Diss. ETH No. 20705

Subspaces in Geometric Computer Vision: Models and Algorithms

A dissertation submitted to

ETH Zürich

for the Degree of

Doctor of Sciences

presented by

Roland Angst
MSc ETH Zürich
born 07 September 1982
citizen of Wil, canton of Zürich, Switzerland

accepted on the recommendation of

Prof. Dr. Marc Pollefeys
examiner

Prof. Dr. Richard Hartley
co-examiner

Prof. Dr. Jean Ponce
co-examiner

Publication year

2012

Abstract

As a highly general concept, linear subspaces represent basic buildings blocks for complex models and abstractions, and are therefore prevalent in science and engineering. In this thesis, we will mostly focus on geometric computer vision. Geometric computer vision addresses the problem of computing 3D reconstructions from imagery data. Models in geometric computer vision heavily build upon subspaces and results from linear and multilinear algebra. Such subspace models enjoy a rich geometric and algebraic structure. Moreover, they are inherently linked to rank- or dimensionality-constraints which can quickly render problems into highly non-trivial challenges.

The major goal and contribution of this thesis is to present a holistic view of subspace models and algorithms mostly for problems as they appear in geometric computer vision. Specifically, the upcoming chapters will present i) non-linear least squares methods tailored to low-rank matrix completion problems, ii) tensor decompositions and Markov Chain Monte Carlo sampling strategies for structure-from-motion problems, iii) convex relaxations of the matrix rank function with a generalization of the trace norm, and iv) Pluecker coordinates which are then used to represent and compute an unknown subspace which intersects multiple known subspaces in unknown intersection points. Several new theoretical insights have been gained by deriving and applying these methods. For example, low-rank tensor factorizations lead to a novel low-rank constraint on feature trajectories of points on a moving object. Even though throughout this thesis these methods and algorithms will mostly be presented and applied to problems in geometric computer vision, many of the introduced concepts and ideas are not limited to this field.

Zusammenfassung

Lineare Unterräume stellen ein sehr generelles Konzept dar und daher dienen sie als Grundbausteine for komplexe Modelle und Abstraktionen. Es überrascht daher nicht, dass Unterräume in den Wissenschaften und im Ingenieurwesen allgegenwärtig sind. Diese Dissertation setzt den Schwerpunkt auf geometrische Aspekte in der Computer Vision. Dieses Forschungsgebiet beschäftigt sich mit dem Problem, aus Bilddaten räumliche 3D Rekonstruktionen zu berechnen. Modelle in diesem Gebiet bauen stark auf Unterräumen und Erkenntnissen sowohl aus der linearen als auch der multilinearen Algebra auf. Nebst starken geometrische und algebraische Strukturen besitzen diese Modelle häufig komplexe Rang- oder Dimensionalitätseigenschaften. Diese Tatsache bedeutet, dass solche Probleme nicht triviale Herausforderungen darstellen.

Das Hauptziel und der Beitrag dieser Dissertation ist die Herleitung und die Präsentation einer ganzheitliche Betrachtungsweise von auf Unterräumen aufbauenden Modellen und Algorithmen, die geometrische Probleme in der Computer Vision behandeln. Die einzelnen Kapitel vertiefen die folgenden Themen: i) nicht lineare Methoden der kleinsten Quadrate zur Vervollständigung von teilweise bekannten Matrizen mit niedrigem Rang, ii) Tensorzerlegungen und Markov Chain Monte Carlo Methoden für sogenannte Structure-from-Motion Probleme, iii) konvexe Relaxierungen des Matrixranges mittels einer Verallgemeinerung der Schatten-1-Matrixnorm, iv) Plücker Koordinaten zur Berechnung eines Unterraums, der mehrere bekannte Unterräume in unbekannten Punkten schneidet. Die Herleitung und Anwendung dieser Methoden führte zu einigen neuen theoretischen Erkenntnissen, wie zum Beispiel zu einer Bedingung für Trajektorien von Punkten auf einem sich bewegenden Objekt, welche solche Trajektorien auf Vektoren innerhalb eines niedrig dimensionalen Unterraumes einschränkt. Obwohl die Methoden und Algorithmen meist anhand von geometrischen Problemen in der Computer Vision eingeführt und dargestellt werden, sind viele der gezeigten Konzepte und Ideen nicht auf dieses Gebiet limitiert.

Acknowledgements

First of all, I thank my advisor Prof. Marc Pollefeys. I am very grateful for his advice and I benefited a lot from his expertise, exceptional intuition and insights. Marc's guidance has given me great freedom to explore and pursue many interesting ideas, a highly important ingredient for becoming an independent researcher. I would also like to thank my co-examiners Prof. Richard Hartley and Prof. Jean Ponce for their interest in my work and their helpful and inspiring suggestions. Being a member of the Computer Vision and Geometry group, I had the opportunity to meet many interesting people, resulting in fruitful discussions about scientific and also about non-scientific topics. Specifically, I would like to mention the senior researchers Dr. Gabriel Brostow, Dr. Friedrich Fraundorfer, Dr. Dongwuk Kyoung, Dr. Luca Ballan, Dr. Christopher Zach, Dr. Roland Memisevic, Dr. Kevin Köser, Dr. Amaël Delaunoy, Dr. Jean-Charles Bazin, Dr. Kalin Kolev, as well as the PhD students Georges Baatz, Andrea Cohen, Christian Häne, Lionel Heng, Dominik Honegger, Bastien Jacquet, Gim Hee Lee, Fabio Maninchedda, Lorenz Meier, Olivier Saurer, Aparna Taneja, Petri Tanskanen, and the long-term visitors Sudipta Sinha, Li Guan, Changchang Wu, David Gallup, Brian Clipp, Oisin Mac Aodha, and Andreas Geiger. Especially, I had the pleasure to share a Swiss-German speaking office with Alexander Schwing, Jens Puwein, and Bernhard Zeisl. Thanks to Georges and Christopher, I have always been in good company, quite often even during conferences. I also need to mention Thorsten Steenbock whose IT support helped me with any hard- or software problem. A special thank goes to Beatrice Gander and Susanne Keller. Being responsible for the group administration, they were not only very helpful in administrative affairs but also represented often the best source of information.

I also thank my family and friends who fortunately distracted me whenever I got stuck too deeply in a scientific problem. Furthermore, I thank Prof. Raquel Urtasun who enabled my research visit at TTIC. Lastly, I acknowledge Google for the Google Europe Fellowship in Computer Vision which supported my research and allowed me to freely explore research ideas.

Contents

1 Overview

1.1 Motivation

A subspace is defined in linear algebra as a subset of an ambient vector space. Such a subset must form again a vector space with the same vector space operations as the original ambient vector space. Besides this purely algebraic point of view, subspaces also offer a very intuitive geometric point of view. For example, a two-dimensional plane in Euclidean 3D space represents a two-dimensional (affine) subspace. This duality between algebraic and geometric interpretation and intuition will reappear several times throughout this thesis.

Subspaces are a highly general concept, representing basic buildings blocks for more complex models and abstractions, and are therefore prevalent in science and engineering. In this thesis, we will mostly, but not exclusively, focus on geometric computer vision. Geometric computer vision addresses the problem of computing 3D reconstructions from imagery data. Geometric computer vision heavily builds upon subspaces and results from linear and multilinear algebra. For example, especially the assumption of a rigid environment translates to strong algebraic multiple-view constraints which inherently build upon subspaces. However, the underlying geometry also allows to reason in more intuitive geometric terms like intersecting planes and rays.

Due to their practical importance, subspaces and basic concepts building upon them are taught early on in any engineering or science curriculum. There exists a wide variety of subspace models and algorithms, usually developed independently by different research communities. However, more recent and complex models involving subspaces enjoy a rich geometric and algebraic structure which clearly go beyond these basic concepts. Moreover, subspaces are inherently linked to rank- or dimensionality-constraints which can quickly render problems into highly non-trivial challenges. The major goal of this thesis is to present a holistic view of subspace models and algorithms for geometric computer vision. This thesis therefore explores more advanced concepts, presents new subspace models mostly for geometric computer vision

problems, and proposes algorithms to address the resulting problem formulations. For example, tensor factorizations, convex relaxations, or Pluecker coordinates will be introduced and applied to model and algorithmically tackle complex interactions between subspaces. We expect that many new concepts and insights are not limited to applications in geometric computer vision.

1.2 Problem Statement

This section presents the main topic of interest of this thesis as a general mathematical optimization problem. Even though in the remainder of the thesis this full generality will be ignored in favour of more specific and applied problem formulations, this generality is nonetheless beneficial to get the 'big-picture' of subspace models and algorithms. We ask the reader to keep this general point of view in mind when reading the subsequent chapters which present theories, formulations, and algorithms mostly applied to structure-from-motion problems. Many concepts and ideas should also be applicable to other problems, even in fields not related to computer vision.

The problems addressed in this thesis can be formulated as a minimization problem of an objective function f subject to a rank constraint

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } \operatorname{rank}(\mathcal{A}(\mathbf{x})) \leq r, \tag{1.1}$$

where $\mathbf{x} \in \mathbb{R}^d$ denotes a vector-valued unknown and $\mathcal{A} : \mathbb{R}^d \to \mathbb{R}^{m \times n}$ is a linear operator which maps this vector to a matrix. For some of the problems considered in this thesis, the vector $\mathbf{x}$ actually corresponds directly to the column-wise vectorization of a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, i.e. $\mathbf{x} = \operatorname{vec}(\mathbf{X})$ and $d = mn$, and the linear operator $\mathcal{A}$ is simply the inverse of the vectorization operator, i.e. $\mathcal{A}$ reshapes the vector again into matrix form. However, the linear operator can also be more complex, as we will see for example in Chap. 6.

The above problem consists of only one single rank constraint. This is for example sufficient to model interactions between two subspaces as the row- and column space spanned by the matrix $\mathcal{A}(\mathbf{x})$. However, as will be presented in Chap. 4, considering the interaction between more than two subspaces requires tensors and also the notion of multirank. This then leads to optimization problems with more than one rank

constraint

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } \operatorname{rank}(\mathcal{A}_i(\mathbf{x})) \leq r_i \text{ for all } i \in [I], \tag{1.2}$$

where $[I] = \{1, \ldots, I\}$. Such a general formulation can model highly complex interactions between the coefficients of the vector $\mathbf{x}$.

The objective function f usually captures a data cost, i.e. how closely the model can explain the measured data. Depending on the application and the preferred point of view, this function can be interpreted as a log-likelihood, log-posterior, an energy, or a an algebraic cost which measures an algebraic error between the model and the data. Throughout this thesis, different point of views will be considered thereby also switching between varying interpretations of the objective function.

Let us assume for the moment that the objective function itself is easy to optimize, i.e. the difficulty is due to the presence of the rank-constraints. Rank constraints are generally very tricky to handle in optimization problems. One obvious approach to eliminate a rank constraint is to introduce additional matrix variables, say $\mathbf{A} \in \mathbb{R}^{m \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times n}$, and to impose an equality constraint $\mathcal{A}(\mathbf{x}) = \mathbf{AB}$ in place of the rank constraint. This results in a bilinear equality constraint in matrix variables $\mathbf{A}$ and $\mathbf{B}$, or as will be seen in Chap. 4 in a multilinear equality constraint in multiple matrix variables. Multilinear problems have interesting algebraic properties but are nevertheless very tricky to optimize. However, this is just one possibility to deal with rank constraints and we will see more approaches throughout this thesis. The major goal of this thesis is to present different views on such rank-constrained problems and propose several solution approaches. The choice of a suitable algorithm is always highly problem and model dependent. Subspaces models and algorithms should therefore not be treated separately and will be presented and discussed jointly and in detail throughout the subsequent chapters. Nevertheless, in the remaining sections of this chapter, a rough overview of subspace models and algorithms will be given in a slightly decoupled presentation. This not only provides an idea what to expect from the upcoming chapters but also helps in structuring the topics covered in this thesis thereby facilitating putting the material in a broader context.

1.3 Subspace Models

Subspace models can conceptually be classified into two classes, namely deterministic models and probablistic models. Deterministic models are mostly inspired by linear or multilinear algebra whereas probabilistic models evolve around probability distributions of the involved variables.

1.3.1 Deterministic Models

Deterministic models do not explicitly model measurement noise (or implicitly assume for simplicity isotropic Gaussian noise which then leads to a standard least squares problem). Hence, deterministic models emphasize the algebraic aspect and the constraints imposed by the algebraic structure. This rich algebraic structure can often be exploited algorithmically, as we will see in Chap. 4, Chap. 5, and Chap. 8.

As mentioned in the previous section, the simplest deterministic subspace models are bilinear models which model the data as the product between two unknown low-rank matrices. Multiway arrays or higher-order tensors offer more flexibility in that they allow capturing the interaction between an arbitrary number of subspaces. This leads to multilinear algebra, or even more generally to algebraic geometry which is concerned amongst others with polynomial multivariate systems of equations. Rank constraints on matrices are actually constraints on the minors of a matrix, and as such these constraints are multilinear polynomials in the entries of the matrix. We refer to Chap. 8 which builds upon results from algebraic geometry for low-rank matrices.

An instructive example for deterministic subspace models is the best rank-r approximation problem of a given matrix $\mathbf{X}$

$$\min_{\mathbf{Y}\in\mathbb{R}^{m\times n}} \|\mathbf{X}-\mathbf{Y}\| \text{ s.t. } \operatorname{rank}(\mathbf{Y}) \leq r \Leftrightarrow \min_{\mathbf{A}\in\mathbb{R}^{m\times r},\mathbf{B}\in\mathbb{R}^{r\times n}} \|\mathbf{X}-\mathbf{AB}\|, \quad (1.3)$$

where the matrix norm can be the spectral norm or the Frobenius norm. Interestingly, even this seemingly simple rank-constrained matrix factorization problem is non-linear and non-convex, but the singular value decomposition nonetheless provides an efficient and globally optimal solution. Unfortunately, in case of the Frobenius norm, a slight change where the objective function only considers a subset of the matrix entries and the complementary set of entries is unknown turns out to be a really challenging problem and the singular value decomposition can no

longer provide the solution. This problem is also known as the matrix completion problem and has recently gained a lot interest in the field of compressive sensing. Chap. 6 deals with this point of view in more detail.

1.3.2 Probabilistic Models

Probabilistic models have gained increased importance during the last few years. Especially graphical models provide a flexible algorithmic framework and an intuitive graphical representation to model even highly complex interactions between random variables. Graphical models intuitively encode the conditional independence between random variables in the topology of a graph. Each node in the graph represents a random variable and the edges between nodes encode the dependence structure between the random variables. Markov random fields and Bayesian networks are the best-known examples for graphical models. Markov random fields are based on undirected graphs, in contrast to Bayesian networks which are based on directed and acyclic graphs. Imposing probability distributions on subsets of the variables completes the statistical model. In the case of Bayesian networks, specifying conditional probabilities is particularly appealing due to the acyclic structure of the graph. In Markov random fields one usually defines a factorized probability distribution where each factor is a function of random variables appearing in a clique in the graph. Thanks to the celebrated Hammersley-Clifford theorem this leads to a valid probability distribution (given some weak conditions, e.g. such as positivity of the probability distribution). We refer to [KF09] for thorough introduction to graphical models.

Probabilistic models are very general and strictly speaking, they subsume deterministic approaches (maybe with the exception of algebraic geometry based methods). For example, the singular value decomposition based solution to the best rank-r approximation problem of Eq. (1.3) is a classical result from linear algebra and hence belongs rather to the deterministic class. However, the square of the Frobenius norm can also be interpreted as a term from a log-likelihood where each entry of the unknown matrix is an independent normally distributed random variable with mean equal to the corresponding entry of $\mathbf{X}$. The low rank constraint has an interpretation of a rather strict prior imposing an upper bound on the allowable model complexity. Analogously,

other deterministic models appearing in this thesis have a similar interpretation in probabilistic terms. However, when trying to understand the algebraic constraints imposed by a subspace model, probabilistic interpretations add an additional layer of complexity and it can be argued that in this case, a proper probabilistic interpretation is initially less important. Once the algebraic aspect of the model is understood, one can still reconsider modeling the data and noise in a probabilistic framework. This thesis puts more emphasis on algebraic aspects and how to algorithmically exploit algebraic constraints imposed by the model. We ask the reader nonetheless to keep in mind that probabilistic generalizations and interpretations are possible, generally asking for completely different algorithms, though. Especially Chap. 7 might serve as an example, how to extend and algorithmically handle an existing low-dimensional subspace model in probabilistic terms.

1.4 Subspace Algorithms

Once a subspace model is specified and several measurements are available, the goal is then usually to infer some variables of interest from the data or to learn some free parameters of the model in order to tune the model to particularities contained in the data. For general non-linear and non-convex inference or learning problems, there are dozens of different algorithms. Huge progress has been made in the last few years especially for probabilistic graphical models in the context of structured prediction: message passing, convex relaxations, graph cuts, efficient branch-and-bound, and Markov Chain Monte Carlo sampling methods are some prominent examples. We refer to [NL11] for an introduction into structured prediction with focus on computer vision applications.

Such methods are very general but unfortunately do not exploit algebraic properties of subspace models. As already mentioned, this thesis puts more emphasis on this algebraic aspect and investigates different algorithms how to exploit algebraic constraints due to the subspace model. As we will see, some methods (Chap. 6, Chap. 7) are strongly related to previously mentioned algorithms for general inference and learning problems. However, other methods (Chap. 4, Chap. 8) are based on an entirely different formulation and theory. The following sections provide a high-level overview of possible algorithmic approaches. Each of those will then be presented in detail in a separate

chapter.

1.4.1 Local Optimization

As shown in Sec. 1.2, rank constraints can be eliminated by introducing additional matrix variables. If the objective function measures the discrepancy between the model and the measured data e.g. with the Frobenius norm, this then often leads to bilinear or multilinear least squares problems. Gauss-Newton or Levenberg-Marquardt methods [NW06] are the method of choice for non-linear least square problems. Interestingly, the multilinear structure opens the door for more efficients methods, such as the Wiberg algorithm. Chap. 3 will present a detailed derivation and analysis of this Wiberg algorithm applied to low-rank matrix factorization problems.

1.4.2 Tensor Factorization

Interactions between multiple subspaces can be modeled with higher-order tensors. Tensor are a generalization of matrices and vectors which are second and first order tensors, respectively. In contrast to matrices, where the rank of the column and row space is always equal, tensors have a multirank and each so called mode can have a different rank. In principle, the non-linear least squares algorithms of Chap. 3 can also be applied to tensors but this disregards specific structure which might be contained in the data. Therefore, Chap. 4 presents how tensors can be used to model complex interactions in a camera network where cameras track feature points on a rigidly moving object. This analysis reveals how to exploit the algebraic properties contained in the data in order to come up with a tensor factorization algorithm which then in turn provides the camera poses, the rigid motion, and the 3D structure of the tracked feature points. Chap. 5 then extends this tensor model to the non-rigid case where deformations are modeled as a low-rank model.

Even though these chapters are highly specific to this multi-camera structure-from-motion problem, we expect that several of the proposed techniques and insights are of relevance for other factorization problems as well.

1.4.3 Convex Relaxation

Non-linear least squares methods, or in general local optimization methods come without guarantees to find an optimal solution. A popular approach in these circumstances is to optimize a convex relaxation of the original non-linear and non-convex problem thereby implicitly hoping that the solution to the convex relaxation will be the same or at least close to the solution of the original problem. Convex linear programming relaxations of discrete integer programming problems, such as labeling problems in computer vision (e.g. for classification, segmentation, or dense stereo) or sparse $L0$-priors (e.g. for dictionary learning), have been shown to be viable alternatives to their non-convex counterparts. A new emerging field known as compressive sensing has derived conditions when such a convex relaxation is guaranteed with high probability to yield the same solution as the non-convex problem. Originally, these results were formulated for vector valued unknowns with $L0$ priors replaced with their convex relaxation, the $L1$ norm. More recently, these results have been generalized to the matrix case where the rank function replaces the $L0$ prior and the trace norm (the sum of singular values) serves as convex relaxation. Chap. 6 proposes a structured trace norm which enables the incorporation of weak prior knowledge, e.g. in a structure-from-motion application one might assume that trajectories move smoothly. This structured trace norm is then used to relax a rank-4 constraint in a structure-from-motion problem.

1.4.4 Sampling

If neither local optimization, nor exploiting problem structure, nor convex relaxations give satisfying results, a last resort is provided by sampling methods (also called the sledge hammer for optimization or inference problems). Sampling methods are quite popular for learning and inference in hierarchical graphical models [Hof11; SM08]. Sampling methods are purely based on probabilistic models. The basic idea is to randomly sample from the target probability distribution and then use these samples for estimating statistics of the variables of interests (e.g. the mean, mode, covariance, ...). Sampling low-dimensional random variables with a certain probability distributions is a well established field. However, sampling high-dimensional random variables suffers from the curse of dimensionality. Even though there are standard

building blocks to sample from high-dimensional spaces, such as Markov Chain Monte Carlo methods, Metropolis-Hastings, and Gibbs-sampling, it is still an art and highly challenging to design efficient sampling algorithms for a specific problem. This is mainly due to the fact that sampling methods often require the specification of a scale parameter which encodes the size of the local neighborhood of interest to sample from. This is tricky in high-dimensional spaces since there might not be a single scale value which is appropriate for the whole space. Adaptive methods try to automatically adjust this scale parameter. Chap. 7 presents an adaptive Markov Chain Monte Carlo method for structure-from-motion problems.

1.4.5 Algebraic Geometry

In strong contrast to sampling methods and on the complete opposite end of the spectrum, algebraic geometry based methods are purely algebraic and have no connection to probabilistic models. As mentioned in Sec. 1.3.1, low rank constraints are actually constraints on the minors of a matrix and hence one single low rank constraints corresponds to a set of multivariate and multilinear polynomials. Algebraic geometry has not only developed deep theories, but also practical algorithms which can handle polynomials system of low order and with few variables [CLO97]. Even more specifically, the minors of a matrix are known as the Plücker coordinates (also known as Grassmann coordinates) and Plücker coordinates possess an extraordinary rich algebraic structure (see Chap. 14-17 in [MS05]). This structure will be exploited in Chap. 8 where we ask ourselves the question how to compute an unknown subspace from a set of given subspaces each of which is known to intersect this unknown subspace. A concrete example is given by four lines in $3D$ which are all known to intersect an unknown line in 4 unknown points. Chap. 8 also presents how a structure-from-sound problem can be formulated and solved in this Plücker coordinate framework.

2 Preliminaries

2.1 Notation

The following notation will be used throughout this thesis. The set of integers from 1 to n will be denoted as $[n] = \{1, \ldots, n\}$. Matrices are written with bold upper case letters $\mathbf{A}$ whereas vectors are bold lower case $\mathbf{a}$. We use calligraphic letters $\mathcal{A}$ for tensors, subspaces, or linear operators. The meaning should be clear from the context. Matrices built up from multiple submatrices are enclosed whenever possible by square brackets $[\cdots]$, vectors built from multiple subvectors by round brackets $(\vdots)$. The identity matrix of dimension $D \times D$ is denoted as $\mathbf{I}_D$. The $m \times n$ matrix where each entry equals 1 is written as $\mathbf{1}_{m \times n}$. The size of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is sometimes indicated as subscript $\mathbf{A}_{m \times n}$ in order to ease the parsing of formulas and to improve readability. The Moore-Penrose pseudo-inverse of a matrix $\mathbf{A}$ is written with the standard symbol as $\mathbf{A}^\dagger$. Concatenation of multiple matrices indexed with a sub- or supscript i is represented with arrows. For example, $[\Downarrow_i \mathbf{A}_i]$ concatenates all the matrices $\mathbf{A}_i$ below each other, implicitly assuming that each of them consists of the same number of columns. The operator $\searrow_i$ stacks multiple matrices $\mathbf{A}_i$ into a block-diagonal matrix $[\searrow_i \mathbf{A}_i]$. Sometimes, the limits of the running index is indicated, e.g. as $[\Rightarrow_{i=1}^{n} \mathbf{A}_i]$.

