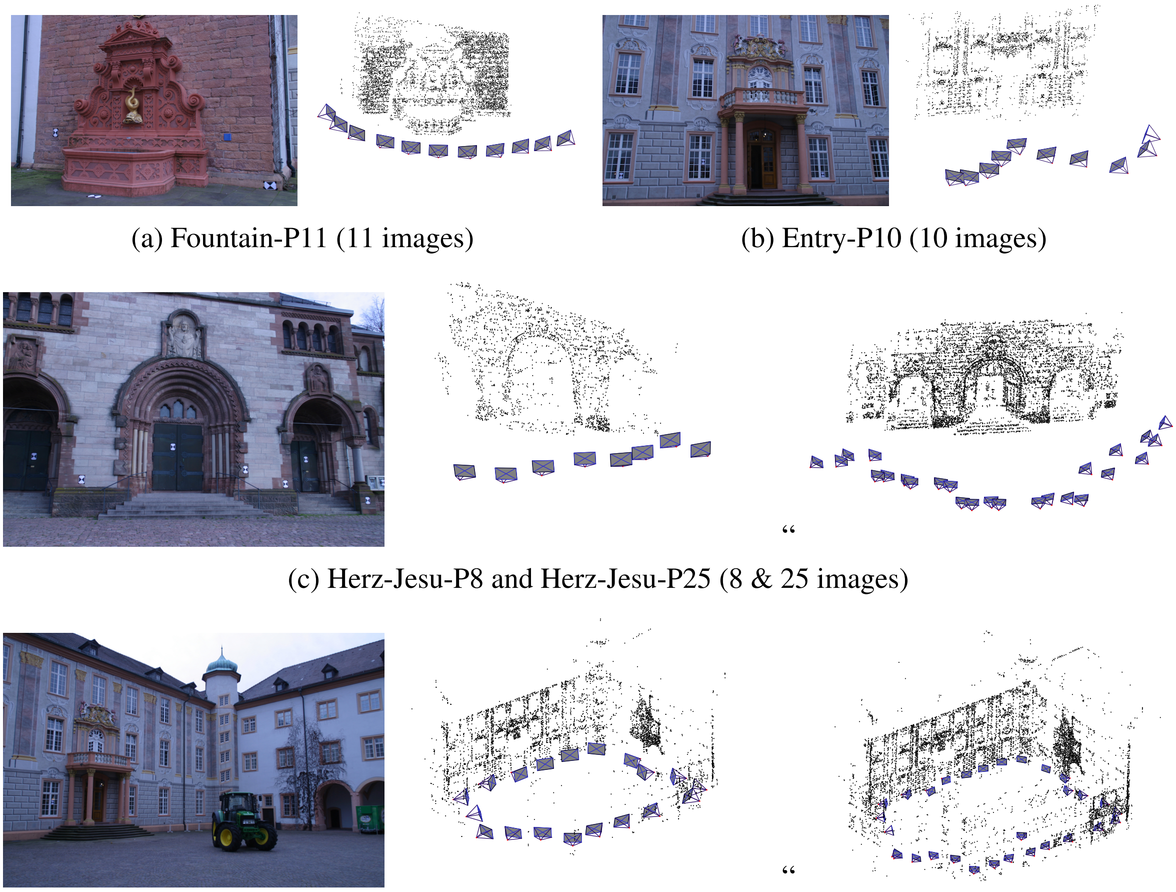
			Fill	pOSE		GPRT	
Sequence	# img	# pts	(%)	SR	$\bar{t}(s)$	SR	$\bar{t}(s)$
House	10	672	42.4	100	4.2	35	5.8
Corridor	11	737	49.8	100	1.7	15	9.0
Dinosaur (trimmed)	36	319	23.1	96	3.0	0	10.1
Dinosaur	36	4983	9.2	98	6.9	0	32.3
Wilshire	190	411	39.3	100	38.1	0	272.2
Blue bear	196	2480	19.3	80	70.7	0	337.7

Table 6.3 Accuracy comparison of the stratified BA strategy using pOSE against other pipelines on Strecha et al. [2008]'s benchmark datasets. The castles are more difficult due to repetitive structures. The reported values are the mean errors in the camera positions (in mm). N/A implies no successful run. The results of Theia are cited from Sweeney [2017], and the results of OpenMVG, Olsson and VisualSfM are cited from Moulon et al. [2013]; Sweeney et al. [2015], both of which report the same values.

Seq. \ Pipeline	pOSE	GPRT	Theia	COLMAP	OpenMVG	Olsson	VisualSfM
Fountain-P11	2.8	2.8–4.5	2.4	2.8	2.5	**2.2**	7.6
Entry-P10	7.1	6.4–7.0	6.0	6.3	**5.9**	6.9	63.0
Herz-Jesu-P8	3.4–3.9	3.8–4.5	**3.1**	4.1	3.5	3.9	19.3
Herz-Jesu-P25	5.2	**5.1**	**5.1**	5.2	5.3	5.7	22.4
Castle-P19	**24.5**–41.1	N/A	25.3	24.9	25.6	76.2	258.0
Castle-P30	**21.7**–26.0	N/A	**21.7**	23.2	21.9	66.8	522.0

registering new views (with bundle adjustment at each iteration). The main difference between them is that VisualSfM does not support Nistér's 5-point algorithm for geometric verification of feature matches and consequently deals with potentially lower quality matches when there are degenerate motions to the conventional 8-point algorithm. Theia, OpenMVG and Olsson and Enqvist's method are all global SfM pipelines, but each have its custom motion averaging method applied for initializing global camera poses. The best results obtained by Theia is cited from its official website[1], while the best results obtained by OpenMVG, VisualSfM and Olsson and Enqvist's pipelines are cited from the literature [Moulon et al., 2013; Sweeney et al., 2015] (both report the same values). The results for COLMAP are not available in the literature, and therefore the reported values are generated using the feature matches obtained in this work.

[1] http://theia-sfm.org/performance.html



(a) Fountain-P11 (11 images)

(b) Entry-P10 (10 images)

(c) Herz-Jesu-P8 and Herz-Jesu-P25 (8 & 25 images)

(d) Castle-P19 and Castle-P30 (19 & 30 images)

Fig. 6.10 Successful reconstructions of Strecha et al. [2008]'s sequences from arbitrary initialization of cameras and 3D points. Tracks are generated from Section 6.4.

Robust kernel The Cauchy loss function described in Section A.2.4 is applied in performing robust projective and metric refinements. This kernel is more robust to outliers than L_1 or Huber loss functions but less than Smooth truncated quadratic or Geman-McClure kernels (see Section A.2.4 for a list of widely-used loss functions).

In this work, Trigg's correction [Triggs et al., 2000] from Section A.2.4 is employed to incorporate the loss function as it is already implemented in the Ceres solver [Agarwal et al., 2014]. Future work could incorporate the joint robust optimization strategy proposed by Zach [2014].

Rotation parameterization Rotation is parameterized using the axis-angle representation from Section A.3.4, employing the Rodrigues formula to rotate points. The Jet library in

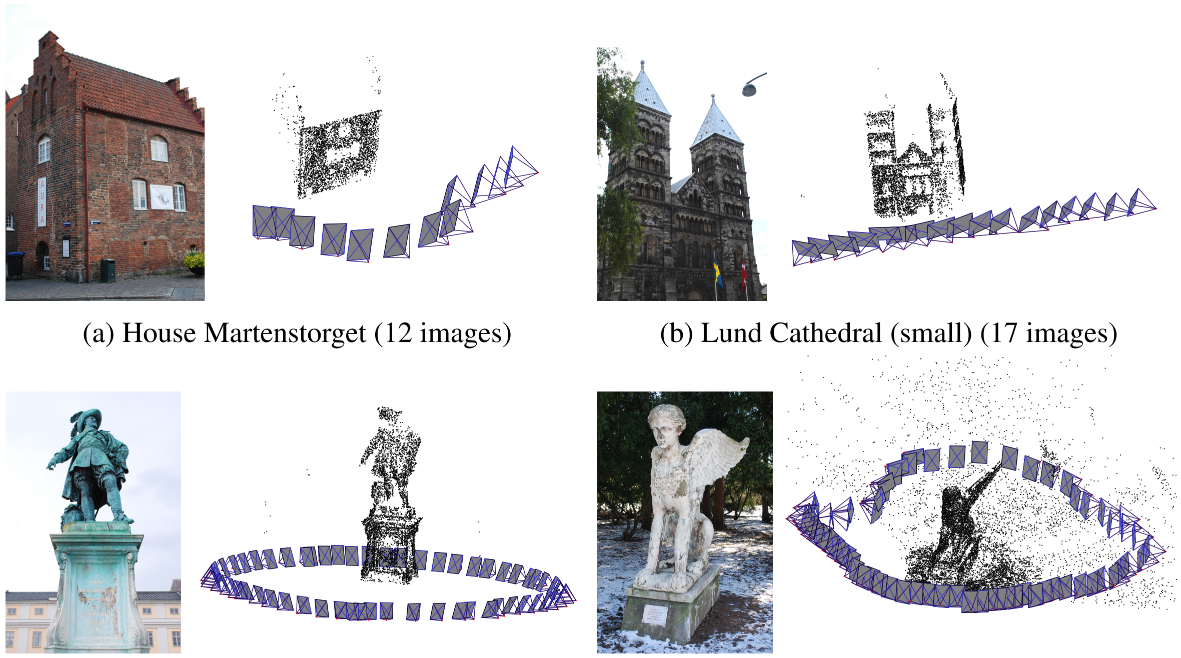
				pOSE		GPRT	
Sequence	# img	# pts	Fill (%)	SR (%)	$\bar{t}$ (s)	SR (%)	$\bar{t}$ (s)
Fountain-P11	11	9181	50.3	100	6.5	15	70.3
Entry-P10	10	4270	55.5	100	5.2	25	27.5
Herz-Jesu-P8	8	3553	60.7	100	3.4	30	16.1
Herz-Jesu-P25	25	12469	27.3	100	12.9	10	86.2
Castle-P19	19	5144	27.2	94	21.4	0	21.3
Castle-P30	30	11531	20.4	94	32.1	0	73.6
House Martenstorget	12	5934	53.2	100	11.3	20	47.8
Lund Cathedral (small)	17	9400	30.6	92	19.2	10	75.6
Gustav II Adolf	57	9562	12.3	100	23.7	20	82.0
Univ. of West. Ontario	57	6742	14.4	100	25.2	5	92.6
Vercingetorix	69	5231	10.3	78	15.0	10	60.0
Lund University Sphinx	70	22770	9.9	96	74.7	0	150.3
Alcatraz courtyard	133	31558	9.7	94	145.5	0	1653.8
Water tower	173	25531	8.0	90	274.6	40	1932.1
Pumpkin	209	25962	4.1	100	147.5	0	869.7

Table 6.4 A list of real SfM datasets and corresponding results using inlier tracks generated by COLMAP [Schönberger and Frahm, 2016]. $\bar{t}$ is mean runtime for executing the stratified BA pipelines.

the Google Ceres Solver [Agarwal et al., 2014] allows automatic differentation of rotation parameters.

Initialization Since VarPro optimally eliminates the 3D structure from the pOSE residual (6.4), only the camera parameters (i.e. $\{\mathtt{P}_i\}$) need to be sampled. Other previous work in matrix factorization [Buchanan and Fitzgibbon, 2005; Chen, 2008; Okatani et al., 2011] simply sampled these from an isotropic Gaussian distribution with mean $\mathbf{0}$ and variance $\mathtt{I}$ in the pixel coordinates. A similar sampling method is employed here but set the the mean of the sampling distribution to be at the centre of the images. Additionally, each row of the sampled camera matrix was normalized to improve numerical stability. This procedure is analogous to taking a sample from the uniform distribution on a 4D hypersphere [Dezert and Musso, 2001].

Camera position error For each run on each dataset, the experiment measured the mean deviation of camera centres from the corresponding baseline values. Since metric reconstructions still have scale ambiguities, the solutions were compared to the ground truth or baselines using the optimal similarity transform (reviewed in Section A.2.1).



(a) House Martenstorget (12 images)

(b) Lund Cathedral (small) (17 images)

(c) Gustav II Adolf (57 images)

(d) Lund University Sphinx (70 images)

Fig. 6.11 Successful reconstructions of smaller Olsson's from arbitrary initialization of cameras and 3D points. Tracks are generated from Section 6.4.

This metric is useful when detecting failure cases where a solution has a low average reprojection error but has one or two cameras positioned incorrectly. (E.g. cameras looking behind a planar scene, which can sometimes occur in Castle-P*s.)

A downside of having metric ambiguity resolved via optimal similiarity transform is that any global translational drift present will go unnoticed. Hence, this measure is likely to output more optimistic value than in reality. Nevertheless, it is a useful metric for comparing reconstructions used by others [Moulon et al., 2013; Sweeney et al., 2015].

6.6 Results and discussions

In the first experiment, the performance of pOSE and GPRT are compared on point tracks known to be free of outliers. Some of these tracks are publicly available classic datasets (e.g. dinosaur), while others are derived from Olsson and Enqvist [2011]'s and Strecha et al. [2008]'s digital camera images piped through a robust incremental SfM pipeline (COLMAP made by Schönberger and Frahm [2016]). pOSE and GPRT are run for a fixed number of runs, each from arbitrarily-sampled cameras and points, and report the fraction of runs each algorithm reaches the best solution up to predefined tolerance value in terms of camera position errors.

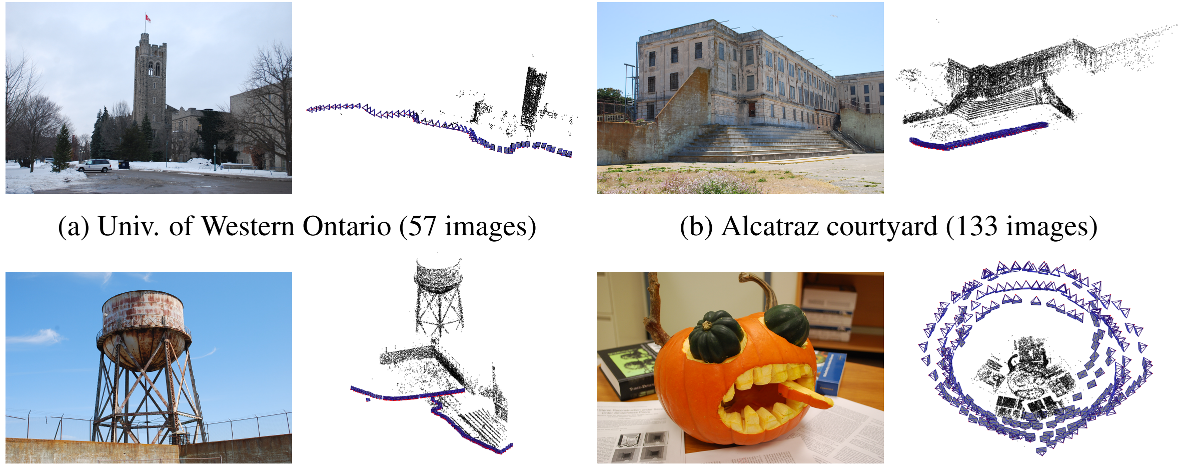
				pOSE		GPRT	
Sequence	# img	# pts	Fill (%)	SR (%)	$\bar{t}$ (s)	SR (%)	$\bar{t}$ (s)
Fountain-P11	11	9181	50.3	100	6.5	30	70.3
Entry-P10	10	4270	55.5	98	5.2	5	27.5
Herz-Jesu-P8	8	3553	60.7	100	3.4	15	16.1
Herz-Jesu-P25	25	12469	27.3	100	12.9	5	86.2
Castle-P19	19	5144	27.2	88	21.4	0	21.3
Castle-P30	30	11531	20.4	100	32.1	0	73.6
House Martenstorget	12	5934	53.2	100	11.3	15	47.8
Lund Cathedral (small)	17	9400	30.6	96	19.2	15	75.6
Gustav II Adolf	57	9562	12.3	86	23.7	35	82.0
Univ. of West. Ontario	57	6742	14.4	98	25.2	35	92.6
Vercingetorix	69	5231	10.3	92	15.0	15	60.0
Lund University Sphinx	70	22770	9.9	76	74.7	5	150.3
Alcatraz courtyard	133	31558	9.7	100	145.5	0	1653.8
Water tower	173	25531	8.0	76	274.6	35	1932.1
Pumpkin	209	25962	4.1	100	147.5	0	869.7

Table 6.5 A list of real SfM datasets and corresponding results using the point tracks generated by Algorithm 6. $\bar{t}$ is mean runtime for executing the stratified BA pipelines.

Table 6.2 and Table 6.4 shows that pOSE has large basin of convergence across various inlier tracks, and that the GPRT constraints are not sufficient to maintain a large basin of convergence for these datasets with missing entries.

In the second experiment, custom tracks are built using the approach in Section 6.4 for each of the image sequences listed in Table 6.5. The peformance of pOSE and GRPT are compared on these point tracks. This experiment is carried out to take into consideration that creating consistent inlier tracks is also an essential non-negligible subproblem in SfM. Similar to the first experiment, each algorithm's success rate is reported on each full sequence along with average runtime in Table 6.5. In addition, since Strecha et al.'s datasets (the first six in Table 6.5) provide ground truth camera poses, the camera position errors of the successful solutions are reported in Table 6.3. The benchmark results show that pOSE with custom tracks yields accurate reconstructions and produces state-of-the-art results on the castle datasets, which have repetitive structures. These results, together with the results in Table 6.5, show that the pOSE-based stratified BA pipeline has wide basins of convergence for optima with accurate reconstructions on small and medium-scale real SfM datasets, if given mostly clean point tracks. Fig. 6.1, Fig. 6.8, Fig. 6.10, Fig. 6.11 and Fig. 6.12 show the successful solutions.

Empirically, the camera position error seems to depends substantially on the generated point tracks (which is stochastic due to RANSAC), the choice of robust kernels used in the



(a) Univ. of Western Ontario (57 images)

(b) Alcatraz courtyard (133 images)

(c) Water tower (173 images)

(d) Pumpkin (209 images)

Fig. 6.12 Successful reconstructions of medium-sized Olsson and Enqvist's sequences from arbitrary initialization of cameras and 3D points. Tracks are generated from Section 6.4.

latter stages and the minimum length of tracks. All these factors change the position of the local optimum within the basin of convergence (sometimes to the better and sometimes to the worse compared to solutions from other SfM pipelines).

In Section 6.5, η was manually set to some value that seems to work well with the tested sequences. Further investigation is necessary to see if there is a single value of η that generalizes well for all sequences, or if it needs to be changed for optimal performance on different types of motion.

6.7 Conclusions

This chapter proposed the pseudo object-space error (pOSE) for projective 3D reconstruction, and it was shown that—by using the variable projection (VarPro) method—pOSE has wide convergence basins for useful optima and can be efficiently implemented. This chapter also presented a stratified framework, which starts from a randomly initialized camera matrices, to obtain ultimately a metric 3D model. It also proposed a combination of algorithms to obtain sufficiently clean correspondences from initial feature matches.

This work has demonstrated competitive results for smaller and medium-scale datasets given sufficiently clean tracks. It is an open question whether the wide convergence basin of our bundle adjustment formulation is confirmed for large datasets, or if alternative strategies such as applying our framework on medium-sized subsets are necessary.

Chapter 7

Conclusions and future work

This dissertation explored ways to widen the basin of convergence for the bundle adjustment type of problems, which is a hard problem due to the bilinear interaction of variables, structured missing data pattern and existence of outliers. The work has been primarily motivated by the recent breaking success of several matrix factorization algorithms converging nearly 100% to the best optimum on some datasets previously considered to be difficult.

To achieve the goal of widening the convergence basin, the thesis started by addressing the problem of fixed rank matrix fatorization with missing data, which is closely related to bundle adjustment as illustrated in Section 2.2.4. Chapter 3 demonstrated that successful matrix factorization algorithms can be unified and viewed as employing a method called variable projection (VarPro). This sparked further investigations with aims to answer the fundamental question of why VarPro succeeds whilst other optimization methods fail on this type of problems. As a result, Chapter 4 revealed that VarPro is actually algorithmically very similar to but different from a standard widely-used joint optimization algorithm, and the major difference is in VarPro's unequal trust region assumption between the camera and point parameters, which allows VarPro to escape the failure mode consistently encountered by the joint optimization-based algorithms. This unification also allowed VarPro to be more efficiently implemented with few modifications.

Chapter 5 investigated whether the advantage of widened convergence basin enjoyed by VarPro is applicable to non-bilinear bundle adjustment problems. Unfortunately, preliminary results showed VarPro's convergence basins of useful optima are almost as narrow as those of joint optimization on nonlinear nonseparable problems, and the basins are only wide for bilinear factorization problems. To bypass this issue, the chapter consequently proposed a stratified strategy in which a bilinear bundle adjustment problem is solved first using VarPro, followed by standard non-bilinear bundle adjustment. In Chapter 6, this scheme was extended to incorporate metric constraints. In order to yield accurate 3D reconstructions, the affine factorization stage was replaced by a hybrid of projective and affine factorizations (referred to as pOSE optimization). It was subsequently shown that the proposed bootstrapping strategy has wide the basins of convergence of useful solutions but only given sufficiently clean image point tracks.

7.1 Limitations

Although this thesis attempted to tackle various aspects of bundle adjustment, matrix factorization and structure-from-motion, there still lies many aspects to which the scope of this work is limited.

- **Limitation of bootstrapping:** It is of valid criticism that this work does not present an algorithm to directly solve metric bundle adjustment from arbitrary intialization, and one may argue that such affine or pOSE initializations are just another set of initializations. This is an unavoidable offset due to VarPro's small basin of convergence when applied on nonseparable problems. Nevertheless, pOSE can yield reconstructions that are visually relatively close to the actual bundle solution.

- **High quality point tracks required:** One of the reasons this work aimed to explore towards initialization-free bundle adjustment (BA) was to streamline the state-of-the-art structure-from-motion (SfM) pipelines. Unfortunately, investigations in Section 6.2 showed good point tracks are essential to yield feasible reconstructions even with the proposed strategy, as variable projection (VarPro)'s wide basin of convergence only seems to hold for non-robust bilinear problems.

- **Success of the variable projection method still in question:** Chatper 4 showed a major failure case for the joint optimization family of algorithms. However, it is still unclear why joint optimization-based algorithms fall into this failure mode in the first place, and most importantly, why variable projection really succeeds. It is suspected (after many empirical runs) that bilinear problems have fewer local minima but this will need to be investigated.

- **Lack of formal proofs in support of the variable projection method:** As briefly mentioned in Chapter 1, there is lack of theory in solving problems with structured missing pattern. The findings from this work are largely based on algorithmic comparisons and empirical evidence. Although some intuitions are demonstrated, no supplementing mathematical proofs have been provided.

- **Larger-scale problems** The experiments in this work focused on small to medium sized datasets. Applying this to even sparser and larger problems could turn out to be problematic, in which case this framework may have to be applied in blocks.
- **Intrinsics required** The work presented here assumed distortion-free images. This could be a reasonable assumption in most cases but may not yield extremely accurate results unless images are undistorted prior to computation.

7.2 Future work

There are several potential avenues for future work.

- **Better tracking strategy:** On the practical side, Section 6.2 has illustrated that structure-from-motion can have wide basin of convergence if the tracks are made sufficiently clean. In other words, it implies that one of the major reasons that make direct bundle adjustment difficult is the presence of severe inconsistent tracks arising from incorrect matches. As mentioned in Section 2.2, generating clean consistent tracks from noisy feature matches is still an active research topic in computer vision. It would be interesting to explore with the aim of developing a reliable gold stand track generation algorithm that can lower the complexity of structure-from-motion.
- **Trivariate optimization for incorporating robustness:** Incorporating robustness using Zach [2014]'s lifting strategy adds a set of new variables to the problem, which can potentially be trilinear depending on the type of the robust kernel and the camera projection model. Finding a good working trivariate optimization strategy is likely to be beneficial for solving various tasks in computer vision.
- **Other nonlinear extension of variable projection:** As mentioned in Section 5.7, Strelow's nonlinear VarPro assumes that the reduced residual is locally linear in the eliminated variables $\mathbf{v}$. This is equivalent to approximating the update in $\mathbf{v}$, $\Delta\mathbf{v}$, as a single Gauss-Newton step. This is one of several assumptions one can make, and it would be interesting to check if formulating $\Delta\mathbf{v}$ as a sum of several or all previous updates taken by $\mathbf{v}$ would improve the convergence behaviour of the nonlinear VarPro method.
- **Better pseudo bundle adjustment problem:** Another interesting question is a search for a better bilinear problem. The general projective reconstruction theorem [Nasihatkon et al., 2015] shows that only a sparse set of regularizers are required when the dataset is noise free. It is likely that there are bilinear formulations that more closely resembles the bundle adjustment solution.

- **Better sampling distribution** At the moment, all the camera and point parameters are sampled from a Gaussian distribution with unit isotropic variance. By understanding how these parameters are affected due to calibration parameters and types of movements, it may be interesting to try a different sampling distribution.

Extending current knowledge on bundle adjustment to more complex systems with reflections such as eye tracking can also be interesting. A highly ambitious goal would be to try and apply the knowledge of matrix factorization discovered here to larger problems such as neural networks.

On the theory side it is still an open question why VarPro performs well for certain problems. This thesis has taken one step towards answering that question, by implicitly showing why joint optimization fails. Also, investigating the theoretical guarantee of VarPro convergence would help to understand what really makes a good optimizer.

References

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *11th European Conference on Computer Vision (ECCV): Part II*, pages 29–42, 2010.

S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org, 2014.

I. Akhter, Y. Sheikh, S. Khan, and T. Kanade. Trajectory space: A dual representation for nonrigid structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(7):1442–1456, July 2011.

M. Armstrong, A. Zisserman, and R. I. Hartley. Self-calibration from image triplets. In *4th European Conference on Computer Vision (ECCV)*, pages 1–16, 1996.

W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9(1):17–29, 1951.

K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 9(5):698–700, May 1987.

S. Avidan and A. Shashua. Threading fundamental matrices. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(1):73–77, 2001.

H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *8th European Conference on Computer Vision (ECCV)*, pages 404–417. Springer Berlin Heidelberg, 2006.

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.

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

P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256, 1992.

D. Bindel, J. Demmel, W. Kahan, and O. Marques. On computing givens rotations reliably and efficiently. *ACM Transactions on Mathematical Software*, 28(2):206–238, jun 2002.

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.

N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research (JMLR)*, 15(1):1455–1459, 2014.

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.

A. M. Buchanan. Investigation into matrix factorization when elements are unknown. Technical report, University of Oxford, 2004.

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.

R. H. Byrd and D. A. Pyne. Some results on the convergence of the iteratively reweighted least squares algorithm for robust regression. Technical report, 1979.

R. Cabral, F. De la Torre, J. P. Costeira, and A. Bernardino. Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 2488–2495, 2013.

E. J. Candès and Y. Plan. Matrix completion with noise. *IEEE*, 98(6):925–936, June 2010.

E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

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 (PAMI)*, 35(1):232–244, 2013.

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.

M. Chandraker, S. Agarwal, D. Kriegman, and S. Belongie. Globally optimal algorithms for stratified autocalibration. *International Journal of Computer Vision (IJCV)*, 90(2):236–254, 2010.

A. Chatterjee and V. M. Govindu. Efficient and robust large-scale rotation averaging. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 521–528, Dec 2013.

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.

P. Chen. Hessian matrix vs. Gauss-Newton Hessian matrix. *SIAM Journal on Numerical Analysis (SINUM)*, 49(4):1417–1435, 2011.

W. Cheney and D. R. Kincaid. *Linear Algebra: Theory and Applications*. Jones and Bartlett Publishers, Inc., 1st edition, 2008.

O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. pages 236–243, 2003.

Y. Dai, H. Deng, and M. He. Dense non-rigid structure-from-motion made easy — a spatial-temporal smoothness based solution. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 4532–4536, 2017.

A. Dalalyan and R. Keriven. l_1-penalized robust estimation for a class of inverse problems arising in multiview geometry. In *Advances in Neural Information Processing Systems*, pages 441–449, 2009.

T. A. Davis. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. Society for Industrial and Applied Mathematics, 2006.

A. Del Bue, J. Xavier, L. Agapito, and M. Paladini. Bilinear modeling via augmented Lagrange multipliers (BALM). *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(8):1496–1508, Aug 2012.

F. Dellaert. Visual slam tutorial: Bundle adjustment. Technical report, 2014.

J. Dezert and C. Musso. An efficient method for generating points uniformly distributed in hyperellipsoids. Technical report, 2001.

E. Eade. Lie groups for 2d and 3d transformations - ethan eade. Technical report, 2017.

A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

O. Enqvist, F. Kahl, and C. Olsson. Non-sequential structure from motion. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 264–271, 2011.

A. Eriksson and A. van den Hengel. Efficient computation of robust low-rank matrix approximations in the presence of missing data using the L1 norm. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 771–778, June 2010.

M. E. Fathy, A. S. Hussein, and M. F. Tolba. Fundamental matrix estimation: A study of error criteria. *Pattern Recognition Letters*, 32(2):383–391, 2011.

O. D. Faugeras and S. Maybank. Motion from point matches: Multiplicity of solutions. *International Journal of Computer Vision*, 4(3):225–246, Jun 1990.

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24 (6):381–395, jun 1981.

A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *5th European conference on Computer Vision (ECCV)*, pages 311–326, 1998.

R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154, 1964.

R. Gherardi and A. Fusiello. Practical autocalibration. In *11th European Conference on Computer Vision (ECCV)*, pages 790–801, 2010.

K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, Jul 2001.

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.

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

G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Edition)*. Johns Hopkins University Press, 1996.

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.

C. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, 1988.

R. I. Hartley. *Euclidean reconstruction from uncalibrated views*, pages 235–256. 1994.

R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(6):580–593, Jun 1997.

R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

R. I. Hartley, J. Trumpf, Y. Dai, and H. Li. Rotation averaging. *International Journal of Computer Vision (IJCV)*, 103(3):267–305, Jul 2013.

M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, Nov 1969.

M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–436, 1952.

A. Heyden and K. Astrom. Euclidean reconstruction from constant intrinsic parameters. In *13th International Conference on Pattern Recognition (ICPR)*, volume 1, pages 339–343, 1996.

A. Heyden and K. Astrom. Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In *1997 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 438–443, 1997.

A. Heyden, R. Berthilsson, and G. Sparr. An iterative factorization method for projective structure and motion from image sequences. *Image and Vision Computing*, 17(13):981–991, 1999.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2012.

A. S. Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM*, 5 (4):339–342, oct 1958.

P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 03 1964.

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.

P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 665–674. ACM, 2013.

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.

C. Julia, A. D. Sappa, F. Lumbreras, J. Serrat, and A. Lopez. Recovery of surface normals and reflectance from different lighting conditions. In *5th International Conference on Image Analysis and Recognition (ICIAR)*, volume 5112 of *Lecture Notes in Computer Science*, pages 315–325. 2008.

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

R. Kennedy, L. Balzano, S. J. Wright, and C. J. Taylor. Online algorithms for factorization-based structure from motion. *Computer Vision and Image Understanding*, 150(C):139–152, sep 2016.

G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *2009 IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2009.

V. Larsson, C. Olsson, E. Bylow, and F. Kahl. Rank minimization with structured data patterns. In *13th European Conference on Computer Vision (ECCV)*, pages 250–265, 2014.

V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate $O(n)$ solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155, 2008.

K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathmatics*, 2(2):164–168, 1944.

H. Li and R. I. Hartley. Five-point motion estimation made easy. In *18th International Conference on Pattern Recognition (ICPR)*, volume 1, pages 630–633, 2006.

H. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

M. I. A. Lourakis and A. A. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1):1–30, 2009.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60:91–110, 2004.

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *7th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2 of *IJCAI'81*, pages 674–679. Morgan Kaufmann Publishers Inc., 1981.

Q.-T. Luong and O. D. Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17(1):43–75, 1996.

S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

L. Magerand and A. Del Bue. Practical projective structure from motion (p2sfm). In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 39–47, 2017.

J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. 3rd edition, 2007.

S. Mahamud and M. Hebert. Iterative projective reconstruction from multiple views. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 430–437, 2000.

J. H. Manton, R. Mahony, and Y. Hua. The geometry of weighted low-rank approximations. *IEEE Transactions on Signal Processing*, 51(2):500–514, 2003.

I. Markovsky. Structured low-rank approximation and its applications. *Automatica*, 44(4): 891–909, 2008.

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.

R. Martí. Multi-start methods. In F. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 355–368. 2003. URL http://dx.nodoi.org/10.1007/0-306-48056-5_12.

D. Martinec and T. Pajdla. Structure from many perspective images with occlusions. *Computer Vision—ECCV 2002*, pages 542–544, 2002.

E. Maset, F. Arrigoni, and A. Fusiello. Practical and efficient multi-view matching. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4578–4586, 2017.

R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research (JMLR)*, 11(Aug):2287–2322, 2010.

T. P. Minka. Old and new matrix algebra useful for statistics. Technical report, Microsoft Research, 2000.

B. Mishra, G. Meyer, F. Bach, and R. Sepulchre. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization (SIOPT)*, 23(4):2124–2149, 2013.

J. Moré and D. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

P. Moulon, P. Monasse, and R. Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *2013 IEEE International Conference on Computer Vision (ICCV)*, pages 3248–3255, 2013.

R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

B. Nasihatkon, R. I. Hartley, and J. Trumpf. A generalized projective reconstruction theorem and depth constraints for projective factorization. *International Journal of Computer Vision (IJCV)*, 115(2):87–114, 2015.

D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–770, 2004.

D. Nistér. Preemptive ransac for live structure and motion estimation. *Machine Vision and Applications*, 16(5):321–329, 2005.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

Y.-i. Ohta, K. Maenobu, and T. Sakai. Obtaining surface orientation from texels under perspective projection. In *7th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 746–751. Morgan Kaufmann Publishers Inc., 1981.

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.

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.

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

J. Oliensis and R. I. Hartley. Iterative extensions of the sturm/triggs algorithm: Convergence and nonconvergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(12):2217–2233, 2007.

C. Olsson and O. Enqvist. Stable structure from motion for unordered image collections. In *17th Scandinavian Conference on Image Analysis (SCIA)*, pages 524–535, 2011.

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.

C. J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(3): 206–218, 1997.

E. Polak and G. Ribiere. Note sur la convergence de méthodes de directions conjuguées. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 3:35–43, 1969.

M. Pollefeys and L. van Gool. Stratified self-calibration with the modulus constraint. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(8):707–724, 1999.

M. Pollefeys, R. Koch, and L. V. Gool. Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision (IJCV)*, 32(1):7–25, 1999.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, 2007.

R. Raguram, J.-M. Frahm, and M. Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *10th European Conference on Computer Vision (ECCV)*, pages 500–513. Springer Berlin Heidelberg, 2008.

A. Rav-Acha, P. Kohli, C. Rother, and A. W. Fitzgibbon. Unwrap mosaics: A new representation for video editing. *ACM Transactions on Graphics (TOG)*, 27(3):17:1–17:11, 2008.

B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

O. Rodrigues. De l'attraction des sphéroïdes. Technical report, 1816.

E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *2005 IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, 2005.

E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *8th European Conference on Computer Vision (ECCV)*, pages 430–443. Springer Berlin Heidelberg, 2006.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

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

P. D. Sampson. Fitting conic sections to "very scattered" data: An iterative refinement of the bookstein algorithm. *Computer Graphics and Image Processing*, 18(1):97–108, 1982.

J. L. Schönberger and J. M. Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

Y. Seo, H. Lee, and S. W. Lee. Outlier removal by convex optimization for l-infinity approaches. In *Pacific-Rim Symposium on Image and Video Technology*, pages 203–214, 2009.

Y. Shen, Z. Wen, and Y. Zhang. Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization. *Optimization Methods Software*, 29(2):239–263, 2014.

D. E. Smith. *History of Modern Mathematics*. Chapman and Hall, 1906.

N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006.

H. Stewénius, D. Nistér, F. Kahl, and F. Schaffalitzky. A minimal solution for relative pose with unknown focal length. In *2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

G. Strang. *Krylov Subspaces and Conjugate Gradients*, 2006.
Available: http://math.mit.edu/classes/18.086/2006/am64.pdf.

C. Strecha, W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

D. Strelow. General and nested Wiberg minimization. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1584–1591, 2012a.

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

P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *4th European Conference on Computer Vision (ECCV)*, pages 709–720. 1996.

J. Svoboda, T. Cashman, and A. Fitzgibbon. QRkit: Sparse, composable QR decompositions for efficient and stable solutions to problems in computer vision. In *2018 IEEE International Conference on Applications of Computer Vision*, 2018.

C. Sweeney. Theia multiview geometry library: Tutorial & reference. http://theia-sfm.org, 2017.

C. Sweeney, T. Sattler, T. Höllerer, M. Turk, and M. Pollefeys. Optimizing the viewing graph for structure-from-motion. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 801–809, 2015.

J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, E. Soto, D. Sweeney, J. Valentin, B. Luff, A. Topalian, E. Wood, S. Khamis, P. Kohli, T. Sharp, S. Izadi, R. Banks, A. Fitzgibbon, and J. Shotton. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. In *ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*, 2016.

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.

P. H. S. Torr, A. Zisserman, and S. J. Maybank. Robust detection of degenerate configurations for the fundamental matrix. In *1995 IEEE International Conference on Computer Vision*, pages 1037–1042, 1995.

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.

K. Usevich and I. Markovsky. Optimization on a Grassmann manifold with application to system identification. *Automatica*, 50(6):1656–1662, 2014.

L. Vandenberghe. *EE103: Applied Numerical Computing lecture note 7*, 2011. http://www.seas.ucla.edu/~vandenbe/103/lectures/qr.pdf.

R. Vidal and R. I. Hartley. Motion segmentation with missing data using PowerFactorization and GPCA. In *2004 IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 310–316, 2004.

G. Wang and Q. M. J. Wu. *Simplified Camera Projection Models*, pages 29–41. Springer London, 2011.

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

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

K. Wilson and N. Snavely. Network principles for sfm: Disambiguating repeated structures with local context. In *2013 IEEE International Conference on Computer Vision (ICCV)*, 2013.

K. Wilson and N. Snavely. Robust global translations with 1DSfM. In *13th European Conference on Computer Vision (ECCV)*, pages 61–75. Springer International Publishing, 2014.

C. Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision (3DV)*, pages 127–134, 2013.

C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3057–3064, 2011.

A. Y. Yang, Z. Zhou, A. G. Balasubramanian, S. S. Sastry, and Y. Ma. Fast l_1-minimization algorithms for robust face recognition. *IEEE Transactions on Image Processing*, 22(8): 3234–3246, 2013.

K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned Invariant Feature Transform. In *14th European Conference on Computer Vision (ECCV)*, pages 467–483, 2016.

G. Yu and J.-M. Morel. ASIFT: An algorithm for fully affine invariant comparison. *Image Processing On Line*, 1:11–38, 2011.

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

C. Zach and G. Bourmaud. Iterated lifting for robust cost optimization. In *2017 British Machine Vision Conference (BMVC)*, 2017.

C. Zach and M. Pollefeys. Practical methods for convex multi-view reconstruction. In *European Conference on Computer Vision*, pages 354–367, 2010.

C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1426–1433, 2010.

Z. Zhang. A flexible new technique for camera calibration. 22:1330–1334, 2000.

Y. Zheng, S. Sugimoto, S. Yan, and M. Okutomi. Generalizing Wiberg algorithm for rigid and nonrigid factorizations with missing components and metric constraints. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2010–2017, 2012.