The span of the columns of a matrix $\mathbf{A}$ is written as $\operatorname{span}(\mathbf{A})$. A matrix whose columns are vectors of a basis for the orthogonal complement of $\operatorname{span}(\mathbf{A})$ is denoted by $[\mathbf{A}]_\perp$, i.e. it holds that $\mathbf{A}^T [\mathbf{A}]_\perp = \mathbf{0}$. The orthogonal projection matrix onto the column space of a matrix $\mathbf{A}$ is denoted as $\mathbb{P}_{\mathbf{A}}$. The projection matrix onto the orthogonal complement of the column space of $\mathbf{A}$ is $\mathbb{P}_{\mathbf{A}}^\perp = \mathbf{I} - \mathbb{P}_{\mathbf{A}}$. The Matlab® standard indexing notation is used for the slicing operation (cutting out certain rows and columns of a matrix), so $\mathbf{A}_{[i:j,k:l]}$ corresponds to the submatrix of $\mathbf{A}$ which is given by selecting rows i to j and columns k to l.

The cross product between two three-vectors can be formulated as a

matrix-vector product

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \mathbf{b} = [\mathbf{a}]_\times \mathbf{b}$$

where $[\mathbf{a}]_\times$ denotes the skew-symmetric cross-product matrix built from the indices of vector $\mathbf{a}$. Additional notations specific to one chapter will be introduced in the corresponding chapter. The notational conventions used in this thesis are also summarized in Tab. 2.1.

Symbol	Meaning
$\mathbf{a}$	Vectors.
$\mathbf{A}$	Matrices.
$\mathcal{A}$	Tensors, subspaces, or linear operators, depending on context.
$\mathrm{vec}(\mathbf{A})$	Column-wise vectorization, in Matlab notation `A(:)`.
$\mathrm{vecs}(\mathbf{A})$	Vectorization of lower triangular part of a matrix.
$\mathcal{A} = \mathcal{S} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$	Three-mode Tucker-tensor decomposition with core tensor $\mathcal{S}$.
$\mathcal{A}_{(i)}$	Flattening of tensor $\mathcal{A}$ along mode i.
$[\Downarrow_i \mathbf{A}_i]$	Vertical stacking of matrices $\mathbf{A}_i \in \mathbb{R}^{m_i \times n}$ below each other. Sometimes the range of the index i will be indicated for clarity reasons in the following way $\left[\Downarrow_{i=1}^{I} \mathbf{A}_i\right]$.
$[\Rightarrow_i \mathbf{A}_i]$	Horizontal stacking of matrices $\mathbf{A}_i \in \mathbb{R}^{m \times n_i}$ next to each other.
$[\searrow_i \mathbf{A}_i]$	Block-diagonal stacking of matrices $\mathbf{A}_i \in \mathbb{R}^{m_i \times n_i}$.
$\mathbf{A} \otimes \mathbf{B}$	Kronecker product.
$\mathbf{A} \odot \mathbf{B}$	Khatri-Rao or Hadamard product, depending on context.
$\mathbf{A}_{m \times n}$	The size of a matrix is sometimes indicated in subscripts.

Continued on next page

Table 2.1 – *Continued from previous page*

Symbol	Meaning
$\mathbf{I}_m$	Identity matrix of dimension m
$\mathbf{1}_{m\times n}$	$m \times n$ matrix consisting entirely of 1-entries.
$[\mathbf{a}]_\times$	skew-symmetric matrix for matrix-based cross-product notation, i.e. for $\mathbf{a}, \mathbf{b} \in \mathbb{R}^4 : \mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b}$.
$\mathbb{S}^n$	Set of symmetric matrices of size $n \times n$.
$\mathbb{S}^n_+$	Set of positive semi-definitve matrices of size $n \times n$.
$\operatorname{span}(\mathbf{A})$	Subspace spanned by the columns of matrix $\mathbf{A}$.
$[\mathbf{A}]_\perp$	Columns of $[\mathbf{A}]_\perp$ are an orthonormal basis for nullspace of matrix $\mathbf{A}$.
$\mathbb{P}_\mathbf{A}$	Orthogonal projection matrix onto column space of $\mathbf{A}$.
$\mathbb{P}^\perp_\mathbf{A} = \mathbf{I} - \mathbb{P}_\mathbf{A}$	Orthogonal projector onto orthogonal complement of $\operatorname{span}(\mathbf{A})$.
$\mathbf{D}_n$	Duplication matrix of dimension n, i.e. for $\mathbf{A} \in \mathbb{S}^n : \mathbf{D}_n \operatorname{vecs}(\mathbf{A}) = \operatorname{vec}(\mathbf{A})$.
$\mathbf{T}_{m,n}$	Commutator matrix for matrices of size $m \times n$, i.e. $\mathbf{T}_{m,n} \operatorname{vec}(\mathbf{A}) = \operatorname{vec}(\mathbf{A}^T)$.
$[n]$	Set of integers from 1 to n, i.e. $[n] = \{1, \ldots, n\}$.
$k \in [K]$	Index of a camera.
$f \in [F]$	Index of a frame.
$n \in [N]$	Index of a point.

Table 2.1: Notations used throughout this thesis

2.2 Multilinear Algebra

Concepts from tensor calculus will be introduced in this section. More specifically the mode-i product and the Tucker tensor decomposition [Tuc66] are defined and several relationships between tensors, the Kronecker product $\otimes$, and the $\mathrm{vec}(\cdot)$-operator are stated. We refer to [LMV00; MN99; KB09] for an introductory text on multilinear algebra, tensor operations and decomposition.

2.2.1 Tensors and the Tucker Tensor Decomposition

Tensors express multilinear relationships between variables and are thus a generalization of entities used in linear algebra, i.e., vectors (1^{st}-order tensors) and matrices (2^{nd}-order tensors). A tensor of order n can be thought of as a n-dimensional array of numbers. Varying the i^{th} index of a tensor while keeping the remaining indices fixed defines the mode-i vectors. An important tool for the analysis and usage of tensors is the mode-i product. The mode-i product $\mathcal{B} = \mathcal{A} \times_i \mathbf{M}$ is a tensor-valued bivariate function of a n^{th}-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$ and a J_i-by-I_i matrix $\mathbf{M}$. The resulting tensor $\mathcal{B}$ is given by left-multiplying all the mode-i vectors by the matrix $\mathbf{M}$. The tensor $\mathcal{B}$ is still of order n, the dimension of the mode-i vectors however changed from I_i to J_i. An efficient and easy way to compute such a mode-i product is to flatten the tensor $\mathcal{A}$ along its i^{th}-mode (which means stacking all the mode-i vectors of $\mathcal{A}$ into one big matrix $\mathcal{A}_{(i)} \in \mathbb{R}^{I_i \times \prod_{j \neq i}^{n} I_j}$) and to left-multiply by the matrix $\mathbf{M}$. This provides the resulting flattened version of $\mathcal{B}_{(i)} = \mathbf{M}\mathcal{A}_{(i)}$. A straight forward reordering of the elements of this flattened tensor leads to the tensor $\mathcal{B}$. Note that the order in which the mode-i vectors are put next to each other is unimportant as long as the reshaping of $\mathcal{B}_{(i)}$ into a tensor $\mathcal{B}$ is performed consistently. Interestingly, the order in which the mode-i products are applied to a tensor does not matter, as long as they are applied along different modes. So for example we have $(\mathcal{A} \times_1 \mathbf{U}_1) \times_2 \mathbf{U}_2 = (\mathcal{A} \times_2 \mathbf{U}_2) \times_1 \mathbf{U}_1$, and thus we simply write $\mathcal{A} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2$.

Given that the measured data can be modeled as a tensor, various algorithms exist to analyze the underlying algebraic structure of the process which generated the measured data and also to decompose the data tensor into more meaningful parts. The most prominent two tensor decompositions are the canonical decomposition (aka. parallel factor

model PARAFAC) [CC70; Har70] and the Tucker decomposition [Tuc66]. A more recent decomposition [LMV00], the higher order singular value decomposition, extends the Tucker decomposition by imposing certain orthogonality constraints. However, the Tucker decomposition without orthogonality constraints is the most suitable tensor decomposition for our purposes because it reveals the underlying mode-i subspaces without imposing unnecessary orthogonality constraints on them. The mode-i subspace is the span of all the mode-i vectors of a tensor. The Tucker decomposition of a n^{th}-order tensor $\mathcal{A}$ expresses the tensor as n mode-i products between a smaller core tensor $\mathcal{S} \in \mathbb{R}^{r_1 \times \cdots r_n}$ and n matrices $\mathbf{M}_i \in \mathbb{R}^{I_i \times r_i}$

$$\mathcal{A} = \mathcal{S} \times_1 \mathbf{M}_1 \times_2 \mathbf{M}_2 \times_3 \cdots \times_n \mathbf{M}_n, \tag{2.1}$$

where the columns of $\mathbf{M}_i$ represent a basis for the mode-i subspace. If for all $r_i < I_i$, the Tucker decomposition provides a dimensionality reduction since the number of parameters decreases when using a smaller core tensor. This representation is exact if each mode-i subspace is indeed only of dimension $r_i < I_i$, otherwise the Tucker decomposition provides a suitable low-dimensional approximation to the original data tensor [LMV00]. Unfortunately, the Tucker decomposition is known to be non-unique since a basis transformation applied to the mode-i subspace can be compensated by the mode-i product of the core tensor with the inverse of this linear transformation $\mathcal{S} \times_i \mathbf{M} = (\mathcal{S} \times_i \mathbf{Q}^{-1}) \times_i [\mathbf{M}_i \mathbf{Q}]$. This fact will become important in Sec. 4.4.

2.2.2 The Kronecker Product and the vec()-operator

The Kronecker product $\otimes$ is closely related to the tensor product, it is not by accident that both products share the very same symbol. The Kronecker product is a matrix-valued bilinear product of two matrices and generalizes the bilinear outer product of vectors $\mathbf{a}\mathbf{b}^T$ to matrices. Throughout this section, let $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$. Then $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{mp \times nq}$ is a block structured matrix where the $(i,j)^{\text{th}}$ block equals the matrix $\mathbf{B}$ scaled by the $(i,j)^{\text{th}}$ element of $\mathbf{A}$. This implies for example that the first column of $\mathbf{A} \otimes \mathbf{B}$ equals the vectorized outer product $\text{vec}\left(\mathbf{B}_{:,1}\mathbf{A}_{:,1}^T\right)$ of the first column of $\mathbf{A}$ and $\mathbf{B}$. Here, the vectorization operator $\text{vec}(\mathbf{A})$ has been used which is defined in matrix calculus as the vector which results by stacking all the columns of matrix $\mathbf{A}$ into a column vector. We also define the cummutation

matrix $\mathbf{T}_{m,n} \in \mathbb{R}^{mn \times mn}$ such that $\operatorname{vec}\left(\mathbf{A}^T\right) = \mathbf{T}_{m,n} \operatorname{vec}(\mathbf{A})$. Note that $\mathbf{T}_{m,n}$ is a permutation matrix. For square matrices $\mathbf{A} \in \mathbb{R}^{n \times n}$, the operator $\operatorname{vecs}(\mathbf{A})$ vectorizes the matrix and eliminates all supradiagonal elements (e.g. for $\mathbf{A} \in \mathbb{R}^{2\times 2}$, $\operatorname{vecs}(\mathbf{A}) = (\mathbf{A}_{1,1}, \mathbf{A}_{2,1}, \mathbf{A}_{2,2})^T \in \mathbb{R}^{3\times 1}$). This is especially useful for symmetric matrices since in this way, $\operatorname{vecs}(\mathbf{A})$ contains all the unique elements of symmetric $\mathbf{A}$. The duplication matrix $\mathbf{D}_n \in \mathbb{R}^{n^2 \times \frac{1}{2}n(n+1)}$ reassembles the vectorization of the symmetric matrix from this operator, i.e.

$$\forall \mathbf{A} \in \mathbb{S}^n : \mathbf{D}_n \operatorname{vecs}(\mathbf{A}) = \operatorname{vec}(\mathbf{A}). \tag{2.2}$$

We refer to Chap. 3 in [MN99] for more details and properties of these operators.

The Kronecker product is helpful in rewriting matrix equations of the form $\mathbf{AXB}^T = \mathbf{C}$ which is equivalent to

$$\operatorname{vec}(\mathbf{C}) = \operatorname{vec}\left(\mathbf{AXB}^T\right) = [\mathbf{B} \otimes \mathbf{A}] \operatorname{vec}(\mathbf{X}). \tag{2.3}$$

If the number of rows and columns of the matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are such that $\mathbf{AC}$ and $\mathbf{BD}$ can be formed, then the mixed-product property of the Kronecker product states that

$$[\mathbf{A} \otimes \mathbf{B}][\mathbf{C} \otimes \mathbf{D}] = [\mathbf{AC}] \otimes [\mathbf{BD}]. \tag{2.4}$$

Closer inspection of Eq. (2.1) reveals a first link between the Kronecker product and tensors: the tensor $\mathcal{A}$ is actually a sum of n-fold outer products $\mathbf{M}_{1,(:,i_1)} \otimes \cdots \otimes \mathbf{M}_{n,(:,i_n)}$ (these products are also known as *decomposable*, *simple*, or *pure* tensors) weighted by the corresponding entry of the core tensor

$$\mathcal{A} = \sum_{1 \le i_1 \le r_i, \cdots, 1 \le i_n \le r_n} \mathcal{S}_{i_1,\cdots,i_n} \mathbf{M}_{1,(:,i_1)} \otimes \cdots \otimes \mathbf{M}_{n,(:,i_n)}. \tag{2.5}$$

The Kronecker product provides a link between the Tucker decomposition and ordinary matrix multiplication. Specifically, given a Tucker decomposition $\mathcal{A} = \mathcal{S} \times_1 \mathbf{M}_1 \times_2 \cdots \times_n \mathbf{M}_n$, the flattened tensor $\mathcal{A}_{(i)}$ along mode i is then given by

$$\mathcal{A}_{(i)} = \mathbf{M}_i \mathcal{S}_{(i)} \left[\mathbf{M}_1 \otimes \cdots \otimes \mathbf{M}_{i-1} \otimes \mathbf{M}_{i+1} \otimes \cdots \otimes \mathbf{M}_n\right]^T,$$

or equivalently $\operatorname{vec}(\mathcal{A}) = [\mathbf{M}_1 \otimes \ldots \otimes \mathbf{M}_n] \operatorname{vec}(\mathcal{S})$ (assuming a consistent vectorization of the tensor entries). The previous equations clearly

show the relation to Eq. (2.3): The core tensor generalizes the matrix $\mathbf{X}$ in Eq. (2.3) by capturing interactions between more than just two subspaces (induced by the column and row span of matrix $\mathbf{C}$ in Eq. (2.3)).

A slight variation of the Kronecker product is the Khatri-Rao product $\mathbf{A} \odot \mathbf{B} \in \mathbb{R}^{mn \times p}$ which is defined for two matrices $\mathbf{A} \in \mathbb{R}^{m \times p}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$ with the same number of columns. Specifically, the Khatri-Rao product takes the columnwise Kronecker product between corresponding pairs of columns $\mathbf{A} \odot \mathbf{B} = [\Rightarrow_i \mathbf{A}_{:,i} \otimes \mathbf{B}_{:,i}]$. The Khatri-Rao product also enjoys a product property for matrices of appropriate dimensions $[\mathbf{C} \otimes \mathbf{D}]\,[\mathbf{A} \odot \mathbf{B}] = [\mathbf{CA} \odot \mathbf{DB}]$. The same symbol $\odot$ is commonly also be used for the Hadamard product between two matrices of equal size $\mathbf{C} = \mathbf{A} \odot \mathbf{B}$. The Hadamard product is the entry-wise product between two matrices, i.e. $\mathbf{C}_{i,j} = \mathbf{A}_{i,j}\mathbf{B}_{i,j}$. Whether $\odot$ denotes the Hadamard or the Khatri-Rao product should be clear from the context.

2.2.3 Solving Multilinear Matrix Equations

Some chapters of this thesis are heavily based on multilinear matrix and tensor notations. As we will see, this representations facilitates reasoning about specific problem instances. Eventually however, often a linear least squares problem has to be solved for the unknowns or the Jacobian with respect to a matrix unknown has to be computed. Linear systems in standard form $\mathbf{Ax} = \mathbf{b}$ can be readily solved with any least-squares method of choice (e.g. with QR-decomposition or singular-value decomposition) and analytical Jacobians allow for more efficient implementation of iterative methods. Thus, there is a need to do matrix calculus, but unfortunately there is no clear consensus on how to do calculus with matrix unknowns. We stick with the concepts introduced in [MN99] and refer the interested reader to this reference for details which go beyond the following brief introduction.

Knowing how to rewrite the three following instances of matrix equations allows to express all the matrix equations mentioned in this thesis in standard form (these identities will become especially handy in App. 4.A.1 and App. 4.A.2).

i) **The Matrix Equation $\mathbf{AXB} = \mathbf{C}$:** The Jacobian of $\mathbf{AXB}$ w.r.t. $\mathbf{x} = \mathrm{vec}(\mathbf{X})$ is $\mathbf{J}_\mathbf{x} = \mathbf{B}^T \otimes \mathbf{A}$ which leads to the linear system in standard form $\mathbf{J}_\mathbf{x}\mathbf{x} = \mathrm{vec}(\mathbf{C})$.

ii) **The equation** $\mathrm{vec}(\mathbf{X} \otimes \mathbf{Y})$**:** Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $\mathbf{Y} \in \mathbb{R}^{p \times q}$. Then the following identities hold:

$$\begin{aligned} \mathrm{vec}(\mathbf{X} \otimes \mathbf{Y}) = & \quad [\mathbf{I}_n \otimes \mathbf{T}_{q,m} \otimes \mathbf{I}_p] (\mathrm{vec}(\mathbf{X}) \otimes \mathrm{vec}(\mathbf{Y})) \\ = & \quad [\mathbf{I}_n \otimes \mathbf{T}_{q,m} \otimes \mathbf{I}_p] \cdot \\ & \quad [\mathbf{I}_{mn} \otimes \mathrm{vec}(\mathbf{Y})] \mathrm{vec}(\mathbf{X}) \qquad (2.6) \\ = & \quad [\mathbf{I}_n \otimes \mathbf{T}_{q,m} \otimes \mathbf{I}_p] \cdot \\ & \quad [\mathrm{vec}(\mathbf{X}) \otimes \mathbf{I}_{pq}] \mathrm{vec}(\mathbf{Y}) \qquad (2.7) \end{aligned}$$

iii) **The Matrix Equation** $\mathbf{X} \otimes \mathbf{Y} = \mathbf{C}$**:** Let $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $\mathbf{Y} \in \mathbb{R}^{p \times q}$. Using the previous identity, we see that the Jacobian w.r.t. the vectorized unknowns $\mathbf{x} = \mathrm{vec}(\mathbf{X})$ and $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$ is

$$\begin{aligned} \mathbf{J}_{\mathbf{x},\mathbf{y}} = & [\mathbf{I}_n \otimes \mathbf{T}_{q,m} \otimes \mathbf{I}_p] \cdot \\ & [\mathbf{I}_{mn} \otimes \mathrm{vec}(\mathbf{Y}), \mathrm{vec}(\mathbf{X}) \otimes \mathbf{I}_{pq}]. \qquad (2.8) \end{aligned}$$

The bilinear matrix equation $\mathbf{X} \otimes \mathbf{Y} = \mathbf{C}$ is thus equivalent to

$$\mathbf{J}_{\mathbf{x},\mathbf{y}} \begin{pmatrix} \mathrm{vec}(\mathbf{X}) \\ \mathrm{vec}(\mathbf{Y}) \end{pmatrix} = \mathrm{vec}(\mathbf{C}). \qquad (2.9)$$

2.3 Projective and Affine Coordinate Systems

The following sections will present a short introduction to projective geometry and camera matrices. It is by no means meant as a self-contained introduction: the main purpose is to derive and motivate the affine camera model which will be used in several places in this thesis. For a detailed introduction and more details about structure-from-motion we refer to the standard textbook [HZ04].

2.3.1 Projective Geometry

Points in projective space $\mathbb{P}^n$ are represented in homogeneous coordinates throughout this thesis. This means that a point corresponds to an equivalence class of non-zero vectors in $\mathbb{R}^{n+1}$. The space $\mathbb{R}^{n+1}$ is partitioned into equivalence classes by defining the equivalence relation $\mathbf{X} \cong \mathbf{Y}$ whenever $\mathbf{X} = \alpha\mathbf{Y}$ for some non-zero $\alpha \in \mathbb{R}$. In other words, the representation of a point in $\mathbb{P}^n$ as a non-zero vector $\mathbf{X} \in \mathbb{R}^{n+1}$ is unique up to an arbitrary non-zero scale factor $\alpha \in \mathbb{R}$. Equivalently, a point in

$\mathbb{P}^n$ equals a one-dimensional subspace of $\mathbb{R}^{n+1}$ and as such, a point is an element of the Grassmannian $G_{1,n+1}$ and its coordinates correspond to the Grassmann coordinates of this one-dimensional subspace. A reader unfamiliar with these terms can safely ignore this interpretation: these terms will become clear in Chap. 8. For notational simplicity, the equivalence relation $\cong$ will usually be written with the standard equality sign $=$ and the interpretation as an equality up to a scalar multiple should be clear from the context.

The choice of the coordinate system in $\mathbb{R}^{n+1}$ is arbitrary and a change of basis leads to a projective transformation of the points $\mathbf{X}' \cong \mathbf{H}\mathbf{X}$ with a regular matrix $\mathbf{H} \in \mathbb{R}^{n+1 \times n+1}$ which is also defined only up to scale, i.e. $\mathbf{H}$ has only $(n+1)^2 - 1$ degrees of freedom.

An advantage of projective geometry is that points at infinity are not distinguished from other points, i.e. points infinitely far away are treated in exactly the same way as finite points. Unfortunately, even though projective coordinates are mathematically very convenient, 3D reconstructions in a general projective coordinate frame can not be properly visualized. It is therefore necessary to upgrade a 3D reconstruction from a projective to an affine or even better to an Euclidean reference frame. This requires the identification of the set of points at infinity. This set turns out to be a plane, namely the so-called plane at infinity. In order to upgrade a reconstruction to an affine coordinate reference system, the coordinate system is chosen such that the plane at infinity is mapped to its canonical coordinate representation $\mathbf{p}_\infty = (0, 0, 0, 1)$. This in turn maps all the points at infinity to points whose last coordinate equals 0. Note that the plane at infinity stays fixed at $\mathbf{p}_\infty = (0, 0, 0, 1)$ (not point-wise, though) under transformations $\mathbf{H}$ of the form

$$\mathbf{H} \cong \begin{bmatrix} \hat{\mathbf{H}} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}, \tag{2.10}$$

with $\hat{\mathbf{H}} \in \mathbb{R}^{3\times 3}$ and $\mathbf{t} \in \mathbb{R}^{3\times 1}$. Transformations of this form are known as affine transformations. Singling out the plane at infinity also enables reasoning about translations $\mathbf{t}$ in coordinate transformations or camera matrices. Indeed, points at infinity are not affected by a (finite) translation of the coordinate system.

Identification of a special conic, known as the Absolute Conic, on the plane at infinity allows to upgrade from an affine to an Euclidean coordinate system. The Absolute Conic equals the set of circular points.

Given circles on an arbitrary plane, they all intersect the plane at infinity in the very same two points, namely the two circular points induced by the chosen plane. This can be understood intuitively by the following reasoning. Circles are conics (quadratic equations) and as such can be represented by a symmetric matrix $\mathbf{Q} \in \mathbb{S}^3$. General conics have 5 degrees of freedom ($\mathbf{Q}$ is only defined up to scale) but a circle is completely specified by 3 points lying on the circle. However, once the Absolute Conic is identified, the two circular points induced by the plane on which the circle lies are known as well. Since these two circular points also lie on the circle, we have 5 points in total which completely specifies the conic of a circle. In summary, only once the Absolute Conic has been identified we can distinguish between circles and general ellipses.

2.3.2 Camera Models

A widely used approximation of a real camera is the pinhole camera model which assumes an infinitesimally small lens (or aperture). In this case, all the rays reaching the image plane of a camera (the sensor in a real digital camera) need to pass through one single point, namely the pinhole which equals the infinitesimally small lens. This pinhole camera model leads to central projections.

The camera image plane is modeled as projective two-space $\mathbb{P}^2$. Central projections can then be modeled mathematically as a mapping from $\mathbb{P}^3$ to $\mathbb{P}^2$. This mapping is given by a full-rank 3×4 matrix $\mathbf{P} \in \mathbb{R}^{3\times4}$ which is known as the camera matrix. The right nullspace $\mathbf{C}$ of a camera matrix $\mathbf{P}$, i.e. $\mathbf{PC} = \mathbf{0}_{3\times1}$, is known as the camera center.

Camera matrices can be classified in several classes. Here, we are mostly interested in the class of *affine cameras* which is a subset of the *cameras at infinity*. As the name suggests, cameras at infinity are cameras whose center is on the plane at infinity. Note that it only makes sense to talk about these kind of cameras if the plane at infinity has been identified. If the plane at infinity is mapped to its canonical position $\mathbf{p}_\infty = (0, 0, 0, 1)$, cameras at infinity are defined through the property that the left-most 3×3 block of $\mathbf{P}$ is singular.

Affine cameras are a specialization of general cameras at infinity whose last row of the camera matrix equals $(0, 0, 0, 1)$, again assuming the plane at infinity at its canonical position. Therefore, points on the plane at infinity will be mapped to points on the line at infinity. The

line at infinity is the intersection between the camera image plane and the plane at infinity. The last row of an affine camera is often neglected in factorization-based structure-from-motion formulations. In this case, an affine camera is modeled with a 2×4 matrix.

The affine camera is an idealized model since the camera center can never be at infinity in practice. The affine camera model therefore provides an approximation to the projective camera model. The accuracy of this approximation depends on two factors. Firstly, the smaller the ratio between the depth variation of the scene and the distance of the camera to the scene, the better the approximation. Secondly, the smaller the distance between an image point and the principal point, the more accurate the affine camera model becomes for this point. More details and properties of affine cameras can be found in Sec.6.3 of [HZ04] and in Sec. 3.8.2.

3 Local Optimization

As mentioned previously, a rank-constraint $\mathcal{A}(\mathbf{x}) \leq r$ in an optimization problem can be eliminated by introducing the explicit parametrization $\mathcal{A}(\mathbf{x}) = \mathbf{U}\mathbf{V}$ with $\mathbf{U} \in \mathbb{R}^{m \times r}$ and $\mathbf{V} \in \mathbb{R}^{r \times n}$. This is a bilinear equality constraint and requires non-linear optimization algorithms. Hence, a straight-forward approach to optimize low-dimensional subspace models is to eliminate all the rank-constraints in this way and to apply off-the-shelf non-linear optimization algorithms. This chapter follows along these lines and investigates the performance of different non-linear least squares algorithms for a simple bilinear matrix factorization problem. More specifically, well-known algorithms such as alternating least squares (ALS), Gauss-Newton, and Levenberg-Marquardt will be shortly introduced. We refer the interested reader to [NW06] for a more detailed introduction to non-linear optimization which explains these standard algorithms, further methods such as conjugate gradients, and a thorough presentation of related issues such as line search or stopping conditions.

Special emphasize will be put in this chapter on the lesser well-known Wiberg algorithm. The Wiberg algorithm is applicable to general non-linear least-squares problems. The results in this chapter will show that the Wiberg algorithm is especially well suited for bilinear or multilinear least squares low-rank matrix factorization problems of the form $\min_{\mathbf{U},\mathbf{V}} \frac{1}{2} \|\mathbf{M} \odot (\mathbf{U}\mathbf{V} - \mathbf{W})\|_F^2$, where $\mathbf{M} \in \{0,1\}^{m \times n}$ is a binary matrix masking unobserved or missing entries in the observation matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, and $\mathbf{U} \in \mathbb{R}^{m \times r}$ and $\mathbf{V} \in \mathbb{R}^{r \times n}$. Simultaneously while we were working on this comparison, [OYD11] found exactly the same behaviour: the Wiberg algorithm performs surprisingly well for bilinear matrix factorization problems. For a comparison between various second-order methods for matrix factorizations except the Wiberg algorithm, we refer to [BF05].

3.1 Non-Linear Least Squares Problems

In non-linear least squares problems, the objective function corresponds to the square of the Euclidean length of the residuum vector

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_2^2 , \tag{3.1}$$

where $\mathbf{r}(\mathbf{x})$ is the residuum vector whose entries usually measure the discrepancy between the measurements and the model with parameters $\mathbf{x}$. In linear least squares problem, the residuum depends linearly on the unknowns, i.e. $\mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{b}$.

Two comments are appropriate at this point. Firstly, the entries in the residuum vector should be on the same order of magnitude. In probabilistic terms, instead of using the Euclidean norm, the length of the residuum vector should be measured with the Mahalanobis distance induced by the covariance matrix of the measurement noise. Secondly, the scaling of the unknowns can have a huge influence on the performance. For example, if one unknown corresponds to an angle measured in radians and another unknown is the distance between two points measured in meters and the distance becomes relatively large, then the convergence behaviour might suffer. This is not an issue for pure Newton's method (because Newton's method is invariant w.r.t. linear transformations of the unknowns) but it will be important for almost all other methods (e.g. Levenberg-Marquardt which is a mix between Gauss-Newton and gradient descent).

These two issues are strongly related to preconditioning for linear systems of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$. Specifically, left preconditioning solves the system $\mathbf{T}(\mathbf{A}\mathbf{x} - \mathbf{b}) = \mathbf{0}$ where the matrix $\mathbf{T}$ is chosen 'sufficiently simple' but still such that the condition number of $\mathbf{T}\mathbf{A}$ is much smaller than the condition number from $\mathbf{A}$. Right preconditioning solves $\mathbf{A}\mathbf{T}\mathbf{y} = \mathbf{b}$ first for $\mathbf{y}$ and then computes $\mathbf{x} = \mathbf{T}\mathbf{y}$. Having said this, in the upcoming chapters, these preconditioning steps will not be mentioned explicitly anymore when dealing with a linear least squares problem or an overdetermined linear system.

3.2 Alternating Least Squares

Alternating least squares is a cyclic block-coordinates descent approach for minimizing multilinear least squares problems. If all but one variables

are fixed in a multilinear problem, the problem becomes a linear least squares problem in the remaining variable and can be solved globally optimal for this variable. As the name suggests, alternating least squares alternately optimizes for one variable while holding the remaining ones fixed. Obviously, even though each alternation step yields the optimal linear least squares solution for one variable, repeated application of this step for different variables comes without any guarantee at all to converge in a global optima of the full problem. A detailed derivation of the linear least squares subproblems of ALS for a trilinear tensor factorization problem is presented in Sec. 4.9. A generalization to arbitrary multilinear least squares problems is straight forward and a further description is therefore omitted at this place.

Due to its simplicity and straight-forward implementation, ALS is the workhorse for matrix and tensor decompositions [KB09]. ALS can work quite well, as for example shown in [HS04] for affine SfM problems. However, as we will see shortly, for matrix completion problems (to which affine SfM problems belong to), the success of ALS largely depends on the pattern of missing entries (and obviously on the initialization).

3.3 Newton's Method

When minimizing a differentiable function, a necessary (but not sufficient) condition is that the gradient has to vanish. Considering the gradient as a multivariate function, one therefore needs to find a zero crossing. Newton's method can be used for finding a zero crossing of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}^m$. Newton's methods is an iterative approach, where at each iteration a linear approximation of the function at the current point is computed and this linear approximation is then used in order to compute an update. Formally, the linear approximation is given by $f(\mathbf{x}_0 + d\mathbf{x}) \approx \mathbf{x}_0 + \mathbf{J}_{\mathbf{x}_0} d\mathbf{x}$, where $\mathbf{J}_{\mathbf{x}_0} \in \mathbb{R}^{m \times n}$ is the Jacobian of f evaluated at point $\mathbf{x}_0$. This approximation is set equal to zero which results in a linear system of equations for the update step (if $m > n$ this system might be overdetermined and is then solved in the linear least squares sense)

$$-\mathbf{J}_{\mathbf{x}_0} d\mathbf{x} = \mathbf{x}_0. \tag{3.2}$$

After solving this linear system, the current point is then updated according to $\mathbf{x}_0 \leftarrow \mathbf{x}_0 + d\mathbf{x}$.

When minimizing a function $f : \mathbb{R}^n \to \mathbb{R}$, Newton's method needs to be applied to the gradient ∇f of f. Hence, the resulting linear system for the update step becomes

$$-\mathbf{H}_{\mathbf{x}_0} d\mathbf{x} = \nabla_{\mathbf{x}_0} f(\mathbf{x}_0), \tag{3.3}$$

where $\mathbf{H}_{\mathbf{x}_0} \in \mathbb{R}^{n \times n}$ is the Hessian matrix of f evaluated at the point $\mathbf{x}_0$. Note that the same equation for the update step can be derived with a second-order Taylor expansion of the objective function.

In general, even though there is a natural step length of $\alpha = 1$ associated with the Newton method, Eq. (3.3) is usually only used to compute the step direction $d\mathbf{x}$ and a subsequent one dimensional line search $\mathbf{x} + \alpha d\mathbf{x}$ is performed over $\alpha \in \mathbb{R}_+$. As already noted in [HS04], for low-rank matrix factorization problems the line search problem reduces to a fourth-degree polynomial in α. This enables an efficient line search where only the roots of one degree four polynomial need to be computed.

Note that the Newton direction defined through Eq. (3.3) is only guaranteed to be a descent direction if the Hessian matrix is positive definite. Otherwise, additional steps need to be taken to ensure a valid descent direction. The Levenberg-Marquardt algorithm presented in Sec. 3.5 is particularly well suited for non-linear least squares problems, but there are other approaches and we refer again to [NW06] for more details.

3.4 Gauss-Newton

The Gauss-Newton methods is a slightly adapted version of Newton's method specifically tailored for small residuum non-linear least squares problems. In order to apply Newton's method to least squares problems, we need the gradient and the Hessian matrix. The gradient of Eq. (3.1) equals $\mathbf{J}^T\mathbf{r}$ where $\mathbf{J}$ equals the Jacobian of the residuum function $\mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m$. The exact Hessian matrix would require the derivatives of this Jacobian matrix. The Gauss-Newton method however approximates the Hessian matrix by neglecting these second-order terms, i.e. $\mathbf{H} \approx \mathbf{J}^T\mathbf{J}$. The underlying assumption is that at the optimal solution, the residuum vector is small and hence, the second-order terms can be neglected since they get multiplied with the small residuum vector. In summary, the Gauss-Newton methods solves

$$-\mathbf{J}^T\mathbf{J} d\mathbf{x} = \mathbf{J}^T\mathbf{r} \tag{3.4}$$

at each iteration in order to compute the update direction. Note that this equation actually corresponds to the normal equation of the overdetermined system $\mathbf{J}d\mathbf{x} = -\mathbf{r}$, and hence for numerical reasons, it is advantageous to solve this overdetermined system (e.g. with QR-decomposition) rather than the normal equations in Eq. (3.4).

An important property of the Gauss-Newton update direction defined by Eq. (3.4) is that it is always a descent direction, given that the Jacobian has full rank and that the gradient $\mathbf{J}^T\mathbf{r}$ is non-zero. This is in strong contrast to the Newton direction and hence the Gauss-Newton direction is always a suitable direction for line search.

3.5 Levenberg-Marquardt

As previously mentioned, the Newton update direction is not guaranteed to be a downhill direction. In contrast, the negative gradient, as used for example in a simple steepest descent algorithm, is always a descent direction. The Levenberg-Marquardt method combines a first-order gradient method with the second-order Gauss-Newton method. Specifically, a scaled identity matrix is added to the Hessian approximation of the Gauss-Newton method

$$-\left(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}\right) d\mathbf{x} = \mathbf{J}^T\mathbf{r}. \tag{3.5}$$

The parameter $\lambda \in \mathbb{R}$ is adjusted depending on whether the update $\mathbf{x} \leftarrow \mathbf{x} + d\mathbf{x}$ would result in an effective decrease of the objective function. It is obvious that an increasing λ gives higher importance to the steepest descent direction given by the gradient $\mathbf{J}^T\mathbf{r}$. Note that Eq. (3.5) are the normal equations for the linear least squares problem

$$\min_{d\mathbf{x}} \frac{1}{2} \left\| \begin{bmatrix} \mathbf{J} \\ \sqrt{\lambda}\mathbf{I} \end{bmatrix} d\mathbf{x} + \begin{pmatrix} \mathbf{r} \\ \mathbf{0} \end{pmatrix} \right\|_2^2, \tag{3.6}$$

which can be solved efficiently (e.g. with QR decomposition) without forming $\mathbf{J}^T\mathbf{J}$. Interestingly, the Levenberg-Marquardt algorithm is equivalent to a trust-region method. As Lemma 10.3 in [NW06] shows, the update direction defined through Eq. (3.5) is equivalent to the trust-region problem

$$\min_{d\mathbf{x}} \|\mathbf{J}d\mathbf{x} + \mathbf{r}\|_2^2 \text{ subject to } \|d\mathbf{x}\| \leq \Delta, \tag{3.7}$$

for some $\Delta > 0$. A non-spherical choice of the trust-region leads to other variations of the Levenberg-Marquardt algorithm, e.g. where the diagonal of $\mathbf{J}^T\mathbf{J}$ is added instead of the identity matrix. An advantage of the Levenberg-Marquardt compared to Gauss-Newton is that it does not require the Jacobian to be full-rank.

3.6 Wiberg Algorithm

The Wiberg algorithm is specifically suited for multilinear least squares problems. Originally developed in the seventies [Wib76], the Wiberg algorithm has been revived in the computer vision community mostly by [SIR95] and [OD07]. More recently, the same authors presented a slightly adapted version of the Wiberg algorithm and an experimental evaluation, which showed the same results as our experiments [OYD11].

The Wiberg algorithm is based on a variable projection approach [GP03] in a Gauss-Newton framework. Variable projection approaches express one variable as a function of the other variables, thereby leading to a more complex objective function in less unknowns. It is a rule of thumb in optimization theory, that eliminating variables at the cost of a more complex objective function generally leads to worse convergence behaviour. Surprisingly, as the experiments in this chapter show, in the case of multilinear matrix factorization problems, elimination of variables seem to perform much better than e.g. standard Levenberg-Marquardt on the full set of variables.

3.6.1 Derivation

The Wiberg algorithm is usually derived for a bilinear least squares matrix factorization problem. One variable is expressed as the linear least squares solution which is considered as a function of the remaining variable. Analogously to the Gauss-Newton method, a small residuum assumption leads to the linear system for the update step. Sec. 4.9 derives the Wiberg algorithm is this way. In this section, the Wiberg algorithm is derived in a completely different and more general way highlighting the similarity to the Schur-complement trick for Gauss-Newton methods.

For concreteness, a bilinear matrix factorization problem is considered as well. However, the derivation in this chapter shows that the

Wiberg algorithm can be applied to any non-linear least squares problem, i.e. there is no restriction to multilinear least squares problems. Let us consider the bilinear low-rank matrix completion problem

$$\min_{\mathbf{U}\in\mathbb{R}^{m\times r},\mathbf{V}\in\mathbb{R}^{r\times n}} \frac{1}{2}\left\|\mathbf{M}\odot(\mathbf{U}\mathbf{V}-\mathbf{W})\right\|_F^2, \tag{3.8}$$

where $\mathbf{M}\in\{0,1\}^{m\times n}$ is a binary matrix masking unobserved or missing entries in the observation matrix $\mathbf{W}\in\mathbb{R}^{m\times n}$. Let us denote the Jacobian matrix $\mathbf{J}=[\mathbf{J}_\mathbf{u},\mathbf{J}_\mathbf{v}]$ with matrix blocks $\mathbf{J}_\mathbf{u}$ and $\mathbf{J}_\mathbf{v}$ corresponding to the Jacobian matrices w.r.t. the vectorized matrices $\mathbf{u}=\mathrm{vec}(\mathbf{U})$ and $\mathbf{v}=\mathrm{vec}(\mathbf{V})$, respectively. Note that $\mathbf{J}_\mathbf{u}$ is solely a function of $\mathbf{v}$ for bilinear problems (and analogously $\mathbf{J}_\mathbf{v}$ is a function of $\mathbf{u}$). With this notation in place, the Gauss-Newton approximation to the Hessian equals

$$\mathbf{H}\approx\mathbf{J}^T\mathbf{J}=\begin{bmatrix}\mathbf{J}_\mathbf{U}^T\mathbf{J}_\mathbf{U} & \mathbf{J}_\mathbf{U}^T\mathbf{J}_\mathbf{V}\\ \mathbf{J}_\mathbf{V}^T\mathbf{J}_\mathbf{U} & \mathbf{J}_\mathbf{V}^T\mathbf{J}_\mathbf{V}\end{bmatrix}. \tag{3.9}$$

The Schur-complement trick eliminates a subset of the variables from the linear system for the update step. This is often done in bundle-adjustement since for these problems, the elimination of one subset of the variables is particularly simple due to the sparseness pattern in the Jacobian matrices. Specifically, the lower-left block of the Gauss-Newton approximation of the Hessian matrix can be zeroed by a left-multiplication

$$\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{J}_\mathbf{V}^T\mathbf{J}_\mathbf{U}\left[\mathbf{J}_\mathbf{U}^T\mathbf{J}_\mathbf{U}\right]^{-1} & \mathbf{I}\end{bmatrix}\mathbf{H}=\begin{bmatrix}\mathbf{J}_\mathbf{U}^T\mathbf{J}_\mathbf{U} & \mathbf{J}_\mathbf{U}^T\mathbf{J}_\mathbf{V}\\ \mathbf{0} & -\mathbf{J}_\mathbf{V}^T\mathbf{J}_\mathbf{U}\left[\mathbf{J}_\mathbf{U}^T\mathbf{J}_\mathbf{U}\right]^{-1}\mathbf{J}_\mathbf{U}^T\mathbf{J}_\mathbf{V}+\mathbf{J}_\mathbf{V}^T\mathbf{J}_\mathbf{V}\end{bmatrix}$$

Using this Schur-complement trick for solving the Gauss-Newton update equations (see Eq. (3.4)) $-\mathbf{J}^T\mathbf{J}\left(d\mathbf{u}^T,d\mathbf{v}^T\right)^T=\mathbf{J}^T\mathbf{r}$ leads to the following least squares problem for the update of $\mathbf{v}$

$$d\mathbf{v}=\arg\min_{d\mathbf{v}}\frac{1}{2}\left\|\mathbb{P}_{[\mathbf{J}_\mathbf{U}]_\perp}\mathbf{J}_\mathbf{V}d\mathbf{v}-\mathbf{r}\right\|_2^2. \tag{3.10}$$

Note that for the bilinear matrix factorization problem in Eq. (3.8) it holds that $\mathbf{r}=\mathbb{P}_{[\mathbf{J}_\mathbf{U}]_\perp}\mathbf{w}$ and hence the least squares problem simplifies to $\frac{1}{2}\left\|\mathbb{P}_{[\mathbf{J}_\mathbf{U}]_\perp}\mathbf{J}_\mathbf{V}d\mathbf{v}-\mathbf{w}\right\|_2^2$. After solving for $d\mathbf{v}$, the update $d\mathbf{u}$ in the standard Gauss-Newton algorithm is then given by

$$d\mathbf{u}=\arg\min_{d\mathbf{u}}\frac{1}{2}\left\|\mathbf{J}_\mathbf{u}d\mathbf{u}-(\mathbf{r}-\mathbf{J}_\mathbf{v}d\mathbf{v})\right\|_2^2. \tag{3.11}$$

For bilinear problems like Eq. (3.8), we have $\mathbf{r} = \mathbf{w} - \mathbf{J}_\mathbf{v}\mathbf{v}$ and hence $d\mathbf{u} = \arg\min_{d\mathbf{u}} \frac{1}{2} \|\mathbf{J}_\mathbf{u} d\mathbf{u} - (\mathbf{w} - \mathbf{J}_\mathbf{v}(\mathbf{v} + d\mathbf{v}))\|_2^2$.

In contrast, the Wiberg algorithm recomputes the Jacobian $\mathbf{J}_\mathbf{u}$ with the updated vector $\mathbf{v} \leftarrow \mathbf{v} + d\mathbf{v}$ and solves for

$$\mathbf{u} = \arg\min_{\mathbf{u}} \frac{1}{2} \|\mathbf{J}_\mathbf{u}\mathbf{u} - \mathbf{w}\|_2^2 . \tag{3.12}$$

In summary, the only difference between standard Gauss-Newton and the Wiberg algorithm is the equation for the update of the variable which was eliminated with the Schur-complement trick (Eq. (3.11) vs. Eq. (3.12)).

3.6.2 Efficient Solution

Due to the matrix factorization ambiguity $\mathbf{U}\mathbf{V} = \mathbf{U}\mathbf{Q}\mathbf{Q}^{-1}\mathbf{V}$ for any regular $\mathbf{Q} \in \mathbb{R}^{r \times r}$, the system matrix $\mathbb{P}_{[\mathbf{J}_\mathbf{U}]_\perp}\mathbf{J}_\mathbf{V}$ in the linear least squares problem in Eq. (3.10) is rank deficient by r^2. Hence, care must be taken when solving this linear least squares problem. In [OD07], it is recommended to use the Moore-Penrose pseudo-inverse which leads to the solution with the smallest L2-norm. Unfortunately, computing this pseudo-inverse is quite costly. Hence, a solution based e.g. on rank-revealing QR decomposition is favorable. For bilinear matrix factorization problems, the Jacobian matrices have a very specific structure which can be exploited in order to derive an even more efficient algorithm.