Y. Zheng, Y. Kuang, S. Sugimoto, K. Åström, and M. Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *2013 IEEE International Conference on Computer Vision*, pages 2344–2351, 2013.

M. Zollhöfer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. W. Fitzgibbon, C. Loop, C. Theobalt, and M. Stamminger. Real-time non-rigid reconstruction using an RGB-D camera. *ACM Transactions on Graphics (TOG)*, 33(4):156:1–156:12, 2014.

Appendix A

Further background materials

This chapter presents more basic background materials that form the basis of this dissertation.

A.1 Linear algebra

In this section, some key results related to vectors and matrices are summarized. Note that semicolons (;) between square brackets represent row stacks, e.g. $[a\ ;\ b] = [a\ ,\ b]^\top$. Also, $[\mathbf{x}]_k$ signifies the k-th element of $\mathbf{x}$.

A.1.1 Definitions

This section defines some quantities that are used throughout the thesis.

Vec operator

Suppose there is a matrix $\mathtt{A} := [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n]$ where $\mathbf{a}_j = [a_{1,j}; a_{2,j}; \cdots; a_{m,j}]$. Then the corresponding vec-operator is defined as $\mathrm{vec}(\mathtt{A}) := [\mathbf{a}_1;\ \mathbf{a}_2;\ \cdots; \mathbf{a}_n]$.

Kronecker product

If there are two matrices $\mathtt{A} \in \mathbb{R}^{m\times n}$ and $\mathtt{B} \in \mathbb{R}^{p\times q}$, the Kronecker product $(\mathtt{A} \otimes \mathtt{B} \in \mathbb{R}^{mp\times nq}$ is defined as

$$\mathtt{A} \otimes \mathtt{B} := \begin{bmatrix} a_{1,1}\mathtt{B} & \cdots & a_{1,n}\mathtt{B} \\ \vdots & \ddots & \vdots \\ a_{m,1}\mathtt{B} & \cdots & a_{m,n}\mathtt{B} \end{bmatrix} \tag{A.1}$$

Hadamard product

Given two matrices $\mathtt{A} \in \mathbb{R}^{m\times n}$ and $\mathtt{B} \in \mathbb{R}^{m\times n}$, the Hadamard product $(\mathtt{A} \odot \mathtt{B}) \in \mathbb{R}^{m\times n}$ is defined as

$$\mathtt{A} \odot \mathtt{B} := \begin{bmatrix} a_{1,1}b_{1,1} & \cdots & a_{1,n}b_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1}b_{m,1} & \cdots & a_{m,n}b_{m,n} \end{bmatrix}. \tag{A.2}$$

Diag operator

Given a vector $\mathbf{x} \in \mathbb{R}^n$ diag : $\mathbb{R}^n \to \mathbb{S}^n$ yields a diagonal matrix that is defined as

$$\operatorname{diag}(\mathbf{x}) := \begin{bmatrix} x_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & x_n \end{bmatrix}. \tag{A.3}$$

Skew symmetric matrix operator

Given a vector $\mathbf{x} \in \mathbb{R}^n$, the skew symmetric matrix operator $[\mathbf{x}]_\times$ yields

$$\operatorname{diag}(\mathbf{x}) := \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \tag{A.4}$$

Matrix exponential

Given a square matrix $\mathtt{A} \in \mathbb{R}^{n\times n}$, the matrix exponential of $\mathtt{A}$ is defined as

$$\exp(\mathtt{A}) := \mathtt{I} + \mathtt{A} + \frac{1}{2}\mathtt{A}^2 + \cdots = \sum_{k=0}^{\infty} \frac{1}{k!}\mathtt{A}^k \tag{A.5}$$

with $\mathtt{A}^0 := \mathtt{I}$. Note that $\exp(\mathtt{AB}) \neq \exp(\mathtt{A})\exp(\mathtt{B})$.

A.1.2 Vec and Kronecker product identities

Below is a list of the key matrix identities used in this work which are obtained or extended from [Magnus and Neudecker, 2007; Minka, 2000]. Given that each small letter in bold represents a column vector, the following rules are illustrated:

$$\operatorname{vec}(\mathtt{A} \odot \mathtt{B}) = \operatorname{diag}(\operatorname{vec}\mathtt{A})\operatorname{vec}(\mathtt{B}) \tag{A.6}$$

$$\operatorname{vec}(\mathtt{AXB}) = (\mathtt{B}^\top \otimes \mathtt{A})\operatorname{vec}(\mathtt{X}) \tag{A.7}$$

$$\operatorname{vec}(\mathtt{AB}) = (\mathtt{B}^\top \otimes \mathtt{I})\operatorname{vec}(\mathtt{A}) = (\mathtt{I}^\top \otimes \mathtt{A})\operatorname{vec}(\mathtt{B}) \tag{A.8}$$

$$(\mathtt{A} \otimes \mathtt{B})(\mathtt{C} \otimes \mathtt{D}) = \mathtt{AC} \otimes \mathtt{BD} \tag{A.9}$$

$$(\mathtt{A} \otimes \mathbf{b})\mathtt{C} = \mathtt{AC} \otimes \mathbf{b}. \tag{A.10}$$

Now define the permutation matrix $\mathtt{K}_{ab}$ such that any arbitrary matrix $\mathtt{C} \in \mathbb{R}^{a\times b}$ has

$$\operatorname{vec}(\mathtt{C}^\top) = \mathtt{K}_{ab}\operatorname{vec}(\mathtt{C}). \tag{A.11}$$

If $\mathtt{A} \in \mathbb{R}^{m\times n}$, $\mathtt{B} \in \mathbb{R}^{p\times q}$ and $\mathbf{b} \in \mathbb{R}^p$, then the following relationships hold:

$$(\mathtt{A} \otimes \mathtt{B}) = \mathtt{K}_{mp}^\top(\mathtt{B} \otimes \mathtt{A})\mathtt{K}_{nq} \tag{A.12}$$

$$(\mathtt{A} \otimes \mathbf{b}) = \mathtt{K}_{mp}^\top(\mathbf{b} \otimes \mathtt{A}). \tag{A.13}$$

A.1.3 Calculus rules

Given that $\mathtt{A}$ is a constant matrix, $\mathtt{Y}(\mathtt{X})$ is a variable matrix and $\mathbf{c}$ is a constant vector, the following rules are observed:

$$\partial[\mathbf{c}^\top\mathbf{x}] = \mathbf{c}^\top\partial[\mathbf{x}] \tag{A.14}$$

$$\partial[\mathbf{c}^\top\mathtt{A}\mathbf{x}] = \mathbf{c}^\top\mathtt{A}\partial[\mathbf{x}] \tag{A.15}$$

$$\partial[\mathtt{A}\mathbf{x}] = \mathtt{A}\partial[\mathbf{x}] \tag{A.16}$$

$$\partial[\mathtt{Y}\mathbf{x}] = \partial[\mathtt{Y}]\mathbf{x} + \mathtt{Y}\partial[\mathbf{x}] \tag{A.17}$$

$$\partial[\mathtt{X}^{-1}] = -\mathtt{X}^{-1}\partial[\mathtt{X}]\mathtt{X}^{-1}. \tag{A.18}$$

Pseudo-inverse $\mathtt{X}^\dagger$ and μ-pseudo inverse $\mathtt{X}^{-\mu}$ are defined as

$$\mathtt{X}^\dagger = (\mathtt{X}^\top\mathtt{X})^{-1}\mathtt{X}^\top \tag{A.19}$$

$$\mathtt{X}^{-\mu} = (\mathtt{X}^\top\mathtt{X} + \mu\mathtt{I})^{-1}\mathtt{X}^\top. \tag{A.20}$$

The derivative of μ-pseudo inverse is then obtained as follows:

$$\begin{aligned}
\partial[\mathtt{X}^{-\mu}] &= \partial[(\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}]\mathtt{X}^\top + (\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}\partial[\mathtt{X}]^\top \qquad \text{(A.21)}\\
&= -(\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}\partial[\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I}](\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}\mathtt{X}^\top + (\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}\partial[\mathtt{X}]^\top \qquad \text{(A.22)}\\
&= -(\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}(\partial[\mathtt{X}^\top]\mathtt{X}+\mathtt{X}^\top\partial[\mathtt{X}])(\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}\mathtt{X}^\top\\
&\qquad + (\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}\partial[\mathtt{X}]^\top \qquad \text{(A.23)}\\
&= -(\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}\partial[\mathtt{X}^\top]\mathtt{X}\mathtt{X}^{-\mu} - (\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}\mathtt{X}^\top\partial[\mathtt{X}]\mathtt{X}^{-\mu}\\
&\qquad + (\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}\partial[\mathtt{X}]^\top \qquad \text{(A.24)}\\
&= -\mathtt{X}^{-\mu}\partial[\mathtt{X}]\mathtt{X}^{-\mu} + (\mathtt{X}^\top\mathtt{X}+\mu\mathtt{I})^{-1}\partial[\mathtt{X}]^\top(\mathtt{I}-\mathtt{X}\mathtt{X}^{-\mu}). \qquad \text{(A.25)}
\end{aligned}$$

The derivative of pseudo-inverse is then simply the above for $\mu = 0$. One may observe the last term is zero for invertible $\mathtt{X}$and $\mu = 0$.

A.1.4 Projector and Householder matrices

If there exists a matrix $\mathtt{X} \in \mathbb{R}^{m\times n}$, the projector matrix which projects a matrix of the same dimension to the column space of $\mathtt{X}$ is given by $\mathtt{X}(\mathtt{X}^\top\mathtt{X})^{-1}\mathtt{X}^\top$. Similarly, the orthogonal projector is given by $\mathtt{I}-\mathtt{X}(\mathtt{X}^\top\mathtt{X})^{-1}\mathtt{X}^\top$. (It is easy to check that this has $\mathtt{X}$ as the nullspace.)

Similarly, for a vector $\mathbf{x} \in \mathbb{R}^n$, the projector and orthogonal projectors are similarly defined as $\mathbf{x}\mathbf{x}^\top/\|\mathbf{x}\|_2^2$ and $\mathtt{I}-\mathbf{x}\mathbf{x}^\top/\|\mathbf{x}\|_2^2$ respectively. Additionally, a vector reflected on the plane perpendicular to $\mathbf{x}$ can be computed by multiplying by the Householder matrix [Householder, 1958] defined as $\mathtt{I}-2\mathbf{x}\mathbf{x}^\top/\|\mathbf{x}\|_2^2$. The proof can be derived from a simple geometric intuition.

A.1.5 Matrix decompositions

Below is a set of matrix decomposition methods used in this work. This section will assume efficient algorithms are available for these methods [Davis, 2006; Svoboda et al., 2018], and will not go over the details of the actual decomposition algorithms used.

LU decomposition

Given a square input matrix $\mathtt{A} \in \mathbb{R}^{n\times n}$ for some n, its LU decomposition forms

$$\mathtt{A} = \mathtt{L}\mathtt{U} \tag{A.26}$$

where $\mathtt{L} \in \mathbb{R}^{n\times n}$ is a lower diagonal matrix and $\mathtt{U} \in \mathbb{R}^{n\times n}$ is an upper diagonal matrix. Since this can sometimes be numerically unstable, LU with partial pivoting is more frequently used, leading to

$$\mathtt{PA} = \mathtt{LU} \tag{A.27}$$

where $\mathtt{P} \in \mathbb{R}^{n \times n}$ is a row-pivoting (permutation) matrix. In both cases, the decomposition is carried out using Gaussian elimination.

Cholesky decomposition

Cholesky decomposition requires a symmetric positive-definite matrix as input. For, $\mathtt{A} \in \mathbb{S}^n$, its Cholesky decomposition yields

$$\mathtt{A} = \mathtt{L}\mathtt{L}^\top = \mathtt{R}^\top\mathtt{R} \tag{A.28}$$

where $\mathtt{L} \in \mathbb{R}^{n \times n}$ is a lower-triangular matrix and $\mathtt{R} \in \mathbb{R}^{n \times n}$ is an upper-triangular matrix.

QR decomposition

Suppose there is an input matrix $\mathtt{A} \in \mathbb{R}^{m \times n}$ for some m and n. For $m > n$,

$$\mathtt{A} = \begin{bmatrix} \mathtt{Q} & \tilde{\mathtt{Q}} \end{bmatrix} \begin{bmatrix} \mathtt{R} \\ \mathtt{0} \end{bmatrix} = \mathtt{QR} \tag{A.29}$$

where $\mathtt{Q} \in \mathbb{R}^{m \times n}$ is an orthonormal column space of $\mathtt{A}$, $\tilde{\mathtt{Q}} \in \mathbb{R}^{m \times m - n}$ is the left nullspace of $\mathtt{A}$ and $\mathtt{R} \in \mathbb{R}^{n \times n}$ is an upper triangular matrix. The most r.h.s. expression is also known as the "economy" QR decomposition. For $m = n$, the QR decomposition simply boils down to $\mathtt{A} = \mathtt{QR}$ with $\mathtt{Q} \in O(n)$.

The most widely used algorithm for QR decomposition, which is also used in MATLAB, employs the Householder reflections [Householder, 1958]. Other algorithms use the Gram-Schmidt process [Cheney and Kincaid, 2008] or Givens rotations [Bindel et al., 2002].

The QR decomposition is also more numerically stable than the LU or Cholesky decompositions (Vandenberghe [2011] illustrates an example).

Singular value decomposition (SVD)

Given an input matrix $\mathtt{A} \in \mathbb{R}^{m \times n}$ for some m and n with $m > n$, its SVD yields

$$\mathtt{A} = \begin{bmatrix} \mathtt{U} & \tilde{\mathtt{U}} \end{bmatrix} \begin{bmatrix} \Sigma \\ \mathtt{0} \end{bmatrix} \mathtt{V}^\top = \mathtt{U}\Sigma\mathtt{V}^\top \tag{A.30}$$

where $\mathtt{U} \in \mathbb{R}^{m\times n}$ is an orthonormal column space of $\mathtt{A}$, $\tilde{\mathtt{U}} \in \mathbb{R}^{m-n\times n}$ is the left nullspace of $\mathtt{A}$, $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix and $\mathtt{V} \in O(n)$. Again, the most r.h.s. expression is referred to as the "reduced" SVD. For $m = n$, the original SVD expression is the same as its reduced form.

The diagonal elements of Σ are called *singular values*, and the number of non-zero singular values signifies $\mathrm{rank}(\mathtt{A})$. In practice, some rounding-off errors may exist, and therefore singular values below some threshold (e.g. 10^{-8}) are set to 0.

A.1.6 Vector and matrix norms

This section briefly goes over some definitions of several norms.

Vector norms

Some major vector norms are defined below. Given an input vector $\mathbf{x} \in \mathbb{R}^n$,

$$\|\mathbf{x}\|_2 := \sqrt{\sum_{k=1}^{n} x_k^2} \tag{A.31}$$

$$\|\mathbf{x}\|_1 := \sum_{k=1}^{n} |x_k| \tag{A.32}$$

$$\|\mathbf{x}\|_0 := \sum_{k=1}^{n} (x_k > 0) \tag{A.33}$$

$$\|\mathbf{x}\|_\infty := \max(|x_1|, \cdots, |x_n|) \tag{A.34}$$

$$\|\mathbf{x}\|_p := \left(\sum_{k=1}^{n} |x_k|^p\right)^{1/p}. \tag{A.35}$$

Matrix norms

For an input matrix $\mathtt{X} \in \mathbb{R}^{m\times n}$, the SVD (see Section A.1.5) of $\mathtt{X}$ is $\mathtt{U}\Sigma\mathtt{V}^\top$. Now, define σ_k to be the k-th largest singular value of $\mathtt{X}$. Then, $\mathtt{X}$ has the following matrix norms:

$$\|\mathtt{X}\|_* := \mathrm{trace}\left(\sqrt{\mathtt{X}^\top\mathtt{X}}\right) = \mathrm{trace}(|\Sigma|) = \sum_{k=1}^{\min(m,n)} |\sigma_k| \tag{A.36}$$

$$\|\mathtt{X}\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} (x_{i,j})^2} = \sqrt{\mathrm{trace}(\mathtt{X}^\top\mathtt{X})} = \sqrt{\mathrm{trace}(\Sigma^2)} = \sqrt{\sum_{k=1}^{\min(m,n)} \sigma_k^2} \tag{A.37}$$

$$\|\mathtt{X}\|_1 := \sum_{i=1}^{m}\sum_{j=1}^{n} |x_{i,j}| \tag{A.38}$$

where $\|\cdot\|_*$ denotes the nuclear norm, $\|\cdot\|_F$ denotes the Frobenius norm and $\|\cdot\|_1$ denotes the L1-norm.

A.1.7 Gram-Schmidt orthogonalization

The Gram-Schmidt process is a method of finding an orthogonal column space of input matrix. It can be used for the QR factorization in Section A.1.5.

First, if the input matrix $\mathtt{A} \in \mathbb{R}^{m\times n}$ is formulated as $[\mathbf{a}_1, \cdots, \mathbf{a}_n]$, the first column space vector $\mathbf{q}_1$ is simply defined as $\mathbf{q}_1 := \mathbf{a}_1/\|\mathbf{a}_1\|_2$.

Now, the second column space vector $\mathbf{q}_2$ is defined such that it forms a basis of $\mathbf{a}_2$ as well as being orthogonal to all previously defined column space vectors, which in this case is only $\mathbf{q}_1$. This can be computed by the equation

$$\mathbf{q}_2 = \frac{\mathbf{a}_2 - \mathbf{q}_1(\mathbf{q}_1 \cdot \mathbf{a}_2)}{\|\mathbf{a}_2 - \mathbf{q}_1(\mathbf{q}_1 \cdot \mathbf{a}_2)\|_2}. \tag{A.39}$$

Applying the same logic to the rest yields a general equation for the k-th column vector:

$$\mathbf{q}_k = \frac{\mathbf{a}_k - \sum_{i=1}^{k-1} \mathbf{q}_i \mathbf{q}_i^\top \mathbf{a}_k}{\|\mathbf{a}_k - \sum_{i=1}^{k-1} \mathbf{q}_i \mathbf{q}_i^\top \mathbf{a}_k\|_2}. \tag{A.40}$$

A.1.8 Krylov subspace

Given a square matrix $\mathtt{A} \in \mathbb{R}^{n\times n}$ and a vector $\mathbf{b} \in \mathbb{R}^n$, the corresponding k-Krylov matrix for $0 < k \leq n$ is defined as

$$\mathtt{K}_k(\mathtt{A}, \mathbf{b}) := \begin{bmatrix} \mathbf{b} & \mathtt{A}\mathbf{b} & \mathtt{A}^2\mathbf{b} & \dots & \mathtt{A}^{k-1}\mathbf{b} \end{bmatrix}. \tag{A.41}$$

A corresponding k-Krylov subspace can be composed by any combination of all column vectors of the k-Krylov matrix.

Arnoldi's method

If $\mathbf{b}$ is spanned by a unit vector $\mathbf{q}_1$, then $\mathtt{A}\mathbf{b}$ has to be spanned by $\mathtt{A}\mathbf{q}_1$. Instead, $\mathbf{q}_2$ can be defined such that it spans $\mathtt{A}b$ along with $\mathbf{q}_1$. Now, this means $\mathtt{A}^2 b$ is spanned by $\mathtt{A}\mathbf{q}_1$ and $\mathtt{A}\mathbf{q}_2$. It was already shown that $\mathtt{A}\mathbf{q}_1$ is spanned by $\mathbf{q}_1$ and $\mathbf{q}_2$. Similarly, $\mathbf{q}_3$ can be defined such that it spans $\mathtt{A}\mathbf{q}_2$ along with $\mathbf{q}_1$ and $\mathbf{q}_2$. To summarize, the subspace spanned by the r-Krylov matrix (A.41)

Algorithm 7 Arnoldi's method

```
$\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|_2$, where $\mathbf{b} \in \mathbb{R}^n$
for $j = 1, \ldots, n-1$ do
    $\mathbf{t} = \mathtt{A}\mathbf{q}_j$
    for $i = 1, \ldots, j$ do
        $h_{i,j} = \mathbf{q}_i^\top \mathbf{t}$
        $\mathbf{t} \leftarrow \mathbf{t} - h_{i,j} q_i$
    end for
    $h_{j+1,j} = \|\mathbf{t}\|_2$
    $\mathbf{q}_j = \mathbf{t}/h_{j+1,j}$
end for
```

can also be spanned by

$$\begin{bmatrix} \mathbf{q}_1 & \mathtt{A}\mathbf{q}_1 & \mathtt{A}\mathbf{q}_2 & \cdots \mathtt{A}\mathbf{q}_{r-1} \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots \mathbf{q}_r . \end{bmatrix} \tag{A.42}$$

These vectors can be obtained by using Arnoldi [1951]'s method, which iteratively finds a new subspace along the Krylov subspace as shown in Algorithm 7. The algorithm is essentially the Gram-Schmit orthogonalization of $[\mathbf{q}_1, \mathtt{A}\mathbf{q}_1, \mathtt{A}\mathbf{q}_2, \cdots, \mathtt{A}\mathbf{q}_{r-1}]$.

Arnoldi's method essentially performs the factorization $\mathtt{AQ} = \mathtt{QH}$, where $\mathtt{Q} := [\mathbf{q}_1, \cdots, \mathbf{q}_n]$ and $\mathtt{H}$ has the following Hessenberg matrix structure

$$\begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,n} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,n} \\ 0 & h_{3,2} & h_{3,3} & \cdots & h_{3,n} \\ 0 & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n,n} \end{bmatrix} . \tag{A.43}$$

Lanczos method

If $\mathtt{A} \in \mathbb{S}^n$, then Arnoldi's method can be simplified to the Lanczos method. Since $\mathtt{Q}^\top \mathtt{AQ} = \mathtt{H}$ is now symmetric, the Hessenberg matrix structure of $\mathtt{H}$ becomes specifically tridiagonal (as the zeros below the tridiagonal part of $\mathtt{H}$ should now be replicated on the upper right as well). This implies that each new subspace $\mathtt{A}\mathbf{q}_j$ can be represented by the following simpler equation

$$\mathtt{A}\mathbf{q}_j = h_{j-1,j}\, \mathbf{q}_{j-1} + h_{j,j}\, \mathbf{q}_j + h_{j+1,j}\, \mathbf{q}_{j+1} \tag{A.44}$$

from which $\mathbf{q}_{j+1}$ can be obtained.

A.1.9 Nearest orthogonal matrix

Given a nearly-orthonormal matrix $\tilde{\mathtt{Q}} \in \mathbb{R}^{3\times 3}$, Arun et al.'s method finds the closest orthogonal matrix $\mathtt{Q} \in O(3)$ in terms of the Frobenius norm. It solves

$$\arg\min_{\mathtt{Q}\in O(3)} ||\mathtt{Q} - \tilde{\mathtt{Q}}||_F^2. \tag{A.45}$$

Expanding (A.45) yields

$$\begin{aligned}\arg\min_{\mathtt{Q}\in O(3)} ||\mathtt{Q} - \tilde{\mathtt{Q}}||_F^2 &= \arg\min_{\mathtt{Q}\in O(3)} \operatorname{trace}\left((\mathtt{Q} - \tilde{\mathtt{Q}})^\top (\mathtt{Q} - \tilde{\mathtt{Q}})\right) \\ &= \arg\min_{\mathtt{Q}\in O(3)} \operatorname{trace}(\tilde{\mathtt{Q}}^\top \tilde{\mathtt{Q}} - 2\tilde{\mathtt{Q}}^\top \mathtt{Q} + \mathtt{I}) \\ &= \arg\min_{\mathtt{Q}\in O(3)} \; - \operatorname{trace}\left(\tilde{\mathtt{Q}}^\top \mathtt{Q}\right) \\ &= \arg\max_{\mathtt{Q}\in O(3)} \; \operatorname{trace}\left(\tilde{\mathtt{Q}}^\top \mathtt{Q}\right) \qquad (A.46)\end{aligned}$$

which is a trace norm maximization problem. By taking the SVD (from Section A.1.5) of $\tilde{\mathtt{Q}}$ to yield $\tilde{\mathtt{Q}} = \mathtt{U}\Sigma\mathtt{V}^\top$, (A.46) is formulated as

$$\arg\max_{\mathtt{Q}\in O(3)} \; \operatorname{trace}\left(\mathtt{V}\Sigma\mathtt{U}^\top \mathtt{Q}\right) = \arg\max_{\mathtt{Q}\in O(3)} \; \operatorname{trace}\left(\Sigma\mathtt{U}^\top \mathtt{Q}\mathtt{V}\right) \tag{A.47}$$

Since $\mathtt{Q}$, $\mathtt{U}$ and $\mathtt{V}$ are all orthogonal matrices in the group $O(3)$, their product is also a a 3×3 orthogonal matrix. This is denoted as $\mathtt{X} \in O(3)$.

Since Σ is a diagonal matrix and $\mathtt{X}$ has no element greater than 1 by definition, the following holds:

$$\operatorname{trace}(\Sigma\mathtt{X}) = \sum_{i=1}^{3} \sigma_{i,i}\mathtt{X}_{i,i} \leq \sum_{i=1}^{3} \sigma_{i,i}. \tag{A.48}$$

This means $\mathtt{X}$ has to be $\mathtt{I}$ for the maximum value, and therefore

$$\mathtt{Q} = \mathtt{U}\mathtt{V}^\top. \tag{A.49}$$

Rotation matrix Each rotation matrix is a 3×3 orthogonal matrix with determinant of $+1$, and hence rotation matrices form a subgroup called $\mathrm{SO}(3)$. To enforce this constraint on the determinant, some suggest using the formula

$$\mathtt{Q} = \mathtt{U}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathtt{U}\mathtt{V}^\top) \end{bmatrix}\mathtt{V}. \tag{A.50}$$

Although (A.50) yields a valid rotation matrix, it is not guaranteed to produce the closest solution [Horn and Johnson, 2012].

A.2 Optimization

This section reviews some optimization methods and algorithms used throughout this dissertation.

A.2.1 Optimal transformation between 3D point correspondences

This section reviews simple methods for finding an optimal Euclidean and similarity transformation matrix between a set of 3D point correspondences. Note that this differs from iterative closest point (ICP) [Besl and McKay, 1992], which also has to generate correspondences between 3D points.

Optimal Euclidean transformation

This is an illustration of the algorithm originally proposed by Arun et al. [1987]. Given two n column-stacks of 3D points $\mathtt{X} \in \mathbb{R}^{3\times n}$ and $\mathtt{Y} \in \mathbb{R}^{3\times n}$ ($n > 3$), the problem is to find

$$\underset{\mathtt{R},\mathbf{t}}{\arg\min} \, ||(\mathtt{R}\mathtt{X} + \mathbf{t}\mathbf{1}^\top) - \mathtt{Y}||_F^2 \tag{A.51}$$

where $\mathtt{R} \in SO(3)$ is the rotation matrix and $\mathbf{t} \in \mathbb{R}^3$ is the translation vector. Although it is possible to yield an optimal $\mathbf{t}$ by using the first optimality condition, the intuition of least squares gives the hint that $\mathbf{t}$ will move in the direction to minimize the mean distance between the point clusters of $\mathtt{Y}$ and rotated $\mathtt{X}$, leading to two overlapping centroids. By denoting the centroids of $\mathtt{X}$ and $\mathtt{Y}$ as $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ respectively, one observes

$$\mathbf{t} = \bar{\mathbf{y}} - \mathtt{R}\bar{\mathbf{x}}. \tag{A.52}$$

In other words, given any rotation $\mathtt{R}$, the translation component will adjust to make the two centroids coincide with each other. Hence, (A.51) boils down to

$$\arg\min_{\mathtt{R}\in SO(3)} \|\mathtt{R}\hat{\mathtt{X}} - \hat{\mathtt{Y}}\|_F^2 = \arg\max_{\mathtt{R}\in SO(3)} \text{tarce}(\mathtt{R}\hat{\mathtt{X}}\hat{\mathtt{Y}}^\top) \tag{A.53}$$

where $\hat{\mathtt{X}} := \mathtt{X} - \bar{\mathbf{x}}\mathbf{1}^\top$ and $\hat{\mathtt{Y}} := \mathtt{Y} - \bar{\mathbf{y}}\mathbf{1}^\top$. Since $\hat{\mathtt{X}}\hat{\mathtt{Y}}^\top$ is just an ordinary 3×3 real matrix, the findings from Section A.1.9 can be reapplied to yield the closest orthogonal matrix via SVD. Additionally, this can be further enforced to be a valid rotation matrix.

Once $\mathtt{R}$ is computed, $\mathbf{t}$ can be computed using (A.52).

Optimal similarity transformation

Now the task is to solve

$$\arg\min_{\mathtt{R},\mathbf{t},\sigma} \|(\sigma\mathtt{R}\mathtt{X} + \mathbf{t}\mathbf{1}^\top) - \mathtt{Y}\|_F^2 \tag{A.54}$$

where σ is the relative scale between the two sets of correspondences. Again, the translational component is adapted to match the centroids of $\mathtt{Y}$ and $(\sigma\mathtt{R}\mathtt{X} + \mathbf{t}\mathbf{1}^\top)$. Hence, given σ and $\mathtt{R}$, the optimal translation is given by

$$\mathbf{t} = \bar{\mathbf{y}} - \sigma\mathtt{R}\bar{\mathbf{x}} \tag{A.55}$$

where $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ are the centroids of $\mathtt{X}$ and $\mathtt{Y}$ respectively. The rest of the variables can be obtained by solving

$$\arg\min_{\mathtt{R},\sigma} \|\sigma\mathtt{R}\hat{\mathtt{X}} - \hat{\mathtt{Y}}\|_F^2 = \arg\max_{\mathtt{R},\sigma} 2\sigma\text{trace}(\mathtt{R}\hat{\mathtt{X}}\hat{\mathtt{Y}}^\top) - \sigma^2\text{trace}(\hat{\mathtt{X}}^\top\hat{\mathtt{X}}) \tag{A.56}$$

where $\hat{\mathtt{X}} := \mathtt{X} - \mathbf{x}\mathbf{1}^\top$ and $\hat{\mathtt{Y}} := \mathtt{Y} - \mathbf{y}\mathbf{1}^\top$. Since $\hat{\mathtt{X}}\hat{\mathtt{Y}}^\top$ is again just a 3×3 real matrix, the solution from Section A.1.9 can be used to yield the closest orthogonal matrix via SVD, which can again be enforced to be a valid rotation matrix.

Given $\mathtt{R}$, the scale factor can be obtained by the equation (derived from the first optimality condition of (A.56)

$$\sigma = \frac{\text{trace}(\mathtt{R}\hat{\mathtt{X}}\hat{\mathtt{Y}}^\top)}{\hat{\mathtt{X}}^\top\hat{\mathtt{X}}}. \tag{A.57}$$

Once $\mathtt{R}$ and σ are computed, $\mathbf{t}$ can be computed using (A.55).

A.2.2 Linear least squares

A linear least squares problem is defined as

$$\arg\min_{\mathbf{x}} \|\mathtt{A}\mathbf{x} - \mathbf{b}\|_2^2 \tag{A.58}$$

with $\mathtt{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$. This section will concentrate on cases when (A.58) is either exactly or over-determined. Different optimization algorithms are available depending on whether (A.58) is homogeneous ($\mathbf{b} = \mathbf{0}$) or inhomogeneous.

Matrix decomposition-based methods

If (A.58) is inhomogeneous, the optimal solution $\mathbf{x}^* \in \mathbb{R}^n$ is defined by the equation

$$\mathbf{x}^* := \mathtt{A}^\dagger \mathbf{b}, \tag{A.59}$$

where $\mathtt{A}^\dagger$ is the pseudo inverse of $\mathtt{A}$ defined in Section A.1.3. (A.59) can be solved directly by employing a matrix decomposition method from Section A.1.5.

Using the LU decomposition with partial pivoting Suppose (A.58) is exactly determined. (i.e. $\mathtt{A}\mathbf{x} = \mathbf{b}$.) Performing LU decomposition with partial pivoting from Section A.1.5 on $\mathtt{A}$ yields $\mathtt{PA}\mathbf{x} = \mathtt{L}(\mathtt{U}\mathbf{x}) = \mathtt{P}\mathbf{b}$. This equation is then solved in two stages, first by using Gaussian elimination to effectively remove $\mathtt{L}$ and obtain $\mathtt{U}\mathbf{x}$, then using the same elimination technique to remove $\mathtt{U}$ and yield $\mathbf{x}$. The computational complexity is about $2n^3/3$ [Golub and Van Loan, 1996].

If (A.58) is overdetermined, one can perform the same procedure on the normal equations $(\mathtt{A}^\top\mathtt{A})\mathbf{x} = \mathtt{A}^\top\mathbf{b}$.

Using the Cholesky decomposition Suppose again that (A.58) is exactly determined (i.e. $\mathtt{A}\mathbf{x} = \mathbf{b}$) and $\mathtt{A}$ is positive definite. The Cholesky decomposition from Section A.1.5 yields $\mathtt{A} = \mathtt{LL}^\top$, where $\mathtt{L}$ is a lower-triangular matrix. The linear system $\mathtt{L}(\mathtt{L}^\top\mathbf{x}) = \mathbf{b}$ is then solved in two stages similar to the LU case above.

Again, overdetermined systems can be treated by using the normal equation $(\mathtt{A}^\top\mathtt{A})\mathbf{x} = \mathtt{A}^\top\mathbf{b}$ as long as $\mathtt{A}^\top\mathtt{A}$ is positive definite.

Using the QR decomposition From Section A.1.5, $\mathtt{A} = \mathtt{QR}$ where $\mathtt{Q} \in \mathbb{R}^{m \times n}$ is a portrait or square matrix comprising orthonormal columns and $\mathtt{R} \in \mathbb{R}^{n \times n}$ is an upper-triangular matrix.

This transforms (A.59) to

$$\begin{aligned}\mathbf{x}^* &= (\mathtt{A}^\top \mathtt{A})^{-1}\mathtt{A}^\top \mathbf{b}\\ &= (\mathtt{R}^\top \mathtt{Q}^\top \mathtt{Q}\mathtt{R})^{-1}\mathtt{R}^\top \mathtt{Q}^\top \mathbf{b}\\ &= (\mathtt{R}^\top \mathtt{R})^{-1}\mathtt{R}^\top \mathtt{Q}^\top \mathbf{b} = \mathtt{R}^{-1}(\mathtt{Q}^\top \mathbf{b}). \end{aligned} \tag{A.60}$$

The computational complexity is about $4n^3/3$ [Golub and Van Loan, 1996], which is approximately twice of that of LU.

Iterative methods with preconditioning

It is also possible to solve (A.58) iteratively as long as the problem is inhomogeneous ($\mathbf{b} \neq 0$). This may be a computationally more efficient option if the linear system is too large and dense. Another plus point for these iterative methods is that intermediate solutions obtained after some number of iterations may still be usable, whereas most matrix decomposition-based methods produces usable outputs only at the end.

For all the methods described below, it will be assumed that (A.58) is preconditioned to lower the condition number of the linear system. Specifically, a symmetric invertible preconditioner $\mathtt{P} := \mathtt{L}\mathtt{L}^\top$ is applied to (A.58) as follows:

$$\begin{aligned} \mathtt{P}^{-1}\mathtt{A}\mathbf{x} &= \mathtt{P}^{-1}\mathbf{b}\\ \Rightarrow \mathtt{L}^{-1}\mathtt{A}\mathbf{x} &= \mathtt{L}^{-1}\mathbf{b}\\ \Rightarrow \mathtt{L}^{-1}\mathtt{A}\mathtt{L}^{-\top}(\mathtt{L}^\top\mathbf{x}) &= \mathtt{L}^{-1}\mathbf{b} && \text{(A.61)}\\ \Rightarrow \hat{\mathtt{A}}\hat{\mathbf{x}} &= \hat{\mathbf{b}} && \text{(A.62)} \end{aligned}$$

where $\hat{\mathbf{x}} := \mathtt{L}^\top\mathbf{x}$, $\hat{\mathtt{A}} := \mathtt{L}^{-1}\mathtt{A}\mathtt{L}^{-\top}$ and $\hat{\mathbf{b}} := \mathtt{L}^{-1}\mathbf{b}$. This ensures the symmetry of $\mathtt{A}$ is preserved after preconditioning.

Jacobi iteration The Jacobi method is one of the simplest iterative algorithms for solving $\mathtt{A}\mathbf{x} = \mathbf{b}$ when $\mathtt{A}$ is a square matrix with non-zero diagonal entries. (A.58) is rewritten as

$$\begin{aligned} \hat{\mathtt{A}}\hat{\mathbf{x}} &= \hat{\mathbf{b}} && \text{(A.63)}\\ \Rightarrow \hat{\mathbf{x}} + \hat{\mathtt{A}}\hat{\mathbf{x}} &= \hat{\mathbf{x}} + \hat{\mathbf{b}}\\ \Rightarrow \hat{\mathbf{x}} &= (\mathtt{I} - \hat{\mathtt{A}})\hat{\mathbf{x}} + \hat{\mathbf{b}}\\ \Rightarrow \hat{\mathbf{x}}_{k+1} &= (\mathtt{I} - \hat{\mathtt{A}})\hat{\mathbf{x}}_k + \hat{\mathbf{b}}. && \text{(A.64)} \end{aligned}$$

Although each update requires stationary variables only and is therefore simple to compute, it can converge slowly.

If the initial parameter $\hat{\mathbf{x}}_1$ is set to be $\hat{\mathbf{b}}$, then $\hat{\mathbf{x}}_2$ can be represented by a linear combination of $\hat{\mathbf{b}}$ and $\hat{\mathtt{A}}\hat{\mathbf{b}}$. Similarly $\hat{\mathbf{x}}_3$ can be represented by a linear combination of$\hat{\mathbf{b}}$, $\hat{\mathtt{A}}\hat{\mathbf{b}}$ and $\hat{\mathtt{A}}^2\hat{\mathbf{b}}$. In other words, $\hat{\mathbf{x}}_k$ spans the k-Krylov subspace $[\hat{\mathbf{b}},\ \hat{\mathtt{A}}\hat{\mathbf{b}},\ \cdots,\ (\hat{\mathtt{A}})^{k-1}\hat{\mathbf{b}}]$. This leads to the question "can one update more efficiently by knowing these basis as a prior?" One solution is to employ one of the Krylov subspace methods reviewed below.

Krylov subspace methods From the previous paragraph, the Jacobi iteration essentially produces updates that are in the Krylov subspace $(\hat{\mathtt{A}}, \hat{\mathbf{b}})$. By using the definition of the Krylov matrix $\mathtt{K}_k \in \mathbb{R}^{n\times k}$ in (A.41), the question is then how to choose the weight on each Krylov matrix basis vector to produce an optimal solution at each iteration. Strang [2006] showed that several iterative algorithms can be classified based on the optimality criteria, two of which are as follows:

1. The preconditioned residual $\hat{\varepsilon}_k = \hat{\mathbf{b}} - \hat{\mathtt{A}}\hat{\mathbf{x}}_k$ is orthogonal to the k-Krylov matrix $\mathtt{K}_k$.

2. The preconditioned residual $\hat{\varepsilon}_k$ has minimum norm for $\hat{\mathbf{x}}_k$ in the subspace spanned by the k-Krylov matrix $\mathtt{K}_k$.

If the first criterion is used, then the algorithm can be interpreted as the *preconditioned conjugate gradient* (PCG). If using the second criterion, the algorithm may be preconditioned *MINRES* or *GMRES* depending on whether $\mathtt{A}$ is symmetric or not.

Note that since $\hat{\mathbf{x}}_k$ is in the column space of $\mathtt{K}_k$ (as weights on each Krylov basis vector is irrelevant), $\hat{\varepsilon}_k = \hat{\mathbf{b}} - \hat{\mathtt{A}}\hat{\mathbf{x}}_k$ is in the column space of $\mathtt{K}_{k+1}$.