In our joint work with Daskalov [Das11], we proposed to use LSQR [PS82], an iterative conjugate gradient style method for linear least squares problems. Analogously to conjugate gradient methods, no explicit representation of the system matrix of the linear system needs to be computed but rather only a function computing the product between the system matrix and a vector needs to be provided. The product between the system matrix $\mathbb{P}_{[\mathbf{J}_\mathbf{U}]_\perp}\mathbf{J}_\mathbf{v}$ and a vector $\mathbf{y}$ can be split in two successive matrix-vector products. Barring missing rows due to missing observations, the Jacobian equals $\mathbf{J}_\mathbf{v} = \mathbf{I}_n \otimes \mathbf{U}$ and is therefore highly sparse. Efficient sparse matrix-vector multiplication can be used to compute $\mathbf{J}_\mathbf{v}\mathbf{y} = \mathbf{z}$. As a projector onto the orthogonal complement of the column space of $\mathbf{J}_\mathbf{u}$, $\mathbb{P}_{[\mathbf{J}_\mathbf{u}]_\perp}$ equals $\mathbb{P}_{[\mathbf{J}_\mathbf{u}]_\perp} = \mathbf{I} - \mathbb{P}_{\mathbf{J}_\mathbf{u}}$ and hence

$$\mathbb{P}_{[\mathbf{J}_\mathbf{u}]_\perp}\mathbf{z} = \mathbf{z} - \mathbb{P}_{\mathbf{J}_\mathbf{u}}\mathbf{z} = \mathbf{z} - \mathbf{O}_{\mathbf{J}_\mathbf{u}}\mathbf{O}_{\mathbf{J}_\mathbf{u}}^T\mathbf{z}, \tag{3.13}$$

where the columns of $\mathbf{O}_{\mathbf{J_u}}$ span an orthogonal basis for the column space of $\mathbf{J_u}$. Thanks to the structure of $\mathbf{J_u}$, the orthogonal projector $\mathbb{P}_{\mathbf{J_u}}$ onto the column space of $\mathbf{J_u}$ can be efficiently computed. Intuitively, the cummutation matrices $\mathbf{T}_{m,n}$ and $\mathbf{T}_{m,r}$ (see Sec. 2.2.2 or Theorem 9 in Chap. 3 in [MN99]) can be used to reorder the entries of $\mathbf{J_u} = \mathbf{V}^T \otimes \mathbf{I}$ (again barring missing rows due missing entries) into block diagonal form $\mathbf{T}_{m,n}\mathbf{J_u}\mathbf{T}_{m,r} = \mathbf{I} \otimes \mathbf{V}^T$. As such, an orthogonal basis for the column space can be computed by orthogonalizing each diagonal block separately and assembling the full orthogonal basis $\mathbf{O}_{\mathbf{J_u}}$ from these individual diagonal blocks. Obviously, this orthogonal basis only needs to be computed once for each linear least squares problem since the matrix $\mathbf{J_u}$ stays the same throughout all the LSQR iterations.

[Han01] has shown the importance of terminating iterative methods like LSQR and conjugate gradients for least squares (CGLS) problems sufficiently early. This is due to the fact that even though initially these methods converge to the solution, in later iterations they diverge due to noise and numerical errors. It was shown that with proper termination criterion, LSQR and CGLS provide solutions for ill-conditioned systems of equal quality as the ones obtained by truncated singular value decomposition. Furthermore, Hanke observed that in practice, the number of required iterations for LSQR and CGLS is smaller than the optimal number of components in a truncated singular value decomposition. Hence, we fixed an upper bound of at most $(n-r)r$ inner LSQR iterations for Eq. (3.10). In practice however, the required precision is often reached before this upper bound. There is no theoretical derivation for this choice but the experiments show that the quality of the solution with this approach is at least as good as the solution given by a truncated singular value decomposition.

3.6.3 Line Search

In the usual derivation of the Wiberg algorithm [SIR95; OD07], the step length for the update direction $d\mathbf{v}$ defined through Eq. (3.10) is implicitly chosen equal to one, i.e. $\mathbf{v} \leftarrow \mathbf{v} + d\mathbf{v}$. This choice can actually lead to steps that increase the objective function value. Hence, we suggest performing a line search $\mathbf{v} + \alpha d\mathbf{v}$ in order to ensure a decrease of the objective function. This is a simple way to ensure convergence, even though it slightly increases the timer required per iteration.

3.7 Synthetic Experiments

In this section, the previously presented algorithms will be evaluated with synthetically generated data. It turned out that Gauss-Newton with line search, Levenberg-Marquardt and Matlab's trust-region-reflective algorithm all performed similarly in terms of recovery rate. Hence, out of these three algorithms, only results for one of them will be reported. Specifically, for small matrices Levenberg-Marquardt will be used whereas for medium and large matrices, Matlab's trust-region-reflective algorithm will be used instead. Alternating least squares and the Wiberg algorithm (with line search) represent the remaining competitors in this comparison. The results reported here are based on joint work with Boris Daskalov [Das11]. Low-rank matrices and patterns for observed entries depend on many parameters and can strongly influence the performance of algorithms. The next paragraphs therefore present in detail how the synthetic data for the empirical evaluation was generated.

3.7.1 The Matrix Completion Problem

The experiments focus on the bilinear low-rank matrix completion problem in Eq. (3.8). Emphasis is put on the rate how often the underlying low-rank matrix has been recovered by different algorithms for varying random input data. This recovery rate is strongly related to the number of available measurements, i.e. to the number of ones in the binary matrix $\mathbf{M}$, and to the pattern of missing entries. This should not come as a surprise since compressive sensing recently established guarantees when a convex relaxation of a rank constraint will yield the same optimal solution as the original non-relaxed problem with very high probability. This probability of success strongly depends on the pattern of missing data and the orientation of the column and row subspace of the underlying matrix. Such guarantees were first established for the matrix completion problem and were subsequently generalized to arbitrary linear measurements [RFP10; Gro11]. In compressive sensing, a specific instance of the general problem in Eq. (1.1) is considered which is known as the affine rank minimization problem

$$\min_{\mathbf{X} \in \mathbb{R}^{m \times n}} \operatorname{rank}(\mathbf{X}) \text{ s.t. } \forall i \in \Omega : \langle \mathbf{M}_i, \mathbf{X} \rangle = w_i, \tag{3.14}$$

where $\langle \mathbf{M}_i, \mathbf{X} \rangle = \text{trace}\left(\mathbf{M}_i^T \mathbf{X}\right)$ is the standard inner product between matrices and Ω is the index set of available measurements. The linear measurement matrices $\mathbf{M}_i$ for $i \in [mn]$ are often assumed to be an orthogonal basis of mn-dimensional space $\mathbb{R}^{m \times n}$. The inner-product equalities then constrain the unknown low-rank matrix to an affine subspace of the embedding mn-dimensional Euclidean space. Hence, the optimal solution lies at the intersection between the variety of matrices with low rank and this affine subspace. The dependency between the measurement matrices and the column and row subspace of the unknown matrix is captured with the so called *restricted isometry property* (There exist several slightly different definitions but they all capture this dependency). Chap. 6 will cover convex relaxations for rank constraints in detail, in the rest of this chapter, we will focus on the non-relaxed matrix completion problem. Moreover, it will be assumed that the correct rank r is known, i.e. the optimization problem in Eq. (3.14) turns into a feasibility problem.

The matrix completion problem is given by choosing the standard basis $\mathbf{e}_i \mathbf{e}_j^T$ with $i \in [m]$ and $j \in [n]$ and $\Omega \subset [m] \times [n]$ where a double index is introduced for notational convenience. Instead of solving the exact matrix completion problem of Eq. (3.14), we will consider the noisy matrix completion problem of Eq. (3.8)

$$\min_{\mathbf{U} \in \mathbb{R}^{m \times r}, \mathbf{V} \in \mathbb{R}^{r \times n}} \frac{1}{2} \sum_{(i,j) \in \Omega} \left\| \langle \mathbf{e}_i \mathbf{e}_j^T, \mathbf{U}\mathbf{V} \rangle - \mathbf{W}_{i,j} \right\|_2^2 \tag{3.15}$$

$$= \min_{\mathbf{U} \in \mathbb{R}^{m \times r}, \mathbf{V} \in \mathbb{R}^{r \times n}} \frac{1}{2} \left\| \mathbf{M} \odot (\mathbf{U}\mathbf{V} - \mathbf{W}) \right\|_F^2, \tag{3.16}$$

with $\mathbf{M} \in \{0,1\}^{m \times n}$ and $\mathbf{M}_{i,j} = 1$ if $(i,j) \in \Omega$ and $\mathbf{M}_{i,j} = 0$ otherwise.

3.7.2 Synthetic Data Generation

As described in the previous section, the probability of successful recovery largely depends on the row and column subspace of the underlying low-rank matrix and the pattern of missing entries. The low-rank measurement matrices $\mathbf{W}$ were generated according to Alg. 1. The steps 1-3 ensure that the row and column space of $\mathbf{W}_0$ are sampled uniformly from the Grassmannian[1] $G_{r,m}$ resp. $G_{r,n}$. These steps make use of the

[1] The Grassmannian $G_{r,n}$ consists of all the r-dimensional subspaces embedded in n-dimensional space. We refer to Chap. 8 for more details.

Algorithm 1: Synthetic data generation

Input: Size (m, n) and rank r of measurement matrix $\mathbf{W}$, diagonal matrix of singular values $\mathbf{\Sigma}$, standard deviation of noise σ_{noise}.

Output: Noisy measurement matrix $\mathbf{W}$.

1 `// Sample the column and row subspaces uniformly from Grassmannian G(r,m) resp. G(r,n).`
2 $\mathbf{U}_{i,j} \sim \text{Normal}(0, 1)$ for $i \in [m], j \in [r]$; $\mathbf{Q}, \mathbf{R} \leftarrow \text{qr}(\mathbf{U})$; $\mathbf{U} \leftarrow \mathbf{Q}$
3 $\mathbf{V}_{i,j} \sim \text{Normal}(0, 1)$ for $i \in [r], j \in [n]$; $\mathbf{Q}, \mathbf{R} \leftarrow \text{qr}(\mathbf{V}^T)$; $\mathbf{V} \leftarrow \mathbf{Q}^T$
4 `// Normalize the singular values.`
5 $\mathbf{\Sigma} \leftarrow \sqrt{mn}\mathbf{\Sigma} / \|\mathbf{\Sigma}\|_F$
6 `// Generate ground truth matrix and noisy observations.`
7 $\mathbf{W}_0 \leftarrow \mathbf{U}\mathbf{\Sigma}\mathbf{V}$
8 $\mathbf{W} \leftarrow \mathbf{W}_0 + \sigma_{noise}\mathbf{1}_{m \times n}$

rank-revealing QR-decomposition. In order to fix an intuitive signal-to-noise ratio, the singular values are normalized in step 5 such that $\sqrt{\frac{1}{mn}\|\mathbf{W}_0\|_F} = 1$. In this way, the signal-to-noise ratio commonly used for matrix completion problems $\frac{\|\mathbf{W}-\mathbf{W}_0\|_F}{\|\mathbf{W}_0\|_F}$ equals to root mean squared error $\frac{\|\mathbf{W}-\mathbf{W}_0\|_F}{\sqrt{mn}}$ whose expectation equals σ_{noise}. All algorithms were fed with the same synthetic data sets and were initialized at the same point.

Choice of Parameters

The matrix size was chosen as $m \times n = 100 \times 300$, the rank was $r = 3$, and the noise was one percent, i.e. $\sigma_{noise} = 0.01$ for all the experiments if not explicitly mentioned otherwise. Rank $r = 3$ was chosen since the rigid affine structure-from-motion problem will lead to a rank-3 matrix factorization problem, as we will see in Sec. 3.8. For each experiment, 200 random synthetic data sets were generated, where the number of known entries was selected such that a prescribed average number of observations per degree of freedom was met. The average number of observations per degree of freedom was selected uniformly at random from the interval $[1.1, 6]$. Note that the degrees of freedom for a $m \times n$ matrix of rank r equals $mr + nr - r^2$ because of the $r \times r$ factorization

ambiguity. For rank-3 matrices of size $m \times n = 100 \times 300$, 1 observation per degree of freedom translates to 4% observed entries out of the mn available ones.

The stopping criterion was based on three parts. Firstly, a threshold for the root mean squared error w.r.t. the observed entries $RMSE = \frac{1}{\sqrt{|\Omega|}} \|\mathbf{M} \odot (\mathbf{UV} - \mathbf{W})\|_F \leq \epsilon$ has been. ϵ was set to 10^{-2}. Secondly, the algorithm was also stopped if the change in this root mean squared error was below 10^{-5} during five consecutive iteration. Lastly, an upper bound of 100 was fixed for the total number of iterations.

Pattern of Known Entries

Two different patterns for the known entries were investigated. Uniform sampling selects known entries (i, j) uniformly at random from the whole set $[m] \times [n]$. Note that if it happened that a row or columns was sampled less often than the rank of the matrix, additional not yet sampled elements are included from this row or column until this lower bound is met.

In diagonal sampling, known entries are restricted around a band which goes from the top left entry to the bottom right entry. This simulates the strong band-diagonal observation pattern in real structure-from-motion data. Inside this band, entries are sampled uniformly at random. We have decided to choose the band width as small as possible such that the number of matrix entries within this band is still at least 1.5 times the requested number of observations.

3.7.3 Balanced Singular Values

For the first experiment, the singular values were chosen roughly equal. Specifically, the ratio between the three non-zero singular values was set to $10 : 9 : 8$ (only the ratio matters, as the singular values are normalized for the data generation). The three algorithms (alternating least squares, Levenberg-Marquardt, and Wiberg) are evaluated on uniformly sampled patterns and diagonal patterns. Fig. 3.1 shows the training error $\frac{1}{\sqrt{|\Omega|}} \|\mathbf{M} \odot (\mathbf{UV} - \mathbf{W})\|_F$ for the 200 randomly generated problem instances. The generalization error is shown in Fig. 3.2 which plots the root mean squared error w.r.t. the ground truth $\frac{1}{\sqrt{mn}} \|\mathbf{UV} - \mathbf{W}\|_F$. The standard deviation of the noise was set to $\sigma_{noise} = 0.01$ and hence, the results reporting an error on the order of 10^{-2} can be considered as

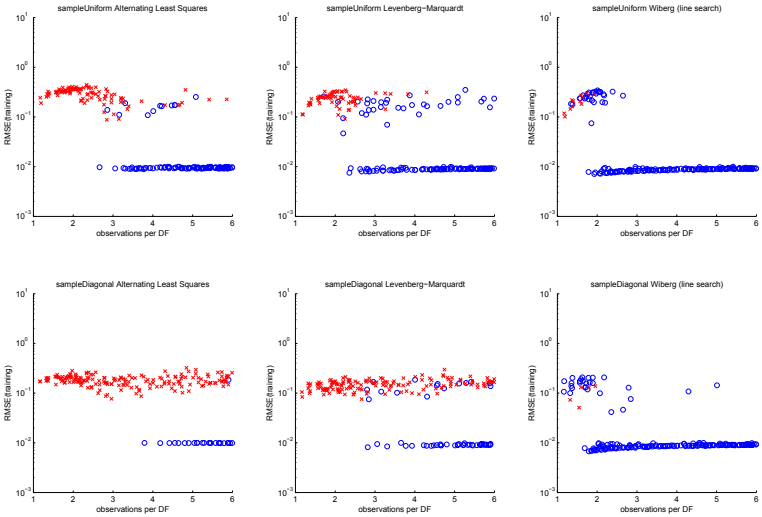


Figure 3.1: Training error $\frac{1}{\sqrt{|\Omega|}} \|\mathbf{M} \odot (\mathbf{UV} - \mathbf{W})\|_F$ for balanced singular values: The red labels indicate instances where the upper limit of 100 iterations has been reached, whereas blue labels denote instances where this upper limit has not been reached.

successfully converged. Lastly, Fig. 3.3 visualizes the required number of iterations for each algorithm.

Number of Iterations

A direct comparison of the required number of iterations can be misleading as the complexity per iteration varies between the different algorithms, especially ALS has a low complexity per iteration. In order to rule out a premature termination of an algorithm, the upper bound on the maximal number of iterations was increased to 1000. However, as the training error in Fig. 3.4 and the required number of iterations in Fig. 3.5 show, this has no significant influence on the results: the number of instances which converged to the global minimum stayed roughly the same even though none of the algorithms reached the upper

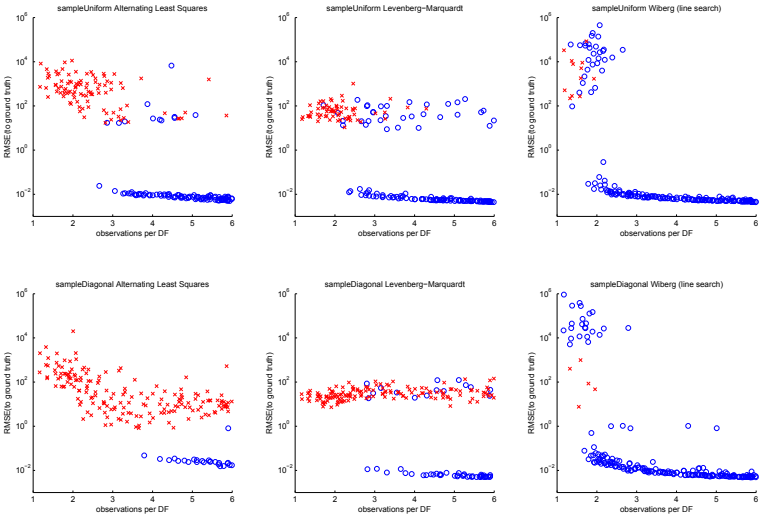


Figure 3.2: Root mean squared error w.r.t. ground truth $\frac{1}{\sqrt{mn}} \|\mathbf{UV} - \mathbf{W}\|_F$ for balanced singular values.

limit of 1000 iterations.

In practice, a commonly observed behaviour was that depending on the initialization, the algorithms either converged very quickly to the global optimum or got stuck in a local minimum early on. This can be seen for example in the two spatially separate point clouds in the plots in Fig. 3.4 and Fig. 3.5. This allows to draw the conclusion that it is advantageous to terminate an algorithm already after a few iterations if it has not yet converged and to restart the algorithm with a different, maybe random, initialization. We could also confirm a well-known property of the alternating least squares algorithm. Namely, this algorithm is known to rapidly decrease the objective function value during the first few iterations. However, the algorithm often seems to get stuck in a plateau if it does not converge to a global optimum in these first few iterations: the objective function value hardly changes and the update steps become increasingly small. At some point, alternating least squares then terminates in a suboptimal

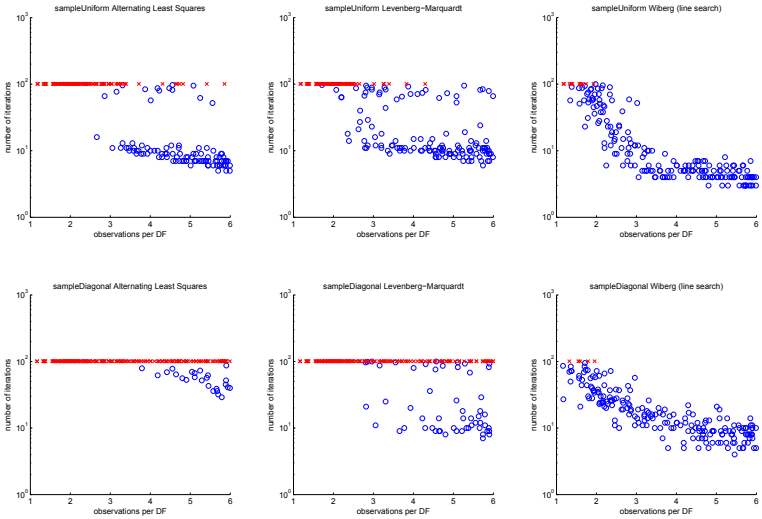


Figure 3.3: Number of iterations: The upper limit of 100 iterations is hardly reached by Wiberg whereas the other two algorithms reach this limit especially for the more difficult problem instances with only very few observations per degree of freedom.

local minima. Based on our experience, instead of reinitializing both matrix factors at the same time, a small random perturbation of just one of the matrix factors, say $\mathbf{V}$, increased the performance of the alternating least squares algorithm considerably. Even though still not competitive with the Wiberg algorithm, the success of introducing randomness in alternating least squares motivated our research with sampling approaches, specifically Gibbs sampling. We refer to Chap. 7 for such sampling techniques. Sampling techniques can deal with outliers in the measurements, a problem for the Wiberg algorithm which is inherently based on a non-robust least-squares cost function.

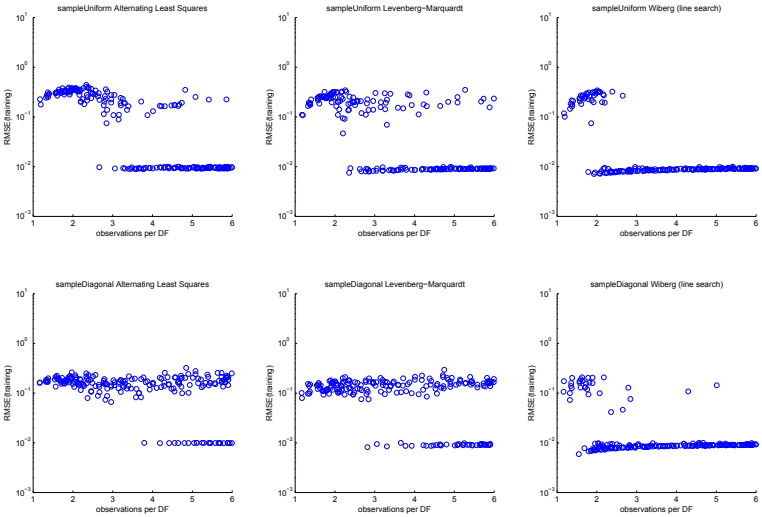


Figure 3.4: Training error $\frac{1}{\sqrt{|\Omega|}} \|\mathbf{M} \odot (\mathbf{UV} - \mathbf{W})\|_F$ for balanced singular values with upper bound of 1000 iterations.

3.7.4 Unbalanced Singular Values

Unbalanced singular values were chosen for a second set of experiments in order to investigate the behaviour of the algorithms for less well-conditioned matrices. The ratio of the three non-zero singular values was chosen as $100 : 10 : 1$. No noise was added in this experiment, i.e. $\sigma_{noise} = 0$, and the threshold for the stopping criterion was set to $\epsilon = 10^{-6}$. The results in Fig. 3.6 show the increased difficulty for unbalanced singular values: the performance of all three algorithms decreased compared to the experiments with balanced singular values. This also highlights the importance of proper data normalization, e.g. for structure-from-motion data. Non-normalized data will lead to highly unbalanced singular values. The Wiberg algorithm still outperforms its competitors by far.

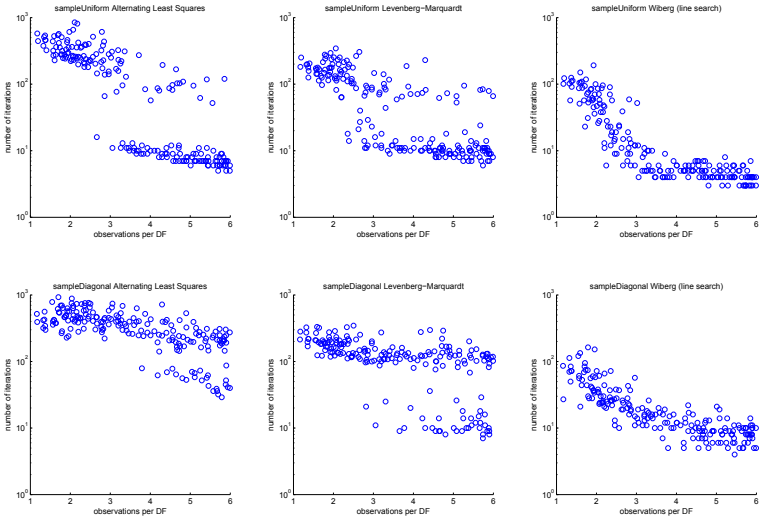


Figure 3.5: Number of iterations with upper bound of 1000 iterations.

3.7.5 Higher Rank

The previous examples focused on a really low rank $r = 3$. In a last set of synthetic experiments, we looked at the performance for matrices with a slightly higher rank, specifically the rank was set to $r = 10$ with a ratio between the singular values $10 : 9 : \ldots : 1$. This leads to more unknowns and Matlab's Levenberg-Marquardt implementation run into memory limitations. Hence, we switched to Matlab's trust-region reflective algorithm for non-linear least squares problems since this uses a less memory-demanding iterative solver for the computation of the step direction.

Based on the results reported in Fig. 3.7, higher rank problems appear to be easier than low rank matrix completion problems. This observation seems counter intuitive as higher rank matrices also induce a higher dimensional search space. An explanation might be that the basins of attractions of local minima are much smaller relative to the basins of attraction of global minima in such larger search spaces.

All the algorithms have a better performance and the difference

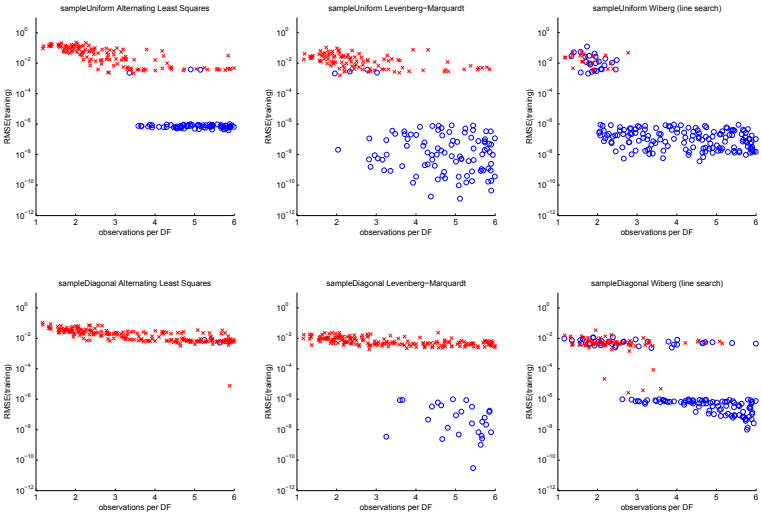


Figure 3.6: Training error for unbalanced singular values.

in performance is less distinctive. The Wiberg algorithm nonetheless slightly outperforms the competing algorithms: The training errors for the Wiberg algorithm reach the signal-to-noise ratio already for slightly more than one observation per degree of freedom. On the other hand Fig. 3.8 shows that for slightly more than one observation per degree of freedom, overfitting can occur and the rank-10 matrix is fit partly to the noise contained in the data. However, severe overfitting only occurs for really sparsely sampled data (e.g. less than 1.4 observations per degree of freedom for the Wiberg algorithm).

3.8 Affine Structure-from-Motion

Tomasi's and Kanade's seminal work [TK92] showed that the structure-from-motion problem with affine cameras can be formulated as a low-rank factorization problem. Feature point trajectories are the input to this factorization algorithm which are computed by detecting and tracking interest points throughout a video sequence. If each interest

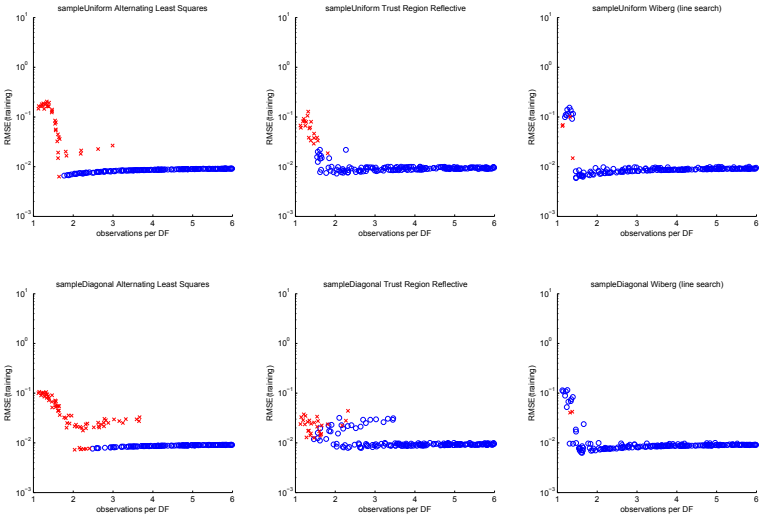


Figure 3.7: Training error for rank 10: Note that here, 1 observation per degree of freedom corresponds to 13% observed entries out of the mn available ones. e.g. 4 observation per degree of freedom correspond to 52% known entries and the band for the diagonal sampling already covers more than 75% of the matrix which means there is hardly any band diagonal structure anymore.

point is seen in all the frames, Tomasi and Kanade have shown that when the tracked feature points are arranged properly in a data matrix, this data matrix is of rank 4 (or 3 if the mean per frame is subtracted) and hence can be factorized for example with a singular value decomposition. The resulting rank-4 matrix factors reveal the camera matrices and the 3D structure of the points. However, in practice, feature points can only be tracked over a short sequence of successive frames resulting in many missing or unknown entries in the data matrix. Hence, there is no factorization algorithm anymore which yields the correct matrix factors. The resulting non-linear problem can however be solved with iterative least-squares methods, as we will see in this section.

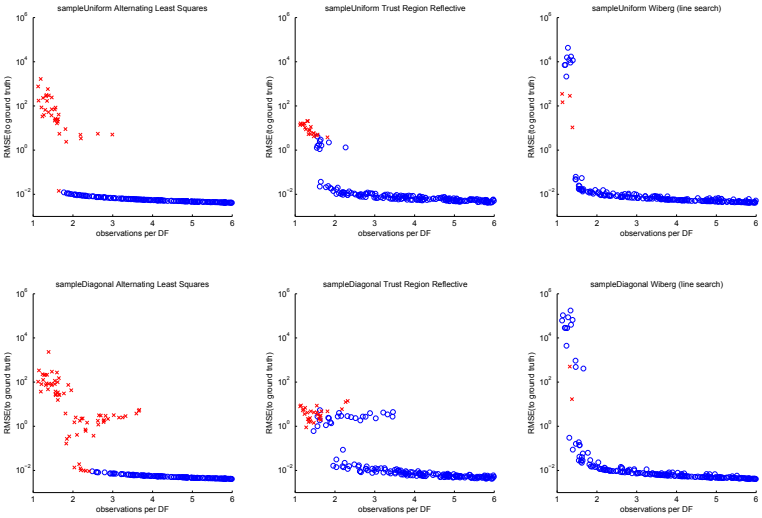


Figure 3.8: Error to ground truth for rank 10.

3.8.1 Derivation

Let us denote the coordinates of the the n-th feature point with homogeneous coordinates $\mathbf{X}_n \in \mathbb{R}^4$ seen in the f-th frame with $\mathbf{x}_{f,n} \in \mathbb{R}^2$. As derived in Sec. 2.3.2, an affine camera can be modeled with a 2×4 camera matrix. Hence, the camera matrix at frame f is given by $\mathbf{P}_f \in \mathbb{R}^{2\times4}$. The observation model is therefore given by $\mathbf{x}_{f,n} = \mathbf{P}_f\mathbf{X}_n$. The observations for all the frames $f \in [F]$ and all the points $n \in [N]$ can be arranged in one data matrix $\mathbf{W} = \left[\Downarrow_{f=1}^{F} \Rightarrow_{n=1}^{N} \mathbf{x}_{f,n}\right] \in \mathbb{R}^{2F\times N}$. Plugging the affine projection model into this data matrix results in a factorized rank-4 representation

$$\mathbf{W} = \left[\Downarrow_{f=1}^{F} \Rightarrow_{n=1}^{N} \mathbf{x}_{f,n}\right] = \left[\Downarrow_f \mathbf{P}_f\right]\left[\Rightarrow_n \mathbf{X}_n\right] = \mathbf{P}\mathbf{X}, \tag{3.17}$$

where $\mathbf{P} = \left[\Downarrow_f \mathbf{P}_f\right] \in \mathbb{R}^{2F\times4}$ and $\mathbf{X} = \left[\Rightarrow_n \mathbf{X}_n\right] \in \mathbb{R}^{4\times N}$. According to Sec. 2.3.2, the definition of affine cameras requires the plane at infinity to be known and to be mapped to its canonical representation, i.e. the coordinate representation of the plane at infinity equals $\mathbf{p}_\infty = (0,0,0,1)$. In this case, the last column of the camera matrices correspond to

the camera translation. Let $\mathbf{P} = \left[\hat{\mathbf{P}}, \mathbf{t}\right]$ with the camera translations $\mathbf{t} = (\Downarrow_f \mathbf{t}_f)$ and $\hat{\mathbf{P}} = \left[\Downarrow_f \hat{\mathbf{P}}_f\right]$ where $\mathbf{t}_f \in \mathbb{R}^{2\times 1}$ and $\hat{\mathbf{P}}_f \in \mathbb{R}^{2\times 3}$. Assuming there is no point at infinity, the scale of the homogeneous representation of each point can be chosen such that $\mathbf{X}^T = \left[\hat{\mathbf{X}}^T, \mathbf{1}_{N\times 1}\right]$. This choice results in a mixed bilinear-linear low-rank model $\mathbf{W} = \hat{\mathbf{P}}\hat{\mathbf{X}} + \mathbf{t}\mathbf{1}_{N\times 1}^T$. Since points can usually only be reliably tracked over a short sequence of frames, the masking matrix $\mathbf{M} \in \{0,1\}^{2F\times N}$ is introduced in order to disregard unobserved entries which leads to the following optimization problem

$$\hat{\mathbf{P}}^*, \hat{\mathbf{X}}^*, \mathbf{t} = \arg\min_{\hat{\mathbf{P}},\hat{\mathbf{X}}} \frac{1}{2} \left\| \mathbf{M} \odot \left[\hat{\mathbf{P}}\hat{\mathbf{X}} + \mathbf{t}\mathbf{1}_{N\times 1}^T - \mathbf{W}\right]\right\|_F^2 \tag{3.18}$$

The residual vector of this least squares problem is given after vectorization

$$\mathbf{r} = \mathbf{I}_\mathbf{M} \operatorname{vec}\left(\hat{\mathbf{P}}\hat{\mathbf{X}} + \mathbf{t}\mathbf{1}_{N\times 1}^T - \mathbf{W}\right) = \mathbf{I}_\mathbf{M} \operatorname{vec}\left(\hat{\mathbf{P}}\hat{\mathbf{X}} + \mathbf{t}\mathbf{1}_{N\times 1}^T\right) - \mathbf{w}, \tag{3.19}$$

where the row-amputated identity matrix $\mathbf{I}_\mathbf{M} \in \mathbb{R}^{m\times 2FN}$ has been introduced. This matrix equals $\operatorname{diag}(\operatorname{vec}(\mathbf{M})) \in \mathbb{R}^{2FN\times 2FN}$ after deleting all the zero-rows, i.e. m equals the number of observations. With this notation in place, the partial Jacobian matrices are given by

$$\mathbf{J}_{\hat{\mathbf{P}}} = \mathbf{I}_\mathbf{M}\left[\hat{\mathbf{X}}^T \otimes \mathbf{I}_{2F}\right],\ \mathbf{J}_\mathbf{t} = \mathbf{I}_\mathbf{M}\left[\mathbf{1}_{N\times 1} \otimes \mathbf{I}_{2F}\right],\ \text{and}\ \mathbf{J}_{\hat{\mathbf{X}}} = \mathbf{I}_\mathbf{M}\left[\mathbf{I}_N \otimes \hat{\mathbf{P}}\right],$$

with the full Jacobian $\mathbf{J} = \left[\mathbf{J}_{\hat{\mathbf{P}}}, \mathbf{J}_\mathbf{t}, \mathbf{J}_{\hat{\mathbf{X}}}\right]$. The equations for the residual vector and the Jacobian matrices are sufficient for applying the alternating least squares, Levenberg-Marquardt, or the Wiberg algorithm. We have chosen to eliminate the points $\hat{\mathbf{X}}$ in the Wiberg algorithm due to reasons explained in the next sections.

3.8.2 Orthogonality Constraints

Due to the factorization ambiguity, once valid matrix factors $\hat{\mathbf{P}}$ and $\hat{\mathbf{X}}$ are found, the reconstruction is unique up to an affine distortion. In order to compute a similarity upgrade, additional constraints need to be imposed to compute a corrective transformation $\mathbf{H} \in \mathbb{R}^{4\times 4}$. Since the plane at infinity has been fixed at its canonical coordinates, the corrective transformations are limited to affine transformations of the form

$$\mathbf{H} = \begin{bmatrix} \hat{\mathbf{H}} & \hat{\mathbf{t}} \\ \mathbf{0}_{3\times 1}^T & 1 \end{bmatrix}, \tag{3.20}$$

A popular choice for such additional constraints is to assume that the camera matrices have square pixels. This leads to self-calibration equations of the form (see also Chap. 19 in [HZ04])

$$\omega_f^* = \mathbf{K}_f \mathbf{K}_f^T = \mathbf{P}_f \underbrace{\mathbf{H} \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1}^T & 0 \end{bmatrix} \mathbf{H}^T}_{=\mathbf{Q}_\infty^*} \mathbf{P}_f^T, \tag{3.21}$$

where $\mathbf{Q}_\infty^*$ is the absolute dual quadric and ω_f^* is the dual image of the absolute conic at frame f which encodes the square pixel assumption.

For affine cameras, the square pixel assumption leads to a scaled orthographic projection (p.171 in [HZ04]). Since affine cameras have no well-defined principal point (i.e. the principal point and the camera translation are interchangeable, see Sec. 6.3.3 in [HZ04]), the principal point can be assumed to be at $(0,0)$ which simplifies the formula for the dual image of the absolute conic considerably (p.465 in [HZ04]). Together with the scaled orthographic projection and the restriction of $\mathbf{H}$ to affine transformation Eq. (3.21) leads to[2]

$$\hat{\mathbf{P}}_f \hat{\mathbf{H}} \hat{\mathbf{H}}^T \hat{\mathbf{P}}_f^T = \begin{bmatrix} \alpha_f & 0 \\ 0 & \alpha_f \end{bmatrix}, \tag{3.22}$$

which provides two linear equations on symmetric $\hat{\mathbf{H}}\hat{\mathbf{H}}^T$. This can be seen by vectorizing the above equation

$$\left[\hat{\mathbf{P}}_f \otimes \hat{\mathbf{P}}_f\right] \mathrm{vec}\left(\hat{\mathbf{H}}\hat{\mathbf{H}}\right) = (\alpha_f, 0, \alpha_f, 0)^T,$$

substituting the unique entries of the symmetric unknown $\hat{\mathbf{H}}\hat{\mathbf{H}}^T \in \mathbb{S}^3$ by the vector $\mathbf{h} = \mathrm{vecs}\left(\hat{\mathbf{H}}\hat{\mathbf{H}}^T\right) \in \mathbb{R}^{6\times 1}$, i.e. $\mathrm{vec}\left(\hat{\mathbf{H}}\hat{\mathbf{H}}\right) = \mathbf{D}_3 \mathrm{vecs}\left(\hat{\mathbf{H}}\hat{\mathbf{H}}\right) = \mathbf{D}_3\mathbf{h}$, and then it holds

$$\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \left[\hat{\mathbf{P}}_f \otimes \hat{\mathbf{P}}_f\right] \mathbf{D}_3 \mathbf{h} = \mathbf{0}_{2\times 1}. \tag{3.23}$$

For multiple frames with sufficiently general motion, the above linear equations lead to an overdetermined linear system in $\mathbf{h}$ which can be solved in the least squares sense. Devectorization of this vector leads to

[2] The third row of the affine camera matrices have been neglected in this formula in order to be have a consistent notation. These rows do not provide any additional constraint anyway.

either a positive or negative definite matrix $\mathbf{Q} = \text{vecs}^{-1}(\mathbf{h}) \in \mathbb{S}^3$ which can be decomposed into corrective transformations $\mathbf{H}$ e.g. via Cholesky decomposition or Eigenvalue decomposition. An alternative to the linear least squares based approach is to consider Eq. (3.22) as constraints of a semi-definite programming instance with unknowns $\mathbf{Q} = \hat{\mathbf{H}}\hat{\mathbf{H}}^T \in \mathbb{S}^3_+$ and α_f.

In summary, traditional factorization based affine SfM approaches first compute an reconstruction in an affine coordinate reference frame. Then a subsequent similarity upgrade is performed by solving the overdetermined linear system due to Eq. (3.23) from which the final corrective transformation can be extracted. These two sequential steps are suboptimal since errors in the affine reconstruction can propagate and amplify in the corrective transformation computation. Indeed, it is not uncommon that the least squares solution $\mathbf{h}$ leads to an indefinite devectorized symmetric matrix $\mathbf{Q}$, clearly an invalid starting point for factorizing $\mathbf{Q}$ into $\mathbf{Q} = \hat{\mathbf{H}}\hat{\mathbf{H}}^T$. In contrast, if non-linear optimization is used, the orthogonality constraints can be built into the objective function, as we will see in the next section.

3.8.3 Orthogonality Constraints as Soft-Penalty

A simple approach to incorporate the orthogonality constraints into the objective function in Eq. (3.18) is to add a soft-penalty term

$$\frac{1}{2m}\left\|\mathbf{M} \odot \left[\hat{\mathbf{P}}\hat{\mathbf{X}} + \mathbf{t}\mathbf{1}_{N\times 1}^T - \mathbf{W}\right]\right\|_F^2 + \frac{\lambda}{F}\sum_{f=1}^{F}\left\|\hat{\mathbf{P}}_f\hat{\mathbf{P}}_f^T - \alpha_f\mathbf{I}_2\right\|_F^2. \quad (3.24)$$

In these experiments we assume also a constant focal length, i.e. one can set $\forall f \in [F] : \alpha_f = 1$. The hyperparemeter λ controls the trade-off between the soft-penalty term for the orthogonality constraints and the reprojection error. The two terms are scaled by the number of observations m respectively by the number of frames F such that the paremeters λ can be chosen independently of the number of observations or the number of frames. This objective function is no longer a bilinear least squares problem since the soft-penalty term for the orthogonality constraints is no longer quadratic in $\hat{\mathbf{P}}$. However, it is still a least squares problem and results in a slightly more complex Jacobian $\mathbf{J}_{\hat{\mathbf{P}}}$ compared to the formulation without orthogonality constraints due to the additional rows corresponding to the soft-penalty terms. The

Jacobians for $\hat{\mathbf{X}}$ and $\mathbf{t}$ however only need to be extended with additional zero-rows and remain the same otherwise (up to the scale factor m^{-1}). The soft-penalty term has no influence for the elimination of the points in the Wiberg algorithm. The resulting objective function will be a highly complex function of $\hat{\mathbf{P}}$ and the soft-penalty term will not add much to this complexity anymore. This is the reason why we chose to eliminate the points $\hat{\mathbf{X}}$ rather than the cameras $\hat{\mathbf{P}}$. Note that the efficient iterative LSQR algorithm presented in Sec. 3.6.2 is still applicable as the additional rows in $\mathbf{J}_{\hat{\mathbf{P}}}$ are highly sparse.

3.8.4 Experiments

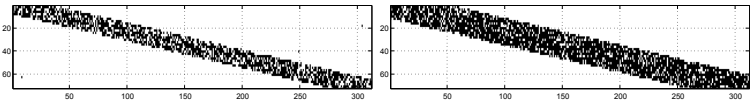
This section evaluates the previously presented non-linear least squares methods on structure-from-motion data. Specifically, the dinosaur sequence[3] is used for the evaluations. This sequence consists of 36 views of a turntable motion of a toy dinosaur. Only points which were visible in at least 6 frames were considered which resulted in 319 remaining feature tracks. Since all the algorithms use a L2-error and hence are not robust w.r.t. outliers, a preprocessing step was applied to eliminate all the outliers: Each feature point was triangulated with the provided ground truth cameras and a feature track was considered as an outlier if the resulting reprojection error was larger than 1 pixel. Out of the 319 feature tracks, 312 survived this preprocessing step.

We have previously seen that unbalanced singular values can harm the performance of the algorithms. Therefore, a proper data normalization is beneficial. We used the standard normalization which centers each point by subtracting the mean of all the points. Then the points are isotropically scaled such that their standard deviation equals one. Such a normalization improved the ratio between the singular values for the dinosaur sequence from $100 : 19 : 15 : 7$ to roughly $100 : 63 : 32 : 28$.

Evaluation on Semi-Synthetic Data

Having triangulated all the points in the preprocessing step allows to compute a complete noise free measurement matrix by reprojecting the outlier-free feature points into the camera views. In this experiment, the ground truth cameras were replaced with their affine counterpart,

[3] Available from `http://www.robots.ox.ac.uk/~vgg/data/data-mview.html`, but originally from the University of Hannover.



(a) 2.5 observations per dof (b) 4.26 observations per dof

Figure 3.9: Random diagonal observations pattern for the dinosaur sequence: Fig. 3.9a shows a sampling pattern with 2.5 observations per degree of freedom whereas Fig. 3.9b uses 4.26 observations per degree of freedom which is equal to the number of observations for the real data used in Sec. 3.8.4.

i.e. affine camera matrices were computed as the best possible affine camera approximation to the given ground truth cameras. This step eliminates the biased error due to the affine camera approximation and allows to investigate the performance for structure-from-motion data in an idealized situation for which the low-rank model is exact. Gaussian noise with $\sigma_{noise} = 1$ pixel was added to the projected points (the resolution of the images is 720×576).

Random sampling patterns were generated from the full observation matrix, similarly to the previously presented synthetic experiments, with the exception that always two consecutive entries in the data matrix $\mathbf{W}$ were deleted, since if the x-coordinate of a point is not observed, then y-coordinate can not be observed either. Fig. 3.9 shows some representative diagonal sampling patterns. Note that for the computation of the number of observations per degree of freedom, the x- and y-coordinate counted as separate observations. Fig. 3.10 shows that the results with semi-synthetic structure-from-motion data look much similar to the results obtained previously with uniform sampling of row and column subspaces from the Grassmannian manifold.

Evaluation on Real Data

In this experiment, the original dinosaur dataset is used, with outliers removed as described previously. Hence, there is an inherent error due to the approximate nature of affine cameras. Out of the complete $2F \times N = 2 \cdot 36 \times 312$ data matrix $\mathbf{W}$, 77% entries were unobserved which translates to 4.26 observations per degree of freedom (the number of degree of freedoms equals $2F \cdot 3 + 3 \cdot N - 3 \cdot 3 + 2F$). The pattern of

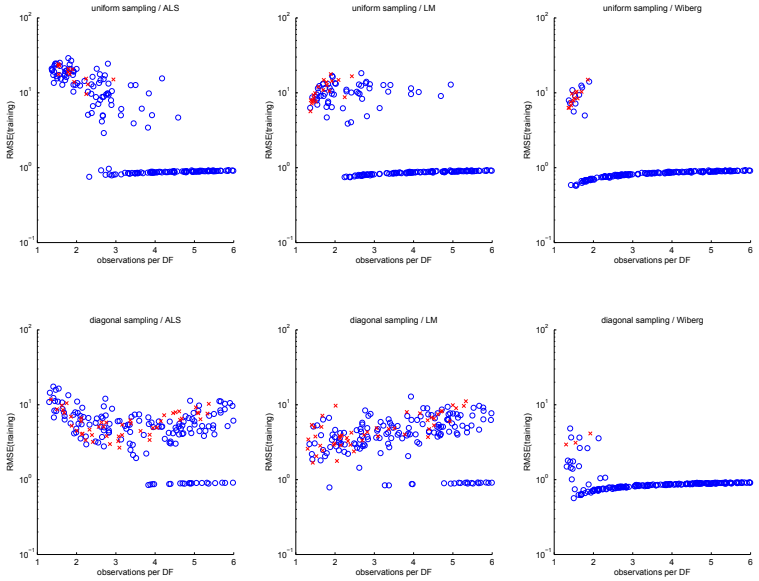


Figure 3.10: Training error $\frac{1}{\sqrt{m}} \|\mathbf{M} \odot (\mathbf{PX} - \mathbf{W})\|_F$ for semi-synthetic dinosaur sequence with noise standard deviation of σ_{noise} = 1 pixel: For the dino-sequence with $F = 36$, $N = 312$, 1 observation per degree of freedom translates to 5.45% observed entries out of the total $2FN$ entries of the complete data matrix $\mathbf{W}$.

known entries is visualized in Fig. 3.11.