Preconditioned conjugate gradient (PCG) This section moves on from the previous paragraph. Preconditioned conjugate gradient can be used to find a root to $\mathtt{A}\mathbf{x} = \mathbf{b}$ if $\mathtt{A} \in \mathbb{S}^n$ and positive definite. First, the fact that the residual $\hat{\varepsilon}_k$ is orthogonal to the subspace spanned by the basis vectors of the k-Krylov matrix $\mathtt{K}_k$ means that it is iteratively solving $\hat{\mathtt{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ for square $\mathtt{A} \in \mathbb{R}^{n\times n}$. After n iterations, it would have solved $\hat{\mathtt{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$, and therefore conjugate gradient can be viewed as minimizing

$$\frac{1}{2}\hat{\mathbf{x}}^\top\hat{\mathtt{A}}\hat{\mathbf{x}} + \hat{\mathbf{b}}^\top\hat{\mathbf{x}} = \frac{1}{2}\mathbf{x}^\top\mathtt{A}\mathbf{x} + \mathbf{b}^\top\mathbf{x}. \tag{A.65}$$

The r.h.s. is obtained by back substituting the original variables.

Since the preconditioned residual $\hat{\varepsilon}_j$ is in $\mathtt{K}_j$ and the residual $\hat{\varepsilon}_k$ is orthogonal to the basis vectors of $\mathtt{K}_k$, having $j < k$ yields

$$\hat{\varepsilon}_j^\top \hat{\varepsilon}_k = 0 \ \forall \ \ j < k \tag{A.66}$$

Since the original residual is related by $\varepsilon_j = \mathtt{L}^{-1}\hat{\varepsilon}_j$,

$$\varepsilon_j^\top \mathtt{L}^{-\top}\mathtt{L}^{-1}\hat{\varepsilon}_k = \varepsilon_j^\top \mathtt{P}^{-1}\hat{\varepsilon}_k = 0 \ \forall \ \ j < k. \tag{A.67}$$

Also, notice that $\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_{j-1}$ lies in the column space of $\mathtt{K}_j$, and $\hat{\varepsilon}_k - \hat{\varepsilon}_{k-1}$ is orthogonal to the basis vectors of $\mathtt{K}_k$. Therefore, $j < k$ leads to

$$(\hat{\mathbf{x}}_j - \hat{\mathbf{x}}_{j-1})^\top (\hat{\varepsilon}_k - \hat{\varepsilon}_{k-1}) = 0 \ \forall \ \ j < k \tag{A.68}$$

$$\Rightarrow (\mathbf{x}_j - \mathbf{x}_{j-1})^\top (\varepsilon_k - \varepsilon_{k-1}) = 0 \ \forall \ \ j < k. \tag{A.69}$$

The second line is derived from using $\hat{\mathbf{x}}_j = \mathtt{L}^\top \mathbf{x}$. Since the original residual $\varepsilon_k := \mathbf{b} - \mathtt{A}\mathbf{x}_k$, (A.69) can be formulated as

$$(\mathbf{x}_j - \mathbf{x}_{j-1})^\top \mathtt{A}(\mathbf{x}_k - \mathbf{x}_{k-1}) = 0 \ \forall \ \ j < k. \tag{A.70}$$

Now, $\mathbf{x}_k$ can be viewed as

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta\mathbf{x}_{k-1} = \mathbf{x}_{k-1} + \hat{\alpha}_{k-1}\mathbf{d}_{k-1}, \tag{A.71}$$

where $\Delta\mathbf{x}_{k-1} \in \mathbb{R}^n$ is the $\mathbf{x}$-update at iteration $k-1$ and $\hat{\alpha}_k \in \mathbb{R}$ and $\mathbf{d}_k \in \mathbb{R}^n$ are the corresponding step length and update direction respectively. Similarly, the original residual $\varepsilon_k \in \mathbb{R}^n$ can be rewritten as

$$\begin{aligned}\varepsilon_k &= \mathbf{b} - \mathtt{A}\mathbf{x}_k = \mathbf{b} - \mathtt{A}\mathbf{x}_{k-1} + \mathtt{A}\mathbf{x}_{k-1} - \mathtt{A}\mathbf{x}_k \\ &= \varepsilon_{k-1} - \mathtt{A}(\mathbf{x}_k - \mathbf{x}_{k-1}) \\ &= \varepsilon_{k-1} - \hat{\alpha}_{k-1}\mathtt{A}\mathbf{d}_{k-1}.\end{aligned} \tag{A.72}$$

Hence, (A.70) can be rewritten as the conjugacy equation

$$\mathbf{d}_{j-1}^\top \mathtt{A}\mathbf{d}_{k-1} = 0 \ \forall \ \ j < k. \tag{A.73}$$

Algorithm 8 Preconditioned conjugate gradient

```
**Inputs:** $\mathbf{x}_0 \in \mathbb{R}^n$, $\mathtt{P}$ (preconditioner)
$k = 0$
$\varepsilon_k = -\mathbf{b} - \mathtt{A}\mathbf{x}_k$
$\mathbf{d}_k = -\varepsilon_k$
**while** not converged **do**
    $\hat{\alpha}_k = \frac{\varepsilon_k^\top \mathtt{P}^{-1} \varepsilon_k}{\mathbf{d}_k \mathtt{A} \mathbf{d}_k}$
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \hat{\alpha}_k \mathbf{d}_k$
    $\varepsilon_{k+1} = \varepsilon_k + \hat{\alpha}_k \mathtt{A} \mathbf{d}_k$
    $\hat{\beta}_k = \frac{\varepsilon_{k+1}^\top \mathtt{P}^{-1} \varepsilon_{k+1}}{\varepsilon_k^\top \mathtt{P}^{-1} \varepsilon_k}$
    $\mathbf{d}_{k+1} = \mathtt{P}^{-1} \varepsilon_{k+1} + \hat{\beta}_k \mathbf{d}_k$
    $k \leftarrow k + 1$
**end while**
**Output:** $\mathbf{x}_k$
```

Since $\hat{\varepsilon}_k$ is orthogonal to the basis vectors of $\mathtt{K}_k$, and the next update direction $\mathbf{d}_k$ has to be conjugate to all the previous directions, $\mathbf{d}_k$ can be formulated as

$$\mathbf{d}_k = \varepsilon_k + \beta \mathbf{d}_{k-1} \tag{A.74}$$

By combining all of this together, the iterative scheme proceeds as follows. Given the current solution $\mathbf{x}_k$ and an update direction $\mathbf{d}_k$ (with $\mathbf{d}_0 = \mathbf{b}$), find the optimum step length $\hat{\alpha}_k$ that minimizes (A.65). Then, compute $\mathbf{x}_{k+1}$ using (A.71) and the new residual ε_{k+1} using (A.72). Finally, $\hat{\mathtt{A}}$-*orthogonalize* ("conjugatize") the preconditioned residual $\hat{\varepsilon}_k + 1$ against all the previous preconditioned directions $\hat{\mathbf{d}}_k$ to find a new conjugate direction. A pseudocode is illustrated in Algorithm 8.

Other derivation of PCG shows that conjugate gradient is a variant of the Lanczos method reviewed in Section A.1.8.

MINRES MINRES is a type of Krylov subspace method briefly mentioned before the preconditioned conjugate gradient. Originally proposed by Paige and Saunders [1975], the algorithm shares the same root of the Lanczos method in Section A.1.8 with conjugate gradient. The algorithm minimizes (A.58) rather than (A.65), effectively choosing the weights of the Krylov matrix basis vectors to minimize the residual rather than to make the residual orthogonal. It is able to handle positive semidefinite system matrices ($\mathtt{A}$). MINRES can also be preconditioned to yield better performance.

Direct linear transform

For the homogeneous case of $\mathbf{b} = 0$, (A.58) contains a degenerate solution $\mathbf{x}^* = \mathbf{0}$ since the scale of the solution can be arbitrarily small. To yield a meaningful result and find the "nearest" nullspace of $\mathtt{A}$, an additional constraint is applied to (A.58) in order to fix the size of the solution. A modified problem solves

$$\arg\min_{\mathbf{x}} \|\mathtt{A}\mathbf{x}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{x}\|_2 = 1. \tag{A.75}$$

By taking the SVD (in Section A.1.5) of $\mathtt{A}$ to yield $\mathtt{A} = \mathtt{U}\Sigma\mathtt{V}^\top$ and noting that $\mathtt{A}^\top\mathtt{A} = \mathtt{V}\Sigma^2\mathtt{V}^\top$, (A.75) can be reformulated as

$$\arg\min_{\mathbf{x}} \|\Sigma\mathtt{V}^\top\mathbf{x}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{x}\|_2 = 1. \tag{A.76}$$

Hence, $\mathbf{x}$ can be simply obtained by taking the last column of $\mathtt{V}$, $\mathbf{v}_n$, which has the smallest singular value associated with it.

A.2.3 Nonlinear least squares

This section introduces some widely available iterative algorithms for solving unconstrained continuous optimization algorithms of the form

$$\arg\min_{\mathbf{x}} \frac{1}{2}\|\varepsilon(\mathbf{x})\|_2^2 \tag{A.77}$$

where $\varepsilon : \mathbb{R}^n \to \mathbb{R}^m$ is a residual vector and $\mathbf{x} \in \mathbb{R}^n$ is a vector of variables. The cost functions optimized here are assumed to be continuous and differentiable everywhere in the parameter space.

Methods for solving a robustified problem of the form $\rho(\|\varepsilon(\mathbf{x})\|_2^2)$ will be illustrated in Section A.2.4.

Gradient descent (GD)

Gradient descent is a first-order optimization algorithm that makes an update against the gradient of the cost function ($\nabla f(\mathbf{x}_k)$) at each iteration. Algorithm 9 illustrates the basic structure.

Note that, for nonlinear least squares problems of the form (A.77), $\nabla f(\mathbf{x}_k)$ can be formulated as $\mathtt{J}^\top(\mathbf{x}_k)\varepsilon(\mathbf{x}_k)$, where $\varepsilon : \mathbb{R}^n \to \mathbb{R}^m$ is the residual vector and $\mathtt{J} : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ defines the Jacobian $\mathtt{J}(\mathbf{x})$ of the residual vector $\varepsilon(\mathbf{x})$.

Algorithm 9 Naive gradient descent algorithm

```
Inputs: $\mathbf{x} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}^n$ (initial learning rate)
repeat
    Update $\alpha$ if necessary (e.g. employing line search or decaying learning rates).
    $\Delta\mathbf{x} \leftarrow -\alpha\nabla f(\mathbf{x})$
    $\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$
until convergence
Output: $\mathbf{x} \in \mathbb{R}^n$
```

Learning rates The set of learning rates $\{\alpha_k\}$ should chosen appropriately; if it is too small it may take a significant time to converge but if it is too large then consequent updates may overshoot and miss the local optimum. Additional line-search along the update direction can help to determine a suitable value for α_k. For certain classes of cost functions (e.g. linear, quadratic), α_k can be obtained analytically by differentiating the cost function (in terms of $\mathbf{x}_k + \alpha_k\nabla f(\mathbf{x}_k)$) with respect to α_k and setting it to 0.

Stochastic gradient descent (SGD) The stochastic gradient descent method, which is one of the widely used algorithms in training large models such as neural nets, formulates $f(\mathbf{x})$ as a linear sum of subsidiary cost functions and then discards some of these randomly when computing an approximate gradient (followed by rescaling of the gradient). When minimizing a least squares function, the original gradient at iteration k is $\mathtt{J}^\top(\mathbf{x}_k)\varepsilon(\mathbf{x}_k)$, and thus performing SGD accounts to just removing some rows of $\mathtt{J}(\mathbf{x}_k)$ and $\varepsilon(\mathbf{x}_k)$ followed by rescaling.

Nonlinear conjugate gradient

The conjugate gradient method illustrated in Section A.2.2 can also be modified and applied to nonlinear optimization problems. The problem is now that the system matrix $\mathtt{A}$ changes along the parameter space, and therefore the conjugacy of update directions is not preserved. Polak and Ribiere [1969], Fletcher and Reeves [1964] and Hestenes and Stiefel [1952] proposed different equations for the update directions. This algorithm is not utilized in the thesis.

Levenberg-Marquardt (LM)

Levenberg-Marquart [Levenberg, 1944; Marquardt, 1963] is a second-order optimization algorithm for solving general nonlinear least squares problems. Given a current solution $\mathbf{x}_k$ the quantity of interest is the update $\Delta\mathbf{x}_k$ to improve upon $\mathbf{x}_k$, forming $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k$. Linearizing

Algorithm 10 Levenberg-Marquardt (LM) algorithm

```
**input:** $\mathbf{x} \in \mathbb{R}^n$
$\lambda \leftarrow 10^{-4}$
**repeat**
    **repeat**
        $\Delta\mathbf{x} \leftarrow \arg\min_{\Delta\mathbf{x}} \|\varepsilon(\mathbf{x}) + \mathrm{J}(\mathbf{x})\Delta\mathbf{x}\|_2^2 + \lambda \|\Delta\mathbf{x}\|_2^2$
        $\lambda \leftarrow 10\lambda$
    **until** $\|\varepsilon(\mathbf{x}+\Delta\mathbf{x})\|_2^2 < \|\varepsilon(\mathbf{x})\|_2^2$
    $\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$
    $\lambda \leftarrow 0.01\lambda$
**until** convergence
**output:** $\mathbf{x} \in \mathbb{R}^n$
```

the residual yields

$$\varepsilon(\mathbf{x}_{k+1}) = \varepsilon(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx \varepsilon_k + \mathrm{J}_k \Delta\mathbf{x}_k, \tag{A.78}$$

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

$$\arg\min_{\Delta\mathbf{x}} \|\varepsilon_k + \mathrm{J}_k \Delta\mathbf{x}\|_2^2, \tag{A.79}$$

Solving (A.79) assumes that the cost is locally quadratic in $\Delta\mathbf{x}_k$, and therefore $\mathbf{x}_k + \Delta\mathbf{x}_k$ may not necessarily decrease the true objective $\|\varepsilon(\mathbf{x})\|_2^2$. LM regularizes the update by adding a penalty term with a damping parameter $\lambda_k \in \mathbb{R}$,

$$\Delta\mathbf{x}_k = \arg\min_{\Delta\mathbf{x}} \|\varepsilon_k + \mathrm{J}_k \Delta\mathbf{x}\|_2^2 + \lambda_k \|\Delta\mathbf{x}\|_2^2. \tag{A.80}$$

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 $\Delta\mathbf{x}_k$ improves the actual cost, the update is accepted and λ_{k+1} is decreased, making (A.80) closer to the GN update in (A.79). 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 Algorithm 10.

The solution of (A.80) is explicitly given by

$$\begin{aligned}
\Delta\mathbf{x}_k &= \arg\min_{\Delta\mathbf{x}} \left\| \begin{bmatrix} \boldsymbol{\varepsilon}_k \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathtt{J}_k \\ \sqrt{\lambda_k}\mathtt{I} \end{bmatrix} \Delta\mathbf{x} \right\|_2^2 \\
&= -\begin{bmatrix} \mathtt{J}_k \\ \sqrt{\lambda_k}\mathtt{I} \end{bmatrix}^\dagger \begin{bmatrix} \boldsymbol{\varepsilon}_k \\ \mathbf{0} \end{bmatrix} \\
&= -(\mathtt{J}_k^\top \mathtt{J}_k + \lambda_k \mathtt{I})^{-1} \mathtt{J}_k^\top \boldsymbol{\varepsilon}_k = \mathtt{J}_k^{-\lambda_k} \boldsymbol{\varepsilon}_k. \qquad \text{(A.81)}
\end{aligned}$$

Computing $\Delta\mathbf{x}_k$ can either be achieved by solving (A.81) directly using a matrix decomposition algorithm such as QR or Cholesky, or via an iterative method such as preconditioned conjugate gradients (PCG).

The Levenberg-Marquardt algorithm is an extension of the Gauss-Newton algorithm introducing a constraint on the **x**-update size [Levenberg, 1944; Marquardt, 1963]. Hence, the algorithm essentially aims to minimize (A.79) subject to a penalty term promoting $\|\delta\|_2^2 = \Delta^2$, where Δ is defined as the trust-region radius and reflects the size of the objective surface which can be approximated as quadratic. (Note that LM cannot have zero damping and therefore cannot avoid contribution from the penalty term promoting $\|\delta\|_2^2 = \Delta^2$.) Marquardt extended this by inserting a scaling matrix $\mathtt{D}$ to the constraint such that minimization would be subject to $\|\mathtt{D}\delta\|_2^2$, but the community frequently prefers Levenberg's original implementation (i.e. $\mathtt{D} = \mathtt{I}$). This dissertation also deals with Levenberg's formulation only.

Damping as a prior Adding the penalty term $\lambda_k\|\Delta\mathbf{x}\|_2^2$ can be thought as adding a prior that the update step $\Delta\mathbf{x}$ has an isotropic Gaussian distribution. This is often referred to as the *Levenberg* damping [Levenberg, 1944]. Large λ_k means lower variance, leading to less likelihood of large updates.

The penalty term can be extended to incorporate non-isotropic Gaussian priors by employing a scaling matrix $\mathtt{D} \in \mathbb{R}^{n\times n}$ within the penalty term to yield $\lambda_k\|\mathtt{D}\Delta\mathbf{x}\|_2^2$. For the case of the Marquardt damping, the prior is non-isotropic but independent.

A trust-region viewpoint (A.80) can be viewed as a Lagrangian-relaxed form of

$$\arg\min_{\Delta\mathbf{x}} \|\boldsymbol{\varepsilon}_k + \mathtt{J}_k\Delta\mathbf{x}\|_2^2 \quad \text{s.t.} \quad \|\Delta\mathbf{x}\|_2^2 \le \delta^2, \qquad \text{(A.82)}$$

where δ is the radius of the region that can be trusted as quadratic. This is because the Lagrangian form of (A.82) is

$$\arg\min_{\Delta\mathbf{x}} \|\varepsilon_k + \mathrm{J}_k\Delta\mathbf{x}\|_2^2 + \lambda_k(\|\Delta\mathbf{x}\|_2^2 - \delta^2) \quad = \arg\min_{\Delta\mathbf{x}} \|\varepsilon_k + \mathrm{J}_k\Delta\mathbf{x}\|_2^2 + \lambda_k\|\Delta\mathbf{x}\|_2^2 \tag{A.83}$$

as δ is fixed at each iteration. In this case, the damping factor λ_k can be thought as the Lagrange multiplier. In theory, this multiplier value should be chosen to satisfy the constraint $\|\Delta\mathbf{x}\|_2^2 \leq \delta^2$, but computing it exactly is a non-trivial problem in λ_k. (One algorithm is proposed by Moré and Sorensen [1983]. See Nocedal and Wright [2006, Chapter 4] for more discussions.)

Instead, the trust region radius δ is updated by rule of thumb based on the value of ρ, which computes the quality of the quadratic model. This is calculated by comparing the actual cost improvement against the estimated improvement. For example, the model quality at iteration k is

$$\rho_k := \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \Delta\mathbf{x}_k)}{h_k(0) - h_k(\Delta\mathbf{x}_k)}, \tag{A.84}$$

where $h_k(\Delta\mathbf{x})$ is the quadratic submodel used at iteration k. Now, the trust region radius δ can be updated based on the model quality as follows:

1. If $\rho_k < 0.25$, the local region is not so quadratic so δ is reduced.
2. If $\rho > 0.75$, the local region is highly quadratic so δ is increased.
3. Otherwise, the radius of the trust region is assumed adequate and δ remains unchanged.

Furthermore, the update $\Delta\mathbf{x}_k$ is accepted if $\rho > 0.25$ but otherwise rejected. In that case, a new step is computed with reduced trust region radius δ.

Newton submodel Although Levenberg-Marquardt (LM) typically uses the Gauss Newton submodel (A.79) with damping, it is possible to apply the same framework to the Newton submodel. Taking the second-order Taylor expansion of the cost function $f(\mathbf{x})$ at $\mathbf{x}_k$ yields

$$f(\mathbf{x}_k + \Delta\mathbf{x}_k) \approx f(\mathbf{x}_k) + \nabla f(\mathbf{x})^\top \Delta\mathbf{x}_k + \frac{1}{2}\Delta\mathbf{x}_k^\top(\nabla\nabla^\top f(\mathbf{x}_k))\Delta\mathbf{x}_k \tag{A.85}$$

$$=: f(\mathbf{x}_k) + \mathbf{g}_k^\top\Delta\mathbf{x}_k + \frac{1}{2}\Delta\mathbf{x}_k^\top \mathtt{H}_k\Delta\mathbf{x}_k. \tag{A.86}$$

where $\mathbf{g}_k := \nabla f(\mathbf{x}_k)$ and $\mathtt{H}_k := \nabla\nabla^\top f(\mathbf{x}_k)$. Given some damping factor λ_k, the Newton subproblem is then

$$\arg\min_{\Delta\mathbf{x}} f(\mathbf{x}_k) + \mathbf{g}_k^\top \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\top \mathtt{H}_k \Delta\mathbf{x} + \lambda_k \|\Delta\mathbf{x}\|_2^2. \tag{A.87}$$

Note that if $f(\mathbf{x}) := \frac{1}{2}\|\varepsilon(\mathbf{x})\|_2^2$, then the gradient and Hessian at iteration k are

$$\mathbf{g}_k = \mathtt{J}_k \varepsilon_k \tag{A.88}$$

$$\mathtt{H}_k = \mathtt{J}_k^\top \mathtt{J}_k + \nabla_{\mathbf{x}} \mathtt{J}(\mathbf{x}_k)\varepsilon_k, \tag{A.89}$$

where $\nabla_{\mathbf{x}} \mathtt{J}(\mathbf{x}_k) \in \mathbb{R}^{m\times n\times n}$ is a 3D tensor. The first matrix on the r.h.s. of (A.89) is called the *Gauss-Newton* matrix. Since this term is guaranteed to be at least positive semidefinite, many second order optimizers approximate the Hessian with the Gauss-Newton matrix (potentially damped). It is reported in the literature [Chen, 2011] and Chapter 3 that using this approximation yields better performance in practice than using the full Hessian.

A.2.4 Robust nonlinear least squares

Standard L^2-norm minimization yields the maximum likelihood estimate (MLE) or maximum a posteriori (MAP) of system parameters given the assumption that noise is approximately normally distributed. However, if noise follows more of a heavier-tailed distribution, the L^2 solution may deviate substantially from the ground truth value, especially when there are gross outliers in the dataset.

One way to tackle this issue is to perform L^1-norm minimization (also known as least absolute deviation) instead, which is less susceptible to outliers. To give a brief intuition behind this, given a cluster of data points, the L^1 solution yields the median of the cluster whereas the L^2 solution provides its mean, which can be sensitive to distant points. Yang et al. [2013] provides a review of recent L^1 minimization solvers.

Another way is to encapsulate the original L^2 objective (A.77) with an additional loss function (also known as robust kernel) in order to "robustify" the objective. Typically, these loss functions are assumed isotropic. If the vector residual $\varepsilon : \mathbb{R}^n \to \mathbb{R}^m$ is a concatenation of p q-dimensional residuals such that $m = pq$ (e.g. 5 reprojection errors yielding 10-dimensional residual), the goal is to solve

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\sum_{i=1}^{p} \rho\left(\|\varepsilon_i(\mathbf{x})\|_2^2\right), \tag{A.90}$$

where $\varepsilon_i : \mathbb{R}^n \to \mathbb{R}^q$ is the i-th residual vector and $\rho : \mathbb{R}^+ \to \mathbb{R}^+$ is an isotropic loss function.

Robust kernels

Many different loss functions have been proposed in the past, and a handful of them have been reviewed in the recent work of Zach and Bourmaud [2017]. Note that each robust kernel is implicitly assuming some form of underlying noise distribution. Some popular choices are illustrated below. For each loss function, $\tau \in \mathbb{R}$ is defined as the radius of the inlier boundary. A graphical illustration is shown in Fig. A.1.

Cauchy loss The Cauchy loss is a non-convex kernel defined as

$$\rho(s) := \tau^2 \log\left(1 + \frac{s}{\tau^2}\right). \tag{A.91}$$

This loss implicitly assumes that the model noise is sampled from a heavier tailed distribution than the standard normal distribution.

Geman-McClure loss The Geman-McClure loss is another non-convex kernel defined as

$$\rho(s) := \frac{\tau^2 s}{s + \tau^2}, \tag{A.92}$$

which is more robust than the Cauchy loss.

Huber loss The Huber loss function [Huber, 1964] is a convex loss function defined as

$$\rho(s) := \begin{cases} s & \text{for } s \leq \tau^2, \\ 2\tau\sqrt{s} - \tau^2 & \text{otherwise,} \end{cases} \tag{A.93}$$

where $\tau \in \mathbb{R}$ is the inlier radius (i.e. $s \leq \tau$ is the inlier region). Huber loss penalizes the inlier region with the L^2 cost and the rest with the L^1 cost, thereby reducing its sensitivity to outliers whilst maintaining its sensitivity to inliers.

Smooth truncated-quadratic loss The smooth truncated-quadratic loss is defined as

$$\rho(s) := \begin{cases} s\left(1 - \frac{s}{2\tau^2}\right) & \text{for } s \leq \tau^2, \\ \frac{1}{2}\tau^2 & \text{otherwise.} \end{cases} \tag{A.94}$$

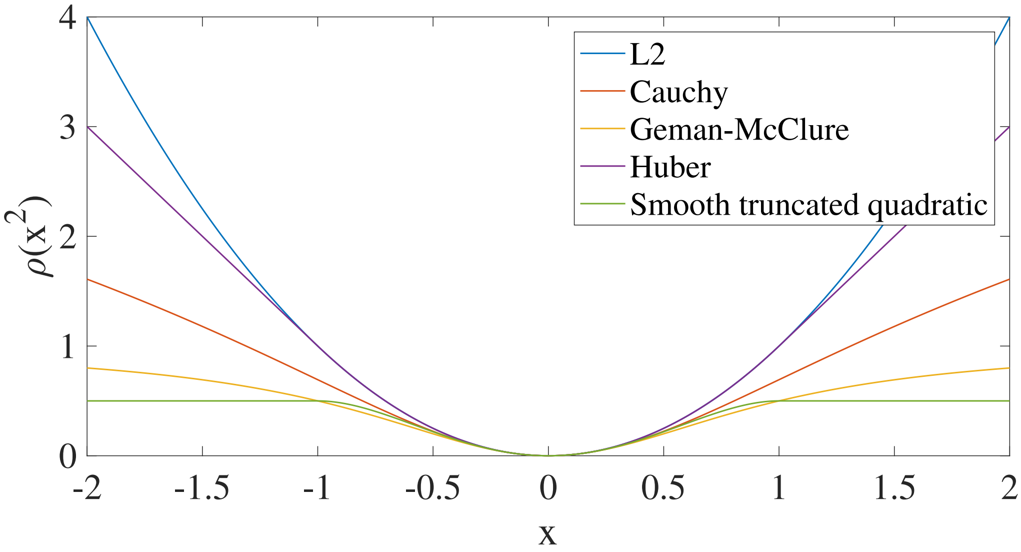


Fig. A.1 Shapes of some widely-used robust kernels. The L^2 norm is most sensitive to outliers whilst the smooth truncated quadratic kernel is least sensitive.

Triggs correction

The Triggs correction [Triggs et al., 2000] uses the chain rule to differentiate through (A.90) and obtain analytic derivatives. At iteration k, the gradient of (A.90) is

$$\sum_{i=1}^{p} \rho'_{ik} \mathtt{J}_{ik}^{\top} \varepsilon_{ik}, \tag{A.95}$$

where $\varepsilon_{ik} := \varepsilon_i(\mathbf{x}_k)$, $\rho'_{ik} := \rho'(\|\varepsilon_i(\mathbf{x}_k)\|_2^2)$ and $\mathtt{J}_{ik} := \mathtt{J}_i(\mathbf{x}_k)$. The Gauss-Newton approximation of the Hessian is then

$$\sum_{i=1}^{p} \mathtt{J}_k^{\top} \left(\rho'_{ik} \mathtt{I} + 2\rho''_{ik} \varepsilon_{ik} \varepsilon_{ik}^{\top} \right) \mathtt{J}_{ik} =: \sum_{i=1}^{p} \mathtt{J}_{ik}^{\top} \mathtt{H}_{ik} \mathtt{J}_{ik}, \tag{A.96}$$

where $\rho''_{ik} := \rho''(\|\varepsilon_i(\mathbf{x}_k)\|_2^2)$. As long as $\mathtt{H}_{ik} \succeq 0$, (A.95) and (A.96) can be used to find the update $\Delta\mathbf{x}_k$. In fact, Huber loss can be implemented efficiently by modifying the Levenberg-Marquardt (LM) algorithm from Section A.2.3 to apply some projection to the Jacobian $\mathtt{J}_k$ at each iteration. When $\mathtt{H}_k$ is indefinite, one can use the approximation $\mathtt{H}_k \approx \rho'_k \mathtt{I}$.

Square rooting

The kernel square-rooting method reformulates (A.90) as

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\sum_{i=1}^{p} \left\| \frac{\varepsilon_i(\mathbf{x})}{\|\varepsilon_i(\mathbf{x})\|_2} \sqrt{\rho\left(\|\varepsilon_i(\mathbf{x})\|_2^2\right)} \right\|_2^2, \tag{A.97}$$

and finds a solution using a second order optimizer like Levenberg-Marquardt from Section A.2.3. It can be thought as projecting the robustified residual to 1D parameter space.

Lifting

Lifting is a technique of projecting a problem to a higher dimension parameter space without changing optima of the problem. In summary, (A.90) can be converted to

$$\min_{\mathbf{x},\mathbf{w}} \frac{1}{2}\sum_{i=1}^{p} \left(w_i^2\|\varepsilon_i(\mathbf{x})\|_2^2 + \kappa^2(w_i^2)\right) = \min_{\mathbf{x},\mathbf{w}} \frac{1}{2}\sum_{i=1}^{p} \left\| \begin{matrix} w_i\varepsilon_i(\mathbf{x}) \\ \kappa(w_i^2) \end{matrix} \right\|_2^2 \tag{A.98}$$

where $\mathbf{w} \in \mathbb{R}^p$ is a vector of newly introduced variables referred to as *weights*. Usually, $\mathbf{w}$ is initialized from $\mathbf{1}$ to allow all the residuals have an equal chance of participating as inliers at the beginning of optimization.

Iteratively reweighted least squares (IRLS) IRLS [Byrd and Pyne, 1979] solves (A.98) using a block coordinate descent approach illustrated in Algorithm 1. In summary, $\mathbf{x}$ is minimized with $\mathbf{w}$ fixed, which is essentially a weighted least squares problem, and then $\mathbf{w}$ is updated using new $\mathbf{x}$.

Joint optimization Zach [2014] proposed to jointly optimize over the weights $\mathbf{w} \in \mathbb{R}^p$ and the system variables $\mathbf{x}$. As will be shown in Section 4.2.1, the two variable sets are stacked together, and Levenberg-Marquardt from Section A.2.3 is applied to minimize (A.98) over $\mathbf{w}$ and $\mathbf{x}$ simultaneously.

A.2.5 Riemannian manifold optimization

This section summarizes a way to incorporate the Riemannian manifold optimization scheme. Full mathematical background can be found in the book of Absil et al. [2008].

A Riemannian manifold is defined as a real and smooth manifold that is equipped with an inner product (Riemannian metric) everywhere on the tangent space at each point. This inner product allows geometric quantities such as angles and lengths of curves to be calculated on the manifold [Boumal et al., 2014].

Various optimization problems in computer vision have system variables lying on Riemannian manifolds. Some simple examples include symmetric eigenvalue problem, structure-from-motion with the homogeneous parameterization [Hartley and Zisserman, 2004] of 3D points and up-to-scale translations in relative pose computation (described in Section A.4.6), in which the parameters lie on a hypersphere, and a reduced instance of bilinear matrix factorization [Boumal and Absil, 2011; Chen, 2008] where one of the matrix factors lies on a Grassmann manifold (which is a subset of Riemannian manifold).

The summary below is guided by the work of Manton et al. [2003] and Boumal and Absil [2011], and is specifically aimed for optimization on Riemannian manifolds embedded in Euclidean space.

1. If an initial point $\mathtt{X}_0$ is outside its underlying manifold, retract (project back to the manifold) and compute the gradient and Hessian.

2. Project the gradient and Hessian (or its approximation) to the tangent space of current $\mathtt{X}$ by multiplying the manifold projection matrix to the original gradient and the original Hessian. (For $\mathrm{SO}(3)$, this step is equivalent to multiplying the gradient and Hessian by the infinitessimal rotation generators.)

3. Solve the **projected** linearized subproblem w/out damping to yield an update on the tangent space.

4. Since the new solution $\mathtt{X} + \Delta\mathtt{X}$ is outside the manifold unless $\Delta\mathtt{X} = \mathtt{0}$, perform retraction using the geodesic distance or an implicit retraction such as the q-factor for the Grassmann and the Stiefel manifolds. (Absil et al. describes taking the q-factor of the updated variable as an efficient way of retracting back onto the observed hemisphere. For $\mathrm{SO}(3)$, this step is equivalent to applying the exponential map onto the new estimate.)

5. Repeat Step 2 to Step 4 until convergence.

Homogeneous parameterization (spherical manifold)

When a vector $\tilde{\mathbf{x}} \in \mathbb{R}^{n+1}$ is in homogeneous coordinates, it can be formulated as lying on a n-dimensional hypersphere, which is an element of the Riemannian manifold.

As briefly described in Section A.1.4, projecting derivatives to the space orthogonal to $\tilde{\mathbf{x}}$ can be achieved by mulitplying by the orthogonal projector matrix $\tilde{\mathbf{x}}^{\perp} \in \mathbb{R}^{n+1\times n}$, which is essentially the nullspace of $\tilde{\mathbf{x}}$. This can be accomplished by either taking the SVD of $\tilde{\mathbf{x}}$ to find

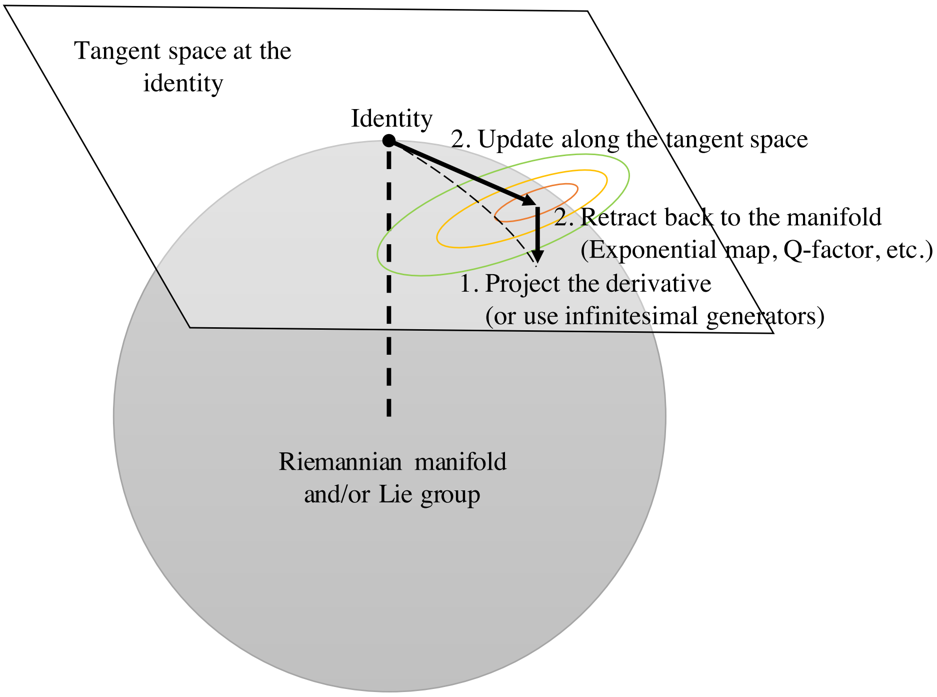

Tangent space at the identity
Identity
2. Update along the tangent space
2. Retract back to the manifold (Exponential map, Q-factor, etc.)
1. Project the derivative (or use infinitesimal generators)
Riemannian manifold and/or Lie group

Fig. A.2 A geometric interpretation of Riemannian manifold and/or Lie group optimization.

nullspace vectors or by using the Householder matrix as described by Hartley and Zisserman [2004].

Grassmann manifold

A Grassmann manifold can be viewed as a subspace spanned by matrix $\mathtt{X} \in \mathbb{R}^{m\times n}$ $(m > n)$ which satisfies the constraint $\tilde{\mathtt{X}}^\top \tilde{\mathtt{X}}$ for some $\tilde{\mathtt{X}}$ generated by a linear sum of the basis vectors of $\mathtt{X}$ (i.e. $\tilde{\mathtt{X}} = \mathtt{XA}$ for some $\mathtt{A} \in \mathbb{R}^{n\times n}$). Formally, this manifold corresponds to the set $\{\text{span}\ (\mathtt{X}) : \mathtt{X} \in \mathbb{R}^{m\times n},\ \mathtt{X}^\top\mathtt{X} = \mathtt{I}\}$. Practically speaking, if $\mathtt{X}$ is involved in a cost function $f(\mathtt{X})$, the following relationship holds for any invertible $\mathtt{A} \in \mathbb{R}^{n\times n}$:

$$f(\mathtt{X}) = f(\mathtt{XA}) \tag{A.99}$$

In other words, if there exists a solution $\mathtt{X}^*$, then the entire set of matrices that spans the column space of $\mathtt{X}^*$ is also a valid solution.

As mentioned by Edelman et al. [1998] and Absil et al. [2008], the orthogonal projector to the tangent space of $\mathtt{X}$ is effectively defined by the equation

$$\mathtt{P} := \mathtt{I}_{mn} - \mathtt{I}_n \otimes \left(\mathtt{X}(\mathtt{X}^\top\mathtt{X})^{-1}\mathtt{X}^\top\right). \tag{A.100}$$

However, projecting the Hessian to the tangent space with (A.100) still suffers from rank deficiency as it does not reduce the actual dimension of the problem. Therefore, one has to constrain the update direction, which can be achieved in two ways. First, one can explicitly downsize the problem [Chen, 2008; Manton et al., 2003] by using projection matrix $(\mathtt{I}_n \otimes \mathtt{X}^{\perp}) \in \mathbb{R}^{mn\times(m-n)}$ (via SVD) instead of $\mathtt{P}$. Second, one can introduce a penalty term [Okatani et al., 2011] on the projected Hessian to constrain the update $\Delta\mathtt{X}$ along the column space of $\mathtt{X}$ (e.g. by using $\|\mathtt{X}\Delta\mathtt{X}\|_2$). This may require the penalty coefficient to be adjusted depending on the scale of the Hessian.

Once $\Delta\mathtt{X}$ is obtained using the projected gradient and Hessian, the new solution $\mathtt{X}+\Delta\mathtt{X}$ can be projected back onto the manifold by using an efficient (non-geodesic) retraction given by

$$\mathtt{X} \leftarrow qf(\mathtt{X}+\Delta\mathtt{X}) \tag{A.101}$$

where qf stands for the Q-factor in the QR decomposition. If $\mathtt{X}$ is a vector, then $qf(\cdot)$ simply boils down to vector normalization.

A.2.6 Handling equality constraints

This section briefly covers some methods for performing optimization with equality constraints as these are utilized in several sections of this thesis. All of the approaches mentioned below solve

$$\arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \quad s.t. \quad \mathbf{h}(\mathbf{x}) = \mathbf{0}, \tag{A.102}$$

where $f(\mathbf{x})$ is the cost function and $\mathbf{h}: \mathbb{R}^n \to \mathbb{R}^m$ is the vector of equality constraints.

Note that a *feasible* solution must satisfy the constraints $\mathbf{h}(\mathbf{x}) = \mathbf{0}$. Also, as shown in Fig. A.3, the gradient of the cost function at the solution $\mathbf{x}^*$ must be orthogonal to the tangent space of $\mathbf{h}(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}^*$. This space is spanned by the column space of $\nabla\mathbf{h}(\mathbf{x}^*)$.

Lagrange multipliers

The method of Lagrange multipliers finds a local minimum of (A.102) by first defining its Lagrange function

$$L(\mathbf{x},\lambda) := f(\mathbf{x}) + \lambda^\top \mathbf{h}(\mathbf{x}), \tag{A.103}$$

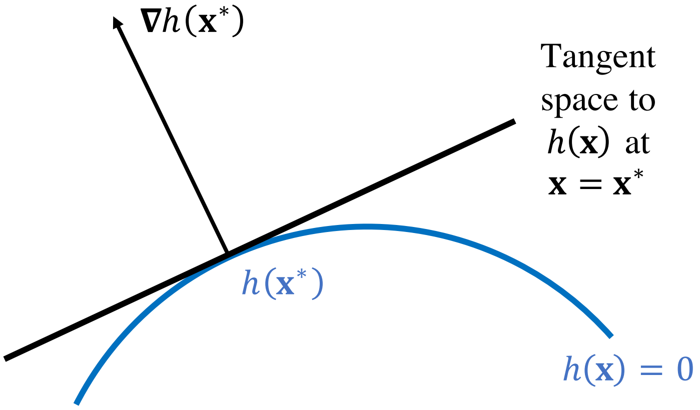

$\nabla h(\mathbf{x}^*)$
Tangent space to $h(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}^*$
$h(\mathbf{x}^*)$
$h(\mathbf{x}) = 0$

Fig. A.3 A geometric interpretation of constrained optimization where $h(\mathbf{x})$ is an equality constraint. Note that the gradient of the cost function at solution $\mathbf{x}^*$ must be orthogonal to the tangent space of $h(\mathbf{x})$ at $x = x^*$, because otherwise $\mathbf{x}^*$ will move to a new position.

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuous differentiable cost function and λ is the vector of Lagrange multipliers. The gradient of (A.103) is

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda) = \nabla_{\mathbf{x}} f(\mathbf{x}) + \nabla_{\mathbf{x}} \mathbf{h}(\mathbf{x})\lambda. \tag{A.104}$$

The first optimality condition tells us that $\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda) = 0$, leading to

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = -\nabla_{\mathbf{x}} \mathbf{h}(\mathbf{x})\lambda. \tag{A.105}$$

Above means that the gradient vector $\nabla_{\mathbf{x}} f(\mathbf{x})$ lies on the subspace spanned by $\nabla_{\mathbf{x}} \mathbf{h}(\mathbf{x})$. In other words, $\nabla_{\mathbf{x}} f(\mathbf{x})$ is orthogonal to the tangent space. This is a necessary condition for optimality, and therefore one may need to confirm the result by looking at the second optimality condition.

Combining (A.104) with $\mathbf{h}(\mathbf{x}) = 0$ yields $(n+m)$ equations for $(n+m)$ parameters.

Quadratic penalty

Instead of using the method of Lagrange multiplier from Section A.2.6, one can decide a minimize

$$\arg\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{h}(\mathbf{x})\|_2^2, \tag{A.106}$$

where $\rho \in \mathbb{R}$ is a penalty coefficient. This parameter controls the trade off between lowering the cost function and fitting the constraints. One advantage of using the penalty term is that it is simple to add into a nonlinear least squares solver, since

$$\frac{1}{2}\|\varepsilon(\mathbf{x})\|_2^2 + \frac{\rho}{2}\|\mathbf{h}(\mathbf{x})\|_2^2 = \frac{1}{2}\left\|\begin{matrix}\varepsilon(\mathbf{x}) \\ \sqrt{\rho}\mathbf{h}(\mathbf{x})\end{matrix}\right\|_2^2 \tag{A.107}$$

However, one of the main disadvantages is that ρ has to become very large to numerically satisfy $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, which can make the Jacobian of (A.107) numerically unstable.

Augmented Lagrange multiplier (ALM)

The method of augmented Lagrange multiplier [Hestenes, 1969] formally solves

$$\arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) + \lambda^\top \mathbf{h}(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{h}(\mathbf{x})\|_2^2, \tag{A.108}$$

where λ is the vector of Lagrange multipliers, $\mathbf{h}(\mathbf{x})$ is the vector of constraints and ρ is the penalty coefficient. (A.108) can be viewed as a shifted quadratic penalty function or a convexification of the Lagrangian function.

Taking the first viewpoint, (A.108) is equivalent to

$$\arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) + \frac{\rho}{2}\left\|\mathbf{h}(\mathbf{x}) + \frac{1}{\rho}\lambda\right\|_2^2. \tag{A.109}$$

Hence, the constraint $\mathbf{h}(\mathbf{x})$ is now approximately satisfied by $-\lambda/\rho$. In terms of Lagrangian multipliers, the first optimality condition yields

$$\begin{aligned}&\nabla f(\mathbf{x}) + \nabla\mathbf{h}(\mathbf{x})\lambda + \rho\nabla\mathbf{h}(\mathbf{x})\mathbf{h}(\mathbf{x}) = 0 \\ \Rightarrow\ &\nabla f(\mathbf{x}) = -\nabla\mathbf{h}(\mathbf{x})(\lambda + \rho\mathbf{h}(\mathbf{x})).\end{aligned} \tag{A.110}$$

.

Del Bue et al. [2012] and Cabral et al. [2013] have proposed iterative schemes for solving (A.108). Essentially, the schemes work in an alternating fashion, minimizing $\mathbf{x}$ for fixed λ and ρ, then updating λ and ρ for fixed $\mathbf{x}$.

Implicit function theorem

The implicit function theorem is useful if the constraint vector $\mathbf{x}$ can be segmented into two sets of variables $\mathbf{u}$ and $\mathbf{v}$ and the constraint vector $\mathbf{h}(\mathbf{u},\mathbf{v})$ has the same dimension as $\mathbf{v}$ as well as being continuously differentiable. Without loss of generality, $\mathbf{x}$ will be defined as $[\mathbf{u};\mathbf{v}]$.

Algorithm 11 Random sample consensus (RANSAC)

```
Inputs: $N$ data points, $t$ (inlier threshold), $k_{max}$
for $k = 0, \dots, k_{max}$ do
    1. Sample $m$ points from $N$ data points.
    2. Estimate model parameters from the sampled points.
    3. Find the number of all data points that are within the inlier threshold $t$.
    4. If the current # inliers > so-far maximum # inliers, save the current model.
end for
Output: model parameters
```

The theorem states that around the local neighbourhood of a feasible point $(\mathbf{u}, \mathbf{v})$ (i.e. $\mathbf{h}(\mathbf{u}, \mathbf{v}) = 0$), $d\mathbf{u}/d\mathbf{v}$ can be obtained analytically if $\nabla_{\mathbf{v}}\mathbf{h}(\mathbf{u}, \mathbf{v})$ is invertible. Moreover, it follows

$$\frac{d\mathbf{v}}{d\mathbf{u}} = -\nabla_{\mathbf{v}}\mathbf{h}(\mathbf{u}, \mathbf{v})^{-1}\nabla_{\mathbf{u}}\mathbf{h}(\mathbf{u}, \mathbf{v}). \tag{A.111}$$

A.2.7 Random sample consensus (RANSAC)

The robust optimization methods listed in Section A.2.4 have a common issue that, depending on the starting point, it is very likely to get stuck in a bad local minimum. This is because many robust kernels weigh large deviations much less than a conventional L^2-norm does, and therefore once the optimizer is stuck in some parameter space, the majority (if not all) of the gradient information outside the inlier radius is lost. This can especially be a problem in cases when a significant proportion of data is corrupted, such that only 10–50% are true inliers.

To circumvent this issue, Fischler and Bolles [1981] proposed a simple scheme called *Random sample consensus* (RANSAC), in which data is repeatedly sampled to fit a set of proposal models, and the model with the most number of inliers is chosen. As this scheme runs for a fixed number of iterations, estimating an optimal number can be derived from the following illustration.

Suppose there are N data points, out of which I are inlier points. If m points are sampled each time, then the probability (p_I) of fetching all inlier points can be obtained analytically. Chum et al. [2003] uses the following approximation to yield

$$p_I = \frac{{}_IC_m}{{}_NC_m} = \frac{I!}{m!(I-m)!}\frac{m!(N-m)!}{N!} = \prod_{j=0}^{m-1}\frac{I-j}{N-j} \approx \left(\frac{I}{N}\right)^m =: \gamma^m, \tag{A.112}$$

where γ is the proportion of inliers. The r.h.s. approximation is valid if I and N are much larger than m. Now, the probability of not being able to sample a set of all inliers is $1 - p_I$, and the probability of doing that k_{max} times is $p_{fail} := (1 - p_I)^k$. Given a threshold probability of failure, the approximate number of iterations required (k) satisfies

$$k_{max} \geq \frac{\log(p_{fail})}{\log(1-p_I)} = \frac{\log(p_{fail})}{\log(1-\gamma^m)}. \tag{A.113}$$

To compute k_{max}, the proportion of inliers (γ) has to be estimated.

Variants

Some of the widely known variants of RANSAC are reviewed below. Many more variants have been extensively reviewed by Raguram et al. [2008].

LORANSAC During the RANSAC iteration in Algorithm 11, Locally Optimized RANSAC (LORANSAC) has an additional step (after step 4) of refining the local model if that model is selected as the "so-far best". Refinement can be in the form of least squares over all inlier-classified points or nested RANSAC over the inliers.

MLESAC MLESAC finds the maximum likelihood estimate of model parameters rather than the model with the highest number of inliers.

PROSAC Progressive RANSAC (PROSAC) sorts data points by some quality measure (e.g. SIFT scores for feature correspondences), then performs RANSAC initially on a small number of high quality data points, gradually increasing the size of the pool.

A.3 Camera projection models

This section introduces several camera projection models and some key geometric concepts that are utilized in later chapters. Most of these illustrations are also available in the review work of Wang and Wu [2011].

For all of the cases illustrated below, the radial lens distortion is assumed to be negligible, allowing the camera projection models to have a linear injective mapping between 3D points and 2D pixels in homogeneous coordinates. Incorporating the radial distortions term requires a nonlinear mapping [Zhang, 2000].

A.3.1 Homogeneous coordinates

This section illustrates the homogeneous coordinate system briefly.

The introduction of homogeneous coordinates dates back to the work of August Ferdinand M̈obius in 1827 [Smith, 1906]. If there is a point $\mathbf{x} \in \mathbb{R}^n$, its homogeneous representation is defined as $\tilde{\mathbf{x}} \in \mathbb{R}^{n+1} := [s\mathbf{x}\ ;\ s]$ for a non-zero $s \in \mathbb{R}$.

Homogeneous line equation Any line passing through $\mathbf{x}$ can be parameterized by $\mathbf{l} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that $\mathbf{l}^\top \mathbf{x} = c$. This can be rewritten in homogeneous form as

$$\begin{bmatrix} \mathbf{l} \\ -c \end{bmatrix}^\top \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} =: \tilde{\mathbf{l}}^\top \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \tag{A.114}$$

$$\Rightarrow \tilde{\mathbf{l}}^\top \tilde{\mathbf{x}} = 0, \tag{A.115}$$

where $\tilde{\mathbf{l}} :\simeq [\mathbf{l}\ ;\ -c]$. (A.115) shows that $\tilde{\mathbf{l}}$ is orthogonal to the column space spanned by the homogeneous points along the line.

2D line equation For the special case of $n = 2$, $\mathbf{x} \in \mathbb{R}^2$ is a 2D point and $\tilde{\mathbf{x}} \in \mathbb{R}^3$ can be viewed as a ray. In this case, $\tilde{\mathbf{l}} \in \mathbb{R}^3$ can be obtained by simply taking the cross product of two homogeneous points on the line. If there are two homogeneous points $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$, then

$$\tilde{\mathbf{l}} = \tilde{\mathbf{x}}_1 \times \tilde{\mathbf{x}}_2. \tag{A.116}$$

If a third homogeneous point $\tilde{\mathbf{x}}_3$ also lies on the same line, then $\tilde{\mathbf{x}}_3$ must be orthogonal to $\tilde{\mathbf{l}}$, i.e. $\tilde{\mathbf{l}}^\top \tilde{\mathbf{x}}_3 = 0$.

2D point to line distance Suppose there is a third point $\mathbf{x}_3 \in \mathbb{R}^n$. Remember that (A.115) in inhomogeneous form is $\mathbf{l}^\top \mathbf{x} = c$, and that $|c|/\|\mathbf{l}\|_2$ signifies the perpendicular distance from the line to the origin. Similarly, $|\mathbf{l}^\top \mathbf{x}_3|/\|\mathbf{l}\|_2$ yields the component of the distance between $\mathbf{x}_3$ and the origin in the direction of $\mathbf{l}$. Hence, the perpendicular distance between $\mathbf{x}_3$ and the line, $d_l(\mathbf{x}_3, \mathbf{l})$, can be computed by the equation

$$d_l(\mathbf{x}_3, \mathbf{l}) := \frac{|\mathbf{l}^\top \mathbf{x}_3 - c|}{\|\mathbf{l}\|_2} = \frac{|\tilde{\mathbf{l}}^\top \tilde{\mathbf{x}}_3|}{\left([\tilde{\mathbf{l}}]_1^2 + [\tilde{\mathbf{l}}]_2^2\right)^{1/2}}, \tag{A.117}$$

where $[\tilde{\mathbf{l}}]_1$ and $[\tilde{\mathbf{l}}]_2$ are the first and second elements of $\tilde{\mathbf{l}}$ respectively.

A graphical illustration is provided in Fig. A.4.

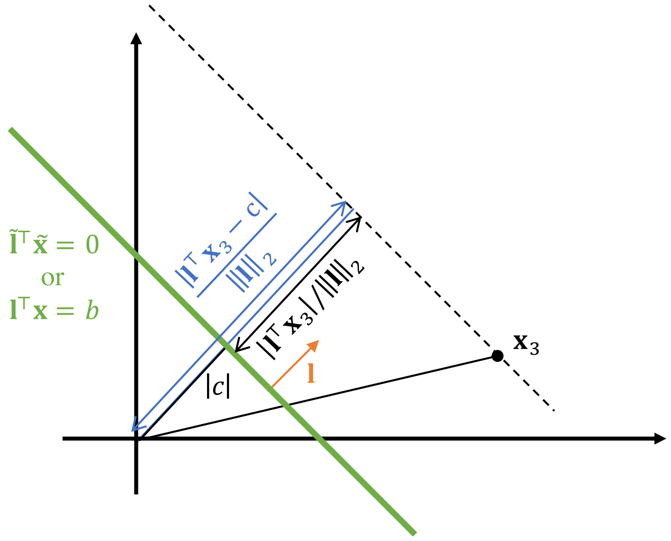

$\tilde{\mathbf{l}}^\top \tilde{\mathbf{x}} = 0$
or
$\mathbf{l}^\top \mathbf{x} = b$
$\frac{|\mathbf{l}^\top \mathbf{x}_3 - c|}{\|\mathbf{l}\|_2}$
$|\mathbf{l}^\top \mathbf{x}_3| / \|\mathbf{l}\|_2$
$|c|$
$\mathbf{l}$
$\mathbf{x}_3$

Fig. A.4 A geometric interpretation of the 2D point-to-line distance.

A.3.2 Calibrated models

Calibrated models refer to the projection models with known camera intrinsics, namely the focal length, pixel scaling, principal point and skew. These are parameterized by a single upper diagonal matrix $\mathtt{K}_i \in \mathbb{R}^{3\times 3}$ defined as

$$\mathtt{K} := \begin{bmatrix} f\,\alpha_u & s & u_0 \\ 0 & f\,\alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} =: \begin{bmatrix} f_u & s & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.118}$$

where f_u and f_v are the effective horizontal and vertical focal lengths respectively (incorporating the focal length f and the pixel scalings via α_u and α_v), s represents the skew of pixels and $[u_0;\ v_0]$ define the principal point. The skew factor is usually treated as 0 for modern digital cameras.

A graphical illustration of the projection models reviewed below is provided in Fig. A.5.

Perspective projection

The perspective model is one of the most widely used projection models in computer vision. Here, all rays passing through the image plane travel straight without any distortion and meet at the optical centre of the camera. In this case, the camera projection matrix injectively mapping a 3D point in world coordinates ($\mathbf{x}_j \in \mathbb{R}^3$) to a 2D point on the image plane ($\mathbf{m}_j \in \mathbb{R}^2$) can be

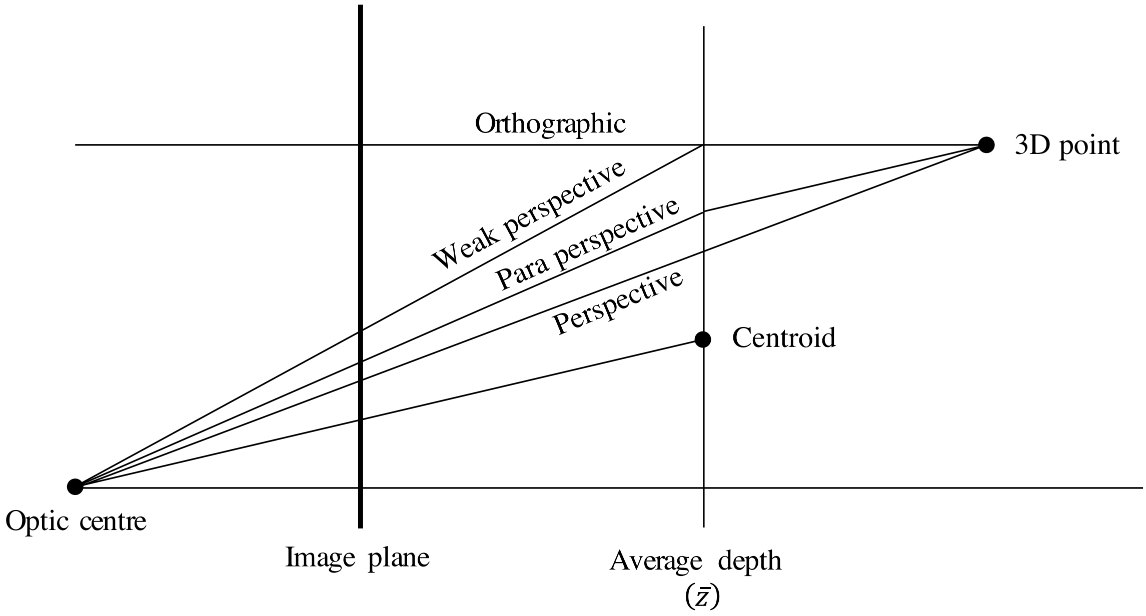


Fig. A.5 A list of calibrated camera projection models. The perspective ray joins the optic centre and the 3D point. The orthographic ray travels orthogonal to the image plane (i.e. no scaling). The weak perspective (or scaled orthographic) ray travels like the orthographic ray until it hits the plane of the "average depth" (because depth is constant), and then it scales down. Finally, the para perspective ray travels parallel to the perspective ray between the optic centre and the centroid of the 3D points until it hits the average depth plane, and then it scales down as well. The para perspective projection can be viewed as a generalization of the weak perspective projection.

decomposed as

$$\mathtt{P_p} = \mathtt{K}[\mathtt{R},\ \mathbf{t}] = \mathtt{KR}[\mathtt{I},\ -\mathbf{c}] \tag{A.119}$$

where $\mathtt{K} \in \mathbb{R}^{3\times 3}$ is the intrinsics matrix of camera as shown in (A.118), $[\mathtt{R}, \mathbf{t}] \in \mathrm{SE}(3)$ represents the transformation from world coordinates to the camera's local coordinates, $\mathbf{c} \in \mathbb{R}^3$ is the optic centre. Note that by inspection, the nullspace of $\mathtt{P}$ is the camera centre $\mathbf{c}$.

If a feature point $\mathbf{x}_j \in \mathbb{R}^3$ is visible in the camera, then it is projected onto the image plane as $\mathbf{m}_j := [u_j, v_j]^\top$, which can be written as

$$u_j = \frac{f_u(\mathbf{r}_1^\top \mathbf{x}_j + t_1)}{\mathbf{r}_3^\top \mathbf{x}_j + t_3} + u_0 =: \frac{f_u(\mathbf{r}_1^\top \mathbf{x} + t_i)}{z_j} + u_0 \tag{A.120}$$

$$v_j = \frac{f_v(\mathbf{r}_2^\top \mathbf{x}_j + t_2)}{\mathbf{r}_3^\top \mathbf{x}_j + t_3} + v_0 =: \frac{f_v(\mathbf{r}_2^\top \mathbf{x} + t_i)}{z_j} + v_0 \tag{A.121}$$

where $\mathbf{r}_i$ is the i-th row vector of $\mathtt{R} \in \mathrm{SO}(3)$, $t_3 \in \mathbb{R}$ is the third element of $\mathbf{t} \in \mathbb{R}^3$ and $z_j \in \mathbb{R}$ is the perspective depth of point j in camera i's viewpoint.

Now, suppose there is a cluster of 3D points viewed by camera i. The average perspective depth of these points, further defined as $\bar{z}$, can be computed using the equation

$$\bar{z} = \mathbf{r}_3^\top \bar{\mathbf{x}} + t_3 \tag{A.122}$$

where $\bar{\mathbf{x}} \in \mathbb{R}^3$ is the centroid of the 3D points. Then z_j can be represented as $\bar{z} + \Delta z_j$, where

$$\Delta z_j = \mathbf{r}_3^\top (\mathbf{x}_j - \bar{\mathbf{x}}). \tag{A.123}$$

Therefore, (A.120) can be rewritten as

$$u_j = \frac{f_u(\mathbf{r}_1^\top \mathbf{x}_j + t_1)}{\bar{z}\left(1 + \frac{\Delta z_j}{\bar{z}}\right)} + u_0 \tag{A.124}$$

In the literature, the words pinhole camera and perspective camera are somtimes used interchangeably. The only minor difference between them is that the perspective model assumes a (virtual and non-inverted) image plane between the optical centre and the 3D point while the pinhole camera has a physical inverted image plane behind the optic centre after rays are focused. For simplicity, this work will also use both terms to refer to the perspective projection.

If there is a significant lens distortion, images should be preprocessed to be undistorted.

Weak perspective projection

Suppose there is a cluster of points viewed by a camera. The weak perspective model makes a strong assumption that the clustered 3D points have comparatively very small depth variations from the average perspective depth (i.e. $\Delta z_j/\bar{z} << 1$).

In the weak perspective model, this assumption is carried over to the zeroth order Taylor expansion of (A.120) and (A.121) in Δz_j about $\bar{z}$, yielding

$$u_j \approx \frac{f_u}{\bar{z}}(\mathbf{r}_1^\top \mathbf{x}_j + t_1) + u_0 \tag{A.125}$$

$$v_j \approx \frac{f_v}{\bar{z}}(\mathbf{r}_2^\top \mathbf{x}_j + t_2) + v_0 \tag{A.126}$$

This can also be written in the matrix form in homogeneous coordinates as

$$\tilde{\mathbf{m}}_j := \begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} = \mathtt{K} \begin{bmatrix} \mathbf{r}_1^\top & t_1 \\ \mathbf{r}_2^\top & t_2 \\ \mathbf{0}^\top & \bar{z} \end{bmatrix} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} \tag{A.127}$$

Note that it is possible to work in inhomogeneous coordinates by writing (A.127) as

$$\mathbf{m}_j := \begin{bmatrix} u_j \\ v_j \end{bmatrix} = \mathtt{K} \begin{bmatrix} \mathbf{r}_1^\top/\bar{z} & t_1/\bar{z} \\ \mathbf{r}_2^\top/\bar{z} & t_2/\bar{z} \end{bmatrix} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} =: \mathtt{KP}_{\mathtt{wp}}\tilde{\mathbf{x}}_j \tag{A.128}$$

where $\mathtt{P}_{\mathtt{wp}} \in \mathbb{R}^{2\times 4}$ is the inhomogeneous weak perspective camera model.

The weak perspective projection is sometimes referred to as scaled-orthographic since the projection requires division by the constant average depth value.

Orthographic projection

The orthographic projection is simply the weak perspective projection without scaling. In other words, there is no average depth division. Hence, moving an orthographic camera back and forth does not change the image observed. In terms of equations,

$$\mathbf{m}_j := \begin{bmatrix} u_j \\ v_j \end{bmatrix} = \mathtt{K} \begin{bmatrix} \mathbf{r}_1^\top & t_1 \\ \mathbf{r}_2^\top & t_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} =: \mathtt{KP}_{\mathtt{orth}}\tilde{\mathbf{x}}_j. \tag{A.129}$$

Para perspective projection

The para perspective model was first proposed by Ohta et al. [1981]. This can be viewed as taking the first order Taylor approximation of (A.124) with respect to the depth variation Δz.

However, this assumes a scene with narrow field of view where all the image points are not so far away from the projection of the centroid. If this does not hold, then the error induced by the para perspective ray can be significant, and even larger than the weak perspective one. (In Fig. A.5, imagine the centroid is moved far below the axis perpendicular to the image plane. Then, the weak perspective ray is actually closer to the perspective ray.) Perhaps due to this reason, the para perspective model is not used as commonly as others. A detailed analysis of the para perspective factorization scheme is given by Poelman and Kanade [1997].

A.3.3 Uncalibrated models

Projective projection

The projective projection model can be viewed as a generalized form of the perspective model from Section A.3.2. It is simply described by a 3×4 real matrix, and there is no internal constraint within the matrix such as the rotation part being in $\mathrm{SO}(3)$.

Given a projective model $\mathtt{P}_{\mathtt{proj}} \in \mathbb{R}^{3\times4}$, the projection $\tilde{\mathbf{m}}_j \in \mathbb{R}^3$ (in homogeneous coordinates) of a feature point $\tilde{\mathbf{x}}_j \in \mathbb{R}^4$ in homogeneous coordinates is obtained by the equation

$$\tilde{\mathbf{m}}_j = \mathtt{P}_{\mathtt{proj}}\tilde{\mathbf{x}}_j. \tag{A.130}$$

In (A.130), any invertible real matrix $\mathtt{H} \in \mathbb{R}^{4\times4}$ can be added to yield an equivalent solution $(\mathtt{P}_{\mathtt{proj}}\mathtt{H})(\mathtt{H}^{-1}\tilde{\mathbf{x}}_j)$. This is termed *projective ambiguity*, which must be resolved to yield a perspective (or approximately perspective) camera of the form $\mathtt{K}[\mathtt{R}\mathbf{t}]$ from the projective one. This resolving process called *metric upgrade* is briefly illustrated in Section 2.2.5.

Affine projection

The affine projection model can be viewed as a generalized version of the weak perspective and the para perspective models. The homogeneous camera matrix has the form

$$\mathtt{P}_{\mathtt{aff}} := \begin{bmatrix} p11 & p12 & p13 & p14 \\ p21 & p22 & p23 & p24 \\ 0 & 0 & 0 & p34 \end{bmatrix}. \tag{A.131}$$

The nullspace of the above matrix has the last element as 0 in homogeneous coordinates, meaning that any affine camera centre is positioned at infinity with the sign not clearly defined. Hence, it is probably more appropriate to treat affine cameras as rays passing through the point and the image plane.

In this model, the projection $\tilde{\mathbf{m}}_j \in \mathbb{R}^3$ of a feature point $\tilde{\mathbf{x}}_j \in \mathbb{R}^4$ in homogeneous coordinates is given by

$$\tilde{\mathbf{m}}_j = \mathtt{P}_{\mathtt{aff}} \tilde{\mathbf{x}}_j. \tag{A.132}$$

It is frequently observed [Gotardo and Martinez, 2011; Tomasi and Kanade, 1992; Zheng et al., 2012] that (A.132) is formulated in inhomogeneous coordinates by dividing all elements of $\mathtt{P}_{\mathtt{aff}}$ by the last element $(p34)$. This yields

$$\mathbf{m}_j = \begin{bmatrix} p11 & p12 & p13 & p14 \\ p21 & p22 & p23 & p24 \end{bmatrix} \begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix}. \tag{A.133}$$

Similar to the projective model, there is affine gauge freedom. i.e. any invertible matrix $\mathtt{H} \in \mathbb{R}^{4\times4}$ of the form

$$\begin{bmatrix} \mathtt{A} & \mathbf{b} \\ \mathbf{0}^\top & d \end{bmatrix} \tag{A.134}$$

can yield an equivalent solution $(\mathtt{P}_{\mathtt{aff}}\mathtt{H})(\mathtt{H}^{-1}\tilde{\mathbf{x}}_j)$. ($d = 1$ if using the inhomogeneous representation.)

A.3.4 Rotation parameterization

Each camera view may have a different local coordinate system from the world coordinates. Changing a coordinate system requires a transformation comprising rotation and translation. This section will briefly summarize some of the widely used parameterizations for rotation.

Euler's rotation theorem states that the rotation of a rigid object involves a single rotation about an axis. The minimal representation involves 3 parameters, but it it is possible to use an overparameterization such as quaternions (4 parameters). Each of the parameterizations reviewed below has a closed form function for mapping the parameters to the corresponding rotation matrix $\mathtt{R} \in \mathrm{SO}(3)$.

Euler angles

The method of Euler angles, first discovered by Leonhard Euler, is a minimal parameterization technique representing rotation $\mathtt{R}$ as a series of three rotational motions—yaw, pitch and roll about some predefined axes in the rotating object's frame. In terms of equations,

$$\mathtt{R} = \mathtt{R}_{\mathbf{x}}(\alpha)\mathtt{R}_{\mathbf{y}}(\beta)\mathtt{R}_{\mathbf{z}}(\gamma) \tag{A.135}$$

where $\mathtt{R}_{\mathbf{x}}$, $\mathtt{R}_{\mathbf{y}}$ and $\mathtt{R}_{\mathbf{z}}$ are the submodular rotation matrices and α, β and γ are variables. The order in which these subrotations are applied can change.

Although this method is intuitively simple to understand, the factorization nature of $\mathtt{R}_{\mathbf{x}}$, $\mathtt{R}_{\mathbf{y}}$ and $\mathtt{R}_{\mathbf{z}}$ makes the problem highly nonlinear in α, β and γ, making it difficult to optimize. Also, there is a singularity condition called *gimbal lock* in which two or more rotating axes coincide and trigger degenerate rotations. This singularity motion can occur quite frequently since it only requires a 90° rotation in one of the axes. Consequently, the use of Euler angles in computer vision is somewhat limited.

Axis-angle

Axis-angle is a widely used minimal parameterization technique (also implemented in Chapter 6), which uses a 3D real vector to represent both the angle and the axis. Given an input vector $\omega \in \mathbb{R}^3$, the rotation angle θ and the axis $\hat{\mathbf{n}}$ are given by

$$\theta := \|\omega\|_2 \tag{A.136}$$

$$\hat{\mathbf{n}} := \frac{\omega}{\|\omega\|_2} =: \hat{\omega}. \tag{A.137}$$

The rotation matrix $\mathtt{R}$ is given by Rodrigues [1816]' formula

$$\mathtt{R} = \mathtt{I} + \left(\frac{\sin\theta}{\theta}\right)[\omega]_\times + \left(\frac{1-\cos\theta}{\theta^2}\right)[\omega]_\times^2. \tag{A.138}$$

Above can be derived from geometrically expressing a point rotating about the axis defined by $\omega/\|\omega\|_2$ by angle θ. Bear in mind that when $\theta \approx 0$, one should use the first order Taylor approximation of(A.138), which is $\mathtt{I} + [\omega]_\times$. Note that there is a singularity point at $\theta = \pi$.

The axis-angle parameterization is implemented in the Ceres solver [Agarwal et al., 2014].

Lie group manifold optimization The axis-angle representation can also be derived algebraically in the context of manifold optimization. As illustrated by Eade [2017], first note that rotation matrices form a compact matrix lie group. This means SO(3) is also a differentiable manifold and has a nice property that its constituent lie algebra $\mathfrak{so}(3)$ describes the tangent space (from Section A.2.5) at the identity. There are 3 tangent space directions, termed *generators*, which are defined as

$$\mathtt{G}_1 := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad \mathtt{G}_2 := \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad \mathtt{G}_3 := \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{A.139}$$

Hence, any small movement from the identity along the tangent plane can be represented by $\sum_{k=1}^{3} \omega_k \mathtt{G}_k$ where $\omega := [\omega_1,\ \omega_2,\ \omega_3]$ is the scale vector. But since $\{\mathtt{G}_k\}$ spans the basis of a skew symmetric matrix defined in (A.4), the movement (lie algebra) can also be written as $[\omega]_\times$. This means that when there is a point $\mathbf{x} \in \mathbb{R}^3$, the infinitesimal rotation at the identity is given by $[\omega]_\times \mathbf{x} =: \|\omega\|_2 \hat{\omega} \times \mathbf{x}$ where $\hat{\omega}$ is a unit vector. This result intuitively shows that the direction of ω corresponds to the axis of rotation.

Each Lie group has its own *exponential* map that maps the lie algebra to the corresponding group. In the view point of manifold optimization, this can be interpreted as moving along the tangent space via lie algebra followed by retracting back to the manifold using the matrix exponential map (A.5). Just to state the results, computing this yields

$$\begin{aligned} \exp([\omega]_\times) &= \mathtt{I} + [\omega]_\times + \frac{1}{2}[\omega]_\times^2 + \cdots \\ &= \mathtt{I} + \left(\frac{\sin\theta}{\theta}\right)[\omega]_\times + \left(\frac{1-\cos\theta}{\theta^2}\right)[\omega]_\times^2 \end{aligned} \tag{A.140}$$

where $\theta := \|\omega\|_2$. (A.140) is identical to (A.138). Expectedly, the derivative of (A.138) with respect to ω at the identity will yield a $3 \times 3 \times 3$ tensor comprising $\mathtt{G}_1$, $\mathtt{G}_2$ and $\mathtt{G}_3$.

The differences between applying the Rodrigues' formula directly by differentiating it and using manifold optimization are as follows. With the latter method, the derivatives are projected to the tangent space (lie algebra) at the identity. Hence, the position of identity has to be adjusted at each iteration to a new estimate such that the tangent space around the new solution can be computed correctly. This requires storing some information about aggregated rotations. Unfortunately, there is a lack of performance comparison between the two strategies.

Unit quaternions

Quternion is a type of number system that can also be used to represent rotations. It has 4 dimensions with one scalar term and three quaternion units ($\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$) which obey some internal rules.

Using similar notations from Section A.3.4, a rotation about an axis $\hat{\mathbf{n}}$ through an angle θ can be represented by a unit quaternion

$$\mathbf{q} = \exp\left(\frac{\theta}{2}\hat{n}_1\mathbf{i} + \hat{n}_2\mathbf{j} + \hat{n}_3\mathbf{k}\right) = \cos\frac{\theta}{2} + (\hat{n}_1\mathbf{i} + \hat{n}_2\mathbf{j} + \hat{n}_3\mathbf{k})\sin\frac{\theta}{2}. \tag{A.141}$$

Its inverse is well-defined as

$$\mathbf{q}^{-1} = \exp\left(\frac{\theta}{2}\hat{n}_1\mathbf{i} - \hat{n}_2\mathbf{j} + \hat{n}_3\mathbf{k}\right) = \cos\frac{\theta}{2} - (\hat{n}_1\mathbf{i} + \hat{n}_2\mathbf{j} + \hat{n}_3\mathbf{k})\sin\frac{\theta}{2}. \tag{A.142}$$

Now, given a point $\mathbf{x} \in \mathbb{R}^3$, one can rotate this point by using the formula $\mathbf{q}(x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k})\mathbf{q}^{-1}$.

One problem with the quaternion representation is that it is essentially an overparameterization unless the unit norm constraint is applied. One way to apply this is to replace $\mathbf{q}$ by $\mathbf{q}/\|\mathbf{q}\|_2$ in the cost function. Although this does fix the rank, it also makes the problem highly nonlinear. Another way is to assume $\mathbf{q}$ lies on a 4D hypersphere, which is a type of the Riemannian manifold, and apply the manifold optimization scheme from Section A.2.5. Although this does seem to work practically, there are always two quaternions that yield the same rotation, and therefore viewing the manifold in this way may not be mathematically pleasing.

Quaternions Zheng et al. [2012] that the weak perspective camera models illustrated in Section A.3.2 can be represented by raw unnormalized quaternions. (A.128) shows that the norm of the unnormalized quaternion gives the inverse of the average scene depth ($\bar{z}$).

A.3.5 Combined parameterization

The rotation and translation parameters can be considered together as being in the SE(3) group [Eade, 2017], which is another compact Lie matrix group with Lie algebra $\mathfrak{se}(3)$. One can apply the same logic described in Section A.3.4 to perform manifold optimization. In brief, derivatives are projected to the tangent space at the identity, and updates are computed along the tangent space. If an iteration is successful, then the position of identity is moved to the new position.

A.4 Perspective two-view geometry

The geometry of two camera views, sometimes called the $epipolar geometry$, plays an essential role in structure-from-motion as it yields useful constraints for the problem. This section will go over the epipolar geometry and illustrate ways to solve.

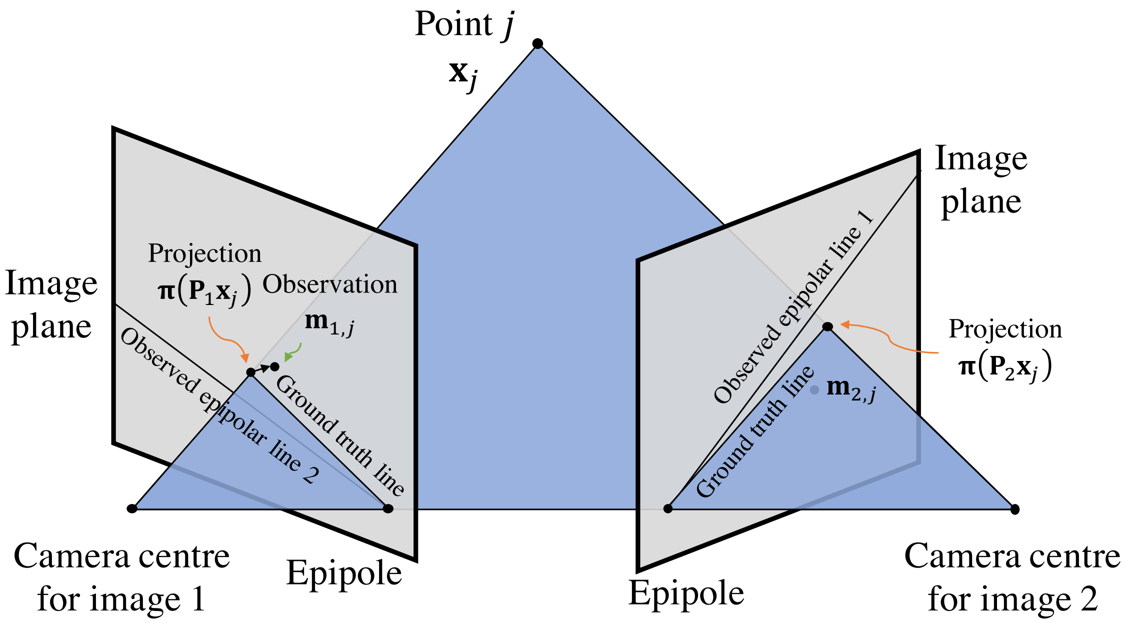


Fig. A.6 Epipolar geometry with noisy measurements. This illustration is provided for a fixed fundamental matrix (otherwise the locations of the epipoles can change as well).