The three algorithms (alternating least squares, Levenberg-Marquardt, and Wiberg) were initialized identically 200 times from random starting points. Convergence was detected if the directional derivative was less than 10^{-8} or if the absolute difference in the objective function between two consecutive iterations was less than 10^{-8}. Since the cost per iteration varies considerably between the algorithms, an upper limit of 3000 iterations was used for alternating least squares whereas for Levenberg-Marquardt and Wiberg an upper limit of 300 iterations was used. Tab. 3.1, Tab. 3.2, Tab. 3.3 summarize performance statistics of

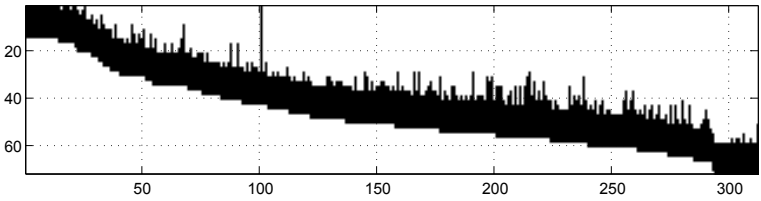


Figure 3.11: Observation pattern for the dinosaur sequence: The pronounced band diagonal pattern in the $2F \times N = 2 \cdot 36 \times 312$ data matrix $\mathbf{W}$ corresponds to 77% missing entries or 4.26 observations per degree of freedom.

Algorithm	mean	median	min	max	converged
ALS	1522.57	1503	96	2973	60.5%
LM	166.362	180	11	299	92.5%
Wiberg	44.315	33	10	180	100%

Table 3.1: Number of required iterations until convergence: The statistics are computed only over the converged runs. The last column indicates the fraction of runs which actually converged. Note that a converged run does not necessarily mean, that the algorithm converged in a global minimum.

these three algorithms. Note that these statistics are only computed over the converged runs. All the measurements were taken on a PC with 2.66GHz Intel Core 2 Quad processor on single-threaded Matlab R2010b. The implementations are not fully optimized, but the timing measurements give a rough idea of the expected order of the running times for fully optimized implementations.

Fig. 3.12 shows the quality of the solutions obtained by the algorithms as cumulative histograms of the 200 runs. These cumulative histograms plot the number of runs which achieved a validation error less than a certain value. The validation error is computed by splitting the available observations uniformly at random into a training set and a validation set (90% training and 10% validation). Hence, actually only $4.26 \cdot 0.9 \approx 3.83$ observations per degree of freedom were availabel for the optimization. The mean reprojection error over the remaining 10% of the points in

Algorithm	mean	median	min	max
ALS	24.3761	28.7625	1.06374	36.8557
LM	22.5693	23.7509	0.815505	39.8557
Wiberg	14.7841	11.0878	3.0743	60.5895

Table 3.2: Overall running time (in seconds).

Algorithm	mean	median	min	max
ALS	0.0115673	0.0115539	0.0110785	0.0123617
LM	0.124534	0.127875	0.0741368	0.13105
Wiberg	0.329286	0.329186	0.302743	0.363601

Table 3.3: Time per iteration (in seconds).

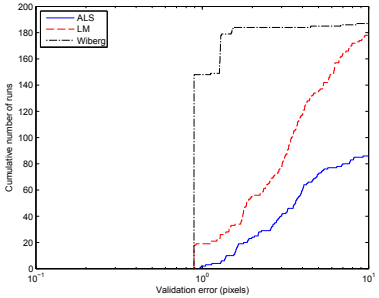
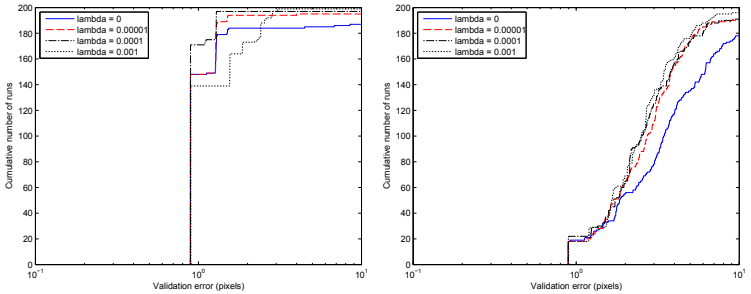


Figure 3.12: Cumulative histogram over 200 runs with random initialization: This histogram shows the number of runs which converged toward a solution with a validation error less than a certain value. The Wiberg algorithm clearly outperforms its two competitors.

the validation set is then used as validation error.



(a) Wiberg algorithm (b) Levenberg-Marquardt

Figure 3.13: Influence of the soft-penalty term: By choosing an appropriate value for λ in Eq. (3.24), the performance of both the Levenberg-Marquardt and the Wiberg algorithm can be slightly improved. In case of the Wiberg algorithm however, a large value for λ harms the performance slightly.

Evaluation on Real Data with Orthogonality Constraints

The results presented in the previous sections are based on solutions w.r.t. an affine coordinate reference frame. A post-processing step according to Sec. 3.8.2 is necessary to upgrade these reconstructions to a similarity frame. In contrast, in this section we follow the soft-penalty based approach presented in Sec. 3.8.3. This means that the solution after the non-linear least squares optimization is already w.r.t. a similarity coordinate frame and no subsequent step is required. Fig. 3.13 shows the cumulative histogram over 200 random initializations. A small performance gain can be achieved by choosing an appropriate weight λ. However, a value slightly too large can also harm the performance.

4 Tensor Factorization for Structure-from-Motion

In this chapter, we will see how the interaction between multiple subspaces can be modeled with a tensor decomposition known as the Tucker decomposition. Once a basis in the domain and range is chosen, a linear mapping can be represented by a matrix. In contrast, tensors model multilinear mappings and fixing a basis in each of the involved vector spaces allows to represent such a multilinear mapping as a multi-dimensional array of numbers. The derivations and formulas in this thesis are always w.r.t. a previously fixed basis if not stated otherwise. Hence, tensors will be identified with multidimensional array of numbers which are also known as multiway arrays. This view has also been used in Sec. 2.2.1 where basic tensor operations have been introduced. Not all properties of matrices generalize to tensors. For example, as we have also seen in Sec. 2.2.1, the modes of a tensor can have different ranks which is in contrast to matrices where the row and column subspace always have the same dimensionality. Tensors therefore offer more flexibility in modeling interactions between multiple subspaces as will be seen in this chapter. This chapter specifically focuses on a multi-camera structure-from-motion application which leads to additional algebraic constraints in this Tucker decomposition which can then be exploited algorithmically. Even though some parts of this chapter will be highly specific to this application, this chapter should be seen as piece from a broader context. Firstly, it shows how interactions between more than two subspaces can be modeled with tensors and how these interactions are encoded in the core tensor of the Tucker tensor decomposition. Secondly, from an algorithmic point of view, this chapter presents how a non-convex problem can sometimes be solved by exploiting problem specific properties. Obviously, if no such properties are present, then the algorithms presented in other chapters of this thesis can be adapted and applied to tensor-based formulations.

4.1 Motivation

Camera networks have gained increased importance in recent years. Existing approaches mostly use point correspondences between different camera views to calibrate such systems. However, it is often difficult or even impossible to establish such correspondences. But even without feature point correspondences between different camera views, if the cameras are temporally synchronized then the data from the cameras are strongly linked together by the motion correspondence: all the cameras observe the same motion. This chapter therefore develops the necessary theory to use this motion correspondence for general rigid as well as planar rigid motions. Given multiple static affine cameras which observe a rigidly moving object and track feature points located on this object, what can be said about the resulting point trajectories? Are there any useful algebraic constraints hidden in the data? Is a 3D reconstruction of the scene possible even if there are no point correspondences between the different cameras? And if so, how many points are sufficient? Is there an algorithm which warrants finding the correct solution to this highly non-convex problem?

This chapter, which is mostly based on our publications [AP09; AP10], addresses these questions and thereby introduces the concept of low-dimensional motion subspaces. The constraints provided by these motion subspaces enable an algorithm which ensures finding the correct solution to this non-convex reconstruction problem. The algorithm is based on multilinear analysis, matrix and tensor factorizations. Our new approach can handle extreme configurations, e.g. a camera in a camera network tracking only one single point. Results on synthetic as well as on real data sequences act as a proof of concept for the presented insights.

4.1.1 Related Work

Factorization-based solutions to the structure from motion (SfM) problem have been heavily investigated and extended ever since Tomasi's and Kanade's seminal work about rigid factorizations [TK92]. Such factorization based approaches enjoy interesting properties: e.g. given an almost affine camera these techniques provide an optimal, closed-

form[1] solution using only non-iterative techniques from linear algebra. The factorization approach, which is based on the singular value decomposition of a data matrix, has been further extended to multi-body motion segmentation [TV07], to perspective cameras [ST96], non-rigid objects [BHB00; Tor+01; Bra01; Bra05; WTW08], and articulated objects [YP08; TR05]. More flexible methods which can deal with missing data entries in the data matrix have been proposed in order to overcome shortcomings of singular value based decompositions which can not cope with such situations [BHB00; HS04; WTW08; GA02]. These approaches are all based on a method known as the alternating least squares method which is a well-known algorithm in the multilinear algebra community (a short introduction to this method will be given in Sec. 4.9).

We propose in Sec. 4.3 to model the data as a tensor rather than as a matrix because this provides valuable insight into the algebraic structure of the factorization and allows to draw from tensor decomposition techniques in related fields. By doing so rigid factorizations can be extended from the monocular setting to the multi-camera setup where the cameras are assumed to be static w.r.t. each other and to be well approximated with an affine camera model. At this point we would like to mention that there is a duality in the motion interpretation: static cameras observing a moving rigid object are completely equivalent to a moving rigid camera rig in an otherwise static rigid world. Therefore, all our reasonings also apply to the case of a moving rigid camera rig. Several methods [Tor+01; BA06] already extended the factorization approach to a two-camera setup and Svoboda et.al. [SMP05] proposed a projective multi-camera self calibration method based on rank-4 factorizations. Unfortunately, these methods all require feature point correspondences between the camera views to be known. Computing correspondences across a wide baseline is a difficult problem in itself and sometimes even impossible to solve (think of two cameras which point at two completely different sides of the rigid object or of two

[1] Throughout this thesis, the term *closed-form solution* denotes a solution which is given by following a fixed number of prescribed, non-iterative steps. Solutions provided by algorithms which iteratively refine a current best guess (e.g. such as the ones presented in Chap. 3) are thus not closed-form solutions. Stretching the notion of closed-form solutions a little further, algorithms involving matrix factorization steps such as the singular value decomposition will still be considered as closed-form, even though nowadays such matrix decompositions are often implemented iteratively.

cameras attached to a camera rig whose fields of view do not intersect at all).

Sequences from two camera views have also been investigated in order to temporally synchronize the cameras [ZI06] or to find correspondences between the camera views. Non-factorization based methods have been proposed to deal with non-overlapping camera views, e.g. hand-eye-calibration [Dan99] or mirror-based [Kum+08]. These methods make strong assumptions about the captured data of each camera since in a first step, a reconstruction for each camera is usually computed separately. Wolf's and Zomet's approach [WZ06] is most closely related to ours. In this work, a two-camera setting is investigated where the points tracked by the second camera are assumed to be expressible as a linear combination of the points in the first camera. This formulation even covers non-rigid deformations and motivated our work on non-rigid factorizations in Chap. 5. However, the available data from the two cameras are treated asymmetrically and are not fused uniformly into one consistent solution. Even worse, if the first sequence can not provide a robust estimate of the whole motion and structure on its own then this method is doomed to failure. In contrast, our method readily fuses partial observations from any number of cameras into one consistent and more robust solution. The monocular structure from planar motion problem has previously attracted some interest [LC05; VO02]. However, these approaches either resort to iterative solutions or require additional information, like the relative position of the plane of rotation w.r.t. the camera.

4.1.2 Chapter Overview

This chapter targets the difficult situation where no feature point correspondences between different camera views are available or where it is even impossible to establish such correspondences due to occlusions: each camera is thus allowed to track its own set of feature points. The only available correspondence between the cameras is the *motion correspondence*: all the cameras observe the same rigid motion. This chapter presents a thorough analysis of the geometric and algebraic structure contained in 2D feature point trajectories in the camera image planes. It unifies our previous analysis for general rigid motions [AP09] with our analysis of planar rigid motion [AP10]. Planar motions are probably the most important special case of rigid motions. Vehicles moving on the

street, traffic surveillance and analysis represent prominent examples. Even data from a camera rig mounted on a moving car behaves according to the above described setting: the camera rig can be considered as stationary and the whole surrounding world as a moving rigid object. Because the car is moving on the ground plane, the motion is restricted to a planar motion.

We decided to use a tensor-based formulation of the affine SfM factorization problem. The reasons which have lead to this decision will be explained shortly in Sec. 4.1.3. For the specific SfM problem at hand, there are two major insights gained by such a tensorial formulation:

i) As a theoretical contribution, the formulation readily reveals that any trajectory seen by any camera is restricted to a low dimensional linear subspace, specifically to a 13D motion subspace for general rigid motions and to a 5D motion subspace for planar rigid motions.

ii) In practical terms, the rank constraint of the motion subspaces together with the knowledge of the algebraic structure contained in the data enables a closed-form solution to the SfM problem, for both the general rigid motions (Sec. 4.6) and the planar rigid motions (Sec. 4.7).

It is interesting to note that even though the multilinear formulation stays almost the same for both the general and the planar rigid motions, the actual algorithm to compute a closed-form solution changes drastically. Our algorithms introduce several new techniques and tricks which might prove useful for other factorization problems, as well.

The applicability of our algorithm is shown on synthetic as well as on real data sequences in Sec. 4.10 and Sec. 4.11. We even show how to calibrate a camera which only tracks one single feature point which is not in correspondence with any other point tracked by any other camera. The chapter concludes in Sec. 4.12 by presenting some ideas for potential future research.

4.1.3 Why Tensors?

As we will derive in Sec. 4.3, the algebraic constraints hidden in feature point trajectories due to a common rigid motion are most easily captured by a trilinear tensor interpretation. This enables an intuitive and concise

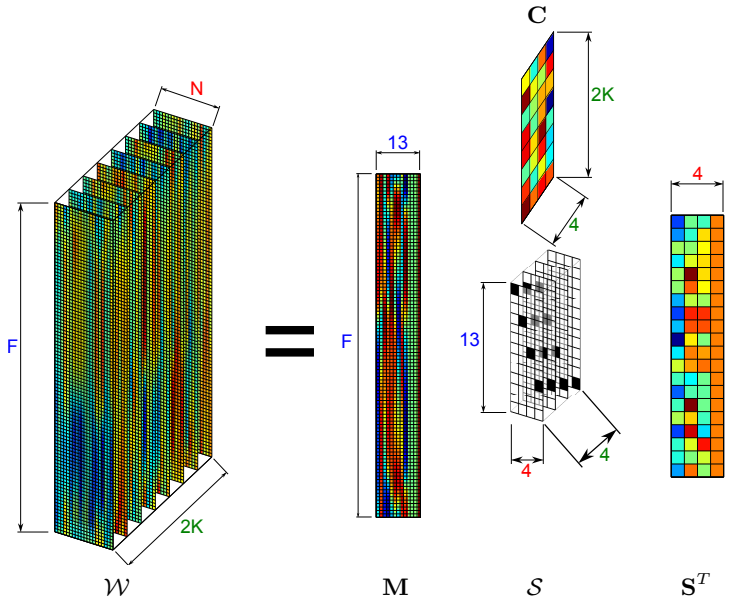


Figure 4.1: The data tensor $\mathcal{W}$ containing the feature point tracks can be viewed as a linear combination of 3$^{\text{rd}}$-order tensors each of which equals the outer product of three vectors (so called simple tensors). The coefficients for this linear combination are stored in the core tensor $\mathcal{S}$, which in our case simply consists of only 13 non-zero entries (visualized as black squares). Stacking all the vectors of these simple tensors according to their mode next to each other reveals the motion matrix $\mathbf{M}$, the camera matrix $\mathbf{C}$, and the coordinates of the points $\mathbf{S}$. The difficulty lies in decomposing the given data tensor into these 13 simple tensors in the presence of missing data entries.

formulation and interpretation of the data, as visualized in Fig. 4.1 (The notation used in this figure will described in detail in Sec. 4.3).

Getting familiar with tensor formulations requires some effort, but

as we will shortly see in more detail in the upcoming sections, tensor notation offers several advantages to formalize and unify the various factorization approaches. Firstly, the derivation of rank constraints on certain matrices follows directly from tensor decomposition analysis (Sec. 4.3). This avoids a cumbersome formulation on how to reorder the coordinates of feature point tracks into a matrix. Secondly, ambiguities (Sec. 4.4) and degenerate cases (Sec. 4.8) are discovered more easily and proofs are simplified. Finally, a tensor-based formulation establishes a link to other fields of research dealing with similar problems, such as microarray data analysis in bioinformatics, blind source separation problems in signal processing, or collaborative filtering in machine learning. A common formulation shared between different communities allows to share ideas more easily. For example iterative schemes (like the ones presented in Sec. 4.9) developed especially for multilinear problems can easily be applied if a more general framework and notation is at hand.

4.2 Notation

K is the total number of static cameras, $k \in [K]$ denotes one specific camera, F is the total number of observed frames and $f \in [F]$ labels one specific frame. The number of tracked feature points in camera k is given by N_k. Coordinates in an affine world coordinate frame will be denoted with a tilde $\tilde{\mathbf{A}}$ whereas coordinates in an Euclidean frame will simply be stated as a bold letter $\mathbf{A}$. As we will see later on, some matrices appearing in our formulation must comply with a certain algebraic structure. A matrix which spans the same subspace as matrix $\mathbf{A}$ but which does not comply with the algebraic structure for $\mathbf{A}$ prescribed by the problem at hand is denoted with a hat $\hat{\mathbf{A}}$.

4.3 Applying Tensor Algebra to SfM Problems

This section applies the techniques introduced in Sec. 2.2 to the structure from motion (SfM) problem for affine cameras. The rigid monocular affine SfM problem was introduced in the seminal work by Tomasi and Kanade [TK92] (see also Sec. 3.8). In the following two sections, this approach is extended to the case of multiple cameras, firstly when the cameras observe general rigid motions, and secondly when the cameras

observe a planar motion. Throughout the derivation, we ask the reader to keep the illustration in Fig. 4.1 in mind which shows a graphical illustration of the tensor decomposition of the structure-from-motion data tensor.

4.3.1 General Rigid Motion: 13D Motion Subspace

For the following derivation, the x- and y-axis of a camera are initially treated separately. Hence, 1D projections of 3D points onto a single camera axis will be considered first. The affine projection $\mathcal{W}_{[k,f,n]}$ of the n^{th} feature point with homogeneous coordinates $\mathbf{s}_n \in \mathbb{R}^{4\times 1}$ undergoing a rigid motion $\begin{bmatrix}\mathbf{R}_f & \mathbf{t}_f\end{bmatrix}$ at frame f, onto the k-th affine camera axis $\mathbf{c}_k^T \in \mathbb{R}^{1\times 4}$ reads like

$$\begin{aligned}\mathcal{W}_{[k,f,n]} &= \mathbf{c}_k^T \begin{bmatrix}\mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0} & 1\end{bmatrix} \mathbf{s}_n = \operatorname{vec}\left(\begin{bmatrix}\mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0} & 1\end{bmatrix}\right)^T \begin{bmatrix}\mathbf{s}_n \otimes \mathbf{c}_k\end{bmatrix} \\ &= \begin{bmatrix}\operatorname{vec}(\mathbf{R}_f)^T & \mathbf{t}_f^T & 1\end{bmatrix} \mathcal{S}_{(f)} \begin{bmatrix}\mathbf{s}_n \otimes \mathbf{c}_k\end{bmatrix}, \end{aligned} \tag{4.1}$$

where $\mathcal{W}_{[k,f,n]} = \mathcal{W}_{[k,f,n]}^T \in \mathbb{R}$ and the Kronecker product property of Eq. (2.3) have been used in the second step. In the last line, we introduced the core tensor $\mathcal{S} \in \mathbb{R}^{4\times 13\times 4}$ flattened along the temporal mode in order to get rid of the zero columns from the vectorized rigid motion. This flattened tensor thus looks like

$$\mathcal{S}_{(f)} = \begin{bmatrix}\mathbf{I}_3 \otimes \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_{3\times 1}\end{bmatrix} & \mathbf{0}_{9\times 4} \\ \mathbf{0}_{4\times 12} & \mathbf{I}_4\end{bmatrix} \in \mathbb{R}^{13\times 16}. \tag{4.2}$$

We recall that by Eq. (2.5), the core tensor captures the interactions between the involved subspaces of the data tensor. The camera axes of all the K cameras can be stacked vertically into a camera matrix $\mathbf{C} = \begin{bmatrix}\Downarrow_k \mathbf{c}_k^T\end{bmatrix} \in \mathbb{R}^{2K\times 4}$ (each camera has a x- and y-axis). In a similar way, the tracked feature points can be stacked into a structure matrix $\mathbf{S} = \begin{bmatrix}\Rightarrow_n \mathbf{s}_n\end{bmatrix} \in \mathbb{R}^{4\times \sum_k N_k}$. By introducing the motion matrix

$$\mathbf{M} = \begin{bmatrix}\Downarrow_f \begin{bmatrix}\operatorname{vec}(\mathbf{R}_f)^T & \mathbf{t}_f^T & 1\end{bmatrix}\end{bmatrix} \in \mathbb{R}^{F\times 13}$$

we finally get the equation for the coordinates of the trajectory of the n^{th} feature point projected onto the k^{th} camera axis $\mathcal{W}_{[k,:,n]} =$

$\mathbf{M}\mathcal{S}_{(f)}[\mathbf{S}_{[:,n]}^T \otimes \mathbf{C}_{[k,:]}]^T$. Fixing a column ordering scheme $\Rightarrow_{n,k}$ consistent with the Kronecker product, we derive the equation for a 3rd-order data tensor $\mathcal{W} \in \mathbb{R}^{2K \times F \times \sum_k N_k}$ flattened along the temporal mode f

$$\mathcal{W}_{(f)} = \left[\Rightarrow_{n,k} \mathcal{W}_{[k,:,n]}\right] = \mathbf{M}\mathcal{S}_{(f)}[\mathbf{S}^T \otimes \mathbf{C}]^T. \tag{4.3}$$

This leads to the following

Observation Any trajectory over F frames of a feature point on an object which transforms rigidly according to $\mathbf{R}_f$ and $\mathbf{t}_f$ at frame f and observed by any static affine camera axis is restricted to lie in a 13-dimensional subspace of a F-dimensional linear space. This subspace is spanned by the columns of the rigid motion matrix

$$\mathbf{M} = \left[\Downarrow_f \left[\mathrm{vec}\,(\mathbf{R}_f)^T \quad \mathbf{t}_f^T \quad 1\right]\right] \in \mathbb{R}^{F \times 13}, \tag{4.4}$$

and is independent of both the camera axis and the coordinates of the feature point.

Eq. (4.3) exactly corresponds to a Tucker decomposition of the data tensor flattened along the temporal mode with a core tensor $\mathcal{S}$. The original tensor is therefore given by consistently reordering the elements of the flattened core tensor into a 3rd order tensor $\mathcal{S} \in \mathbb{R}^{4 \times 13 \times 4}$ and by applying the three mode-i products between the core tensor and the mode-f, mode-k, and mode-n subspaces $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{S}^T$, respectively:

$$\mathcal{W} = \mathcal{S} \times_f \mathbf{M} \times_k \mathbf{C} \times_n \mathbf{S}^T \tag{4.5}$$

Note that f, k, and n are used for readability reasons as labels for the mode-i product along the temporal mode, the mode of the camera axes, or the mode of the feature points, respectively. This derivation clearly shows the trilinear nature of the image coordinates of the projected feature trajectories.