A.4.1 Epipolar constraint

Suppose an unknown feature point $\mathbf{x}_j \in \mathbb{R}^3$ is viewed by two cameras with camera intrinsics $\mathtt{K}_1 \in \mathbb{R}^{3\times3}$ and $\mathtt{K}_2 \in \mathbb{R}^{3\times3}$ respectively. As a result, the first camera observes $\mathbf{m}_{1,j} \in \mathbb{R}^2$ pixel coordinates and similarly, the second camera observes $\mathbf{m}_{2,j} \in \mathbb{R}^2$. Both of these are known values. These quantities in homogeneous coordinates are defined as $\tilde{\mathbf{m}}_{1,j} \in \mathbb{R}^3$ and $\tilde{\mathbf{m}}_{2,j} \in \mathbb{R}^3$ respectively.

Without loss of generality, assume the local coordinate system of camera 1 is identical to that of the world. Since camera 1 observes $\mathbf{m}_{1,j}$ on the image plane, this corresponds to $\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j}$ in camera 1's coordinates.

Suppose the local coordinate system of camera 2 requires some unknown transformation of world (or camera 1) coordinates, parameterized as rotation by $\mathtt{R} \in \mathrm{SO}(3)$ followed by some translation $\mathbf{t} \in \mathbb{R}^3$. In this situation, the optical centre of camera 1, which is $\mathbf{0}$, would be observed as $\mathbf{t}$ on the image plane of camera 2. This is formally defined as the *epipole*. Also, the point $\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j}$ on camera 1's image plane is observed as

$$\mathtt{R}\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j} + \mathbf{t} \tag{A.143}$$

on the image plane of camera 2 in homogeneous coordinates.

Epipole and epipolar line Now, the ray from the optical centre of camera 1 passing through $\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j}$ will be observed as a 2D line on the image plane of camera 2, and this is defined as the *epipolar line*. Since the epipolar line passes through the epipole $\mathbf{t}$ and the "transferred" image point (A.143), Section A.3.1 indicates this line can be parameterized as a homogeneous equation $\tilde{\mathbf{u}}^\top\tilde{\mathbf{l}} = 0$, where $\tilde{\mathbf{u}}$ is a point along the epipolar line and

$$\begin{aligned}\tilde{\mathbf{l}} &:= \mathbf{t} \times \mathtt{R}\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j} + \mathbf{t}\\ &= [\mathbf{t}]_\times \mathtt{R}\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j}.\end{aligned} \tag{A.144}$$

$[\mathbf{t}]_\times$ is a linear map representation of the original cross product operator.

Finally, if the measurements are noise-free, the projection of $\mathbf{x}_j$ on the image plane of camera 2 ($\tilde{\mathbf{m}}_{2,j}$ in pixels and $\mathtt{K}_2^{-1}\tilde{\mathbf{m}}_{2,j}$ in camera 2's coordinates) must also lie on the epipolar line. i.e.

$$\begin{aligned}&(\mathtt{K}_2^{-1}\tilde{\mathbf{m}}_{2,j})^\top\tilde{\mathbf{l}} = 0\\ \Rightarrow &\tilde{\mathbf{m}}_{2,j}^\top \mathtt{K}_2^{-\top}[\mathbf{t}]_\times \mathtt{R}\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j} = 0.\end{aligned} \tag{A.145}$$

(A.145) is known as the epipolar constraint.

Essential matrix

Longuet-Higgins [1981] termed the quantity $[\mathbf{t}]_\times\mathtt{R} \in \mathbb{R}^{3\times 3}$ as the *essential matrix* $\mathtt{E}$. This has 5 degrees of freedom (3 for rotation and 2 for translation) and is rank 2 due to scale ambiguity in translation. As a result, $\mathtt{E}$ has a property that its two non-zero singular values are the same.

(A.145) can now be formulated as

$$\tilde{\mathbf{u}}_{2,j}^\top \, \mathtt{E} \, \tilde{\mathbf{u}}_{1,j} = 0, \tag{A.146}$$

where $\tilde{\mathbf{u}}_{1,j} := \mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j}$ and $\tilde{\mathbf{u}}_{2,j} := \mathtt{K}_2^{-1}\tilde{\mathbf{m}}_{2,j}$ are normalized points. These can also be described as rays from each camera's origin through $\mathbf{x}_j$ in homogeneous coordinates.

Fundamental matrix

Similar to the way the perspective camera model generalizes to the projective model in Section A.3, the essential matrix may be generalized as a rank-2 3 × 3 real matrix formally defined as the *fundamental matrix* $\mathtt{F}$, which was proposed by Luong and Faugeras [1996]. Given the essential matrix $\mathtt{E}$, the fundamental matrix can be computed by the equation

$$\mathtt{F} = \mathtt{K}_2^{-\top}\,\mathtt{E}\,\mathtt{K}_1^{-1}. \tag{A.147}$$

Hence, (A.145) can be written as

$$\tilde{\mathbf{m}}_{2,j}^{\top}\,\mathtt{F}\,\tilde{\mathbf{m}}_{1,j} = 0. \tag{A.148}$$

This has 8 degrees of freedom again due scale ambiguity. A fundamental matrix does not necessarily have two equal non-zero singular values.

A.4.2 Homography constraint

Both essential and fundamental matrices in Section A.4.1 are derived from the epipolar constraint (A.145), which is by nature a point-to-line geometric constraint. In certain situations, there may exist a stronger point-to-point transformation constraint. This arises when

1. two (potentially different) cameras are viewing a plane, or
2. two images are observed by the same camera undergoing a purely rotational motion.

Viewing a plane

Assume camera 1 is in world coordinates. The viewed plane can be formulated by the standard 3D plane equation

$$\hat{\mathbf{n}}^{\top}\mathbf{x}_j = d, \tag{A.149}$$

where $\hat{\mathbf{n}} \in \mathbb{R}^3$ is the plane normal, $\mathbf{x}_j \in \mathbb{R}^3$ is the feature point j in world coordinates and d is the shortest distance between the plane and the origin. Note $\mathbf{x}_j \simeq \mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j}$ from Section A.4.1.

Now suppose that transforming the coordinate system of camera 1 to that of camera 2 requires some rotation by $\mathtt{R} \in \mathrm{SO}(3)$ and translation by $\mathbf{t} \in \mathbb{R}^3$. i.e.

$$\tilde{\mathbf{m}}_{2,j} = \mathtt{R}\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j} + \mathbf{t}, \tag{A.150}$$

where $\tilde{\mathbf{m}}_{1,j} \in \mathbb{R}^3$ and $\tilde{\mathbf{m}}_{2,j} \in \mathbb{R}^3$ are the projections of $\mathbf{x}_j$ in homogeneous coordinates on the image plane of camera 1 and that of camera 2 respectively. Combining (A.149) with (A.150) yields

$$\begin{aligned}\tilde{\mathbf{m}}_{2,j} &= \mathtt{K}_2\mathtt{R}\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j} + \frac{1}{d}\mathbf{t}(\hat{\mathbf{n}}^\top \mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j})\\ &= \mathtt{K}_2\left(\mathtt{R}+\frac{1}{d}\mathbf{t}\hat{\mathbf{n}}^\top\right)\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j} \qquad\qquad =: \mathtt{H}\tilde{\mathbf{m}}_{1,j},,\end{aligned} \tag{A.151}$$

where $\mathtt{H} \in \mathbb{R}^{3\times 3}$ is the *homography* matrix that has 8 degrees of freedom due to scale ambiguity. This represents a direct invertible transformation between the projections of $\mathbf{x}_j$ in camera 1 and 2.

Purely rotational camera motion

Homography can also arise for the case of a camera purely rotating around the scene, e.g. panorama. The intrinsics of this camera is denoted as $\mathtt{K}$.

Suppose without loss of generality that view 1 is in world coordinates whilst view 2 has a coordinate system that requires a linear transformation of $[\mathtt{R}_2, \mathbf{0}]$ applied to the world coordinates. If a feature point $\mathbf{x}_j$ is visible in both views, the projections of this point in view 1 and view 2 are $\tilde{\mathbf{m}}_{1,j} = \mathtt{K}_1\mathbf{x}_j$ and $\tilde{\mathbf{m}}_{2,j} = \mathtt{K}_2\mathtt{R}_2\mathbf{x}_j$ respectively due to zero translational component. Hence, the homography matrix between the two projections is given by

$$\tilde{\mathbf{m}}_{2,j} = \mathtt{K}_2\mathtt{R}_2\mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j} =: \mathtt{H}\tilde{\mathbf{m}}_{1,j}, \tag{A.152}$$

where $\mathtt{H} \in \mathbb{R}^{3\times 3}$ is a *homography* matrix with 8 degrees of freedom.

A.4.3 Geometric errors

In presence of noise in data, the constraints described above are not likely to be abided exactly. This section briefly goes over some error distance measures that can be used to quantify how closely the constraints are met.

Epipolar distance

In presence of noise, it is almost likely that the projection of $\mathbf{x}_j$ on the image plane of camera 2 ($\mathtt{K}_2^{-1}\tilde{\mathbf{m}}_{2,j}$) does not lie on the epipolar line. By referring to Section A.3.1 again, the perpendicular between the epipolar line and the projection $\tilde{\mathbf{m}}_{2,j}$ can be found by computing

$$\frac{\tilde{\mathbf{m}}_{2,j}^\top \mathtt{K}_2^{-\top}\ \mathtt{E}\ \mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j}}{\left([\mathtt{K}_2^{-\top}\ \mathtt{E}\ \mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j}]_1^2 + [\mathtt{K}_2^{-\top}\ \mathtt{E}\ \mathtt{K}_1^{-1}\tilde{\mathbf{m}}_{1,j}]_2^2\right)^{1/2}} = \frac{\tilde{\mathbf{m}}_{2,j}^\top\ \mathtt{F}\ \tilde{\mathbf{m}}_{1,j}}{\left([\mathtt{F}\ \tilde{\mathbf{m}}_{1,j}]_1^2 + [\mathtt{F}\ \tilde{\mathbf{m}}_{1,j}]_2^2\right)^{1/2}}. \tag{A.153}$$

The above distance is measured in pixels.

Since (A.153) only measures the epipolar distance in one image, it is possible to extend it to incoporate epipolar distances in both images. This yields the symmetric epipolar distance defined as

$$\frac{1}{2}\tilde{\mathbf{m}}_{2,j}^{\top}\,\mathtt{F}\,\tilde{\mathbf{m}}_{1,j}\left(\frac{1}{\left([\mathtt{F}\,\tilde{\mathbf{m}}_{1,j}]_1^2+[\mathtt{F}\,\tilde{\mathbf{m}}_{1,j}]_2^2\right)^{1/2}}+\frac{1}{\left([\mathtt{F}^{\top}\,\tilde{\mathbf{m}}_{2,j}]_1^2+[\mathtt{F}^{\top}\,\tilde{\mathbf{m}}_{2,j}]_2^2\right)^{1/2}}\right). \tag{A.154}$$

Reprojection error

The reprojection error is the distance between the estimated 2D projection and its actual observation on the image plane of the camera concerned. Assuming again that camera 1 is in world coordinates (i.e. identity rotation and no translation), the projection of the point $\mathbf{x}_j \in \mathbb{R}^3$ on the image plane of camera 1 is $\mathtt{K}_1\mathbf{x}_j$ in homogeneous coordinates. If the actual observation is defined (similarly to other sections) as $\mathbf{m}_{1,j} \in \mathbb{R}^2$, the 2D error on the image plane of camera 1 is computed as

$$\pi(\mathtt{K}_1\mathbf{x}_j) - \mathbf{m}_{1,j}, \tag{A.155}$$

where $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ is the projection function defined as $\pi([x,\ y,\ z]^{\top}) := [x/z,\ y/z]^{\top}$.

If camera 2's coordinate system requires rotation of the world coordinates by $\mathtt{R}$ followed by some translation $\mathbf{t}$, the projection of $\mathbf{x}_j$ on the image plane of camera 2 is $\mathtt{K}_2(\mathtt{R}\mathbf{x}_j+\mathbf{t})$. Hence, the reprojection error between the measurement and the projection is

$$\pi(\mathtt{K}_2(\mathtt{R}\mathbf{x}_j+\mathbf{t})) - \mathbf{m}_{2,j}. \tag{A.156}$$

Finally, the norm of the symmetric reprojection error is simply

$$\sqrt{\frac{1}{2}\left(\|\pi(\mathtt{K}_1\mathbf{x}_j)-\mathbf{m}_{1,j}\|_2^2+\|\pi(\mathtt{K}_2(\mathtt{R}\mathbf{x}_j+\mathbf{t}))-\mathbf{m}_{2,j}\|_2^2\right)}. \tag{A.157}$$

Note that the reprojection error is favourable as it fully incorporate the underlying perspective projection model with additive noise on the image plane. However, computing (A.157) requires an explicit estimate of the 3D point location unless there exists a homography between the two views considered.

Homography case Homography is essentially an invertible transformation matrix that can transfer one set of 2D observations to another. If there exists a homography matrix $\mathtt{H} \in \mathbb{R}^{3\times 3}$ that transfers observations in camera 1 to camera 2 such that $\tilde{\mathbf{m}}_{2,j} = \mathtt{H}\tilde{\mathbf{m}}_{1,j}$, the reprojection error on the image plane of camera 2 is

$$\|\pi(\mathtt{H}\tilde{\mathbf{m}}_{1,j}) - \mathbf{m}_{2,j}\|_2, \tag{A.158}$$

and the reprojection error on the image plane of camera 1 is

$$\|\pi(\mathtt{H}^{-1}\tilde{\mathbf{m}}_{2,j}) - \mathbf{m}_{1,j}\|_2. \tag{A.159}$$

The norm of the symmetric error is then

$$\sqrt{\frac{1}{2}\left(\|\pi(\mathtt{H}^{-1}\tilde{\mathbf{m}}_{2,j}) - \mathbf{m}_{1,j}\|_2^2 + \|\pi(\mathtt{H}\tilde{\mathbf{m}}_{1,j}) - \mathbf{m}_{2,j}\|_2^2\right)}. \tag{A.160}$$

Sampson error

Although the reprojection error most closely incorporates the underlying projection model, it has an inevitable drawback that the 3D position of matched feature points must be estimated unless there exists a homography between the two views (planar scene or rotation-only motion). This estimation process, called *triangulation*, can involve comparatively costly nonlinear optimization (as will be discussed in Section A.4.7).

Sampson [1982] proposed an alternative error metric, which closely resembles the reprojection error when the error value is small.

Suppose there is a pair of matched observations $\mathbf{m}_{1,j} \in \mathbb{R}^2$ and $\mathbf{m}_{2,j} \in \mathbb{R}^2$ which does not exactly satisfy the epipolar constraint (A.145) due to some additive sensor noise. In other words, there will be some pixelwise perturbation $\delta_1 \in \mathbb{R}^2$ and $\delta_2 \in \mathbb{R}^2$ that will correct the observations accordingly so satisfy the epipolar constraint. This can be formulated this as

$$\varepsilon(\delta_1, \delta_2) = \begin{bmatrix} \mathbf{m}_{2,j} + \delta_2 \\ 1 \end{bmatrix}^\top \mathtt{F} \begin{bmatrix} \mathbf{m}_{1,j} + \delta_1 \\ 1 \end{bmatrix} = 0. \tag{A.161}$$

The question is, what are the optimal δ_1 and δ_2? Sampson used a single joint Gauss-Newton step (briefly reviewed in Section A.2.3) to derive these, which essentially uses a first order approximation of $\varepsilon(\delta_1, \delta_2)$. First, define $\delta := [\delta_1;\ \delta_2]$. Then, the first order approximation of (A.161) yields

$$\varepsilon(\delta) \approx \varepsilon_0 + \mathtt{J}_0\delta, \tag{A.162}$$

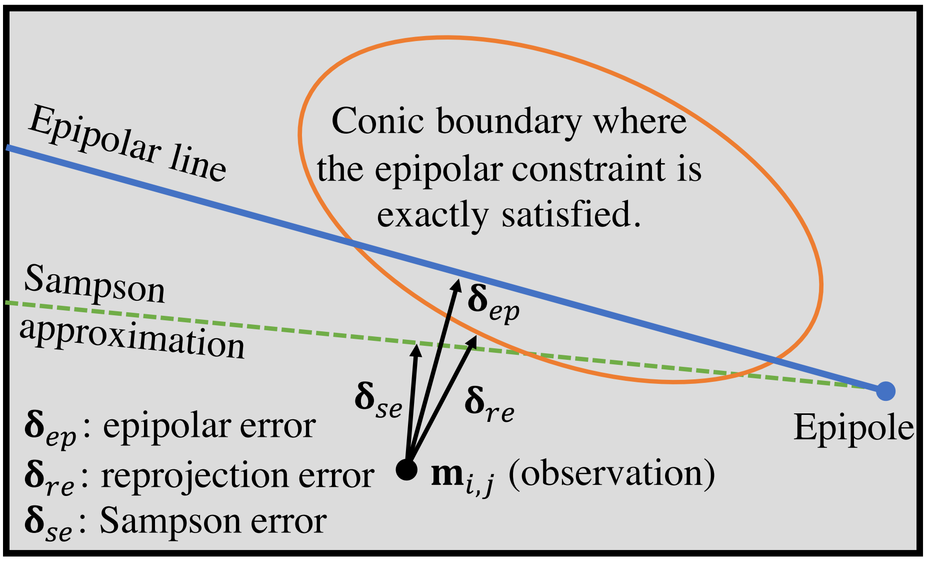

Epipolar line
Conic boundary where the epipolar constraint is exactly satisfied.
Sampson approximation
$\boldsymbol{\delta}_{ep}$
$\boldsymbol{\delta}_{se}$
$\boldsymbol{\delta}_{re}$
Epipole
$\boldsymbol{\delta}_{ep}$: epipolar error
$\boldsymbol{\delta}_{re}$: reprojection error
$\boldsymbol{\delta}_{se}$: Sampson error
$\mathbf{m}_{i,j}$ (observation)

Fig. A.7 Graphical illustration of different geometric errors illustrated in Section A.4.3. The epipolar error is the minimum distance shift of $\mathbf{m}_{i,j}$ required to satisfy the epipolar constraint (see Section A.4.1) assuming that the measurement in the other image is noise-free. On the other hand, the reprojection error finds the minimum distance shifts required to satisfy the epipolar constraint when both measurements are allowed to vary (i.e. forms an ellipse). Sampson error is the first order approximation of this reprojection error.

where ε_0 is the current error ($[\mathbf{m}_{2,j};1]^\top \mathtt{F}[\mathbf{m}_{1,j};1]$) and $\mathtt{J}_\delta \in \mathbb{R}^{1\times 4}$ is the Jacobian with respect to δ. Hence, $\mathtt{J}_\delta$ can be written as

$$\mathtt{J}_\delta := \left[[\mathbf{m}_{2,j}^\top,\ 1]\mathtt{F} \quad [\mathbf{m}_{1,j}^\top,\ 1]\mathtt{F}^\top \right], \tag{A.163}$$

and the ordinary Gauss-Newton step would have been

$$\delta = -\mathtt{J}_\delta^\dagger \varepsilon_0. \tag{A.164}$$

However, the problem here is that (A.162) is an underdetermined system, leading to infinitely many solutions. Hence, Sampson proposed to find the smallest step δ which satisfies (A.164). This can be solved using the method of Lagrange multipliers in Section A.2.6, minimizing

$$\arg\min_{\delta} \frac{1}{2}\|\delta\|_2^2 + \lambda(\mathtt{J}_\delta \delta + \varepsilon_0) \tag{A.165}$$

for nonzero λ. Setting the gradient w.r.t. δ and λ to zeros produces

$$\delta + \lambda\,\mathrm{J}_{\delta}^{\top} = 0, \tag{A.166}$$

$$\mathrm{J}_{\delta}\delta + \varepsilon_0 = 0. \tag{A.167}$$

Substituting (A.166) to (A.167) yields

$$-\lambda\,\mathrm{J}_{\delta}\mathrm{J}_{\delta}^{\top} + \varepsilon_0 = 0 \tag{A.168}$$

$$\Rightarrow \lambda = (\mathrm{J}_{\delta}\mathrm{J}_{\delta}^{\top})^{-1}\varepsilon_0, \tag{A.169}$$

and finally substituting (A.169) back into (A.166) yields

$$\delta = -\mathrm{J}_{\delta}\mathrm{J}_{\delta}\mathrm{J}_{\delta}^{\top^{-1}}\varepsilon_0. \tag{A.170}$$

This is defined as the Sampson error, and its norm is calculated as

$$\begin{aligned}
\|\delta\|_2 &= \sqrt{\varepsilon_0^{\top}(\mathrm{J}_{\delta}\mathrm{J}_{\delta}^{\top})^{-1}\mathrm{J}_{\delta}^{\top}\mathrm{J}_{\delta}(\mathrm{J}_{\delta}\mathrm{J}_{\delta}^{\top})^{-1}\varepsilon_0} \\
&= \sqrt{\frac{\varepsilon_0^{\top}\varepsilon_0}{\mathrm{J}_{\delta}\mathrm{J}_{\delta}^{\top}}} \qquad \text{(A.171)} \\
&= \frac{\left|[\mathbf{m}_{2,j};1]^{\top}\mathrm{F}[\mathbf{m}_{1,j};1]\right|}{\sqrt{\|\mathrm{F}[\mathbf{m}_{1,j};1]\|_2^2 + \|\mathrm{F}^{\top}[\mathbf{m}_{2,j};1]\|_2^2}} \qquad \text{(A.172)} \\
&= \frac{\left|[\mathbf{m}_{2,j};1]^{\top}\mathrm{F}[\mathbf{m}_{1,j};1]\right|}{\sqrt{[\mathrm{F}[\mathbf{m}_{1,j};1]]_1^2 + [\mathrm{F}[\mathbf{m}_{1,j};1]]_2^2 + [\mathrm{F}^{\top}[\mathbf{m}_{2,j};1]]_1^2 + [\mathrm{F}^{\top}[\mathbf{m}_{2,j};1]]_2^2}}. \qquad \text{(A.173)}
\end{aligned}$$

Although (A.173) looks similar to the symmetric epipolar distance shown in (A.154), Fathy et al. [2011] observed that the two exhibit different behaviours, with the latter yielding a biased estimate of the reprojection error.

A.4.4 Minimal solvers

This section introduces some mainstream algorithms that solve for the geometric constraints illustrated above. These solvers are termed minimal as each use the minimum number of correspondences required to estimate the underlying geometry. Note that all these algorithms minimize some algebraic distance, which is not necessarily geometrically meaningful.

Developing new minimal solvers is an active topic of research in computer vision.

8-point algorithm

It was Longuet-Higgins [1981] who first developed the 8-point algorithm. As in previous sections, define $\tilde{\mathbf{m}}_{i,j}$ as the projection of the feature point $\mathbf{x}_j$ on the image plane of view i in homogeneous coordinates. By utilizing the identities from Section A.1.2, (A.148) can be rewritten as follows:

$$\tilde{\mathbf{m}}_{2,j}^\top \, \mathtt{F} \, \tilde{\mathbf{m}}_{1,j} = \operatorname{vec}(\tilde{\mathbf{m}}_{2,j}^\top \, \mathtt{F} \, \tilde{\mathbf{m}}_{1,j}) = (\tilde{\mathbf{m}}_{1,j}^\top \otimes \tilde{\mathbf{m}}_{2,j}^\top) \operatorname{vec} \mathtt{F}. \tag{A.174}$$

Note that each correspondence gives a single epipolar constraint, and therefore minimum of 8 correspondences are needed to fulfil the 8 degrees of freedom residing in a fundamental matrix. Bear in mind that the homogeneous scale factor must be consistent across the correspondences and therefore is usually set to 1. This yields

$$\begin{bmatrix} \tilde{\mathbf{m}}_{2,1}^\top \otimes \tilde{\mathbf{m}}_{1,1}^\top \\ \vdots \\ \tilde{\mathbf{m}}_{2,8}^\top \otimes \tilde{\mathbf{m}}_{1,8}^\top \end{bmatrix} \operatorname{vec} \mathtt{F} = \mathbf{0}, \tag{A.175}$$

which is a homogeneous least squares problem of the form $\mathtt{A}\mathbf{x} = \mathbf{0}$ discussed in Section A.2.2. $\mathtt{F}$ is typically computed using the direct linear transformation (DLT) algorithm reviewed in Section A.2.2.

Enforcing the fundamental matrix constraint The solution obtained from (A.175) may be full rank as no direct rank 2 constraint is imposed. To yield a proper fundamental matrix of rank 2, one needs to solve

$$\arg\min_{\mathtt{F}'} \|\mathtt{F} - \mathtt{F}'\|_F^2 \quad s.t. \operatorname{rank}(\mathtt{F}') = 2. \tag{A.176}$$

This can be solved using the solution illustrated in Section 2.3.2, which is equivalent to taking the SVD of $\mathtt{F}$ and then re-multiplying the SVD factors with the third singular value set to 0.

Normalization for numerical stability Hartley [1997] showed that normalization of observation values (in pixels) is critical in obtaining high quality fundamental matrices. This is because (A.147) implies that the top left 2×2 sub-block of a fundamental matrix is numerically much smaller than the bottom right value (as camera intrinsics $\mathtt{K}$ typically have high focal lengths). To achieve this, the cluster of observed 2D projections in each view is first translated such that its centroid is at the origin and then scaled to set the average distance between each observation and the origin to be $\sqrt{2}$. For each cluster of observations, such transformation can be represented by an upper diagonal matrix $\mathtt{T}_i$ (similar to the camera intrinsics matrix without skew) defined as

$$\mathtt{T}_i := \begin{bmatrix} s_i & 0 & 0 \\ 0 & s_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -c_{i,x} \\ 0 & 1 & -c_{i,y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_i & 0 & -s_i c_{i,x} \\ 0 & s_i & -s_i c_{i,y} \\ 0 & 0 & 1 \end{bmatrix}, \tag{A.177}$$

where i represents the view (camera) number, s_i is the inverse scaling factor and $[c_x,\ c_y]^\top$ is the centroid of the observations.

After the fundamental matrix is computed, it is de-normalized by using the equation $T_1^{-1}\mathtt{F}T_2^{-1}$.

Degeneracies Some specific motions may induce rank deficient system matrices in (A.175). Widely known deficiencies are points lying on a plane or a quadric surface, and these degenerate conditions are thoroughly investigated by Torr et al. [1995].

In practice, the 8-point algorithm may be used in parallel with the homography estimation (4-point) algorithm as a fail safe mechanism. Wu [2013]'s VisualSfM also uses this approach.

7-point algorithm

Since a fundamental matrix is rank-2 by definition, this actually adds one extra constraint that can decrease the number of minimum correspondences required to 7.

Suppose the same situation as in Section A.4.4 except that there are now only 7 correspondences, yielding the following linear system

$$\begin{bmatrix} \tilde{\mathbf{m}}_{2,1}^\top \otimes \tilde{\mathbf{m}}_{1,1}^\top \\ \vdots \\ \tilde{\mathbf{m}}_{2,7}^\top \otimes \tilde{\mathbf{m}}_{1,7}^\top \end{bmatrix} \operatorname{vec}\mathtt{F} = \mathbf{0}, \tag{A.178}$$

Similar to (A.175), the system is homogeneous, having the form $\mathtt{A}\mathbf{x} = 0$ where $\mathtt{A} \in \mathbb{R}^{7\times 9}$. This means $\operatorname{null}(\mathtt{A}) \in \mathbb{R}^{9\times 2}$, meaning that any linear combination of the two nullspace vectors yield a valid solution to (A.178). The 3×3 matrix form of these nullspace vectors will be called $\mathtt{F}_1$ and $\mathtt{F}_2$ respectively for convenience.

Now, imposing the rank constraint yields

$$\det(\mathtt{F}_1 + \lambda \mathtt{F}_2) = a\lambda^3 + b\lambda^2 + c\lambda + d = 0, \tag{A.179}$$

which is a third-order polynomial in λ that can be solved analytically. This is also known as a cubic singularity condition [Li and Hartley, 2006]. Solving (A.179) yields either 1 or 3 real roots. For the latter case, more correspondences are required to determine the geometrically meaningful solution. Also, if $\lambda = 0$, then the rank of the resulting fundamental matrix is 1, which again requires more correspondences to better constrain the problem.

As for the 8-point algorithm, a normalization step may be incorporated to yield numerically more stable results. Also, the trick for enforcing the essential matrix constraint can also be applied.

5-point algorithm

Unlike the above two algorithms which estimate the fundamental matrix, the 5-point algorithm [Nister, 2004] directly estimates the essential matrix reviewed in Section A.4.1, which has 2 less degrees of freedom. This requires a prior knowledge of the camera intrinsic parameters $\mathtt{K}_1$ and $\mathtt{K}_2$.

As well as abiding by $\det(\mathtt{E}) = 0$, Faugeras and Maybank [1990] showed that the following cubic constraint must be satisfied by $\mathtt{E}$ to be a valid essential matrix.

$$2\mathtt{E}\mathtt{E}^\top\mathtt{E} - \mathrm{trace}(\mathtt{E}\mathtt{E}^\top)\mathtt{E} = 0. \tag{A.180}$$

(A.180) has 9 equations but only 2 are linearly independent.

Nister [2004] proposed to solve through these constraints in the following manner. First, each of the 5 correspondences introduces an epipolar constraint (A.145), which can be rewritten as

$$(\tilde{\mathbf{u}}_{1,j}^\top \otimes \tilde{\mathbf{u}}_{2,j}^\top)\,\mathrm{vec}(\mathtt{E}) = 0. \tag{A.181}$$

Combining these for all 5 correspondences yields a homogeneous 5×9 linear system

$$\begin{bmatrix} \tilde{\mathbf{u}}_{1,1}^\top \otimes \tilde{\mathbf{u}}_{2,1}^\top \\ \vdots \\ \tilde{\mathbf{u}}_{1,5}^\top \otimes \tilde{\mathbf{u}}_{2,5}^\top \end{bmatrix} \mathrm{vec}(\mathtt{E}) = \mathbf{0}. \tag{A.182}$$

Similar to the 8-point algorithm, solving (A.182) using the direct linear transformation A.2.2 generates 4 solution bases each of which forms a nullspace of the 5×9 system matrix. A general solution can be represented as

$$\mathtt{E} = \alpha\mathtt{E}_1 + \beta\mathtt{E}_2 + \gamma\mathtt{E}_3 + \mathtt{E}_4, \tag{A.183}$$

where $\mathtt{E}_i$ corresponds to the i-th solution basis. $\mathtt{E}_4$ does not have a specified scale since the overall scale of the solution is irrelevant.

Now, substituting (A.183) to $\det(\mathtt{E}) = 0$ and (A.180) forms 10 cubic polynomials in α, β and γ. These can be parameterized as

$$\mathtt{B}\mathbf{b} = \mathbf{0}, \tag{A.184}$$

where $\mathtt{B} \in \mathbb{R}^{10\times 20}$ is a 10×20 coefficient matrix and $\mathbf{b} \in \mathbb{R}^{20}$ is a monomial vector comprising the intermediate polynomial power terms α^3, β^3, γ^3, $\alpha\beta^2$, etc. Nistér's trick here is to eliminate α and β from the equation and form a univariate polynomial in terms of γ. In brief, this is achieved as follows:

1. Perform a Gauss-Jordan elimination [Press et al., 2007] of $\mathtt{B}$ to make it in the form $[\mathtt{I},\ \mathtt{C}]$, where $\mathtt{C}$ is a dense matrix corresponding to intermediate power terms that are linear in α, β and 1 (e.g. $\alpha z^2, \beta z, z+2$).

2. Choose 6 equations from the Gauss-Jordan eliminated system $[\mathtt{I},\ \mathtt{C}]$ and combined them in a specific way to yield 3 equations of the form

$$\mathtt{D}\begin{bmatrix}\alpha\\ \beta\\ 1\end{bmatrix} = \mathbf{0}, \tag{A.185}$$

   where $\mathtt{D}$ is a matrix of polynomials in γ. (This does not discard any useful information since only 2 of 10 original equations are linearly independent.)

3. Since $\mathtt{D}$ is rank-deficient, use $\det(\mathtt{D}) = 0$ to compose a 10th order polynomial in γ, and solve it by computing the eigenvalues of the companion matrix [Horn and Johnson, 2012]. This may yield up to 10 real roots.

4. For each of the 10 roots, back substitute z into (A.185) to recover corresponding α and β.

More thorough details are available in Nister [2004]'s original paper. As above can yield up to 10 solutions, the one which has the highest "credibility" over all correspondences is chosen. For each solution, the credibility can be measured either in terms of the number of inlier correspondences given the model or in terms of the sum of epipolar distance.

There are other variants of the 5-point algorithm proposed by Stewénius et al. [2005] and Li and Hartley [2006].

Homography estimation

If two cameras are viewing a plane or a camera undergoes a pure rotation, one may use the homography estimation algorithm, which requires only 4 correspondences.

From Section A.4.2, the homography constraint (A.151) can be modified to

$$\begin{aligned}\tilde{\mathbf{m}}_{2,j} \times \mathtt{H}\,\tilde{\mathbf{m}}_{1,j} &= [\tilde{\mathbf{m}}_{2,j}]_\times\,\mathtt{H}\,\tilde{\mathbf{m}}_{1,j} = \mathbf{0} && \text{(A.186)}\\ &= \mathrm{vec}([\tilde{\mathbf{m}}_{2,j}]_\times\,\mathtt{H}\,\tilde{\mathbf{m}}_{1,j}) = (\tilde{\mathbf{m}}_{1,j}^\top \otimes [\tilde{\mathbf{m}}_{2,j}]_\times)\,\mathrm{vec}(\mathtt{H}), && \text{(A.187)}\end{aligned}$$

which yields up to 2 linearly independent constraints as the rank of $[\tilde{\mathbf{m}}2, j]_\times$ is 2. Hence, only the first two rows of $[\tilde{\mathbf{m}}_{2,j}]_\times$ may be used, which is denoted as $[\tilde{\mathbf{m}}_{2,j}]_{\times,1:2}$. (A.187) can be thought as constraining the discrepancy between camera 1's transferred ray through $\mathbf{x}_j$ and camera 2's observed ray through $\mathbf{x}_j$.

Now, stacking (A.187) for each correspondence yields a 8 × 9 rank-deficient linear system equation

$$\begin{bmatrix}\tilde{\mathbf{m}}_{1,1}^\top \otimes [\tilde{\mathbf{m}}_{2,1}]_{\times,1:2}\\ \vdots \\ \tilde{\mathbf{m}}_{1,4}^\top \otimes [\tilde{\mathbf{m}}_{2,4}]_{\times,1:2}\end{bmatrix}\mathrm{vec}(\mathtt{H}) = \mathbf{0}. \qquad \text{(A.188)}$$

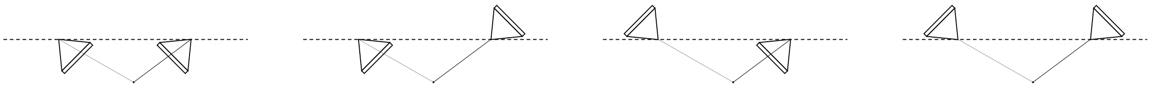
Again, this can be solved by using the direct linear transformation (DLT) algorithm from Section A.2.2, which effectively solves (A.188) with an additional constraint $\|\mathtt{H}\|_F^2 = 1$. The homogeneous scale for each correspondence must be the same. (e.g. set to 1.)

Similar to the 8-point and 7-point algorithms, normalizing each cluster of projections improves the quality of homography matrices obtained.

A.4.5 Robust estimation of geometric model

The minimal solver algorithms described above only uses a handful of points, which may contain outlying matches that can critically deteriorate the quality of the estimated geometric quantity. The 8-point algorithm and the homography estimation algorithm described above could be generalized to include more matches, but still this would not help if a large proportion of matches are wrong.

To circumvent this issue, a type of random sample consensus (RANSAC) algorithm reviewed in Section A.2.7 may be incorporated on top of the aforementioned minimal solvers. For example, if there are n matches, sample the minimum of correspondences (e.g. 5 or 8) required to fit a two-view model, estimate model parameters and then check how many of the total correspondences are considered as inliers given the model. This step is repeated until the



(a) Both cameras in front (b) Right camera behind (c) Left camera behind (d) Both cameras behind

Fig. A.8 Possible configurations arising from an essential matrix or homography. For the perspective camera model, all observed inlier points must lie in front of the participating cameras' image planes. This is known as the *cheirality* constraint.

probability of failing to sample a set of inliers goes below some threshold. The model with the largest number of inliers is used.

In practice, employing RANSAC substantially improves the quality of the estimated geometry since repetitive sampling helps to avoid bad local minima. However, the model may still yield an incorrect estimate of geometry if there are repetitive structures or symmetries in the scene.

Refinement

The solution outputted from RANSAC may be further refined by minimizing the Sampson error (in Section A.4.3), which can be viewed as a first order approximation of the reprojection error but does not require explicit estimation of the cameras and 3D points. This is typically carried out by parameterizing the essential matrix $[\mathbf{t}]_\times \mathtt{R}(\theta)$ with 6 or 7 parameters depending on the rotation parameterization used.

A.4.6 Relative camera poses

Once a geometric quantity is obtained using one of the aforementioned algorithms, it is possible to retrieve relative rotation and up-to-scale translation between two views. It will be assumed that camera intrinsics are known.