4.3.2 Planar Rigid Motion: 5D Motion Subspace

The rotation around an axis $\mathbf{a}$ by an angle α can be expressed as a rotation matrix $\mathbf{R}_{\mathbf{a},\alpha} = \cos\alpha \mathbf{I}_3 + (1 - \cos\alpha)\mathbf{a}\mathbf{a}^T + \sin\alpha\,[\mathbf{a}]_\times$. Rotation matrices $\mathbf{R}_{\mathbf{a},\alpha}$ around a fixed axis $\mathbf{a}$ are thus restricted to a three dimensional subspace in nine dimensional Euclidean ambient space

$$\mathrm{vec}\,(\mathbf{R}) = \left[\mathrm{vec}\,(\mathbf{I}_3) \quad \mathrm{vec}\left(\mathbf{a}\mathbf{a}^T\right) \quad \mathrm{vec}\left([\mathbf{a}]_\times\right)\right] \begin{pmatrix} \cos\alpha \\ 1 - \cos\alpha \\ \sin\alpha \end{pmatrix}.$$

Let the columns of $\mathbf{V} = [\mathbf{a}]_\perp \in \mathbb{R}^{3\times 2}$ denote an orthonormal basis for the orthogonal complement of the rotation axis $\mathbf{a}$, i.e. these columns span the plane orthogonal to the rotation axis. A rigid motion induced by this plane (i.e. the rotation is around the plane normal and the translations are restricted to shifts inside the plane) is then given by

$$\begin{pmatrix} \mathrm{vec}\left(\mathbf{R}_{\mathbf{a},\alpha}\right) \\ \mathrm{vec}\left(\mathbf{V}\mathbf{t}\right) \\ 1 \end{pmatrix} = \begin{bmatrix} \mathrm{vec}\left(\mathbf{I}_3\right) & \mathrm{vec}\left(\mathbf{a}\mathbf{a}^T\right) & \mathrm{vec}\left([\mathbf{a}]_\times\right) & \mathbf{0}_{9\times 2} \\ \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 1} & \mathbf{0}_{3\times 1} & \mathbf{V} \\ 1 & 1 & 0 & \mathbf{0}_{1\times 2} \end{bmatrix} \begin{pmatrix} \cos\alpha \\ 1-\cos\alpha \\ \sin\alpha \\ \mathbf{t} \end{pmatrix}, \tag{4.6}$$

which shows that *any* rigid motion in this plane is restricted to a five dimensional subspace of 13-dimensional (or 16 if zero-entries are not disregarded) Euclidean space. Interestingly, by noting that the space of symmetric rank-1 matrices $\mathrm{vec}\left(\mathbf{a}\mathbf{a}^T\right)$ considered as a linear space is 6 dimensional, we see that rotations around at least *five different axes of rotation* are required to span the full 13-dimensional space (the vector space of skew-symmetric matrices $[\mathbf{a}]_\times$ is 3 dimensional and thus rotations around 3 different axes already span this space, whereas the identity matrix is also symmetric and therefore only 5 remaining linear degrees of freedom of the 3×3-symmetric rank-1 matrices must be provided by additional rotation axes).

Plugging the representation of Eq. (4.6) into the motion matrix of Eq. (4.4) leads to

$$\begin{aligned} \mathbf{M} = &[\Downarrow_f \left(\cos\alpha_f, (1-\cos\alpha_f), \sin\alpha_f, \mathbf{t}_f^T\right)] \cdot \\ &\begin{bmatrix} \mathrm{vec}\left(\mathbf{I}_3\right)^T & \mathbf{0}_{1\times 3} & 1 \\ \mathrm{vec}\left(\mathbf{a}\mathbf{a}^T\right)^T & \mathbf{0}_{1\times 3} & 1 \\ \mathrm{vec}\left([\mathbf{a}]_\times\right)^T & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{2\times 9} & \mathbf{V}^T & \mathbf{0}_{2\times 1} \end{bmatrix}, \end{aligned} \tag{4.7}$$

which shows that the temporally varying variables separate from the temporally constant variables, namely the axis of rotation $\mathbf{a}$ and the plane of rotation $\mathbf{V}$. When combining Eq. (4.7) with Eq. (4.3), the temporally constant matrix built from the axis of rotation and plane of

rotation can be absorbed into a new core tensor

$$\mathcal{C}_{(f)} = \begin{bmatrix} \mathrm{vec}\,(\mathbf{I}_3)^T & \mathbf{0}_{1\times3} & 1 \\ \mathrm{vec}\left(\mathbf{a}\mathbf{a}^T\right)^T & \mathbf{0}_{1\times3} & 1 \\ \mathrm{vec}\left([\mathbf{a}]_\times\right)^T & \mathbf{0}_{1\times3} & 0 \\ \mathbf{0}_{2\times9} & \mathbf{V}^T & \mathbf{0}_{2\times1} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{I}_3 \otimes \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times1} \end{bmatrix} & \mathbf{0}_{9\times4} \\ \mathbf{0}_{4\times12} & \mathbf{I}_4 \end{bmatrix}}_{=\mathcal{S}_{(f)}} \in \mathbb{R}^{5\times16} \quad (4.8)$$

and the number of columns in the new planar motion matrix

$$\mathbf{M} = [\Downarrow_f (\cos\alpha_f, (1-\cos\alpha_f), \sin\alpha_f, \mathbf{t}_f^T)] \quad (4.9)$$

hence reduces to 5. The resulting matrix is exactly the same as the data tensor flattened along the temporal mode $\mathbf{W} = \mathcal{W}_{(f)} = \mathbf{M}\mathcal{C}_{(f)}\left[\mathbf{S}\otimes\mathbf{C}^T\right]$, in contrast to the general rigid motion this time the matrix is only of rank 5. The data tensor is thus described again as a Tucker tensor decomposition $\mathcal{W} = \mathcal{C}\times_k\mathbf{C}\times_f\mathbf{M}\times_n\mathbf{S}^T \in \mathbb{R}^{2K\times F\times N}$ with slightly modified motion matrix $\mathbf{M}\in\mathbb{R}^{F\times5}$ (see Eq. (4.9)) and core tensor $\mathcal{C}\in\mathbb{R}^{4\times5\times4}$ as given in Eq. (4.8). We summarize these findings in

Observation Any trajectory over F frames of a feature point on an object which transforms rigidly in a plane (with plane normal $\mathbf{a}$ and orthogonal complement $\mathbf{V} = [\mathbf{a}]_\perp \in \mathbb{R}^{3\times2}$) according to $\mathbf{R}_{\mathbf{a},\alpha_f}$ and $\mathbf{V}\mathbf{t}_f$ at frame f and observed by any static affine camera axis is restricted to lie in a 5-dimensional subspace of a F-dimensional linear space. This subspace is spanned by the columns of the planar rigid motion matrix

$$\mathbf{M} = [\Downarrow_f (\cos\alpha_f, (1-\cos\alpha_f), \sin\alpha_f, \mathbf{t}_f^T)] \in \mathbb{R}^{F\times5}, \quad (4.10)$$

and is independent of both the camera axis and the coordinates of the feature point.

4.4 Ambiguities

Due to inherent gauge freedoms, a matrix or tensor factorization (and thus also a 3D reconstruction) is never unique. This section presents ambiguities arising in multi-camera SfM problems. Thanks to the tensor formulation, the derivation of such ambiguities is rather straight-forward, at least for the general rigid motion case. Note that these gauge freedoms will lead to rank deficient linear systems (e.g. see Sec. 4.6) since there is an infinite number of valid solutions. In order to solve these linear systems, it is paramount to know the dimensionality of their nullspace

which is revealed by an analysis of the ambiguities. Furthermore, the algorithm for planar rigid motions in Sec. 4.7 is strongly based on exploiting the ambiguities due to the inherent gauge freedom in order to derive a closed-form solution.

4.4.1 Multi-Camera Rigid Motion

The Tucker decomposition is known to be non-unique since a basis transformation applied to the mode-i subspace can be compensated by the mode-i product of the core tensor with the inverse of this linear transformation $\mathcal{A} \times_i \mathbf{M}_i = (\mathcal{A} \times_i \mathbf{Q}_i) \times_i \mathbf{M}_i \mathbf{Q}_i^{-1}$ (see Sec. 2.2.1). This would obviously result in changing the entries of the known core tensor (Eq. (4.2)). However, affine transformations of the camera matrix and points can be compensated by a suitable transformation of the motion subspace keeping the known core tensor thus unchanged. Let

$$\mathbf{Q}_C = \begin{bmatrix} \mathbf{R}_C & \mathbf{t}_C \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \text{ and } \mathbf{Q}_S = \begin{bmatrix} \mathbf{R}_S & \mathbf{t}_S \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$$

denote two affine transformations of the global camera reference frame and the global point reference frame, respectively. The factorization is obviously ambiguous

$$\mathcal{W}_{[:,f,:]} = \mathbf{C}\mathbf{Q}_C^{-1}\mathbf{Q}_C \begin{bmatrix} \mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \mathbf{Q}_S \mathbf{Q}_S^{-1}\mathbf{S}. \tag{4.11}$$

In tensor notation, this equation looks like

$$\begin{aligned} \mathcal{W} = &\left(\mathcal{S} \times_k \mathbf{Q}_C \times_f \mathbf{Q}_M \times_n \mathbf{Q}_S^T\right) \\ &\times_k \mathbf{C}\mathbf{Q}_C^{-1} \times_f \mathbf{M}\mathbf{Q}_M^{-1} \times_n \left[\mathbf{S}^T \mathbf{Q}_S^{-T}\right], \end{aligned}$$

where $\mathbf{Q}_M$ denotes an appropriate transformation of the motion matrix. Now the question is, how does this transformation $\mathbf{Q}_M$ have to look like in order to compensate for the affine transformations of the cameras and the points, i.e. such that the core tensor does not change? We can simply solve for $\mathbf{Q}_M$ in the equation $\mathcal{S} = \mathcal{S} \times_f \mathbf{Q}_M \times_k \mathbf{Q}_C \times_n \mathbf{Q}_S^T$ which leads to

$$\mathbf{Q}_M = \mathcal{S}_{(f)}\left[\mathbf{Q}_S^{-T} \otimes \mathbf{Q}_C^{-1}\right]^T \mathcal{S}_{(f)}^T. \tag{4.12}$$

The inverse of this transformation is then applied to the motion matrix $\mathbf{M} \leftarrow \mathbf{M}\mathbf{Q}_M^{-1}$ which compensates for the affine transformations of the

cameras and points. Note that even if we are working in an Euclidean reference frame (which means that the motion matrix fulfills certain rotational constraints) and the transformation applied to the camera and point matrices are Euclidean transformations then the implied motion matrix $\mathbf{M}\mathbf{Q}_M^{-1}$ still fulfills the rotational constraints. This clearly shows that the factorization is only unique up to two affine resp. two Euclidean transformations $\mathbf{Q}_S$ and $\mathbf{Q}_C$ which should not come as a surprise since Eq. (4.1) is a product involving three factors and hence two ambiguities arise between these factors as shown in Eq. (4.11).

4.4.2 Multi-Camera Planar Rigid Motion

A similar reasoning as in the previous section also applies in the case of planar rigid motions. The factorization is again ambiguous

$$\mathcal{W}_{[:,f,:]} = \mathbf{C}\mathbf{Q}_C^{-1}\mathbf{Q}_C \begin{bmatrix} \mathbf{R}_{\mathbf{a},\alpha_f} & \mathbf{V}\mathbf{t}_f \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \mathbf{Q}_S\mathbf{Q}_S^{-1}\mathbf{S}. \tag{4.13}$$

The planar case however requires a distinction between two components of a transformation: Any transformation can be decomposed into a component which solely acts on the space spanned by the plane of motion and a component which captures the remaining transformation. We refer to the former component as a transformation inside the plane whereas the latter is called a transformation outside the plane. Analogously to the fact that the plane at infinity is invariant under affine transformations, the plane of rotation is invariant (not point-wise, though!) under transformations inside the plane.

Interestingly in contrast to general rigid motions, only transformations $\mathbf{Q}_C$ and $\mathbf{Q}_S$ which are restricted to similarity transformations inside the plane of motion can be compensated by a corresponding transformation $\mathbf{Q}_M$ of the reference frame of the motion matrix without changing the core tensor $\mathcal{C}$. In mathematical terms, the overconstrained system $\mathcal{C} \times_k \mathbf{Q}_C \times_f \mathbf{Q}_M \times_n \mathbf{Q}_S^T = \mathcal{C}$ can be solved exactly for $\mathbf{Q}_M$, i.e. $\mathbf{Q}_M = \mathcal{C}_{(f)}\left[\mathbf{Q}_S^{-1} \otimes \mathbf{Q}_C^{-T}\right]\mathcal{C}_{(f)}^*$ if the transformations $\mathbf{Q}_C$ and $\mathbf{Q}_S$ are restricted to similarity transformations inside the plane of motion. Since the first three columns of $\mathbf{M}\mathbf{Q}_M^{-1}$ should still lead to proper rotations, the scaling factor of the similarity transformations of the cameras and points must cancel each other. The reconstruction restricted to the plane of motion is thus unique up to two similarity transformations with reciprocal scaling (one for the cameras and one for the points).

Only transformations, whose restrictions to the plane of motion are similarity transformations with reciprocal scalings, seem to allow a solution to $\mathcal{C} \times_k \mathbf{Q}_C \times_f \mathbf{Q}_M \times_n \mathbf{Q}_S^T = \mathcal{C}$. This fact will be important later on in our algorithm: Let us assume that a motion matrix has been found whose restriction to the plane of motion has proper algebraic structure, then we are guaranteed that the reconstruction restricted to this plane is uniquely determined up to a similarity transformation, which is a stronger guarantee than just being unique up to an affine transformation.

Transformations of the points or cameras outside the plane of rotation can not be compensated by a transformation of the motion. A out-of-plane transformation of the cameras has to be compensated directly by a suitable transformation of the points. Let

$$\mathbf{Z}_{\mathbf{a},\lambda} = \begin{bmatrix}\mathbf{V} & \mathbf{a}\end{bmatrix} \operatorname{diag}(\mathbf{I}_2, \lambda) \begin{bmatrix}\mathbf{V} & \mathbf{a}\end{bmatrix}^T$$

be a scaling along the rotation axis, $\mathbf{R}$ an arbitrary rotation matrix, and $\mathbf{t}_\parallel = \mathbf{a}\beta$ a translation along the rotation axis. With the camera and point transformations

$$\mathbf{Q}_C = \begin{bmatrix}\mathbf{R}\mathbf{Z}_{\mathbf{a},\lambda} & -\mathbf{R}\mathbf{Z}_{\mathbf{a},\lambda}\mathbf{t}_\parallel \\ \mathbf{0}_{1\times 3} & 1\end{bmatrix} \quad \text{and} \quad \mathbf{Q}_S = \begin{bmatrix}\mathbf{Z}_{\mathbf{a},\lambda}^{-1}\mathbf{R}^T & \mathbf{t}_\parallel \\ \mathbf{0}_{1\times 3} & 1\end{bmatrix}$$

it can be shown that $\mathcal{C}_{\mathbf{a},\mathbf{V}} \times_k \mathbf{Q}_C \times_n \mathbf{Q}_S^T = \mathcal{C}_{\mathbf{Ra},\mathbf{RV}}$ where $\mathcal{C}_{\mathbf{a},\mathbf{V}}$ denotes the core tensor with rotation axis $\mathbf{a}$ and orthogonal complement $\mathbf{V}$. Note that neither the scaling nor the translation along the rotation axis influences the core tensor or the motion matrix. Hence, there is a scaling and translation ambiguity along the axis of rotation.

In the problem we are targeting, there are no point correspondences between different cameras. In this situation there is a *per camera* scale and translation ambiguity along the rotation axis. There is still only one global out-of-plane rotation ambiguity: the transformation of the plane of rotation is still linked to the other cameras through the commonly observed planar motion, even in the presence of missing correspondences. Fortunately, as we will see later, the scale ambiguity along the rotation axis can be resolved by using orthogonality and equality of norm constraints on the camera axes. The translation ambiguities along the rotation axis however can not be resolved without correspondences between different camera views. Nevertheless, by registering the centroids of the points observed by each camera to the

same height along the rotation axis, a solution close to the ground truth can usually be recovered.

4.5 Rank-4 vs. Rank-8 Factorization

We first state the classical factorization approach for rigid motions for one single camera [TK92] in our tensor formulation. We recall the dual interpretation: either we can think of the data being generated by a moving rigid object observed by a static camera or by a moving camera observing a stationary rigid object. Let us first stick with the latter interpretation. The rigid motion is then absorbed in a temporally varying sequence of camera matrices $\mathbf{C}_f = \mathbf{C}\begin{bmatrix}\mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0}_{1\times 3} & 1\end{bmatrix} \in \mathbb{R}^{2\times 4}$. The projections of the points $\mathbf{S} = [\Rightarrow_n \mathbf{s}_n] \in \mathbb{R}^{4\times N}$ are collected in a matrix

$$\mathbf{W} = [\Downarrow_f \mathbf{C}_f][\Rightarrow_n \mathbf{s}_n] \in \mathbb{R}^{2F\times N}, \tag{4.14}$$

which is maximally of rank 4 due to the rank theorem. So where is the connection to our previously derived tensor formulation? By closer inspection of Eq. (4.14), we note that this data matrix actually exposes the structure matrix and thus equals the transpose of the previously described data tensor flattened along the mode of the points $\mathbf{W} = \mathcal{W}_{(n)}^T = [\mathbf{M} \otimes \mathbf{C}]\,\mathcal{S}_{(n)}^T\mathbf{S}$ with the motion matrix as given in Eq. (4.4). Here, the duality of the interpretation whether the camera or the structure is moving is clearly revealed.

Decomposing this data matrix using the rank-4 singular value decomposition

$$\mathbf{W} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = [\mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}][\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{V}^T] = \hat{\mathbf{P}}\hat{\mathbf{S}} \tag{4.15}$$

results in an affine reconstruction of the moving camera matrices $\hat{\mathbf{P}} = \mathbf{U}\mathbf{\Lambda}^{\frac{1}{2}}$ and the structure matrix $\hat{\mathbf{S}} = \mathbf{\Lambda}^{\frac{1}{2}}\mathbf{V}^T$. Similarly as described in Sec. 3.8.2, the affine reconstruction can be upgraded via a corrective transformation to a similarity reconstruction by computing the least squares solution of an overconstrained system of linear equations which ensures the orthogonality and equality of norms conditions between the two rows of the camera rotation matrix. [Bra01] also nicely describes the algorithm to compute this corrective transformation.

Later, we will see that the flattening along the temporal mode $\mathcal{W}_{(f)}$ is more revealing than the flattening along the mode of the points. This

is related to [Akh+11]'s notion of duality between trajectory and shape space for non-rigid motions. Note that our tensor formulation shows these results more explicitly and easily, together with all the ambiguities due to the inherent gauge freedoms. We note, that in the single camera case, $\mathcal{W}_{(f)}$ will maximally be of rank 8 instead of 13. This is due to the fact that the rank of a Kronecker product between two matrices equals the product of the rank of its two factors. The rank of matrix $\mathbf{C}$ is only two in the single camera case, and hence the matrix $\mathbf{S}^T \otimes \mathbf{C}$ in Eq. (4.3) is of rank 8. Interestingly, for rigid planar motions, even a single camera already spans the complete 5D motion space since the rank of $\mathcal{W}_{(f)}$ will be upper bounded by 5 anyway.

The major question in the upcoming two sections is how to find a corrective transformation when multiple cameras are observing the very same rigid body motion, but no feature point correspondences between different cameras are available. This corrective transformation should ensure orthogonality constraints on rotation or camera matrices as well as ensure a correct algebraic Kronecker structure. This problem is closely linked to the problem of finding a corrective transformation in non-rigid SfM formulations using blend shapes. We refer to Sec. 5.4.3 and [XCK04] for some insights on how to compute the corrective transformation in closed-form in the case of the non-rigid blend shape formulation.

4.6 Rank-13 Factorization

With the previously derived formulation, our problem can be restated in the following way. Given the knowledge of certain elements of the 3^{rd}-order tensor $\mathcal{W}$, compute the underlying mode-f, mode-n, and mode-k subspaces ($\mathbf{M}$, $\mathbf{S}^T$, and $\mathbf{C}$, respectively) which generate the data tensor according to Eq. (4.5). Our problem thus essentially boils down to a multilinear tensor factorization problem. If there is no missing data, i.e. $\forall k, f, n : \mathcal{W}_{[k,f,n]}$ is known, the computation of the Tucker decomposition [Tuc66] is straight forward and the unknown subspaces $\mathbf{M}$, $\mathbf{S}$, and $\mathbf{C}$ are directly revealed by this decomposition. Missing entries in the data tensor however prevent the application of the standard Tucker decomposition algorithm. If the pattern of missing entries is completely unstructured, we must resort to iterative factorization methods. The problem of matrix factorization in the presence of missing or outlying measurements pops up in many different fields of research,

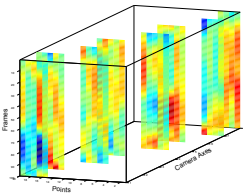
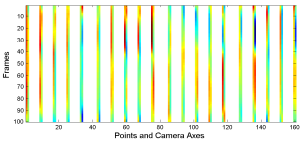
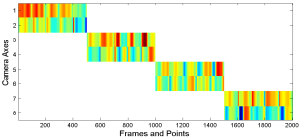
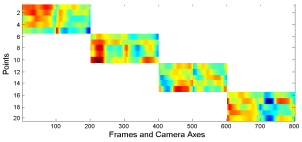
Formal Representation	Graphical Illustration
$\mathcal{W} = \mathcal{S} \times_k \mathbf{C} \times_f \mathbf{M} \times_n \mathbf{S}^T$ $\mathcal{W} \in \mathbb{R}^{2K \times F \times \sum_k N_k}$	
$\mathcal{W}_{(f)} = \mathbf{M}\mathcal{S}_{(f)}(\mathbf{S}^T \otimes \mathbf{C})^T$ $\mathcal{W}_{(f)} \in \mathbb{R}^{F \times 2K \sum_k N_k}$	
$\mathcal{W}_{(k)} = \mathbf{C}\mathcal{S}_{(k)}(\mathbf{S}^T \otimes \mathbf{M})^T$ $\mathcal{W}_{(k)} \in \mathbb{R}^{2K \times F \sum_k N_k}$	
$\mathcal{W}_{(n)} = \mathbf{S}^T\mathcal{S}_{(n)}(\mathbf{C} \otimes \mathbf{M})^T$ $\mathcal{W}_{(n)} \in \mathbb{R}^{\sum_k N_k \times 2KF}$	

Figure 4.2: If there are no feature point correspondences between different camera views then the data tensor $\mathcal{W}$ has many missing data entries (missing data entries are visualized transparently). Along the third order data tensor itself, its three flattened versions are shown as well. Note that only $\mathcal{W}_{(f)}$ has completely known columns which allows to compute a basis of the motion subspace span $(\mathbf{M})$. Due to the block-diagonal structure of the known data entries, the remaining two flattened tensors cannot be used to compute a globally consistent subspace for the camera matrices or the coordinates of the points.

Algorithm 2: Closed-form algorithm for general rigid motions.