Enforcing the essential matrix constraint

If a 7 or 8-point algorithm was used, the outputted fundamental matrix $\mathtt{F}$ only satisfies the constraint $\det(\mathtt{F}) = 0$. An essential matrix $\mathtt{E}$ has an additional constraint that the two non-zero singular values of $\mathtt{F}$ are equal. To enforce this, a near-Essential matrix $\tilde{\mathtt{E}}$ is obtained by computing

$$\tilde{\mathtt{E}} = \mathtt{K}_2^\top \, \mathtt{F} \, \mathtt{K}_1, \tag{A.189}$$

which is a simple reformulation of (A.147). Then, $\tilde{\mathtt{E}}$ is decomposed using the SVD (from Section A.1.5) to yield $\tilde{\mathtt{E}} = \tilde{\mathtt{U}}\tilde{\Sigma}\tilde{\mathtt{V}}^\top$ with $\tilde{\mathtt{U}} \in O(3)$ and $\tilde{\mathtt{V}} \in O(3)$. Since the essential matrix can be decomposed as $\mathtt{U}\,\mathrm{diag}([\sigma,\sigma,0])\mathtt{V}^\top$ (where $\mathtt{U} \in O(3)$ and $\mathtt{V} \in O(3)$ and $\sigma \in \mathbb{R}$ is a scale factor) finding the nearest essential matrix in the least squares sense amounts to solving

$$\underset{\mathtt{U}\in O(3),\sigma\in\mathbb{R},\mathtt{V}\in O(3)}{\arg\min} \quad \|\tilde{\mathtt{U}}\tilde{\Sigma}\tilde{\mathtt{V}}^\top - \sigma\mathtt{U}\,\mathrm{diag}([1,1,0])\mathtt{V}^\top\|_F^2, \tag{A.190}$$

and using a similar reformulation technique to (A.46) yields

$$\underset{\mathtt{U}\in O(3),\sigma\in\mathbb{R},\mathtt{V}\in O(3)}{\arg\max} \quad \sigma\ \mathrm{trace}(\tilde{\Sigma}\tilde{\mathtt{V}}^\top\mathtt{V}\,\mathrm{diag}([1,1,0])\mathtt{U}^\top\tilde{\mathtt{U}}) - \sigma^2. \tag{A.191}$$

Since any product of two orthogonal matrices is also an orthogonal matrix, two more definitions $\hat{\mathtt{U}} := \tilde{\mathtt{U}}^\top\mathtt{U}$ and $\hat{\mathtt{V}} := \tilde{\mathtt{V}}^\top\mathtt{V}$ are defined. Then, (A.191) can be rewritten as

$$\underset{\hat{\mathtt{U}}\in O(3),\sigma\in\mathbb{R},\hat{\mathtt{V}}\in O(3)}{\arg\max} \quad \sigma\ \mathrm{trace}(\tilde{\Sigma}(\hat{\mathbf{v}}_1\hat{\mathbf{u}}_1^\top + \hat{\mathbf{v}}_2\hat{\mathbf{u}}_2^\top) - \sigma^2, \tag{A.192}$$

$$\underset{\hat{\mathtt{U}}\in O(3),\sigma\in\mathbb{R},\hat{\mathtt{V}}\in O(3)}{\arg\max} \quad \sigma\ (\tilde{\sigma}_{1,1}\hat{u}_{1,1}\hat{v}_{1,1} + \tilde{\sigma}_{2,2}\hat{u}_{2,2}\hat{v}_{2,2}) - \sigma^2, \tag{A.193}$$

where $\hat{\mathbf{u}}_j$ and $\hat{\mathbf{v}}_j$ represent the j-th columns of $\hat{\mathtt{U}}$ and $\hat{\mathtt{V}}$ respectively and $\hat{u}_{i,j}$ and $\hat{v}_{i,j}$ represent the (i,j)-th element of $\hat{\mathtt{U}}$ and $\hat{\mathtt{V}}$ respectively. It is now clear that $\hat{u}_{1,1}$, $\hat{v}_{1,1}$, $\hat{u}_{2,2}$ and $\hat{v}_{2,2}$ must all be 1 to maximize the trace term. This leads to $\hat{\mathtt{U}} = \mathtt{I}$ and $\hat{\mathtt{V}} = \mathtt{I}$, and therefore $\mathtt{U} = \tilde{\mathtt{U}}$ and $\mathtt{V} = \tilde{\mathtt{V}}$.

The problem then just simply breaks down to computing

$$\sigma = \underset{\sigma\in\mathbb{R}}{\arg\max}\, \sigma(\tilde{\sigma}_{1,1} + \tilde{\sigma}_{2,2}) - \sigma^2 = \frac{1}{2}(\tilde{\sigma}_{1,1} + \tilde{\sigma}_{2,2}). \tag{A.194}$$

In summary, taking the SVD of $\mathtt{K}_2^\top\mathtt{F}\mathtt{K}_1$ to yield $\mathtt{U}\Sigma\mathtt{V}^\top$, then computing $\mathtt{U}\,\mathrm{diag}([110])\mathtt{V}^\top$ yields the closest essential matrix with two unitary singular values.

Four possible poses

The essential matrix can be parameterized as $\mathtt{E} = [\mathbf{t}]_\times\mathtt{R} =: \mathtt{S}\mathtt{R}$, where $\mathtt{S} \in \mathbb{R}^{3\times 3}$ is a skew-symmetric matrix. Golub and Van Loan [1996] proves that any 3 × 3 skew symmetric matrix can be parameterized as

$$\mathtt{S} \simeq \mathtt{U} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathtt{U}^\top =: \mathtt{UZU}^\top = \mathtt{U}\mathrm{diag}([1,1,0]) \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathtt{U}^\top =: \mathtt{U}\mathrm{diag}([1,1,0])\mathtt{WU}^\top. \tag{A.195}$$

Note that $\mathtt{W}^\top\mathtt{W} = \mathrm{diag}([1,1,0])$ and $\mathtt{W}^2 = \mathrm{diag}([-1,-1,0])$. Since the SVD of the $\mathtt{E}$ matrix yields $\mathtt{U}\mathrm{diag}([1,1,0])\mathtt{V}^\top$, comparing this with the up-to-scale equation (A.195) implies that $\mathtt{R}$ can be $\mathtt{UWV}^\top$ or $\mathtt{UW}^\top\mathtt{V}^\top$. There is a 180° rotation between the two poses.

Also, above tells us that the left nullspace of $\mathtt{E}$ is $\mathbf{u}_3$, which is the 3rd column vector of $\mathtt{U}$. As this equals the left nullspace of $[\mathbf{t}]_\times$, which is $\mathbf{t}$, $\mathbf{t} \simeq \mathbf{u}_3$. Since there is a scale ambiguity in the scene, the possible translation directions are $\mathbf{u}_3$ and $-\mathbf{u}_3$.

Fig. A.8 illustrates the four potential relative poses between two cameras. However, only (a) is geometrically meaningful since 3D points must lie in front of the observing cameras. Selecting a geometrically valid view is illustrated in Section A.4.8.

Pose from homography If a homography matrix is available instead of an essential matrix, then the relative pose decomposition depends on the nature of the motion. If the motion is panoramic (i.e. no translation), then (A.152) implies $\mathtt{R} = \mathtt{K}_2^{-1}\mathtt{HK}_1$. In reality, $\mathtt{H}$ will be noisy, and therefore one needs to find the closest rotation matrix to $\mathtt{K}_2^{-1}\mathtt{HK}_1$ using Arun et al.'s method described in Section A.1.9.

On the other hand, if the homography arises from a planar scene, then there are similar closed-form expressions for finding 2 possible rotations and 2 translations. Details can be found in Schönberger and Frahm [2016]'s COLMAP.

A.4.7 Triangulation

Triangulation is the process of estimating the 3D positions of feature points given camera parameters. One thing to note is that triangulation can be processed separately for each point. Two widely used methods are shown below, and although the illustrations are based on two views, it can easily be extended to more views.

Minimizing an algebraic error One way is to minimize the discrepancy between the observed rays and the estimated rays. i.e. for each point $\mathbf{x}_j \in \mathbb{R}^3$, solve

$$\min_{\tilde{\mathbf{x}}_j \in \mathbb{R}^4} \left\| \begin{matrix} \tilde{\mathbf{m}}_{1,j} \times \mathtt{P}_1 \tilde{\mathbf{x}}_j \\ \tilde{\mathbf{m}}_{1,j} \times \mathtt{P}_2 \tilde{\mathbf{x}}_j \end{matrix} \right\|_2^2 \quad \text{s.t.} \quad \|\tilde{\mathbf{x}}_j\|_2^2 = 1. \tag{A.196}$$

The r.h.s. constraint comes from the fact that $\tilde{\mathbf{x}}_j$ is in homogeneous coordinates. As (A.196) is a linear homogeneous equation, one can solve using the direct linear transformation from Section A.2.2.

Minimizing the reprojection error One can find 3D points that minimize the reprojection error (A.157). Since the projection function $\mathbf{p}i : \mathbb{R}^3 \to \mathbb{R}^2$ is nonlinear, this requires a nonlinear optimizer such as Levenberg-Marquardt from Section A.2.3. Since nonlinear optimizers require an initial estimate as input, a solution obtained from minimizing the aforementioned algebraic error may be inputted for refinement.

A.4.8 Cheirality constraint

Fig. A.8 implies that there is only 1 out of 4 configurations that is geometrically meaningful. To find the correct one, each configuration is used to triangulate all inlier correspondences, and the one with the maximum number of 3D points in front of two cameras is selected as a valid model.

At this stage, the triangulation process does not have to yield accurate results since it is only used to determine the sign of depths. Therefore, minimizing an algebraic error using the DLT from Section A.2.2 would suffice. Correspondences with negative depths may optionally be discarded.

A.4.9 Mini bundle adjustment

Once rough estimates of the relative pose and 3D points are obtained, they can be refined simultaneously via bundle adjustment, which minimizes the reprojection error described in Section A.4.3. The first camera matrix can be set constant to $[\mathtt{I}, \mathbf{0}]$ since the world coordinates can be set arbitrarily.

Appendix B

Derivations for matrix factorization

B.1 Derivatives for joint optimization

B.1.1 Jacobian $\mathtt{J}$

The Jacobian matrix for joint optimization is defined as $\mathtt{J} := \partial \varepsilon / \partial \mathbf{x}$ where $\mathbf{x} = [\mathbf{u}; \mathbf{v}]$. Applying the identity from Section 2.1.2 yields

$$\mathtt{J} := \begin{bmatrix} \dfrac{\partial \varepsilon_1(\mathbf{u},\mathbf{v})}{\partial \mathbf{u}} & \dfrac{\partial \varepsilon_1(\mathbf{u},\mathbf{v})}{\partial \mathbf{v}} \\ \dfrac{\partial \varepsilon_2(\mathbf{u})}{\partial \mathbf{u}} & \\ & \dfrac{\partial \varepsilon_3(\mathbf{v})}{\partial \mathbf{v}} \end{bmatrix} = \begin{bmatrix} \hat{\mathtt{V}} & \tilde{\mathtt{U}} \\ \{\sqrt{\mu}\mathtt{I}\} & \\ & \{\sqrt{\mu}\mathtt{I}\} \end{bmatrix}. \tag{B.1}$$

B.1.2 Gradient $\mathbf{g}$

The gradient is defined as $\mathbf{g} := [\partial f / \partial \mathbf{x}]^\top$ where again $\mathbf{x} = [\mathbf{u}; \mathbf{v}]$. Since $f(\mathbf{u},\mathbf{v}) := \|\varepsilon(\mathbf{u},\mathbf{v})\|_2^2$, $\mathbf{g}$ can be obtained by computing $2\mathtt{J}^\top \varepsilon$, leading to

$$\frac{1}{2}\mathbf{g} = \begin{bmatrix} \hat{\mathtt{V}}^\top & \sqrt{\mu}\mathtt{I} & \\ \tilde{\mathtt{U}}^\top & & \sqrt{\mu}\mathtt{I} \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \sqrt{\mu}\mathbf{u} \\ \sqrt{\mu}\mathbf{v} \end{bmatrix} = \begin{bmatrix} \hat{\mathtt{V}}^\top \varepsilon_1 + \{\mu \mathbf{u}\} \\ \tilde{\mathtt{U}}^\top \varepsilon_1 + \{\mu \mathbf{v}\} \end{bmatrix}. \tag{B.2}$$

B.1.3 Hessian $\mathtt{H}$

Hessian is defined as $\mathtt{H} := \partial\mathbf{g}/\partial\mathbf{x}$ with $\mathbf{x} = [\mathbf{u};\mathbf{v}]$. This gives

$$\frac{1}{2}\mathtt{H} = \frac{1}{2}\begin{bmatrix}\frac{\partial\mathbf{g}}{\partial\mathbf{u}} & \frac{\partial\mathbf{g}}{\partial\mathbf{v}}\end{bmatrix} = \begin{bmatrix} \hat{\mathtt{V}}^\top\frac{\partial\varepsilon_1}{\partial\mathbf{u}} + \mu\mathtt{I} & \hat{\mathtt{V}}^\top\frac{\partial\varepsilon_1}{\partial\mathbf{v}} + \frac{(\partial\hat{\mathtt{V}})^\top\varepsilon_1}{\partial\mathbf{v}} \\ \tilde{\mathtt{U}}^\top\frac{\partial\varepsilon_1}{\partial\mathbf{u}} + \frac{(\partial\tilde{\mathtt{U}})^\top\varepsilon_1}{\partial\mathbf{u}} & \tilde{\mathtt{U}}^\top\frac{\partial\varepsilon_1}{\partial\mathbf{v}} + \mu\mathtt{I}\end{bmatrix}. \tag{B.3}$$

Applying tricks from the work of Minka [2000] and Magnus and Neudecker [2007] give

$$\begin{aligned}
\partial[\tilde{\mathtt{U}}]^\top\varepsilon_1 &= (\mathtt{I}\otimes\partial\mathtt{U}^\top)\tilde{\mathtt{W}}^\top\varepsilon_1 && \text{(B.4)}\\
&= (\mathtt{I}\otimes\partial\mathtt{U}^\top)\operatorname{diag}(\operatorname{vec}(\mathtt{W}))\operatorname{vec}(\mathtt{R}) && // \ \mathtt{R}(\mathtt{U},\mathtt{V}) := \mathtt{U}\mathtt{V}^\top - \mathtt{M} \quad \text{(B.5)}\\
&= (\mathtt{I}\otimes\partial\mathtt{U}^\top)\operatorname{vec}(\mathtt{W}\odot\mathtt{R}) && \text{(B.6)}\\
&= \operatorname{vec}(\partial\mathtt{U}^\top(\mathtt{W}\odot\mathtt{R})) && // \ \operatorname{vec}(\mathtt{AXB}) = (\mathtt{B}^\top\otimes\mathtt{A})\operatorname{vec}(\mathtt{X}) \quad \text{(B.7)}\\
&= ((\mathtt{W}\odot\mathtt{R})^\top\otimes\mathtt{I}_r)\operatorname{vec}(\partial\mathtt{U}^\top) && \text{(B.8)}\\
&= \mathtt{Z}^\top\mathtt{K}_{mr}\partial\mathbf{u}, && // \ \mathtt{Z} := (\mathtt{W}\odot\mathtt{R})\otimes\mathtt{I} \quad \text{(B.9)}
\end{aligned}$$

and similarly

$$\begin{aligned}
\partial[\hat{\mathtt{V}}]^\top\varepsilon_1 &= (\partial\mathtt{V}^\top\otimes\mathtt{I})\tilde{\mathtt{W}}^\top\varepsilon_1 = (\partial\mathtt{V}^\top\otimes\mathtt{I})\operatorname{vec}(\mathtt{W}\odot\mathtt{R}) && \text{(B.10)}\\
&= \operatorname{vec}((\mathtt{W}\odot\mathtt{R})\partial\mathtt{V}) && // \ \operatorname{vec}(\mathtt{AXB}) = (\mathtt{B}^\top\otimes\mathtt{A})\operatorname{vec}(\mathtt{X}) \quad \text{(B.11)}\\
&= (\mathtt{I}_r\otimes(\mathtt{W}\odot\mathtt{R}))\operatorname{vec}(\partial\mathtt{V}) && \text{(B.12)}\\
&= \left(\mathtt{K}_{mr}^\top((\mathtt{W}\odot\mathtt{R})\otimes\mathtt{I}_r)\mathtt{K}_{nr}\right)\mathtt{K}_{nr}^\top\operatorname{vec}(\partial\mathtt{V}^\top) && \text{(B.13)}\\
&= \mathtt{K}_{mr}^\top\mathtt{Z}\partial\mathbf{v}. && // \ \mathtt{Z} := (\mathtt{W}\odot\mathtt{R})\otimes\mathtt{I} \quad \text{(B.14)}
\end{aligned}$$

These two terms arise only when computing exact Hessian since they do not appear in $\mathtt{J}^\top\mathtt{J}$. Combining results from above and noting the damping factor $\langle\lambda\mathtt{I}\rangle$ yield

$$\frac{1}{2}\mathtt{H} = \begin{bmatrix}\hat{\mathtt{V}}^\top\hat{\mathtt{V}} + \{\mu\mathtt{I}\} + \langle\lambda\mathtt{I}\rangle & \hat{\mathtt{V}}^\top\tilde{\mathtt{U}} + [\mathtt{K}_{mr}^\top\mathtt{Z}]_{FN} \\ \tilde{\mathtt{U}}^\top\hat{\mathtt{V}} + [\mathtt{Z}^\top\mathtt{K}_{mr}]_{FN} & \tilde{\mathtt{U}}^\top\tilde{\mathtt{U}} + \{\mu\mathtt{I}\} + \langle\lambda\mathtt{I}\rangle\end{bmatrix} \tag{B.15}$$

where $\mathtt{K}_{mr}$ and $\mathtt{Z}$ are defined in Section 3.1. This is equivalent to Hessian for Damped Newton [Buchanan and Fitzgibbon, 2005].

B.2 Derivatives for variable projection

As shown in Section 3.1.4, the analytic expression for $\mathbf{v}^*(\mathbf{u}) := \arg\min_{\mathbf{v}} f(\mathbf{u},\mathbf{v})$ is

$$\mathbf{v}^*(\mathbf{u}) = \tilde{\mathtt{U}}^{-\mu}\tilde{\mathbf{m}}. \tag{B.16}$$

B.2.1 The derivative of $\mathbf{v}^*(\mathbf{u})$

Substituting (B.16) to the original objective yields

$$\varepsilon_1^*(\mathbf{u},\mathbf{v}^*(\mathbf{u})) = \tilde{\mathtt{U}}\mathbf{v}^* - \tilde{\mathbf{m}} = -(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{-\mu})\tilde{\mathbf{m}}. \tag{B.17}$$

Applying the μ-pseudo inverse rule in Section A.1.3 produces

$$\begin{aligned}
\partial[\mathbf{v}^*] &= -\tilde{\mathtt{U}}^{-\mu}\partial[\tilde{\mathtt{U}}]\tilde{\mathtt{U}}^{-\mu}\tilde{\mathbf{m}} \\
&\qquad + (\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}} + \mu\mathtt{I})^{-1}\partial[\tilde{\mathtt{U}}]^\top(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{-\mu})\tilde{\mathbf{m}} && \text{(B.18)} \\
&= -\tilde{\mathtt{U}}^{-\mu}\partial[\tilde{\mathtt{U}}]\mathbf{v}^* - (\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}} + \mu\mathtt{I})^{-1}\partial[\tilde{\mathtt{U}}]^\top\varepsilon_1^* && \text{(B.19)} \\
&&& \textit{// noting (B.16) and (B.17)} \\
&= -\tilde{\mathtt{U}}^{-\mu}\hat{\mathtt{V}}^*\partial\mathbf{u} - (\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}} + \mu\mathtt{I})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}\partial\mathbf{u}, && \text{(B.20)} \\
&&& \textit{// using bilinearity and noting (B.9)}
\end{aligned}$$

and hence

$$\frac{d\mathbf{v}^*}{d\mathbf{u}} = -(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}} + \mu\mathtt{I})^{-1}(\tilde{\mathtt{U}}^\top\hat{\mathtt{V}}^* + \mathtt{Z}^{*\top}\mathtt{K}_{mr}). \tag{B.21}$$

B.2.2 Jacobian $\mathtt{J}_{\mathbf{u}}$

Jacobian for variable projection is defined as

$$\mathtt{J}_{\mathbf{u}} := \frac{d\varepsilon^*(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{d\mathbf{u}} = \begin{bmatrix} \dfrac{d\varepsilon_1^*(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{d\mathbf{u}} \\ \dfrac{d\varepsilon_2^*(\mathbf{u})}{d\mathbf{u}} \\ \dfrac{d\varepsilon_3^*(\mathbf{v}^*(\mathbf{u}))}{d\mathbf{u}} \end{bmatrix}. \tag{B.22}$$

Noting that

$$\frac{d\varepsilon_1^*(\mathbf{u},\mathbf{v}^*(\mathbf{u}))}{d\mathbf{u}} = \left(\frac{\partial\varepsilon_1^*}{\partial\mathbf{u}}\right) + \left(\frac{\partial\varepsilon_1^*}{\partial\mathbf{v}^*}\right)\frac{d\mathbf{v}^*}{d\mathbf{u}}, \tag{B.23}$$

this yields

$$\mathtt{J}_{\mathbf{u}} = \begin{bmatrix} \hat{\mathtt{V}}^* + \tilde{\mathtt{U}}\frac{d\mathbf{v}^*}{d\mathbf{u}} \\ \{\sqrt{\mu}\mathtt{I}\} \\ \left\{\sqrt{\mu}\frac{d\mathbf{v}^*}{d\mathbf{u}}\right\} \end{bmatrix}. \tag{B.24}$$

B.2.3 Gradient $\mathbf{g}^*$

The gradient for variable projection, $\mathbf{g}^*$, can be obtained by simplifying $2\mathtt{J}^{*\top}\boldsymbol{\varepsilon}^*$:

$$\frac{1}{2}\mathbf{g}^* = \left(\hat{\mathtt{V}}^* + \tilde{\mathtt{U}}\frac{d\mathbf{v}^*}{d\mathbf{u}}\right)^\top \varepsilon_1^* + \left\{\mu\mathbf{u} + \mu\left(\frac{d\mathbf{v}^*}{d\mathbf{u}}\right)^\top \mathbf{v}^*\right\} \quad // \ \varepsilon_2^* = \mathbf{u}, \quad \varepsilon_3^* = \mathbf{v}^*(\mathbf{u}) \tag{B.25}$$

B.2.4 The Gauss-Newton matrix $\mathtt{H}_{GN}^*$

Unlike joint optimization, Hessian for variable projection with regularization $\mathtt{H}^* := d\mathbf{g}^*/d\mathbf{u}$ is complex due to the need to compute the expression

$$\frac{d\left[\frac{d\mathbf{v}^*}{d\mathbf{u}}\right]^\top}{d\mathbf{u}}\mathbf{v}^*. \tag{B.26}$$

Boumal and Absil [2011] were able to bypass this issue by taking the directional derivative of the gradient (rather than just the derivative).

Instead, the damped Gauss-Newton matrix, $\mathtt{H}_{GN}^*$, can be obtained from $\mathtt{J}^{*\top}\mathtt{J}^*$:

$$\frac{1}{2}\mathtt{H}_{GN}^* = \left(\hat{\mathtt{V}}^* + \tilde{\mathtt{U}}\frac{d\mathbf{v}^*}{d\mathbf{u}}\right)^\top \left(\hat{\mathtt{V}}^* + \tilde{\mathtt{U}}\frac{d\mathbf{v}^*}{d\mathbf{u}}\right) + \left\{\mu\mathtt{I} + \mu\left(\frac{d\mathbf{v}^*}{d\mathbf{u}}\right)^\top\left(\frac{d\mathbf{v}^*}{d\mathbf{u}}\right)\right\} + \langle\lambda\mathtt{I}\rangle \tag{B.27}$$

A summary of this section can be found in Table 3.2. It will be shown in Section B.3.5 that obtaining analytic Hessian for $\mu = 0$ is less tricky.

B.3 Derivatives for un-regularized VarPro

This section shows various analytic derivatives for $\mu = 0$, and describes their relations to Ruhe and Wedin [1980]'s algorithms. Setting $\mu = 0$ simplifies $\mathbf{v}^*(\mathbf{u})$ from (B.16) to

$$\mathbf{v}^*(\mathbf{u}) = \tilde{\mathtt{U}}^\dagger \tilde{\mathbf{m}}. \tag{B.28}$$

B.3.1 The derivative of $\mathbf{v}^*(\mathbf{u})$

Removing μ-related terms from (B.21) gives

$$\begin{aligned} \frac{d\mathbf{v}^*}{d\mathbf{u}} &= -(\tilde{\mathtt{U}}^\top \tilde{\mathtt{U}})^{-1}(\tilde{\mathtt{U}}^\top \hat{\mathtt{V}}^* + \mathtt{Z}^{*\top}\mathtt{K}_{mr}) \qquad &(B.29)\\ &= -\tilde{\mathtt{U}}^\dagger \hat{\mathtt{V}}^* - (\tilde{\mathtt{U}}^\top \tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}. &(B.30) \end{aligned}$$

B.3.2 Jacobian $\mathtt{J}_{\mathbf{u}1}$

Removing μ-related terms from (B.24) and noting (B.30) yields

$$\begin{aligned} \mathtt{J}_{\mathbf{u}1} &= \hat{\mathtt{V}}^* + \tilde{\mathtt{U}}\frac{d\mathbf{v}^*}{d\mathbf{u}} \qquad &(B.31)\\ &= (\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger)\hat{\mathtt{V}}^* - \tilde{\mathtt{U}}^{\dagger\top}\mathtt{Z}^{*\top}\mathtt{K}_{mr} &(B.32) \end{aligned}$$

B.3.3 Gradient $\mathbf{g}_1^*$

Deleting μ-related terms and noting (B.25), (B.32) and (B.28) produces:

$$\begin{aligned} \frac{1}{2}\mathbf{g}_1^* &= \hat{\mathtt{V}}^{*\top}(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger)\boldsymbol{\varepsilon}_1^* + \mathtt{K}_{mr}^\top \mathtt{Z}^* \tilde{\mathtt{U}}^\dagger \boldsymbol{\varepsilon}_1^* \qquad &(B.33)\\ &= \hat{\mathtt{V}}^{*\top}\boldsymbol{\varepsilon}_1^* \qquad \textit{// } \tilde{\mathtt{U}}^\dagger \boldsymbol{\varepsilon}_1^* = 0 \ \textit{ since } \ \boldsymbol{\varepsilon}_1^* = \tilde{\mathtt{U}}\mathbf{v}^* - \tilde{\mathbf{m}}. &(B.34) \end{aligned}$$

Above can be rearranged in matrix form to give $\nabla_{\mathtt{U}} f_1^*(\mathtt{U})$:

$$\begin{aligned} \frac{1}{2}\nabla_{\mathtt{U}} f_1^*(\mathtt{U}) &= \mathrm{unvec}(\hat{\mathtt{V}}^{*\top}\boldsymbol{\varepsilon}_1^*) \qquad &(B.35)\\ &= (\mathtt{W} \odot \mathtt{R}^*)\mathtt{V}^* \qquad \textit{// similar to (B.12)} &(B.36) \end{aligned}$$

B.3.4 The Gauss-Newton matrix $\mathtt{H}^*_{GN1}$

The Gauss-Newton matrix can be obtained by computing $\mathtt{J}_1^{*\top}\mathtt{J}_1^*$:

$$\begin{aligned}
\frac{1}{2}\mathtt{H}^*_{GN1} &= \hat{\mathtt{V}}^{*\top}(\mathtt{I}-\tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\hat{\mathtt{V}}^* + \mathtt{K}_{mr}^{\top}\mathtt{Z}^*\tilde{\mathtt{U}}^{\dagger}\tilde{\mathtt{U}}^{\dagger\top}\mathtt{Z}^{*\top}\mathtt{K}_{mr} \\
&\quad + \hat{\mathtt{V}}^{*\top}(\mathtt{I}-\tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\tilde{\mathtt{U}}^{\dagger\top}\mathtt{Z}^{*\top}\mathtt{K}_{mr} \\
&\qquad + \mathtt{K}_{mr}^{\top}\mathtt{Z}^*\tilde{\mathtt{U}}^{\dagger}(\mathtt{I}-\tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\hat{\mathtt{V}}^* \qquad /\!/ \textit{ noting } \mathtt{I}-\tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger} = (\mathtt{I}-\tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})^{\top} = (\mathtt{I}-\tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})^2 \\
&= \hat{\mathtt{V}}^{*\top}(\mathtt{I}-\tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\hat{\mathtt{V}}^* + \mathtt{K}_{mr}^{\top}\mathtt{Z}^*(\tilde{\mathtt{U}}^{\top}\tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr} \;/\!/ \textit{ noting } \tilde{\mathtt{U}}^{\dagger}\tilde{\mathtt{U}}^{\dagger\top} = (\tilde{\mathtt{U}}^{\top}\tilde{\mathtt{U}})^{-1} \textit{ and } \tilde{\mathtt{U}}^{\dagger}\tilde{\mathtt{U}} = \mathtt{I}
\end{aligned} \tag{B.37}$$

B.3.5 The Hessian matrix $\mathtt{H}_1^*$

Taking the partial derivative of $\mathbf{g}_1^*$ yields

$$\begin{aligned}
\frac{1}{2}\partial[\mathbf{g}_1^*] &= \partial[\hat{\mathtt{V}}^*]^{\top}\varepsilon_1^* + \hat{\mathtt{V}}^{*\top}\partial[\varepsilon_1^*] \\
&= \mathtt{K}_{mr}^{\top}\mathtt{Z}^*\partial\mathbf{v}^* + \hat{\mathtt{V}}^{*\top}\tilde{\mathtt{U}}\partial\mathbf{v}^* + \hat{\mathtt{V}}^{*\top}\hat{\mathtt{V}}^*\partial\mathbf{u}, \qquad /\!/ \textit{ noting bilinearity and Equation (B.14)}
\end{aligned} \tag{B.38}$$

and hence

$$\frac{1}{2}\mathtt{H}_1^* = \hat{\mathtt{V}}^{*\top}\hat{\mathtt{V}}^* + \hat{\mathtt{V}}^{*\top}\tilde{\mathtt{U}}\frac{d\mathbf{v}^*}{d\mathbf{u}} + \mathtt{K}_{mr}^{\top}\mathtt{Z}^*\frac{d\mathbf{v}^*}{d\mathbf{u}} \tag{B.39}$$

$$\begin{aligned}
&= \hat{\mathtt{V}}^{*\top}\hat{\mathtt{V}}^* - \hat{\mathtt{V}}^{*\top}\tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger}\hat{\mathtt{V}}^* - \hat{\mathtt{V}}^{*\top}\tilde{\mathtt{U}}^{\dagger\top}\mathtt{Z}^{*\top}\mathtt{K}_{mr} - \mathtt{K}_{mr}^{\top}\mathtt{Z}^*\tilde{\mathtt{U}}^{\dagger}\hat{\mathtt{V}}^* \\
&\quad - \mathtt{K}_{mr}^{\top}\mathtt{Z}^*(\tilde{\mathtt{U}}^{\top}\tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}.
\end{aligned} \tag{B.40}$$

Comparing with (B.37) and dampening yields:

$$\begin{aligned}
\frac{1}{2}\mathtt{H}_1^* &= \hat{\mathtt{V}}^{*\top}(\mathtt{I}-\tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\hat{\mathtt{V}}^* + [-1]_{FN} \times \mathtt{K}_{mr}^{\top}\mathtt{Z}^*(\tilde{\mathtt{U}}^{\top}\tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr} \\
&\qquad - [\hat{\mathtt{V}}^{*\top}\tilde{\mathtt{U}}^{\dagger\top}\mathtt{Z}^{*\top}\mathtt{K}_{mr} + \mathtt{K}_{mr}^{\top}\mathtt{Z}^*\tilde{\mathtt{U}}^{\dagger}\hat{\mathtt{V}}^*]_{FN} + \langle\lambda\mathtt{I}\rangle.
\end{aligned} \tag{B.41}$$

B.3.6 Column-wise derivatives

Some algorithms are based on what is referred here as column-wise derivatives [Chen, 2008; Gotardo and Martinez, 2011]. The unregularized objective $f_1^*(\mathbf{u}, \mathbf{v}^*(\mathbf{u}))$ may be viewed as the the sum of squared norm of individual column vectors in $\mathtt{R}^*(\mathbf{u}, \mathbf{v}^*(\mathbf{u}))$. In other words,

$$\begin{aligned} f_1^*(\mathbf{u}, \mathbf{v}^*(\mathbf{u})) := \|\mathtt{R}^*\|_F^2 &= \sum_{j=1}^{n} \|\mathbf{r}_j^*\|_2^2 && \text{(B.42)} \\ &= \sum_{j=1}^{n} \|\mathbf{w}_j \odot (\mathtt{U}\mathbf{v}_j^* - \mathbf{m}_j)\|_2^2 \\ &= \sum_{j=1}^{n} \|\operatorname{diag}(\mathbf{w}_j)(\mathtt{U}\mathbf{v}_j^* - \mathbf{m}_j)\|_2^2 \\ &= \sum_{j=1}^{n} \|\tilde{\mathtt{W}}_j(\mathtt{U}\mathbf{v}_j^* - \mathbf{m}_j)\|_2^2 && \text{(B.43)} \end{aligned}$$

where $\mathbf{w}_j$ is the j-th column of $\mathtt{W}$, $\mathbf{v}_j^*$ is the j-th row of $\mathtt{V}^*(\mathtt{U})$ transposed and $\mathbf{m}_j$ is the j-th column of $\mathtt{M}$. For each j, the non-zero rows of $\tilde{\mathtt{W}}_j$ form $\tilde{\mathtt{W}}_j \in \mathbb{R}^{p_j \times m}$ where p_j is the number of visible elements in the j-th column. Using (B.42) and defining $\tilde{\mathtt{U}}_j := \tilde{\mathtt{W}}_j\mathtt{U}$ and $\tilde{\mathbf{m}}_j := \tilde{\mathtt{W}}_j\mathbf{m}_j$ yields

$$\varepsilon_{1j}^* = \tilde{\mathtt{U}}_j\mathbf{v}_j^* - \tilde{\mathbf{m}}_j \tag{B.44}$$

such that $f_{1j}^*(\mathbf{u}, \mathbf{v}^*(\mathbf{u})) = \|\varepsilon_{1j}^*(\mathbf{u}, \mathbf{v}^*(\mathbf{u}))\|_2^2$. From this, one can observe that each row of $\mathtt{V}^*$ is independent of other rows and thus $\mathbf{v}_j^*$ can be obtained as

$$\mathbf{v}_j^* = \arg\min_{\mathbf{v}_j} \|\tilde{\mathtt{U}}_j\mathbf{v}_j^* - \tilde{\mathbf{m}}_j\|_2^2 = \tilde{\mathtt{U}}_j^\dagger\tilde{\mathbf{m}}_j, \tag{B.45}$$

leading to

$$\mathbf{r}_j^* = -(\mathtt{I} - \tilde{\mathtt{U}}_j\tilde{\mathtt{U}}_j^\dagger)\tilde{\mathbf{m}}_j. \tag{B.46}$$

The derivative of $\mathbf{v}_j^*(\mathbf{u})$

Before computing the derivative of $\mathbf{v}_j^*$, define

$$\hat{\mathtt{V}}_j^* := \tilde{\mathtt{W}}_j({\mathbf{v}_j^*}^\top \otimes \mathtt{I}_m) \in \mathbb{R}^{p_j \times mr} \tag{B.47}$$

$$\mathtt{Z}_j^* := (\mathbf{w}_j \odot \mathbf{r}_j^*) \otimes \mathtt{I}_r \in \mathbb{R}^{mr \times r}. \tag{B.48}$$

Computing the derivative similar to (B.19) yields

$$\begin{aligned}\partial[\mathbf{v}_j^*] &= -\tilde{\mathtt{U}}_j^\dagger \partial[\tilde{\mathtt{U}}_j]\mathbf{v}_j^* - (\tilde{\mathtt{U}}_j^\top \tilde{\mathtt{U}}_j)^{-1}\partial[\tilde{\mathtt{U}}_j]^\top \boldsymbol{\varepsilon}_{1j}^* \\ &= -\tilde{\mathtt{U}}_j^\dagger \hat{\mathtt{V}}_j^* \partial\mathbf{u} - (\tilde{\mathtt{U}}_j^\top \tilde{\mathtt{U}}_j)^{-1}{\mathtt{Z}_j^*}^\top \mathtt{K}_{mr}\partial\mathbf{u}, \end{aligned} \tag{B.49}$$

and hence

$$\begin{aligned}\frac{d\mathbf{v}_j^*}{d\mathbf{u}} &= -\tilde{\mathtt{U}}_j^\dagger \hat{\mathtt{V}}_j^* - (\tilde{\mathtt{U}}_j^\top \tilde{\mathtt{U}}_j)^{-1}{\mathtt{Z}_j^*}^\top \mathtt{K}_{mr} \\ &= -\tilde{\mathtt{U}}_j^\dagger \hat{\mathtt{V}}_j^* - (\tilde{\mathtt{U}}_j^\top \tilde{\mathtt{U}}_j)^{-1}(\mathtt{I} \otimes (\mathbf{w}_j \odot \mathbf{r}_j^*)^\top) \qquad \textit{// using (A.13)} \\ &= -\tilde{\mathtt{U}}_j^\dagger \hat{\mathtt{V}}_j^* - (\tilde{\mathtt{U}}_j^\top \tilde{\mathtt{U}}_j)^{-1} \otimes (\mathbf{w}_j \odot \mathbf{r}_j^*)^\top \end{aligned} \tag{B.50}$$

Let us now define $\mathbf{u}_j := \mathrm{vec}(\tilde{\mathtt{W}}_j\mathtt{U}) = (\mathtt{I} \otimes \tilde{\mathtt{W}}_j)\mathbf{u} \in \mathbb{R}^{p_j r}$. Taking the derivative with respect to $\mathbf{u}_j$ yields

$$(\mathtt{I} \otimes \tilde{\mathtt{W}}_j)\frac{d\mathbf{u}}{d\mathbf{u}_j} = \mathtt{I}. \tag{B.51}$$

Since $\tilde{\mathtt{W}}_j\tilde{\mathtt{W}}_j^\top = \mathtt{I}$,

$$\frac{d\mathbf{u}}{d\mathbf{u}_j} = \mathtt{I} \otimes \tilde{\mathtt{W}}_j^\top. \tag{B.52}$$

This gives

$$\frac{d\mathbf{v}_j^*}{d\mathbf{u}_j} = \frac{d\mathbf{v}_j^*}{d\mathbf{u}}\frac{d\mathbf{u}}{d\mathbf{u}_j} = \frac{d\mathbf{v}_j^*}{d\mathbf{u}}(\mathtt{I} \otimes \tilde{\mathtt{W}}_j^\top) \tag{B.53}$$

The column-wise Jacobian matrix $\mathtt{J}_{\mathbf{u}1j}$

As far as the notation is concerned, the only difference between (B.30) and (B.50) is the existence of subscript j. Since Jacobian is also independent for each column, one can write

$$\begin{aligned}
\mathtt{J}_{\mathbf{u}1j} &= \hat{\mathtt{V}}_j^* + \tilde{\mathtt{U}}_j \frac{d\mathbf{v}_j^*}{d\mathbf{u}} \\
&= (\mathtt{I} - \tilde{\mathtt{U}}_j \tilde{\mathtt{U}}_j^\dagger)\hat{\mathtt{V}}_j^* - \tilde{\mathtt{U}}_j^{\dagger\top}\mathtt{Z}_j^{*\top}\mathtt{K}_{mr} \\
&= (\mathtt{I} - \tilde{\mathtt{U}}_j \tilde{\mathtt{U}}_j^\dagger)\hat{\mathtt{V}}_j^* - \tilde{\mathtt{U}}_j^\top \left((\tilde{\mathtt{U}}_j^\top \tilde{\mathtt{U}}_j)^{-1} \otimes (\mathbf{w}_j \odot \mathbf{r}_j^*)^\top \right). \qquad \text{(B.54)}
\end{aligned}$$

Also,

$$\frac{d\varepsilon_{1j}^*}{d\mathbf{u}_j} = \mathtt{J}_{\mathbf{u}1j}\frac{d\mathbf{u}}{d\mathbf{u}_j} = \mathtt{J}_{\mathbf{u}1j}(\mathtt{I} \otimes \tilde{\mathtt{W}}_j^\top) \qquad \text{(B.55)}$$

The column-wise gradient vector $\mathbf{g}_1^*$

Noting $\mathbf{g}_1^* := df_1^*/d\mathbf{u}$ yields

$$\begin{aligned}
\frac{1}{2}\mathbf{g}_1^* &= \frac{1}{2}\frac{d}{d\mathbf{u}}\sum_{j=1}^n \|\varepsilon_{1j}^*\|_2^2 = \sum_{j=1}^n \left(\frac{d\varepsilon_{1j}^*}{d\mathbf{u}}\right)^\top \varepsilon_{1j}^* \\
&= \sum_{j=1}^n \mathtt{J}_{1j}^{*\top}\varepsilon_{1j}^* = \sum_{j=1}^n \hat{\mathtt{V}}_j^{*\top}\varepsilon_{1j}^* \qquad \textit{// using (B.46)} \qquad \text{(B.56)}
\end{aligned}$$

Another way of looking by Chen [2008] is

$$\begin{aligned}
\frac{1}{2}\mathbf{g}_1^* &= \sum_{j=1}^n \left(\frac{d\varepsilon_{1j}^*}{d\mathbf{u}_j}\frac{d\mathbf{u}_j}{d\mathbf{u}}\right)^\top \varepsilon_{1j}^* = \sum_{j=1}^n \left(\frac{d\mathbf{u}_j}{d\mathbf{u}}\right)^\top (\mathtt{I} \otimes \tilde{\mathtt{W}}_j)\mathtt{J}_{1j}^{*\top}\varepsilon_{1j}^* \qquad \textit{// noting (B.55)} \\
&= \sum_{j=1}^n (\mathtt{I} \otimes \tilde{\mathtt{W}}_j)^\top (\mathtt{I} \otimes \tilde{\mathtt{W}}_j)\mathtt{J}_{1j}^{*\top}\varepsilon_{1j}^* \qquad \textit{// noting } \mathbf{u}_j = (\mathtt{I} \otimes \tilde{\mathtt{W}}_j)\mathbf{u} \\
&:= \frac{1}{2}\sum_{j=1}^n (\mathtt{I} \otimes \tilde{\mathtt{W}}_j^\top)\mathbf{g}_{1j}^* \qquad \text{(B.57)}
\end{aligned}$$

where

$$\frac{1}{2}\mathbf{g}_{1j}^* := \left(\frac{d\varepsilon_{1j}^*}{d\mathbf{u}_j}\right)^\top \varepsilon_{1j}^*. \qquad \text{(B.58)}$$

The column-wise Hessian matrix $\mathtt{H}_1^*$

From the definition $\mathtt{H}_1^* := d\mathbf{g}_1^*/d\mathbf{u}$, the summation in $\mathbf{g}_1^*$ stays in $\mathtt{H}_1^*$. Noting the shape of $\mathtt{J}_{1j}^{*\top}\mathtt{J}_{1j}^{*\top}$, (B.41) can be converted to

$$\frac{1}{2}\mathtt{H}_1^* = \sum_{j=1}^{n}\Bigg[\hat{\mathtt{V}}_j^{*\top}(\mathtt{I}-\tilde{\mathtt{U}}_j\tilde{\mathtt{U}}_j^{\dagger})\hat{\mathtt{V}}_j^* + [-1]_{FN} \times \mathtt{K}_{mr}^\top\mathtt{Z}_j^*(\tilde{\mathtt{U}}_j^\top\tilde{\mathtt{U}}_j)^{-1}\mathtt{Z}_j^{*\top}\mathtt{K}_{mr} \\ - [\hat{\mathtt{V}}_j^{*\top}\tilde{\mathtt{U}}_j^{\dagger\top}\mathtt{Z}_j^{*\top}\mathtt{K}_{mr} + \mathtt{K}_{mr}^\top\mathtt{Z}_j^*\tilde{\mathtt{U}}_j^{\dagger}\hat{\mathtt{V}}_j^*]_{FN}\Bigg] + \langle\lambda\mathtt{I}\rangle. \tag{B.59}$$

All the $\mathtt{Z}_j^{*\top}\mathtt{K}_{mr}$ terms can be replaced by $\mathtt{I}\otimes(\mathbf{w}_j\odot\mathbf{r}_j^*)$ using (A.13). Then, Chen's notation yields

$$\frac{1}{2}\mathtt{H}_1^* := \frac{1}{2}\frac{d\mathbf{g}_1^*}{d\mathbf{u}} + \langle\lambda\mathtt{I}\rangle \tag{B.60}$$

$$= \frac{1}{2}\sum_{j=1}^{n}\left[(\mathtt{I}\otimes\tilde{\mathtt{W}}_j^\top)\frac{d\mathbf{g}_{1j}^*}{d\mathbf{u}_j}\frac{d\mathbf{u}_j}{d\mathbf{u}}\right] + \langle\lambda\mathtt{I}\rangle$$

$$= \frac{1}{2}\sum_{j=1}^{n}\left[(\mathtt{I}\otimes\tilde{\mathtt{W}}_j^\top)\mathtt{H}_{1j}^*(\mathtt{I}\otimes\tilde{\mathtt{W}}_j)\right] + \langle\lambda\mathtt{I}\rangle, \tag{B.61}$$

where

$$\mathtt{H}_{1j}^* := \frac{d\mathbf{g}_{1j}^*}{d\mathbf{u}_j} = (\mathtt{I}\otimes\tilde{\mathtt{W}}_j)\frac{d(\hat{\mathtt{V}}_j^{*\top}\boldsymbol{\varepsilon}_{1j}^*)}{d\mathbf{u}_j} \qquad \textit{// from (B.57)} \tag{B.62}$$

$$= (\mathtt{I}\otimes\tilde{\mathtt{W}}_j)\frac{d\mathbf{g}_1^*}{d\mathbf{u}}\frac{d\mathbf{u}}{d\mathbf{u}_j} = (\mathtt{I}\otimes\tilde{\mathtt{W}}_j)\frac{d\mathbf{g}_1^*}{d\mathbf{u}}(\mathtt{I}\otimes\tilde{\mathtt{W}}_j^\top). \tag{B.63}$$

Hence $\mathtt{H}_{1j}^*$ is the condensed form of j-th layer of the Hessian matrix.

B.4 Manifold optimization on unreguralized VarPro

For un-regularized VarPro, the objective has the property $f_1^*(\mathtt{U}) = f_1^*(\mathtt{UA})$ for any invertible $\mathtt{A}$. Hence, $\mathtt{U}$ lies on a Grassmann manifold embedded in Euclidean space which is a type of Riemannian manifold with its own metrics.

By reviewing Section A.2.5, the tangent space-projection matrix is $\mathtt{I}-\mathtt{UU}^\top$. As vectorized quantities are used here, the projection matrix must also be converted to support them. Since the following relationship holds

$$\begin{aligned}\mathbf{g}_p^* &= \operatorname{vec}\left((\mathtt{I}-\mathtt{UU}^\top)\frac{df_1^*(\mathbf{u}))}{d\mathtt{U}}\right)\\ &= (\mathtt{I}\otimes(\mathtt{I}-\mathtt{UU}^\top))\operatorname{vec}\left(\frac{df_1^*(\mathbf{u})}{d\mathtt{U}}\right)\\ &= (\mathtt{I}\otimes(\mathtt{I}-\mathtt{UU}^\top))\mathbf{g}_1^*, \qquad \text{(B.64)}\end{aligned}$$

one can deduce that $\mathtt{P}_p = \mathtt{I}\otimes(\mathtt{I}-\mathtt{UU}^\top) = \mathtt{I}\otimes\mathtt{U}_\perp\mathtt{U}_\perp^\top$.

B.4.1 Projected gradient

The projected gradient $\mathbf{g}_p^*$ can be obtained by multiplying the tangent space-projection matrix $\mathtt{P}_p$ to the original gradient $\mathbf{g}_1^*$. Doing so initially gives us

$$\begin{aligned}\mathbf{g}_p^* &:= (\mathtt{I}\otimes(\mathtt{I}-\mathtt{UU}^\top))\mathbf{g}_1^* && \text{(B.65)}\\ &= \mathbf{g}_1^* - 2(\mathtt{I}\otimes\mathtt{UU}^\top)\hat{\mathtt{V}}^{*\top}\boldsymbol{\varepsilon}_1^*\\ &= \mathbf{g}_1^* - 2(\mathtt{I}\otimes\mathtt{UU}^\top)\operatorname{vec}((\mathtt{W}\odot\mathtt{R}^*)\mathtt{V}^*) && \textit{// noting (B.36)}\\ &= \mathbf{g}_1^* - 2\operatorname{vec}(\mathtt{UU}^\top(\mathtt{W}\odot\mathtt{W}\odot(\mathtt{UV}^{*\top}-\mathtt{M}))\mathtt{V}^*) && \text{// } (\mathtt{B}^\top\otimes\mathtt{A})\operatorname{vec}(\mathtt{X}) = \operatorname{vec}(\mathtt{AXB})\\ &= \mathbf{g}_1^* - 2(\mathtt{V}^{*\top}\otimes\mathtt{U})\operatorname{vec}(\mathtt{U}^\top(\mathtt{W}\odot\mathtt{R}^*)) && \text{(B.66)}\end{aligned}$$

Vectorizing $\mathtt{U}^\top(\mathtt{W}\odot\mathtt{R}^*)$ yields

$$\begin{aligned}\operatorname{vec}(\mathtt{U}^\top(\mathtt{W}\odot\mathtt{R}^*)) &= (\mathtt{I}\otimes\mathtt{U}^\top)\operatorname{vec}(\mathtt{W}\odot\mathtt{R}^*) && \text{(B.67)}\\ &= (\mathtt{I}\otimes\mathtt{U}^\top)\operatorname{diag}(\operatorname{vec}(\mathtt{W}))\operatorname{vec}(\mathtt{R}^*)\\ &= (\mathtt{I}\otimes\mathtt{U}^\top)\tilde{\mathtt{W}}^\top\boldsymbol{\varepsilon}_1^*\\ &= \tilde{\mathtt{U}}^\top(\tilde{\mathtt{U}}\mathbf{v}^* - \tilde{\mathbf{m}}) = \tilde{\mathtt{U}}^\top(\tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger\tilde{\mathbf{m}} - \tilde{\mathbf{m}}) = 0 && \text{(B.68)}\end{aligned}$$

giving

$$\mathbf{g}_p^* = \mathbf{g}_1^*. \qquad \text{(B.69)}$$

This means that the gradient is already on the tangent space of $\mathtt{U}$ and thus no additional computation is required.

B.4.2 Projected Hessian

Projected Hessian $\mathtt{H}_p^*$ can be obtained by multiplying the tangent space-projection matrix $\mathtt{P}_p$ to the derivative of the projected gradient $\mathbf{g}_p^*$. Since $\mathbf{g}_p^* = \mathbf{g}_1^*$, the original undamped Hessian should be projected to the tangent space of $\mathtt{U}$:

$$\mathtt{H}_p^* := \mathtt{P}_p \frac{d\mathbf{g}_1^*}{d\mathbf{u}} = (\mathtt{I} - \mathtt{I} \otimes \mathtt{U}\mathtt{U}^\top) \frac{d\mathbf{g}_1^*}{d\mathbf{u}} \tag{B.70}$$

There are 4 terms present in $\mathtt{H}_1^*$ as shown in (B.41). Noting that

$$\begin{aligned}(\mathtt{I} \otimes \mathtt{U}^\top)(\hat{\mathtt{V}}^{*\top}(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger)\hat{\mathtt{V}}^*) &= (\mathtt{V}^{*\top} \otimes \mathtt{U}^\top)\tilde{\mathtt{W}}^\top(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger)\hat{\mathtt{V}}^* \qquad \textit{// noting } \hat{\mathtt{V}}^* = \tilde{\mathtt{W}}(\mathtt{V}^* \otimes \mathtt{I})\\ &= (\mathtt{V}^{*\top} \otimes \mathtt{I})(\mathtt{I} \otimes \mathtt{U}^\top)\tilde{\mathtt{W}}^\top(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger)\hat{\mathtt{V}}^*\\ &= (\mathtt{V}^{*\top} \otimes \mathtt{I})\tilde{\mathtt{U}}^\top(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger)\hat{\mathtt{V}}^* = 0,\end{aligned} \tag{B.71}$$

one can see that the term $\hat{\mathtt{V}}^{*\top}(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger)\hat{\mathtt{V}}^*$ remains the same after projection. Since

$$\begin{aligned}(\mathtt{I} \otimes \mathtt{U}^\top)(\mathtt{K}_{mr}^\top\mathtt{Z}^*) &= \mathtt{K}_{mr}^\top(\mathtt{U}^\top \otimes \mathtt{I})\mathtt{K}_{mr}\mathtt{K}_{mr}^\top((\mathtt{W} \odot \mathtt{R}^*) \otimes \mathtt{I})\\ &= \mathtt{K}_{mr}^\top(\mathtt{U}^\top(\mathtt{W} \odot \mathtt{R}^*) \otimes \mathtt{I}) = 0, \qquad \textit{// noting Equations (B.67) and (B.68)}\end{aligned} \tag{B.72}$$

this yields the projected Hessian matrix

$$\begin{aligned}\frac{1}{2}\mathrm{Hess} f^*(\mathbf{u}) = \hat{\mathtt{V}}^{*\top}(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger)\hat{\mathtt{V}}^* + [-1]_{FN} \times \mathtt{K}_{mr}^\top\mathtt{Z}^*(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}\\ - \mathtt{P}_p[\hat{\mathtt{V}}^{*\top}\tilde{\mathtt{U}}^{\dagger\top}\mathtt{Z}^{*\top}\mathtt{K}_{mr}]_{FN} - [\mathtt{K}_{mr}^\top\mathtt{Z}^*\tilde{\mathtt{U}}^\dagger\hat{\mathtt{V}}^*]_{FN}.\end{aligned} \tag{B.73}$$

Once again, this shows that projected Gauss-newton $\mathtt{H}_{pGN}^*$ is analytically equivalent to original Gauss-Newton $\mathtt{H}_{1GN}^*$ as the first two terms are the analytically the same. However, $\mathrm{Hess} f^*(\mathbf{u})$ is asymmetric. One way to tackle this is to consider the manifold constraint of the subproblem.

B.4.3 The projected subproblem

The undamped Newton solution $\Delta\mathbf{u}$ to the subproblem mentioned in (C.35) satisfies

$$\mathrm{Hess} f^*(\mathbf{u})\Delta\mathbf{u} = -\mathbf{g}_p^* \tag{B.74}$$

where $\Delta\mathbf{u}$ lies on the tangent space of $\mathtt{U}$. This means that

$$\mathtt{U}^\top\Delta\mathtt{U} = (\mathtt{I}\otimes\mathtt{U}^\top)\Delta\mathbf{u} = 0 \tag{B.75}$$

which leads to $(\mathtt{I}\otimes(\mathtt{I}-\mathtt{U}\mathtt{U}^\top))\Delta\mathbf{u} = \mathtt{P}_p\Delta\mathbf{u} = \Delta\mathbf{u}$. Equation (B.74) now becomes

$$\text{Hess} f^*(\mathbf{u})\mathtt{P}_p\Delta\mathbf{u} = -\mathbf{g}_p^* \tag{B.76}$$

Noting Equations (B.71) and (B.72), the matrix product to the left of $\Delta\mathbf{u}$ yields

$$\begin{aligned}\text{Hess} f^*(\mathbf{u})\mathtt{P}_p = {} & \hat{\mathtt{V}}^{*\top}(\mathtt{I}-\tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger)\hat{\mathtt{V}}^* + [-1]_{FN}\times\mathtt{K}_{mr}^\top\mathtt{Z}^*(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr} \\ & -\mathtt{P}_p[\hat{\mathtt{V}}^{*\top}\tilde{\mathtt{U}}^{\dagger\top}\mathtt{Z}^{*\top}\mathtt{K}_{mr}]_{FN} - [\mathtt{K}_{mr}^\top\mathtt{Z}^*\tilde{\mathtt{U}}^\dagger\hat{\mathtt{V}}^*]_{FN}\mathtt{P}_p.\end{aligned} \tag{B.77}$$

Note that the expression for Gauss-Newton is still preserved. Now the symmetric projected Hessian with damping $\mathtt{H}_p^*$ is defined as

$$\mathtt{H}_p^* := \text{Hess} f^*(\mathbf{u})\mathtt{P}_p + \langle\lambda\mathtt{I}\rangle = \mathtt{P}_p\frac{d\mathbf{g}_p^*}{d\mathbf{u}}\mathtt{P}_p + \langle\lambda\mathtt{I}\rangle. \tag{B.78}$$

B.4.4 Constraining the subproblem

The subproblem mentioned in (C.35) is rank-deficient since the update is orthogonal to $\mathtt{U}$ and consequently its dimension is r^2 less than that of $\mathbf{u}\in\mathbb{R}^{mr}$. There are two approaches to tackling this issue.

Reducing the subproblem dimension

The first approach, proposed by Manton et al. [2003], is to reduce the entire subproblem dimension by projecting it to the orthogonal space of $\mathtt{U}$ [Chen, 2008]. Noting that $\mathtt{I}-\mathtt{U}\mathtt{U}^\top = \mathtt{U}_\perp\mathtt{U}_\perp^\top$, The reduced projection matrix is defined as

$$\mathtt{P}_r := \mathtt{I}\otimes\mathtt{U}_\perp^\top \in \mathbb{R}^{(m-r)r\times mr} \tag{B.79}$$

which satisfies $\mathtt{P}_r\mathtt{P}_p = \mathtt{P}_r$. Defining the reduced update $\Delta\mathbf{u}_\perp \in \mathbb{R}^{mr-r^2}$ yields $\mathtt{P}_r{}^\top\Delta\mathbf{u}_\perp = \Delta\mathbf{u}_p$. Hence, projecting the subproblem to the reduced dimension yields

$$\begin{aligned}
&\mathtt{P}_r\mathtt{H}_p^*\Delta\mathbf{u}_p &&= -\mathtt{P}_r\mathbf{g}_p^* && \text{(B.80)}\\
\Rightarrow&\mathtt{P}_r\left(\mathtt{P}_p\frac{d\mathbf{g}_p^*}{d\mathbf{u}}\mathtt{P}_p + \langle 2\lambda\mathtt{I}\rangle\right)\Delta\mathbf{u}_p &&= -\mathtt{P}_r\mathbf{g}_p^* &&\\
\Rightarrow&\mathtt{P}_r\left(\frac{d\mathbf{g}_p^*}{d\mathbf{u}}\mathtt{P}_p + \langle 2\lambda\mathtt{I}\rangle\right)(\mathtt{P}_r{}^\top\Delta\mathbf{u}_\perp) &&= -\mathtt{P}_r\mathbf{g}_p^* &&\\
\Rightarrow&\mathtt{P}_r\left(\frac{d\mathbf{g}_1^*}{d\mathbf{u}} + \langle 2\lambda\mathtt{I}\rangle\right)\mathtt{P}_r{}^\top\Delta\mathbf{u}_\perp &&= -\mathtt{P}_r\mathbf{g}_1^* &&\\
\Rightarrow&\mathtt{P}_r\mathtt{H}_1^*\mathtt{P}_r{}^\top\Delta\mathbf{u}_\perp &&= -\mathtt{P}_r\mathbf{g}_1^*, && \text{(B.81)}\\
\Rightarrow&\left(\mathtt{P}_r\frac{d\mathbf{g}_1}{d\mathbf{u}}\mathtt{P}_r{}^\top + \langle 2\lambda\mathtt{I}\rangle\right)\Delta\mathbf{u}_\perp &&= -\mathtt{P}_r\mathbf{g}_1^*. \quad // \ \mathtt{P}_r\mathtt{P}_r{}^\top = (\mathtt{I}\otimes\mathtt{U}_\perp^\top)(\mathtt{I}\otimes\mathtt{U}_\perp) = \mathtt{I} && \text{(B.82)}
\end{aligned}$$

As the update is now forced to be orthogonal to current $\mathtt{U}$, the reduced problem is better-conditioned than the original at the expense of increased computational complexity. This approach is incorporated to Chen [2008]'s LM_M series of algorithms.

Relaxing the orthogonality constraint

Whilst the first approach enforces the Grassmann manifold constraint $\mathtt{U}^\top\Delta\mathtt{U} = 0$ directly, the second approach relaxes this constraint by adding a term that promotes the orthogonality of the update to the subproblem such that

$$\Delta\mathbf{u}_p = \arg\min_{\delta\perp\mathbf{u}} \mathbf{g}_p^{*\top}\delta + \frac{1}{2}\delta^\top\mathtt{H}_p^*\delta + \langle\alpha\|\mathtt{U}^\top\Delta\|_F^2\rangle_P \tag{B.83}$$

where $\Delta \in \mathbb{R}^{m\times r}$ is the unvectorized form of δ. Differentiating the added term with respect to δ gives

$$\begin{aligned}
\frac{d\|\mathtt{U}^\top\Delta\|_F^2}{d\delta} = \frac{d\|\operatorname{vec}(\mathtt{U}^\top\Delta)\|_2^2}{d\delta} &= \frac{d\|(\mathtt{I}\otimes\mathtt{U}^\top)\delta\|_2}{d\delta} && \text{(B.84)}\\
&= \frac{d(\delta^\top(\mathtt{I}\otimes\mathtt{U}\mathtt{U}^\top)\delta)}{d\delta} = (\mathtt{I}\otimes\mathtt{U}\mathtt{U}^\top)\delta. && \text{(B.85)}
\end{aligned}$$

This is equivalent to updating Hessian as

$$\mathtt{H}_p^* \leftarrow \mathtt{H}_p^* + \langle \alpha \mathtt{I} \otimes \mathtt{U}\mathtt{U}^\top \rangle_P. \tag{B.86}$$

When $\alpha = 1$, the additional term in $\langle$angle-bracketed red$\rangle_P$ is in fact analytically equal to the rank-filling term introduced in Damped Wiberg [Okatani et al., 2011] but the above illustration gives better interpretation and unification in viewpoint of manifold optimization.

B.4.5 Overall remark

The aforementioned analytic results imply that performing un-regularized variable projection using Gauss Newton or its variants automatically incorporates the Grassmann manifold optimization framework. This is simply because both the gradient and the Gauss-Newton matrix (or its approximation discussed in Section B.5.1) do not change when they are projected to the tangent space of $\mathtt{U}$. The algorithms that implements orthorgonalization via orth or q-factor can be interpreted as performing manifold retraction.

Some algorithms such as Damped Wiberg [Okatani et al., 2011] and LM_M_GN [Chen, 2008] have enjoyed greater success by implicitly or explicitly incorporating the Grassmann manifold constraint $\mathtt{U}^\top \Delta \mathtt{U} = 0$.

B.5 Connections to existing algorithms

B.5.1 Ruhe and Wedin's algorithms

Applying Ruhe and Wedin's algorithms to our unregularized problem yields the following results for the set of matrices defined in their original paper [Ruhe and Wedin, 1980]:

$$\mathtt{B} = \frac{\partial [\tilde{\mathtt{U}}]^\top \mathbf{v}^*}{\partial \mathbf{u}} = \hat{\mathtt{V}}^* \qquad \text{// noting bilinearity} \tag{B.87}$$

$$\mathtt{C} = \tilde{\mathtt{U}}^{\dagger\top} \frac{\partial [\tilde{\mathtt{U}}]^\top \varepsilon_1^*}{\partial \mathbf{u}} = \tilde{\mathtt{U}}^{\dagger\top} \mathtt{Z}^{*\top} \mathtt{K}_{mr}. \qquad \text{// analogous to (B.9)} \tag{B.88}$$

$$\mathtt{P}_F := \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger \tag{B.89}$$

$$\mathtt{Q}_F := \mathtt{I} - \mathtt{P}_F = \mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger \tag{B.90}$$

$$\mathbf{g} = \tilde{\mathbf{m}} \tag{B.91}$$

$$\mathtt{Q}_F \mathbf{g} = (\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger)\tilde{\mathbf{m}} \qquad \text{// from (B.17) noting } \mu = 0 \tag{B.92}$$

Transforming $\mathbf{u}$-update of Algorithm 1 (RW1) to our notation gives

$$
\begin{aligned}
\delta\mathbf{u} &= (\mathtt{Q}_F\mathtt{B} + \mathtt{P}_F\mathtt{C})^{\dagger}\mathtt{Q}_F\mathbf{g} && \text{(B.93)}\\
&= -\left((\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\hat{\mathtt{V}}^{*} + \tilde{\mathtt{U}}^{\dagger\top}\mathtt{Z}^{*\top}\mathtt{K}_{mr}\right)^{\dagger}\boldsymbol{\varepsilon}_1^{*} && \text{(B.94)}\\
&= -(\mathtt{J}_{RW1}^{*\top}\mathtt{J}_{RW1}^{*})^{-1}\mathtt{J}_{RW1}^{*\top}\boldsymbol{\varepsilon}_1^{*} && \text{(B.95)}\\
&= -(\mathtt{J}_{RW1}^{*\top}\mathtt{J}_{RW1}^{*})^{-1}\hat{\mathtt{V}}^{*\top}\boldsymbol{\varepsilon}_1^{*}. \qquad \textit{// from (B.34)} && \text{(B.96)}
\end{aligned}
$$

which is in fact Gauss-Newton on VarPro with or without manifold projection.

Algorithm 2 (RW2) approximates $\mathtt{J}_{RW1}^{*}$ by neglecting the $\mathtt{P}_F\mathtt{C}$ term which is ultimately the same as having $d\mathbf{v}^{*}/d\mathbf{u} \approx -\tilde{\mathtt{U}}^{\dagger}\hat{\mathtt{V}}^{*}$. This yields

$$
\begin{aligned}
\delta\mathbf{u} &= (\mathtt{Q}_F\mathtt{B})^{\dagger}\mathtt{Q}_F\mathbf{g} && \text{(B.97)}\\
&= -\left((\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\hat{\mathtt{V}}^{*}\right)^{\dagger}\boldsymbol{\varepsilon}_1^{*} && \text{(B.98)}\\
&= -\left(\hat{\mathtt{V}}^{*\top}(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\hat{\mathtt{V}}^{*}\right)^{-1}\hat{\mathtt{V}}^{*\top}(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\boldsymbol{\varepsilon}_1^{*} \\
&\qquad\qquad \textit{// } (\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger}) = (\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})^{\top} = (\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})^{2} && \text{(B.99)}\\
&= -\left(\hat{\mathtt{V}}^{*\top}(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\hat{\mathtt{V}}^{*}\right)^{-1}\hat{\mathtt{V}}^{*\top}\boldsymbol{\varepsilon}_1^{*} \qquad \textit{// noting } \boldsymbol{\varepsilon}_1^{*} = -(\mathtt{I} - \tilde{\mathtt{U}}\tilde{\mathtt{U}}^{\dagger})\tilde{\mathbf{m}}. && \text{(B.100)}
\end{aligned}
$$

The approximated matrix used here is analytically the same as that of CSF [Gotardo and Martinez, 2011].

Finally, Algorithm 3 (RW3) ignores the variable change and solve for one variable whilst fixing the other. This is done by having $\mathtt{J}_{RW1}^{*\top} \approx \hat{\mathtt{V}}^{*}$ which leads to

$$
\delta\mathbf{u} = -\left(\hat{\mathtt{V}}^{*\top}\hat{\mathtt{V}}^{*}\right)^{-1}\hat{\mathtt{V}}^{*\top}\boldsymbol{\varepsilon}_1^{*}.
$$

The equation for updated $\mathbf{u}$ ($\mathbf{u}^{*}(\mathbf{v}^{*})$) is then

$$
\begin{aligned}
\mathbf{u}^{*} &= \mathbf{u} - \left(\hat{\mathtt{V}}^{*\top}\hat{\mathtt{V}}^{*}\right)^{-1}\hat{\mathtt{V}}^{*\top}\boldsymbol{\varepsilon}_1^{*}. && \text{(B.101)}\\
&= \mathbf{u} - \left(\hat{\mathtt{V}}^{*\top}\hat{\mathtt{V}}^{*}\right)^{-1}\hat{\mathtt{V}}^{*\top}(\hat{\mathtt{V}}^{*}\mathbf{u} - \tilde{\mathbf{m}}) && \text{(B.102)}\\
&= \hat{\mathtt{V}}^{\dagger}\tilde{\mathbf{m}} && \text{(B.103)}
\end{aligned}
$$

which is the same as the alternation (ALS) step for updating $\mathbf{u}$. Since the expression for $\mathbf{v}^{*}(\mathbf{u})$ is already equivalent to that in alternation, this makes Algorithm 3 (RW3) the unregularized block coordinate-descent mentioned in Section 3.1.4.

As a side note, it is worth noting that the overview of algorithms mentioned on page 325 of Ruhe and Wedin [1980]'s paper does not fully match the algorithm descriptions on page 324 of the same paper. The description on page 325 seems to have one additional matrix $\mathtt{H}$ which is believed to be a typo, and therefore the connections made here are based on the algorithm description on page 324.

The summary of the findings is shown in Table 3.3.

B.5.2 Chen's algorithms

Chen [2008]'s algorithms apply column-wise derivatives (see Section B.3.6) with respect to $\mathbf{u}_j$ where $\mathbf{u}_j$ is a set of rows of $\mathtt{U}$ used for the computation of the j-th layer. The derivation contains several redundant Kronecker products which can be removed by applying the matrix identities in Section A.1.2.

The LM_S series apply no constraint, and the LM_M series use the constraint mentioned in Section B.4.4. The inverses are computed using the backslash in MATLAB.

B.5.3 CSF

CSF [Gotardo and Martinez, 2011] also applies column-wise derivatives (see Section B.3.6), but unlike the seris of LM_S and LM_M, CSF uses the derivatives with respect to $\mathbf{u}$ rather than $\mathbf{u}_j$ (see Section B.3.6. No subproblem constraint is used, and the inverses are computed using the backslash in MATLAB.

B.5.4 Damped Wiberg

The analysis of the Damped Wiberg framework is shown in Table 3.3. The analytic output of derivations in the work of Okatani and Deguchi [2007] and Okatani et al. [2011] are essentially the same but are written in three different ways to the derivations in Section B.3:

1. The definitions of $\mathbf{u}$ and $\mathbf{v}$ are swapped.

2. Instead of $\mathbf{u} := \mathrm{vec}(\mathtt{U})$, the authors use $\mathbf{u} := \mathrm{vec}(\mathtt{U}^\top)$ introducing an additional permutation in the $\mathbf{u}$-update equation. In other words,

$$\frac{1}{2}\mathtt{H}^*_{\mathrm{DW}} = \mathtt{K}_{mr}\hat{\mathtt{V}}^{*\top}(\mathtt{I} - [\tilde{\mathtt{U}}\tilde{\mathtt{U}}^\dagger]_{RW2})\hat{\mathtt{V}}^*\mathtt{K}_{mr}^\top + \langle \mathtt{K}_{mr}(\mathtt{I} \otimes \mathtt{U}\mathtt{U}^\top)\mathtt{K}_{mr}^\top \rangle_P + \langle \lambda \mathtt{I} \rangle \tag{B.104}$$

3. The Gauss-Newton matrix is not fully derived but approximated in the same way as in Ruhe and Wedin's Algorithm 2.

Damped Wiberg uses QR decomposition that seems to improve both the convergence rate and the speed as the numerically-stable Q-blocks and R-blocks are used repeatedly.

B.5.5 RTRMC

RTRMC uses regularized variable projection with full Hessian and Grassmann manifold projection. To overcome the issue mentioned in Section B.2.4, the authors incorporate an indirect sub-problem solver which only requires directional derivatives.

This algorithm uses a different regularizer as mentioned in Section 2.3.3. In case when the regularizer is turned off, algorithm's gradient and projected Hessian matches those derived in Section B.4.1 and Section B.4.2. To show this, note that the unregularized projected gradient is

$$\frac{1}{2}\mathbf{g}_p^* := \operatorname{vec}((\mathtt{W}\odot\mathtt{R}^*)\mathtt{V}^*). \tag{B.105}$$

Hence, grad $f_1^*(\mathtt{U}) := \nabla_{\mathtt{U}} f_1^*(\mathtt{U})$ is simply the unvectorized quantity $(\mathtt{W}\odot\mathtt{R}^*)\mathtt{V}^*$.

Noting (3.5), the Hessian operator Hess $f^*(\mathtt{U})[\eta]$ can be obtained by taking the directional derivative of grad$f^*(\mathtt{U})$ in the direction η:

$$\begin{aligned} D\text{grad } f_1^*(\mathtt{U})[\eta] = (&\mathtt{W}\odot\mathtt{W}\odot\eta\mathtt{V}^{*\top})\mathtt{V}^* \\ &+(\mathtt{W}\odot\mathtt{W}\odot\mathtt{U}\, D\mathtt{V}^{*\top}(\mathtt{U})[\eta])\mathtt{V}^* \\ &+(\mathtt{W}\odot\mathtt{R}^*)\, D\mathtt{V}^*(\mathtt{U})[\eta]. \end{aligned} \tag{B.106}$$

Vectorizing the first term produces

$$\begin{aligned} \operatorname{vec}((\mathtt{W}\odot\mathtt{W}\odot\eta\mathtt{V}^{*\top})\mathtt{V}^*) &= (\mathtt{V}^{*\top}\otimes\mathtt{I})\operatorname{vec}(\mathtt{W}\odot\mathtt{W}\odot\eta\mathtt{V}^{*\top}) \\ &= (\mathtt{V}^{*\top}\otimes\mathtt{I})\tilde{\mathtt{W}}^\top\tilde{\mathtt{W}}\operatorname{vec}(\eta\mathtt{V}^{*\top}) \\ &= (\mathtt{V}^{*\top}\otimes\mathtt{I})\tilde{\mathtt{W}}^\top\tilde{\mathtt{W}}(\mathtt{V}^{*\top}\otimes\mathtt{I})\operatorname{vec}(\eta) \\ &= \hat{\mathtt{V}}^{*\top}\hat{\mathtt{V}}^*\operatorname{vec}(\eta), \end{aligned} \tag{B.107}$$

which is the RW3 (ALS) term. Vectorizing the second term generates

$$\begin{aligned} \operatorname{vec}(\mathtt{W}\odot\mathtt{W}\odot\mathtt{U}\, D\mathtt{V}^{*\top}(\mathtt{U})[\eta])\mathtt{V}^* &= (\mathtt{V}^{*\top}\otimes\mathtt{I})\operatorname{vec}(\mathtt{W}\odot\mathtt{W}\odot\mathtt{U}\, D\mathtt{V}^{*\top}(\mathtt{U})[\eta]) \\ &= (\mathtt{V}^{*\top}\otimes\mathtt{I})\tilde{\mathtt{W}}^\top\tilde{\mathtt{W}}\operatorname{vec}(\mathtt{U}\, D\mathtt{V}^{*\top}(\mathtt{U})[\eta]) \\ &= (\mathtt{V}^{*\top}\otimes\mathtt{I})\tilde{\mathtt{W}}^\top\tilde{\mathtt{W}}(\mathtt{I}\otimes\mathtt{U})\operatorname{vec}(\, D\mathtt{V}^{*\top}(\mathtt{U})[\eta]) \\ &= \hat{\mathtt{V}}^{*\top}\tilde{\mathtt{U}}\operatorname{vec}(\, D\mathtt{V}^{*\top}(\mathtt{U})[\eta]), \end{aligned} \tag{B.108}$$

and vectorizing the last term yields

$$\begin{aligned}
\operatorname{vec}((\mathtt{W}\odot\mathtt{R}^*)\, D\mathtt{V}^*(\mathtt{U})[\eta]) &= (\mathtt{I}\otimes(\mathtt{W}\odot\mathtt{R}^*))\operatorname{vec}(D\mathtt{V}^*(\mathtt{U})[\eta]) \\
&= (\mathtt{I}\otimes(\mathtt{W}\odot\mathtt{R}^*))\mathtt{K}_{nr}^\top\operatorname{vec}(D\mathtt{V}^{*\top}(\mathtt{U})[\eta]) \qquad \textit{// using (A.11)} \\
&= \mathtt{K}_{mr}^\top((\mathtt{W}\odot\mathtt{R}^*)\otimes\mathtt{I})\mathtt{K}_{nr}\mathtt{K}_{nr}^\top\operatorname{vec}(D\mathtt{V}^{*\top}(\mathtt{U})[\eta]) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \textit{// using (A.12)} \\
&= \mathtt{K}_{mr}^\top\mathtt{Z}^*\operatorname{vec}(D\mathtt{V}^{*\top}(\mathtt{U})[\eta]) \qquad \text{(B.109)} \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(B.110)}
\end{aligned}$$

$\operatorname{vec}(D\mathtt{V}^{*\top}(\mathtt{U})[\eta])$ can be obtained by reformulating the equation given by Boumal and Absil [2011]. Bearing in mind that here the regularization is set to zero and that the second decomposition matrix $\mathtt{V}$ is transposed in this dissertation, the equation (3.62) in the work of Boumal and Absil [2011] gives us

$$\begin{aligned}
\operatorname{vec}(\, D\mathtt{V}^{*\top}(\mathtt{U})[\eta]) &= -(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\operatorname{vec}(\eta^\top(\mathtt{W}\odot\mathtt{R}^*)+\mathtt{U}^\top(\mathtt{W}\odot\mathtt{W}\odot(\eta\mathtt{V}^{*\top})) \qquad \text{(B.111)} \\
&= -(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}((\mathtt{W}\odot\mathtt{R}^*)^\top\otimes\mathtt{I})\operatorname{vec}(\eta^\top) \\
&\quad -(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}(\mathtt{I}\otimes\mathtt{U}^\top)(\mathtt{W}\odot\mathtt{W}\odot\eta\mathtt{V}^{*\top}) \\
&= -(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}\operatorname{vec}(\eta)-(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\tilde{\mathtt{U}}^\top\tilde{\mathtt{W}}(\mathtt{V}^*\otimes\mathtt{I})\operatorname{vec}(\eta) \\
&= -((\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}+(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\tilde{\mathtt{U}}^\top\hat{\mathtt{V}}^*)\operatorname{vec}(\eta). \qquad \text{(B.112)}
\end{aligned}$$

Hence, (B.108) reduces to

$$-\hat{\mathtt{V}}^{*\top}\tilde{\mathtt{U}}((\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}+(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\tilde{\mathtt{U}}^\top\hat{\mathtt{V}}^*)\operatorname{vec}(\eta), \tag{B.113}$$

and (B.109) to

$$-\mathtt{K}_{mr}^\top\mathtt{Z}^*((\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\mathtt{Z}^{*\top}\mathtt{K}_{mr}+(\tilde{\mathtt{U}}^\top\tilde{\mathtt{U}})^{-1}\tilde{\mathtt{U}}^\top\hat{\mathtt{V}}^*)\operatorname{vec}(\eta). \tag{B.114}$$

It is then easy to check that adding all three terms give $\mathtt{H}^*\operatorname{vec}(\eta)$, which is the full Hessian for unregularized VarPro.

B.6 Additional symmetry in the Gauss-Newton matrix

The additional symmetry is easier to observe when the problem is formulated in terms of column-wise derivatives. Hence, this section begins by presenting a summary of column-wise derivatives from Section B.3.6. Then follows the illustration of search for an additional symmetry in the Gauss-Newton matrix.

B.6.1 Summary of column-wise derivatives

We may view the unregularized objective $f_1^*(\mathbf{u},\mathbf{v}^*(\mathbf{u}))$ as the the sum of squared norm of individual column vectors of $\mathtt{R}^*(\mathbf{u},\mathbf{v}^*(\mathbf{u}))$. In other words,

$$f_1^*(\mathbf{u},\mathbf{v}^*(\mathbf{u})) := \sum_{j=1}^{n} \|\tilde{\mathtt{W}}_j(\mathtt{U}\mathbf{v}_j^* - \mathbf{m}_j)\|_2^2 \tag{B.115}$$

where $\mathbf{w}_j$ is the j-th column of $\mathtt{W}$, $\mathbf{v}_j^*$ is the j-th row of $\mathtt{V}^*(\mathtt{U})$ transposed and $\mathbf{m}_j$ is the j-th column of $\mathtt{M}$. For each j, the nonzero rows of $\tilde{\mathtt{W}}_j$ forms $\tilde{\mathtt{W}}_j \in \mathbb{R}^{p_j \times m}$ where p_j is the number of visible elements in j-th column. Defining $\tilde{\mathtt{U}}_j := \tilde{\mathtt{W}}_j\mathtt{U}$ and $\tilde{\mathbf{m}}_j := \tilde{\mathtt{W}}_j\mathbf{m}_j$ yields

$$\varepsilon_{1j}^* = \tilde{\mathtt{U}}_j\mathbf{v}_j^* - \tilde{\mathbf{m}}_j. \tag{B.116}$$

such that $f_{1j}^*(\mathbf{u},\mathbf{v}^*(\mathbf{u})) = \|\varepsilon_{1j}^*(\mathbf{u},\mathbf{v}^*(\mathbf{u}))\|_2^2$. From this, we can observe that each row of $\mathtt{V}^*$ is independent of other rows and thus $\mathbf{v}_j^*$ can be obtained as

$$\mathbf{v}_j^* = \arg\min_{\mathbf{v}_j} \|\tilde{\mathtt{U}}_j\mathbf{v}_j^* - \tilde{\mathbf{m}}_j\|_2^2 = \tilde{\mathtt{U}}_j^{\dagger}\tilde{\mathbf{m}}_j \tag{B.117}$$

Before moving on to derivations, we define:

$$\hat{\mathtt{V}}_j^* := \tilde{\mathtt{W}}_j(\mathbf{v}_j^{*\top} \otimes \mathtt{I}_m) \in \mathbb{R}^{p_j \times mr} \tag{B.118}$$

$$\mathtt{Z}_j^* := (\mathbf{w}_j \odot \mathbf{r}_j^*) \otimes \mathtt{I}_r \in \mathbb{R}^{mr \times r} \tag{B.119}$$

The column-wise Jacobian matrix $\mathtt{J}_{\mathbf{u}1j}$

Since the Jacobian is also independent for each column, we may write

$$\mathtt{J}_{\mathbf{u}1j} = (\mathtt{I} - \tilde{\mathtt{U}}_j\tilde{\mathtt{U}}_j^{\dagger})\hat{\mathtt{V}}_j^* - \tilde{\mathtt{U}}_j^{\dagger\top}\mathtt{Z}_j^{*\top}\mathtt{K}_{mr} \tag{B.120}$$

$$= (\mathtt{I} - \tilde{\mathtt{U}}_j\tilde{\mathtt{U}}_j^{\dagger})\hat{\mathtt{V}}_j^* - \tilde{\mathtt{U}}_j^{\top}\left((\tilde{\mathtt{U}}_j^{\top}\tilde{\mathtt{U}}_j)^{-1} \otimes (\mathbf{w}_j \odot \mathbf{r}_j^*)^{\top}\right). \tag{B.121}$$

The column-wise Hessian matrix $\mathtt{H}_1^*$

The matrix derivative form of $\mathtt{H}_1^*$ can be converted to the column-wise derivative form:

$$\begin{aligned}\frac{1}{2}\mathtt{H}_1^* = \sum_{j=1}^{n} \Bigg[& \hat{\mathtt{V}}_j^{*\top}(\mathtt{I} - [\tilde{\mathtt{U}}_j\tilde{\mathtt{U}}_j^{\dagger}]_{RW2})\hat{\mathtt{V}}_j^* + [-1]_{FN} \times [\mathtt{K}_{mr}^{\top}\mathtt{Z}_j^*(\tilde{\mathtt{U}}_j^{\top}\tilde{\mathtt{U}}_j)^{-1}\mathtt{Z}_j^{*\top}\mathtt{K}_{mr}]_{RW1} \\ & - [\hat{\mathtt{V}}_j^{*\top}\tilde{\mathtt{U}}_j^{\dagger\top}\mathtt{Z}_j^{*\top}\mathtt{K}_{mr} + \mathtt{K}_{mr}^{\top}\mathtt{Z}_j^*\tilde{\mathtt{U}}_j^{\dagger}\hat{\mathtt{V}}_j^*]_{FN}\Bigg] + \langle\lambda\mathtt{I}\rangle \end{aligned} \tag{B.122}$$

All $\mathtt{Z}_j^{*\top}\mathtt{K}_{mr}$ can be replaced by $\mathtt{I} \otimes (\mathbf{w}_j \odot \mathbf{r}_j^*)^{\top}$ using (A.13).

B.6.2 Finding the symmetry

The above result denotes that the Hessian matrix for the unregularized variable projection formulation can be formulated as sum of n layers where n is the number of columns in the measurement matrix $\mathtt{M}$ (see Section B.3.6). As mentioned in Section B.4.2, the Gauss-Newton matrix is analytically preserved when projected to the tangent space of $\mathtt{U}$. Hence, we only need to compute the original Gauss-Newton matrix which is

$$\frac{1}{2}\mathtt{H}_{GN1}^* = \sum_{j=1}^{n}\left[\hat{\mathtt{V}}_j^{*\top}(\mathtt{I} - [\tilde{\mathtt{U}}_j\tilde{\mathtt{U}}_j^{\dagger}]_{RW2})\hat{\mathtt{V}}_j^* + [\mathtt{K}_{mr}^{\top}\mathtt{Z}_j^*(\tilde{\mathtt{U}}_j^{\top}\tilde{\mathtt{U}}_j)^{-1}\mathtt{Z}_j^{*\top}\mathtt{K}_{mr}]_{RW1}\right] + \langle\lambda\mathtt{I}\rangle.$$

Simplifying the first term yields

$$\begin{aligned}\hat{\mathtt{V}}_j^{*\top}\hat{\mathtt{V}}_j^* &= (\mathbf{v}_j^* \otimes \mathtt{I})\tilde{\mathtt{W}}_j^{\top}\tilde{\mathtt{W}}_j(\mathbf{v}_j^{*\top} \otimes \mathtt{I}) && \text{(B.123)} \\ &= (\mathbf{v}_j^* \otimes \tilde{\mathtt{W}}_j^{\top})(\mathbf{v}_j^{*\top} \otimes \tilde{\mathtt{W}}_j) && \textit{// using (A.10)} \\ &= \mathbf{v}_j^*\mathbf{v}_j^{*\top} \otimes \tilde{\mathtt{W}}_j^{\top}\tilde{\mathtt{W}}_j && \text{(B.124)}\end{aligned}$$

which is the Kronecker product of two symmetric matrices. Such product is not only symmetric but it has another factor of symmetry. In other words, the entire matrix can be recovered by its upper triangular elements which themselves can be recovered by its upper triangular constituents.

Before moving onto the second term, defining $\tilde{\mathtt{U}}_{j\mathtt{Q}} := qf(\tilde{\mathtt{U}}_j)$ yields $\tilde{\mathtt{U}}_j\tilde{\mathtt{U}}_j^{\dagger} = \tilde{\mathtt{U}}_{j\mathtt{Q}}\tilde{\mathtt{U}}_{j\mathtt{Q}}^{\top}$. Now simplifying the second term gives us

$$\begin{aligned}-\hat{\mathtt{V}}_j^{*\top}(\tilde{\mathtt{U}}_j\tilde{\mathtt{U}}_j^{\dagger\top})\hat{\mathtt{V}}_j^* &= -(\mathbf{v}_j^* \otimes \tilde{\mathtt{W}}_j^{\top})(\tilde{\mathtt{U}}_{j\mathtt{Q}}\tilde{\mathtt{U}}_{j\mathtt{Q}}^{\top})(\mathbf{v}_j^{*\top} \otimes \tilde{\mathtt{W}}_j) \\ &= -\mathbf{v}_j^*\mathbf{v}_j^{*\top} \otimes \tilde{\mathtt{W}}_j^{\top}\tilde{\mathtt{U}}_{j\mathtt{Q}}\tilde{\mathtt{U}}_{j\mathtt{Q}}^{\top}\tilde{\mathtt{W}}_j && \textit{// using (A.10)} && \text{(B.125)}\end{aligned}$$

which is again the Kronecker product of two symmetric matrices.

Given that $\tilde{\mathtt{U}}_j = \tilde{\mathtt{U}}_{j\mathtt{Q}}\tilde{\mathtt{U}}_{j\mathtt{R}}$, we can transform $(\tilde{\mathtt{U}}_j^\top \tilde{\mathtt{U}}_j)^{-1}$ to $\tilde{\mathtt{U}}_{j\mathtt{R}}^{-1}\tilde{\mathtt{U}}_{j\mathtt{R}}^{-\top}$. The last term then becomes

$$\begin{aligned}
\mathtt{K}_{mr}^\top \mathtt{Z}_j^*(\tilde{\mathtt{U}}_j^\top \tilde{\mathtt{U}}_j)^{-1}\mathtt{Z}_j^{*\top}\mathtt{K}_{mr} &= (\mathtt{I}\otimes(\mathbf{w}_j\odot\mathbf{r}_j^*))\tilde{\mathtt{U}}_{j\mathtt{R}}^{-1}\tilde{\mathtt{U}}_{j\mathtt{R}}^{-\top}(\mathtt{I}\otimes(\mathbf{w}_j\odot\mathbf{r}_j^*)^\top) \\
&\qquad\qquad\qquad\qquad \textit{// using (A.13)} \\
&= \tilde{\mathtt{U}}_{j\mathtt{R}}^{-1}\tilde{\mathtt{U}}_{j\mathtt{R}}^{-\top}\otimes(\mathbf{w}_j\odot\mathbf{r}_j^*)(\mathbf{w}_j\odot\mathbf{r}_j^*)^\top \qquad \textit{// using (A.10)} \qquad \text{(B.126)}
\end{aligned}$$

which again is the Kronecker product of two symmetric matrices. This shows that the Gauss-Newton matrix computation can be sped up by factor of 4 at most by exploiting this additional symmetry (in addition to the original Hessian symmetry).

Appendix C

Derivations for projective bundle adjustment

C.1 Analytic expressions of the required derivatives

Section 5.5 proposed to formulate the projective bundle adjustment problem as follows:

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

$$= \min_{\{\mathbf{p}_i\},\{\mathbf{q}_i\},\{s_i\},\{\mathbf{x}_j\},\{t_j\}} \sum_{(i,j)\in\Omega} \|\varepsilon_{i,j}(\mathbf{p}_i,\mathbf{q}_i,s_i,\mathbf{x}_j,t_j,\mu_i)\|_2^2, \tag{C.2}$$

where the unknowns are listed in Table 1.

In Section 5.3.1, we showed that only three derivatives are required to compile all algorithms listed in Table 5.4. In this section, we present the analytic form of these derivatives obtained by using the techniques illustrated in the work of Minka [2000] and Magnus and Neudecker [2007].

C.1.1 Jacobian with respect to the camera parameters

As shown in Table 5.3, the Jacobian with respect to the camera parameters ($\partial\varepsilon/\partial\tilde{\mathbf{p}}$) consists of three smaller Jacobians which are $\partial\varepsilon/\partial\mathbf{p}$, $\partial\varepsilon/\partial\mathbf{q}$ and $\partial\varepsilon/\partial\mathbf{s}$. We first obtain the analytic derivatives of a single residual $\varepsilon_{i,j}$ with respect to $\mathbf{p}_i$, $\mathbf{q}_i$ and s_i:

$$\frac{\partial \varepsilon_{i,j}}{\partial \mathbf{p}_i} = \frac{\partial \pi_{i,j}}{\partial \mathbf{p}_i} = \frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \frac{\partial \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix}}{\partial \mathbf{p}_i} = \frac{\begin{bmatrix} \tilde{\mathbf{x}}_j^\top & \\ & \tilde{\mathbf{x}}_j^\top \end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \tag{C.3}$$

$$\frac{\partial \varepsilon_{i,j}}{\partial \mathbf{q}_i} = \frac{\partial \pi_{i,j}}{\partial \mathbf{q}_i} = \frac{\partial}{\partial \mathbf{q}_i} \left[\frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \right] \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix} = \frac{-\mu_i \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix} \mathbf{x}_j^\top}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2} \tag{C.4}$$

$$\frac{\partial \varepsilon_{i,j}}{\partial s_i} = \frac{\partial \pi_{i,j}}{\partial s_i} = \frac{\partial}{\partial s_i} \left[\frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \right] \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix} = \frac{-t_j \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2} \tag{C.5}$$

Using above results, we can construct $\partial \varepsilon_{i:}/\partial \mathbf{p}_i$, $\partial \varepsilon_{i:}/\partial \mathbf{q}_i$ and $\partial \varepsilon_{i:}/\partial s_i$ by stacking all the residuals of visible points for frame i. Further noting that $\varepsilon_{i:}$ only depends on camera i leads to

$$\frac{\partial \varepsilon}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial \varepsilon_{1:}}{\partial \mathbf{p}_1} & \cdots & \frac{\partial \varepsilon_{1:}}{\partial \mathbf{p}_f} \\ \vdots & \ddots & \vdots \\ \frac{\partial \varepsilon_{f:}}{\partial \mathbf{p}_1} & \cdots & \frac{\partial \varepsilon_{f:}}{\partial \mathbf{p}_f} \end{bmatrix} = \begin{bmatrix} \frac{\partial \varepsilon_{1:}}{\partial \mathbf{p}_1} & & \\ & \ddots & \\ & & \frac{\partial \varepsilon_{f:}}{\partial \mathbf{p}_f} \end{bmatrix} =: \texttt{blkdiag}\left(\frac{\partial \varepsilon_{1:}}{\partial \mathbf{p}_1}, \cdots, \frac{\partial \varepsilon_{f:}}{\partial \mathbf{p}_f} \right), \tag{C.6}$$

where `blkdiag` is a standard MATLAB function for creating a block-diagonal matrix from input submatrices. Similarly, we obtain

$$\frac{\partial \varepsilon}{\partial \mathbf{q}} = \texttt{blkdiag}\left(\frac{\partial \varepsilon_{1:}}{\partial \mathbf{q}_1}, \cdots, \frac{\partial \varepsilon_{f:}}{\partial \mathbf{q}_f} \right) \tag{C.7}$$

$$\frac{\partial \varepsilon}{\partial \mathbf{s}} = \texttt{blkdiag}\left(\frac{\partial \varepsilon_{1:}}{\partial s_1}, \cdots, \frac{\partial \varepsilon_{f:}}{\partial s_f} \right). \tag{C.8}$$

We now have sufficient building blocks for computing $\mathtt{J}_{\tilde{\mathbf{p}}}$.

C.1.2 Jacobian with respect to the point parameters

As shown in Table 5.3, the Jacobian with respect to the point parameters ($\partial \varepsilon/\partial \tilde{\mathbf{x}}$) consists of two smaller Jacobians which are $\partial \varepsilon/\partial \mathbf{x}$ and $\partial \varepsilon/\partial \mathbf{t}$. Similar to Section C.1.1, we first obtain the analytic derivatives of a single residual $\varepsilon_{i,j}$ with respect to $\mathbf{x}_j$ and t_j:

$$\frac{\partial \varepsilon_{i,j}}{\partial \mathbf{x}_j} = \frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \frac{\partial \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix}}{\partial \mathbf{x}_j} + \frac{\partial}{\partial \mathbf{x}_j} \left[\frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \right] \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix} \tag{C.9}$$

$$= \frac{\begin{bmatrix} p_{i,1} & p_{i,2} & p_{i,3} \\ p_{i,5} & p_{i,6} & p_{i,7} \end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} - \frac{\mu_i \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix} \mathbf{q}_i^\top}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2} \tag{C.10}$$

$$\frac{\partial \varepsilon_{i,j}}{\partial t_j} = \frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \frac{\partial \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix}}{\partial t_j} + \frac{\partial}{\partial t_j} \left[\frac{1}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} \right] \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix} \tag{C.11}$$

$$= \frac{\begin{bmatrix} p_{i,4} \\ p_{i,8} \end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} - \frac{s_i \begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2}. \tag{C.12}$$

This then yields

$$\frac{\partial \varepsilon_{i,j}}{\partial \tilde{\mathbf{x}}_j} = \begin{bmatrix} \dfrac{\partial \varepsilon_{i,j}}{\partial \mathbf{x}_j} & \dfrac{\partial \varepsilon_{i,j}}{\partial t_j} \end{bmatrix} = \frac{\begin{bmatrix} \mathbf{p}_{i,1}^\top \\ \mathbf{p}_{i,2}^\top \end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j} - \frac{\begin{bmatrix} \mathbf{p}_{i,1}^\top \tilde{\mathbf{x}}_j \\ \mathbf{p}_{i,2}^\top \tilde{\mathbf{x}}_j \end{bmatrix} \begin{bmatrix} \mu_i \mathbf{q}_i^\top & s_i \end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j) + s_i t_j)^2}. \tag{C.13}$$

Using the above result, we can construct $\partial \varepsilon_{:j} / \partial \tilde{\mathbf{x}}_j$ and by stacking all the residuals of visible frames for point j. Further noting that $\varepsilon_{:j}$ only depends on point j leads to

$$\frac{\partial \varepsilon_{p,m}}{\partial \tilde{\mathbf{x}}} = \begin{bmatrix} \dfrac{\partial \varepsilon_{:1}}{\partial \tilde{\mathbf{x}}_1} & \cdots & \dfrac{\partial \varepsilon_{:1}}{\partial \tilde{\mathbf{x}}_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \varepsilon_{:n}}{\partial \tilde{\mathbf{x}}_1} & \cdots & \dfrac{\partial \varepsilon_{:n}}{\partial \tilde{\mathbf{x}}_n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \varepsilon_{:1}}{\partial \tilde{\mathbf{x}}_1} & & \\ & \ddots & \\ & & \dfrac{\partial \varepsilon_{:n}}{\partial \tilde{\mathbf{x}}_n} \end{bmatrix} =: \texttt{blkdiag}\left(\frac{\partial \varepsilon_{:1}}{\partial \tilde{\mathbf{x}}_1}, \cdots, \frac{\partial \varepsilon_{:n}}{\partial \tilde{\mathbf{x}}_n} \right), \tag{C.14}$$

where `blkdiag` is a standard MATLAB function for creating a block-diagonal matrix from input submatrices. We can then obtain $\mathtt{J}_{\tilde{\mathbf{x}}} := \mathtt{K}_{p2f}[\partial \varepsilon_{p,m} / \partial \tilde{\mathbf{x}}]$, where the permutation matrix $\mathtt{K}_{p2f}$ is used to reorder the residuals in the camera-major form.

C.1.3 Additional derivative for RW1

For the RW1 algorithm, we need to compute $\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]\varepsilon/\partial\tilde{\mathbf{p}}$. The derivative of a matrix pseudo-inverse is provided in (A.25). Applying this yields:

$$\begin{aligned}
\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]\varepsilon &= (-\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger} + (\mathtt{J}_{\tilde{\mathbf{x}}}^{\top}\mathtt{J}_{\tilde{\mathbf{x}}})^{-1}\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]^{\top}(\mathtt{I} - \mathtt{J}_{\tilde{\mathbf{x}}}\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}))\varepsilon && \text{(C.15)}\\
&= (\mathtt{J}_{\tilde{\mathbf{x}}}^{\top}\mathtt{J}_{\tilde{\mathbf{x}}})^{-1}\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]^{\top}\varepsilon - \mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}\varepsilon - (\mathtt{J}_{\tilde{\mathbf{x}}}^{\top}\mathtt{J}_{\tilde{\mathbf{x}}})^{-1}\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]^{\top}\mathtt{J}_{\tilde{\mathbf{x}}}\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}\varepsilon. && \text{(C.16)}
\end{aligned}$$

We note that $\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}\varepsilon = \Delta\tilde{\mathbf{x}}$, which is the last inner iteration update for points ($\tilde{\mathbf{x}}$). At this inner iteration stage, the points should have already converged to $\tilde{\mathbf{x}}^{*}(\tilde{\mathbf{p}})$ using a second-order solver and therefore $\Delta\tilde{\mathbf{x}}$ should be close to 0. Hence, (C.16) simplies to

$$\lim_{\Delta\tilde{\mathbf{x}}\to 0}\left[\partial[\mathtt{J}_{\tilde{\mathbf{x}}}^{\dagger}]\varepsilon\right] = (\mathtt{J}_{\tilde{\mathbf{x}}}^{\top}\mathtt{J}_{\tilde{\mathbf{x}}})^{-1}\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]^{\top}\varepsilon. \qquad \text{(C.17)}$$

In essence, we need to be able to compute $\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]^{\top}\varepsilon/\partial\tilde{\mathbf{p}}$.

We start by deriving $\partial[\partial\varepsilon_{i,j}/\partial\tilde{\mathbf{x}}_j]^{\top}\varepsilon_{i,j}$ with respect to the i-th camera parameters.

$$\begin{aligned}
\frac{\partial}{\partial\mathbf{p}_i}\left[\frac{\partial\varepsilon_{i,j}}{\partial\tilde{\mathbf{x}}_j}\right]^{\top}\varepsilon_{i,j} &= \frac{\partial}{\partial\mathbf{p}_i}\left[\frac{\begin{bmatrix}\mathbf{p}_{i,1}^{\top}\\ \mathbf{p}_{i,2}^{\top}\end{bmatrix}}{\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j} - \frac{\begin{bmatrix}\mathbf{p}_{i,1}^{\top}\tilde{\mathbf{x}}_j\\ \mathbf{p}_{i,2}^{\top}\tilde{\mathbf{x}}_j\end{bmatrix}\begin{bmatrix}\mu_i\mathbf{q}_i^{\top} & s_i\end{bmatrix}}{(\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j)^2}\right]^{\top}\varepsilon_{i,j} && \text{(C.18)}\\
&= \frac{1}{\partial\mathbf{p}_i}\left[\frac{[\varepsilon_{ij1}\partial\mathbf{p}_{i,1}+\varepsilon_{ij2}\partial\mathbf{p}_{i,2}]}{\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j}\right. && \text{(C.19)}\\
&\qquad\left. - \frac{\begin{bmatrix}\mu_i\mathbf{q}_i\\ s_i\end{bmatrix}[\varepsilon_{ij1}\tilde{\mathbf{x}}_j^{\top}\partial\mathbf{p}_{i,1}+\varepsilon_{ij2}\tilde{\mathbf{x}}_j^{\top}\partial\mathbf{p}_{i,2}]}{(\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j)^2}\right] && \text{(C.20)}\\
&= \frac{\varepsilon_{i,j}^{\top}\otimes\mathtt{I}_4}{\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j} - \frac{\begin{bmatrix}\mu_i\mathbf{q}_i\\ s_i\end{bmatrix}(\varepsilon_{i,j}^{\top}\otimes\tilde{\mathbf{x}}_j^{\top})}{(\mu_i(\mathbf{q}_i^{\top}\mathbf{x}_j)+s_it_j)^2} && \text{(C.21)}
\end{aligned}$$

$$\frac{\partial}{\partial \mathbf{q}_i}\left[\frac{\partial \varepsilon_{i,j}}{\partial \tilde{\mathbf{x}}_j}\right]^\top \varepsilon_{i,j} = \frac{\partial}{\partial \mathbf{q}_i}\left[\frac{\begin{bmatrix}\mathbf{p}_{i,1}^\top\\ \mathbf{p}_{i,2}^\top\end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j)+s_i t_j} - \frac{\begin{bmatrix}\mathbf{p}_{i,1}^\top\tilde{\mathbf{x}}_j\\ \mathbf{p}_{i,2}^\top\tilde{\mathbf{x}}_j\end{bmatrix}\begin{bmatrix}\mu_i\mathbf{q}_i^\top & s_i\end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j)+s_i t_j)^2}\right]^\top \varepsilon_{i,j} \tag{C.22}$$

$$= -\frac{\mu_i\begin{bmatrix}\mathbf{p}_{i,1} & \mathbf{p}_{i,2}\end{bmatrix}\varepsilon_{i,j}\mathbf{x}_j^\top}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j)+s_i t_j)^2} - \frac{\begin{bmatrix}\mathbf{p}_{i,1}^\top\tilde{\mathbf{x}}_j & \mathbf{p}_{i,2}^\top\tilde{\mathbf{x}}_j\end{bmatrix}\varepsilon_{i,j}\begin{bmatrix}\mu_i \mathtt{I}_3\\ \mathbf{0}_{1\times 4}\end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j)+s_i t_j)^2} \tag{C.23}$$

$$+\frac{2\mu_i\begin{bmatrix}\mathbf{p}_{i,1}^\top\tilde{\mathbf{x}}_j & \mathbf{p}_{i,2}^\top\tilde{\mathbf{x}}_j\end{bmatrix}\varepsilon_{i,j}\begin{bmatrix}\mu_i\mathbf{q}\\ s_i\end{bmatrix}\mathbf{x}_j^\top}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j)+s_i t_j)^3} \tag{C.24}$$

$$\frac{\partial}{\partial s_i}\left[\frac{\partial \varepsilon_{i,j}}{\partial \tilde{\mathbf{x}}_j}\right]^\top \varepsilon_{i,j} = \frac{\partial}{\partial s_i}\left[\frac{\begin{bmatrix}\mathbf{p}_{i,1}^\top\\ \mathbf{p}_{i,2}^\top\end{bmatrix}}{\mu_i(\mathbf{q}_i^\top \mathbf{x}_j)+s_i t_j} - \frac{\begin{bmatrix}\mathbf{p}_{i,1}^\top\tilde{\mathbf{x}}_j\\ \mathbf{p}_{i,2}^\top\tilde{\mathbf{x}}_j\end{bmatrix}\begin{bmatrix}\mu_i\mathbf{q}_i^\top & s_i\end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j)+s_i t_j)^2}\right]^\top \varepsilon_{i,j} \tag{C.25}$$

$$= -\frac{t_j\begin{bmatrix}\mathbf{p}_{i,1} & \mathbf{p}_{i,2}\end{bmatrix}\varepsilon_{i,j}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j)+s_i t_j)^2} - \frac{\begin{bmatrix}\mathbf{p}_{i,1}^\top\tilde{\mathbf{x}}_j & \mathbf{p}_{i,2}^\top\tilde{\mathbf{x}}_j\end{bmatrix}\varepsilon_{i,j}\begin{bmatrix}\mathbf{0}_{3\times 1}\\ 1\end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j)+s_i t_j)^2} \tag{C.26}$$

$$+\frac{2t_j\begin{bmatrix}\mathbf{p}_{i,1}^\top\tilde{\mathbf{x}}_j & \mathbf{p}_{i,2}^\top\tilde{\mathbf{x}}_j\end{bmatrix}\varepsilon_{i,j}\begin{bmatrix}\mu_i\mathbf{q}\\ s_i\end{bmatrix}}{(\mu_i(\mathbf{q}_i^\top \mathbf{x}_j)+s_i t_j)^3}. \tag{C.27}$$

Then, by noting the block-diagonal structure of $\partial\varepsilon_{p,m}/\partial\tilde{\mathbf{x}}$ in (C.14), we have

$$\frac{\partial}{\partial \mathbf{p}}[\mathtt{J}_{\tilde{\mathbf{x}}}]^\top\varepsilon = \frac{\partial}{\partial \mathbf{p}}\left[\frac{\partial\varepsilon_{p,m}}{\partial\tilde{\mathbf{x}}}\right]^\top\varepsilon_{p,m} = \begin{bmatrix}\frac{\partial}{\partial \mathbf{p}}\left[\frac{\partial\varepsilon_{:1}}{\partial\tilde{\mathbf{x}}_1}\right]^\top\varepsilon_{:1}\\ \vdots\\ \frac{\partial}{\partial \mathbf{p}}\left[\frac{\partial\varepsilon_{:n}}{\partial\tilde{\mathbf{x}}_n}\right]^\top\varepsilon_{:n}\end{bmatrix} = \begin{bmatrix}\sum_{i\in\Omega_1}\frac{\partial}{\partial \mathbf{p}}\left[\frac{\partial\varepsilon_{i,1}}{\partial\tilde{\mathbf{x}}_1}\right]^\top\varepsilon_{i,1}\\ \vdots\\ \sum_{i\in\Omega_n}\frac{\partial}{\partial \mathbf{p}}\left[\frac{\partial\varepsilon_{in}}{\partial\tilde{\mathbf{x}}_n}\right]^\top\varepsilon_{in}\end{bmatrix} \tag{C.28}$$

$$= \begin{bmatrix}\sum_{i\in\Omega_1}\frac{\partial}{\partial \mathbf{p}_1}\left[\frac{\partial\varepsilon_{i,1}}{\partial\tilde{\mathbf{x}}_1}\right]^\top\varepsilon_{i,1} & \cdots & \sum_{i\in\Omega_1}\frac{\partial}{\partial \mathbf{p}_f}\left[\frac{\partial\varepsilon_{i,1}}{\partial\tilde{\mathbf{x}}_1}\right]^\top\varepsilon_{i,1}\\ \vdots & \ddots & \vdots\\ \sum_{i\in\Omega_n}\frac{\partial}{\partial \mathbf{p}_1}\left[\frac{\partial\varepsilon_{in}}{\partial\tilde{\mathbf{x}}_n}\right]^\top\varepsilon_{in} & \cdots & \sum_{i\in\Omega_n}\frac{\partial}{\partial \mathbf{p}_f}\left[\frac{\partial\varepsilon_{in}}{\partial\tilde{\mathbf{x}}_n}\right]^\top\varepsilon_{in}\end{bmatrix}, \tag{C.29}$$

where Ω_j denotes the set of visible frames for point j. In other words, we just need to sum the quantities $\partial[\partial\varepsilon_{i,j}/\partial\tilde{\mathbf{x}}_j]^\top\varepsilon_{i,j}/\partial\mathbf{p}_i$ (which is derived in (C.21)) appropriately. Similar expressions can be obtained for $\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]^\top\varepsilon/\partial\mathbf{q}$ and $\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]^\top\varepsilon/\partial\mathbf{s}$, which are altogether combined to form $\partial[\mathtt{J}_{\tilde{\mathbf{x}}}]^\top\varepsilon/\partial\tilde{\mathbf{p}}$.

C.2 Further details on constraining the scale freedoms

In this section, we describe briefly how to incorporate the Riemannian manifold optimization framework for constraining the scale freedoms.

C.2.1 Tangent space projectors for the homogeneous variable projection algorithms

For the variable projection (RW) algorithms, the point parameters $\tilde{\mathbf{x}}$ are updated in the inner loop and the camera parameters $\tilde{\mathbf{p}}$ are updated in the outer loop. For the homogeneous set of algorithms, the tangent space projector for the inner loop (i.e. over the points) is

$$\mathtt{P}_{r1} = \begin{bmatrix} \tilde{\mathbf{x}}_1^\perp & & \\ & \ddots & \\ & & \tilde{\mathbf{x}}_n^\perp \end{bmatrix} = \mathtt{blkdiag}(\tilde{\mathbf{x}}_1^\perp, \cdots, \tilde{\mathbf{x}}_n^\perp) \in \mathbb{R}^{3n\times 4n}, \tag{C.30}$$

where $\tilde{\mathbf{x}}_j^\perp$ is the left null space of $\tilde{\mathbf{x}}_j$. Hence, the reduced update is in $\mathbb{R}^{3n}$, which needs to be projected back to $\mathbb{R}^{4n}$ by multiplying $\mathtt{P}_{r1}^\top$ to the reduced update vector.

The tangent space projector for the outer loop (over the cameras) is

$$\mathtt{P}_{r2} = \begin{bmatrix} \tilde{\mathbf{p}}_1^\perp & & \\ & \ddots & \\ & & \tilde{\mathbf{p}}_f^\perp \end{bmatrix} = \mathtt{blkdiag}(\tilde{\mathbf{p}}_1^\perp, \cdots, \tilde{\mathbf{p}}_n^\perp) \in \mathbb{R}^{11f\times 12f}, \tag{C.31}$$

where $\tilde{\mathbf{p}}_i := [\mathbf{p}_i^\top, \mathbf{q}_i^\top, s_i]^\top$ and $\tilde{\mathbf{p}}_i^\perp$ is the left null space of $\tilde{\mathbf{p}}_i$. Similarly, the reduced update is in $\mathbb{R}^{11f}$, which needs to be projected back to $\mathbb{R}^{12f}$ by multiplying $\mathtt{P}_{r2}^\top$ to the reduced update vector.

C.2.2 Tangent space projector for the joint algorithms

Assuming that the parameters are stacked as $[\tilde{\mathbf{p}}_1, \cdots, \tilde{\mathbf{p}}_f, \tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_n]$, the corresponding tangent space projector for the homogeneous joint algorithms is

$$\mathtt{P}_{r3} = \begin{bmatrix} \tilde{\mathbf{p}}_1^{\perp} & & & & & \\ & \ddots & & & & \\ & & \tilde{\mathbf{p}}_f^{\perp} & & & \\ & & & \tilde{\mathbf{x}}_1^{\perp} & & \\ & & & & \ddots & \\ & & & & & \tilde{\mathbf{x}}_n^{\perp} \end{bmatrix} \in \mathbb{R}^{(11f+3n)\times(12f+4n)}. \tag{C.32}$$

Hence, the reduced update will be in $\mathbb{R}^{(11f+3n)}$, which needs to be projected back to $\mathbb{R}^{(12f+4n)}$ by multiplying $\mathtt{P}_{r3}^{\top}$ to the reduced update vector.

C.3 Details on constraining the gauge freedom of VarPro

In Section 5.3.3, the gauge freedom is penalized using the rank-filling technique introduced by Okatani et al. [2011]. This can be regarded as a relaxed form of the Riemannian manifold optimization framework illustrated in Section A.2.5.

Since the subproblem is already projected when constraining the scale freedoms (see Section 5.3.2), we instead add a gauge freedom penalty term to the subproblem which discourages updates along the column space of $\mathtt{P} := [\mathtt{P}_1^{\top}, \cdots, \mathtt{P}_f^{\top}]^{\top}$.

C.3.1 Homogeneous affine, hom. projective and inhom. projective

As illustrated in Table 5.2, the gauge can be any invertible matrix $\in \mathbb{R}^{4\times 4}$. In each iteration, we wish to prevent any update in the direction spanning $\mathrm{col}(\mathtt{P})$.

Let us generalize the update in cameras ($\mathtt{P}$) such that

$$\mathtt{P} + \Delta\mathtt{P} = \mathtt{P}\mathtt{A} + \mathtt{P}^{\perp}\mathtt{B}, \tag{C.33}$$

where $\mathtt{P}^{\perp}$ is the left null space of $\mathtt{P}$. Then, by definition,

$$\mathtt{P}^{\top}\mathtt{P} + \mathtt{P}^{\top}\Delta\mathtt{P} = \mathtt{P}^{\top}\mathtt{P}\mathtt{A} + \mathtt{P}^{\top}\mathtt{P}^{\perp}\mathtt{B} = \mathtt{P}^{\top}\mathtt{P}\mathtt{A}. \tag{C.34}$$

Since we wish to prevent any update along the column space of $\mathtt{P}$, this fixes $\mathtt{A}$ to be $\mathtt{I}$, leading to the linear constraint $\mathtt{P}^\top \Delta\mathtt{P} = \mathtt{0}$. Hence, instead of solving (C.35), we solve the relaxed objective

$$\Delta\mathbf{p} = \arg\min_{\delta} \mathbf{g}\delta + \frac{1}{2}\delta^\top \mathtt{H}\delta + \|\mathtt{P}^\top \Delta\mathtt{P}\|_2^2, \tag{C.35}$$

where $\Delta\mathbf{p}$ is $\Delta\mathtt{P}$ vectorized.

C.3.2 Homogeneous affine

As illustrated in Table 5.2, the gauge for an homogeneous affine camera is any invertible matrix $\in \mathbb{R}^{4\times 4}$ with the last row set to $[0,0,0,1]$. Similar to C.3.1, In each update, we wish to prevent any update in the direction spanning the column space of $\mathtt{P}$.

Let us generalize the update in cameras ($\mathtt{P}$) such that

$$\mathtt{P} + \Delta\mathtt{P} = \mathtt{P}\begin{bmatrix} \mathtt{C} & \mathbf{d} \\ \mathbf{0}^\top & 1 \end{bmatrix} + \mathtt{P}^\perp \mathtt{B}, \tag{C.36}$$

where $\mathtt{P}^\perp$ is the left null space of $\mathtt{P}$. Then, by definition,

$$\mathtt{P}^\top\mathtt{P} + \mathtt{P}^\top\Delta\mathtt{P} = \mathtt{P}^\top\mathtt{P}\mathtt{A} + \mathtt{P}^\top\mathtt{P}^\perp\mathtt{B} = \mathtt{P}^\top\mathtt{P}\mathtt{A}. \tag{C.37}$$

Now, we can replace $\mathtt{A}$ with the homogeneous affine gauge matrix

$$\mathtt{A} = \begin{bmatrix} \mathtt{C} & \mathbf{d} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{C.38}$$

$$\Rightarrow \mathtt{P}^\top\mathtt{P} + \mathtt{P}^\top\Delta\mathtt{P} = \mathtt{P}^\top\mathtt{P}\begin{bmatrix} \mathtt{C} & \mathbf{d} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{C.39}$$

Separating $\mathtt{P}$ into $[\mathtt{P}_{1:3}, \mathbf{p}_4]$ gives

$$\begin{bmatrix} \mathtt{P}_{1:3}^\top \\ \mathbf{p}_4^\top \end{bmatrix} [\mathtt{P}_{1:3}, \mathbf{p}_4] + \begin{bmatrix} \mathtt{P}_{1:3}^\top \\ \mathbf{p}_4^\top \end{bmatrix} [\Delta\mathtt{P}_{1:3}, \Delta\mathbf{p}_4] = \begin{bmatrix} \mathtt{P}_{1:3}^\top \\ \mathbf{p}_4^\top \end{bmatrix} [\mathtt{P}_{1:3}\mathtt{C}, \mathtt{P}_{i:3}\mathbf{d} + \mathbf{p}_4] \tag{C.40}$$

$$\Rightarrow \begin{bmatrix} \mathtt{P}_{1:3}^\top\mathtt{P}_{1:3} & \mathtt{P}_{1:3}^\top\mathbf{p}_4 \\ \mathbf{p}_4^\top\mathtt{P}_{1:3} & \mathbf{p}_4^\top\mathbf{p}_4 \end{bmatrix} + \begin{bmatrix} \mathtt{P}_{1:3}^\top\Delta\mathtt{P}_{1:3} & \mathtt{P}_{1:3}^\top\Delta\mathbf{p}_4 \\ \mathbf{p}_4^\top\Delta\mathtt{P}_{1:3} & \mathbf{p}_4^\top\Delta\mathbf{p}_4 \end{bmatrix} = \begin{bmatrix} \mathtt{P}_{1:3}^\top\mathtt{P}_{1:3}\mathtt{C}, \mathtt{P}_{1:3}^\top\mathtt{P}_{1:3}\mathbf{d} + \mathtt{P}_{1:3}^\top\mathbf{p}_4 \\ \mathbf{p}_4^\top\mathtt{P}_{1:3}\mathtt{C}, \mathbf{p}_4^\top\mathtt{P}_{1:3}\mathbf{d} + \mathbf{p}_4^\top\mathbf{p}_4 \end{bmatrix} \tag{C.41}$$

Similar to Section C.3.1, we fix $\mathtt{C} = \mathtt{I}_3$ and $\mathbf{d} = \mathbf{0}$ to prevent the update along the column space of $\mathtt{P}$. This forms 4 linear constraints some of which are inter-dependent. We have empirically found that constraining the following two independent constraints yields the best performance:

(applying all 4 seems to over-constrain the problem.)

$$\mathtt{P}_{1:3}^{\top}\Delta\mathtt{P}_{1:3} = \mathtt{0}, \tag{C.42}$$

$$\mathbf{p}_4^{\top}\Delta\mathbf{p}_4 = 0. \tag{C.43}$$

We now solve the relaxed objective

$$\Delta\mathbf{p} = \arg\min_{\delta} \mathbf{g}\delta + \frac{1}{2}\delta^{\top}\mathtt{H}\delta + \|\mathtt{P}_{1:3}^{\top}\Delta\mathtt{P}_{1:3}\|_2^2 + \|\mathbf{p}_4^{\top}\Delta\mathbf{p}_4\|_2^2 \tag{C.44}$$

where $\Delta\mathbf{p}$ is $\Delta\mathtt{P}$ vectorized.