Input: Feature points trajectories in measurement matrix $\mathbf{W} \in \mathbb{R}^{F \times 2 \sum_k N_k}$

Output: Rigid motion $\mathbf{M} \in \mathbb{R}^{F \times 13}$, camera matrix $\mathbf{C} \in \mathbb{R}^{2K \times 4}$, and points $\mathbf{S} \in \mathbb{R}^{4 \times \sum_k N_k}$

1 Rank-13 factorization according to Eq. (4.16) ; `// Sec. 4.6.1`
2 Stratified upgrade ; `// Sec. 4.6.2`
3 Affine upgrade ; `// Sec. 4.6.2`
4 Affine cameras ; `// Sec. 4.6.2`
5 Enforcing Kronecker-structure ; `// Sec. 4.6.2`
6 Similarity upgrade ; `// Sec. 4.6.2`

such as bioinformatics, signal processing, and collaborative filtering. This chapter however focuses on how a certain pattern of missing entries can be exploited together with the very specific structure of the SfM problem in order to derive a closed-form solution. This solution is based on several sequential steps which are summarized by Alg. 2 from high level whereas the upcoming sections provide a detailed description of each individual step. Less importance is put on iterative approaches in this chapter, since we have already seen several of those for bilinear matrix factorizations in the previous chapter. Sec. 4.9 however shows how to extend alternating least squares and the Wiberg algorithm to the trilinear problem we are faced with in this chapter. We also refer to [AXS08] for an algebraically oriented approach for bilinear matrix factorizations if the pattern of missing entries follows a certain pattern known as the Young diagram and references therein for other approaches.

4.6.1 Missing Correspondences

Let us now consider the setting where each camera k observes its own set of feature points $\mathbf{S}^k \in \mathbb{R}^{4 \times N_k}$ and hence, there are no correspondences available between different camera views. In this case, each camera no longer observes every point, i.e., there are tuples (k, n) for which the value $\mathcal{W}_{[k,f,n]}$ is unknown. Without loss of generality we can assume that the points in $\mathbf{S}$ are given by stacking the individual $\mathbf{S}^k$ next to each other $\mathbf{S} = \left[\Rightarrow_k \mathbf{S}^k \right]$. As we can see in Fig. 4.2, only the flattened tensor

$\mathcal{W}_{(f)}$ along the temporal mode f contains some columns whose entries are all known, amongst many completely unknown columns. These known columns however still span the complete 13-dimensional mode-f subspace. Analogously to the well-known rank-4 factorization approach, this rank-13 constraint can be used to robustly decompose the known $2\sum_k N_k$ columns of the flattened data tensor $\mathcal{W}_{(f)}$ with a singular value decomposition into a product of two rank-13 matrices. Formally, the data tensor $\mathcal{W}^k \in \mathbb{R}^{2\times F\times N_k}$ of each camera is flattened along the temporal mode and the resulting matrices $\mathbf{W}^k = \mathcal{W}^k_{(f)} = \mathbf{M}\mathcal{S}_{(f)}[\mathbf{S}^k \otimes \mathbf{C}^{k\,T}]$ are concatenated column-wise in a combined data matrix $\mathbf{W} = \left[\Rightarrow_k \mathbf{W}^k\right]$. A rank-13 matrix factorization (e.g. with SVD $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$) reveals the two factors $\hat{\mathbf{M}} = \mathbf{U} \in \mathbb{R}^{F\times 13}$ and $\hat{\mathbf{A}} = \mathbf{\Sigma}\mathbf{V}^T \in \mathbb{R}^{13\times 2\sum_k N_k}$ which fulfill

$$\mathbf{W} = \mathbf{M}\mathcal{S}_{(f)}\left[\Downarrow_k \mathbf{S}^{k\,T} \otimes \mathbf{C}^k\right]^T = [\hat{\mathbf{M}}\mathbf{Q}][\mathbf{Q}^{-1}\hat{\mathbf{A}}]. \tag{4.16}$$

This factorization separates the temporally varying component (the motion) from temporally static component (the points and the cameras). The factorization is possible since all the cameras share the same temporally varying component as all of them observe the same rigid motion. However, as indicated with an unknown 13-by-13 transformation matrix $\mathbf{Q}$, the factorization provided by the singular value decomposition does not conform to the correct algebraic structure of the flattened tensor along the temporal mode. For example, the second factor $\mathcal{S}_{(f)}\left[\Downarrow_k \mathbf{S}^{k\,T} \otimes \mathbf{C}^k\right]^T$ must have a specific algebraic structure induced by the Kronecker-product but a general rank-13 factorization will yield a matrix $\hat{\mathbf{A}}$ which does not conform to this structure. The main problem is therefore to find a corrective transformation $\mathbf{Q}$ which establishes a correct algebraic structure in $\hat{\mathbf{M}}\mathbf{Q}$ and in $\mathbf{Q}^{-1}\hat{\mathbf{A}}$.

4.6.2 Stratified Corrective Transformation

Inspired by stratified upgrades for projective structure-from-motion reconstructions (where the plane at infinity is fixed first and after that, the image of the absolute conic is computed, see Chapter 10.4 in [HZ04] for more details.), we propose to use a stratified approach to compute the unknown corrective transformation matrix $\mathbf{Q} = \mathbf{Q}_{aff}\mathbf{Q}^{-1}_{kron}\mathbf{Q}_{metric}$. $\mathbf{Q}_{aff}$ isolates the camera translations from the remaining rows of $\hat{\mathbf{A}}$ and thus resembles an affine upgrade. The correct algebraic Kronecker-structure of an affine version $\tilde{\mathbf{A}}$ of $\mathbf{A}$ is enforced by $\mathbf{Q}^{-1}_{kron}$, whereas

$\mathbf{Q}_{metric}$ finally performs a similarity upgrade. The following subsections present each step in detail.

Affine Upgrade

The first step in computing $\mathbf{Q}$ consists in transforming the last column of the motion matrix $\hat{\mathbf{M}}$ such that it conforms to the one vector $\mathbf{1}$ of $\mathbf{M}$. We will call this step the affine upgrade. Specifically, we solve for $\mathbf{q}_{aff}$ in $\mathbf{1}_{F\times 1} = \hat{\mathbf{M}}\mathbf{q}_{aff}$. A guaranteed non-singular affine upgrade is then given by $\mathbf{Q}_{aff} = \begin{bmatrix}[\mathbf{q}_{aff}]_\perp & \mathbf{q}_{aff}\end{bmatrix}$, where $[\mathbf{q}_{aff}]_\perp$ denotes an orthogonal basis for the nullspace of $\mathbf{q}_{aff}$. We can not gain any more knowledge by analyzing $\hat{\mathbf{M}}$, yet. We therefore turn our attention toward $\hat{\mathbf{A}}$.

Computing Affine Cameras

As previously mentioned, in this step we look for a transformation $\mathbf{Q}_{kron}$ which ensures the correct algebraic structure of an affine reconstruction

$$\tilde{\mathbf{A}} = \mathbf{Q}_{kron}\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}} = \mathcal{S}_{(f)}\left[\Downarrow_k \tilde{\mathbf{S}}^{k\,T} \otimes \tilde{\mathbf{C}}^k\right]^T. \tag{4.17}$$

This is a bilinear problem in the unknowns $\mathbf{Q}_{kron}$, $\tilde{\mathbf{S}}$, and $\tilde{\mathbf{C}}$. Since the product between $\mathbf{Q}_{kron}^{-1}$ and $\mathbf{Q}_{metric}$ should not modify the last column of $\mathbf{Q}_{aff}$ anymore, the last column of $\mathbf{Q}_{metric}^{-1}\mathbf{Q}_{kron}$ has to be equal to $(\mathbf{0}_{1\times 12}, 1)^T$. This together with the fact that the structure of $\mathbf{Q}_{metric}$ must follow the structure in Eq. (4.12) implies that the last column of $\mathbf{Q}_{kron}$ equals $(\mathbf{0}_{1\times 12}, 1)^T$ and thus only the first twelve columns of $\mathbf{Q}_{kron}$ are actually unknown. By realizing that the last row of $\tilde{\mathbf{S}}$ corresponds to the one vector we see that the last four rows of Eq. (4.17) actually describe an over-determined linear problem in the $4\cdot 12+2K\cdot 4$ unknowns of $\mathbf{Q}_{kron,[10:13,1:12]}$ and $\tilde{\mathbf{C}}$. The resulting system can be solved linearly in the least-squared sense (see App. 4.A.1 for details).

Camera Self-Calibration

A self calibration approach can be used by assuming zero skew, principal point at origin, and a known aspect ratio equal to 1. This yields a diagonal intrinsic camera matrix $\mathbf{K}^k = \mathrm{diag}\,(s_k, s_k, 1)$ and we can use self-calibration methods using the dual quadric, see [HZ04] section 19.3.

Since we computed affine camera matrices, the dual quadric is of the form

$$\mathbf{\Omega}_\infty^* = \begin{bmatrix} \mathbf{\Omega}_{3\times3} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{1\times4} & \end{bmatrix},$$

which means we only require at least three rather than four cameras observing the scene to compute symmetric $\mathbf{\Omega}_\infty^*$. However, if there is no prior knowledge about the intrinsic camera matrices available, if there are only two cameras, or if we are not only interested in the camera matrices but also in the rigid transformations and the points observed by the cameras, the steps explained in the next subsections are necessary.

Enforcing the Kronecker-Structure

Once an affine camera matrix $\tilde{\mathbf{C}}$ is known, the originally bilinear problem reduces to a linear one

$$\tilde{\mathbf{A}} = \mathcal{S}_{(f)}[\Downarrow_k \tilde{\mathbf{S}}^{k^T} \otimes \tilde{\mathbf{C}}^k]^T = \mathbf{Q}_{kron}\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}} \tag{4.18}$$

in the unknowns $\tilde{\mathbf{S}}^k$ and $\mathbf{Q}_{kron}$. This is again an over-determined linear problem with $3\sum_k N_k + 9 \cdot 12$ unknowns since the last four rows and the last column of $\mathbf{Q}_{kron}$ are already known and the last column of $\tilde{\mathbf{S}}$ should equal the constant one vector (App. 4.A.2 provides details on how to set up and solve this system).

Metric Upgrade

There is not enough information contained in

$$\mathcal{S}_{(f)}(\tilde{\mathbf{S}}^T \otimes \tilde{\mathbf{C}})^T = \mathbf{Q}_{kron}\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}}$$

to perform the metric upgrade and we thus have to turn our attention again to the rigid motion matrix $\hat{\mathbf{M}}$. However, in contrast to Sec. 4.6.2, an affine reconstruction with a valid Kronecker structure of the rigid motion is now available. Thus, the metric correction matrix $\mathbf{Q}_{metric}$ must fulfill the algebraic structure derived in Eq. (4.12). We are therefore looking for affine transformations $\mathbf{Q}_C$ and $\mathbf{Q}_S$ such that

$$\begin{aligned} \mathbf{M} &= \left[\Downarrow_f \left((\mathrm{vec}\,(\mathbf{R}_f))^T, \mathbf{t}_f^T, 1\right)\right] \\ &= \underbrace{\left[\Downarrow_f \left((\mathrm{vec}\left(\tilde{\mathbf{R}}_f\right))^T, \tilde{\mathbf{t}}_f^T, 1\right)\right]}_{=\tilde{\mathbf{M}}=\hat{\mathbf{M}}\mathbf{Q}_{aff}\mathbf{Q}_{kron}^{-1}} \underbrace{\mathcal{S}_{(f)}[\mathbf{Q}_S^T \otimes \mathbf{Q}_C]^T \mathcal{S}_{(f)}^T}_{=\mathbf{Q}_{metric}} \end{aligned} \tag{4.19}$$

conforms to an Euclidean rigid motion matrix. Let

$$\mathbf{Q}_S = \begin{bmatrix} \mathbf{T}_S^{-1} & \mathbf{t}_S \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{Q}_C = \begin{bmatrix} \mathbf{T}_C & \mathbf{t}_C \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}.$$

Using the Kronecker product property of Eq. (2.3), the above equation Eq. (4.19) is equivalent to the set of equations

$$\mathbf{R}_f = \mathbf{T}_C \tilde{\mathbf{R}}_f \mathbf{T}_S^{-1} \tag{4.20}$$

$$\mathbf{t}_f = \mathbf{T}_C \tilde{\mathbf{R}}_f \mathbf{t}_S + \mathbf{T}_C \tilde{\mathbf{t}}_f + \mathbf{t}_C \tag{4.21}$$

for $f \in [F]$. Eq. (4.20) is in turn equivalent to $\mathbf{T}_C \tilde{\mathbf{R}}_f = \mathbf{R}_f \mathbf{T}_S$. Since $\mathbf{R}_f$ is a rotation matrix we have

$$\left[\mathbf{T}_C \tilde{\mathbf{R}}_f\right]^T \left[\mathbf{T}_C \tilde{\mathbf{R}}_f\right] = \left[\mathbf{R}_f \mathbf{T}_S\right]^T \left[\mathbf{R}_f \mathbf{T}_S\right] \tag{4.22}$$

$$= \tilde{\mathbf{R}}_f^T \mathbf{T}_C^T \mathbf{T}_C \tilde{\mathbf{R}}_f = \mathbf{T}_S^T \mathbf{T}_S. \tag{4.23}$$

This set of equations is linear in symmetric $\mathbf{T}_C^T \mathbf{T}_C$ and $\mathbf{T}_S^T \mathbf{T}_S$ and can be solved by similar techniques as the ones presented in Sec. 3.8.2 or [Bra01; Bra05] for the rigid case. Each frame provides 6 constraints on the 12 unknowns and a solution for $\mathbf{T}_C^T \mathbf{T}_C$ and $\mathbf{T}_S^T \mathbf{T}_S$ can be found given sufficiently many frames are available. A final eigenvalue decomposition of these symmetric matrices finally yields the matrices $\mathbf{T}_S$ and $\mathbf{T}_C$. These matrices are then used to render Eq. (4.21) linear in the unknowns, i.e., the translations $\mathbf{t}_S$, $\mathbf{t}_C$, and $\mathbf{t}_f$. This provides $3F$ constraints on the $3F + 6$ unknowns. The resulting linear system therefore has a six dimensional solution space which accounts for the six degrees of freedoms for choosing $\mathbf{t}_C$ and $\mathbf{t}_S$. Note that we have not made use of any orthogonality constraints on the camera axes. These orthogonality constraints implicitly imply a scaled orthographic camera model, whereas our factorization algorithm can deal with general affine cameras. On the other hand, if the dual quadric $\mathbf{Q}_\infty^*$ has been used to perform the similarity upgrade of the cameras (implicitly assuming a special form for the intrinsic calibration matrix), $\mathbf{T}_C$ can be fixed to the identity and is no longer required to be treated as an unknown. All the experiments in Sec. 4.10 are computed without the camera self-calibration approach based on $\mathbf{Q}_\infty^*$.

4.7 Rank-5 Factorization

In contrast to a rank-13 motion subspace, one camera is sufficient in order to span the complete 5 dimensional motion subspace of a planar motion. This leads to the following idea: Intuitively, a separate reconstruction can be made for each camera. These separate reconstructions are unique up to the ambiguities mentioned previously. This especially means that the reconstruction of each camera restricted to (or projected onto) the plane of rotation is a *valid* similarity reconstruction, i.e. the individual reconstructions are expressed in varying coordinate reference frames which, however, only differ from each other by similarity transformations. Using knowledge from the 5D-motion subspace, these reconstructions can then be aligned in a consistent world reference frame. If the additional assumption is made that the two camera axes of each camera are orthogonal and have equal norm (the norm can vary between different cameras) then the coordinate frame of the reconstruction can be upgraded to a similarity frame in all three dimensions. We thus end up with a consistent 3D-reconstruction.

There is a major drawback in the above algorithmic sketch. The fact that all the cameras observe the very same rigid motion is only used in the final step to align all the individual reconstructions. It is a desirable property that the information from all the cameras should be fused right at the first stage of the algorithm in order to get a more robust reconstruction. Furthermore, in order to compute the initial reconstruction of a camera, this camera needs to track at least two points. If the camera tracks only one feature point, a reconstruction based solely on this camera is *not* possible: at least two points are necessary to span the 5D-motion subspace. The algorithm which is presented in the upcoming sections on the other hand does not suffer from these shortcomings. The algorithm fuses the information from all the cameras right at the first stage and works even when each camera tracks only one single point. Last but not least, the algorithm provides a closed-form solution. Again, Alg. 3 provides an overview of the multiple sequential steps whereas the following sections give detailed explanations.

Algorithm 3: Closed-form algorithm for planar rigid motions.

Input: Feature points trajectories in measurement matrix $\mathbf{W} \in \mathbb{R}^{F \times 2\sum_k N_k}$

Output: Rigid motion $\mathbf{M} \in \mathbb{R}^{F \times 5}$, camera matrix $\mathbf{C} \in \mathbb{R}^{2K \times 4}$, and points $\mathbf{S} \in \mathbb{R}^{4 \times \sum_k N_k}$

1 Rank-5 factorization according to Eq. (4.24) ; `// Sec. 4.7.1`
2 Trigonometric upgrade ; `// Sec. 4.7.2`
3 Metric upgrade in plane of motion ; `// Sec. 4.7.3`
4 Projection onto plane of motion ; `// Sec. 4.7.3`
5 Per-camera metric reconstruction inside plane of motion ; `// Sec. 4.7.3`
6 Registration in common coordinate system in plane of motion ; `// Sec. 4.7.3`
7 Metric upgrade of component outside of plane of motion ; `// Sec. 4.7.3`

4.7.1 Rank-5 Factorization

In a similar spirit to Sec. 4.6, we can fuse the data from all the cameras in order to compute a consistent estimate of the motion matrix. But this time, a rank-5 factorization of the combined data matrix $\mathbf{W} = [\Rightarrow_k \mathcal{W}^k_{(f)}]$ reveals the correct column span $\text{span}(\mathbf{M}) = \text{span}(\hat{\mathbf{M}})$ of the motion matrix

$$\mathbf{W} = \hat{\mathbf{M}}\hat{\mathbf{A}} = \underbrace{\left[\Downarrow_f \cos\alpha_f \quad 1-\cos\alpha_f \quad \sin\alpha_f \quad t_{f,1} \quad t_{f,2}\right]}_{=\hat{\mathbf{M}}\mathbf{Q}} \cdot \underbrace{\mathcal{C}_{(f)}\left[\Rightarrow_k \mathbf{S}^k \otimes \mathbf{C}^{k\,T}\right]}_{=\mathbf{Q}^{-1}\hat{\mathbf{A}}}, \tag{4.24}$$

where we have introduced the corrective transformation $\mathbf{Q} \in \mathbb{R}^{5\times5}$ in order to establish the correct algebraic structure. If all the cameras only track two points in total, the combined data matrix $\mathbf{W}$ will then only consist of four columns and thus a rank-5 factorization is obviously impossible. Luckily, we know that the first two columns of the motion matrix in Eq. (4.9) should sum to the constant one vector. Hence, in these situations, only a rank 4 factorization of the data matrix

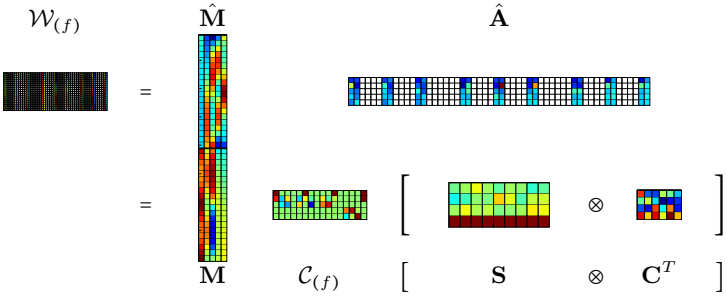


Figure 4.3: Visual representation of the rank-5 factorization. Missing data entries due to missing correspondences between different cameras are depicted transparently.

$\mathbf{W} \in \mathbb{R}^{2F\times 4}$ is performed, the resulting motion matrix is augmented with the constant one vector $\hat{\mathbf{M}} \leftarrow [\hat{\mathbf{M}}, \mathbf{1}_{F\times 1}]$ and the second factor is adapted correspondingly $\hat{\mathbf{A}} \leftarrow [\hat{\mathbf{A}}^T, \mathbf{0}_{2N\times 1}]^T$. The rest of the algorithm remains the same.

The corrective transformation is again computed in a piecewise (or stratified) way. Specifically, the corrective transformation is split into three separate transformations $\mathbf{Q} = \mathbf{Q}_{trig}\mathbf{Q}_{orient}^{-1}\mathbf{Q}_{transl}^{-1}$ where the transformation $\mathbf{Q}_{trig}$ establishes the correct trigonometric structure on the first three columns of the motion matrix, $\mathbf{Q}_{orient}$ aligns the orientations of the cameras in a consistent similarity reference frame, and $\mathbf{Q}_{transl}$ is related to correctly translate the reconstruction. The individual steps are described in detail in the next sections.

4.7.2 Trigonometric Structure

The first three columns of $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4, \mathbf{q}_5]$ can be solved for in the following way: since $\hat{\mathbf{M}}_{[f,:]}\mathbf{q}_i\mathbf{q}_i^T\hat{\mathbf{M}}_{[f,:]}^T = \mathbf{M}_{[f,i]}^2$ we have

$$1 = \hat{\mathbf{M}}_{[f,:]}\left[(\mathbf{q}_1+\mathbf{q}_2)(\mathbf{q}_1+\mathbf{q}_2)^T\right]\hat{\mathbf{M}}_{[f,:]}^T = (\cos\alpha_f + (1-\cos\alpha_f))^2 \tag{4.25}$$

$$1 = \hat{\mathbf{M}}_{[f,:]}\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right]\hat{\mathbf{M}}_{[f,:]}^T = \cos^2\alpha_f + \sin^2\alpha_f.$$

These observations lead to F constraints on symmetric rank-2 matrix $\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T$, symmetric rank-1 matrix $(\mathbf{q}_1+\mathbf{q}_2)(\mathbf{q}_1+\mathbf{q}_2)^T$, or symmetric

rank-3 matrix $b\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right] + (1-b)(\mathbf{q}_1+\mathbf{q}_2)(\mathbf{q}_1+\mathbf{q}_2)^T$ with $b \in \mathbb{R}$:

$$\begin{aligned} 1 &= \hat{\mathbf{M}}_{[f,:]}\left[(\mathbf{q}_1+\mathbf{q}_2)(\mathbf{q}_1+\mathbf{q}_2)^T\right]\hat{\mathbf{M}}_{[f,:]}^T \\ &= \hat{\mathbf{M}}_{[f,:]}\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right]\hat{\mathbf{M}}_{[f,:]}^T \\ &= \hat{\mathbf{M}}_{[f,:]}\left[b\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right] + (1-b)\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_2\mathbf{q}_2^T\right]\right]\hat{\mathbf{M}}_{[f,:]}^T \end{aligned} \tag{4.26}$$

These F equations are linear in the unknown symmetric matrices and result in a one dimensional solution space (since there is a valid solution for any $b \in \mathbb{R}$). App. 4.A.3 shows how to extract the solution vectors $\mathbf{q}_1$, $\mathbf{q}_2$, and $\mathbf{q}_3$ from this one dimensional solution space. Once this is done, the corrective transformation $\mathbf{Q}_{trig} = \left[\mathbf{q}_1 \quad \mathbf{q}_2 \quad \mathbf{q}_3 \quad [\mathbf{q}_1,\mathbf{q}_2,\mathbf{q}_3]_\perp\right]$ is applied to the first factor $\hat{\mathbf{M}}\mathbf{Q}_{trig}$ which establishes the correct trigonometric structure in the first three columns. The inverse of this transformation is applied to the second factor $\tilde{\mathbf{A}} = \mathbf{Q}_{trig}^{-1}\hat{\mathbf{A}}$. Note that the structure of the first three columns of the motion matrix should not get modified anymore and hence any further corrective transformation must have upper block-triangular structure with an identity matrix of dimension 3 in the upper left corner. The inverse of such an upper block-triangular matrix has exactly the same non-zero pattern, i.e.

$$\mathbf{Q}_{transl}\mathbf{Q}_{orient} = \begin{bmatrix}\mathbf{I}_3 & \mathbf{Q}_{3\times2}\\ \mathbf{0}_{2\times3} & \mathbf{I}_2\end{bmatrix}\begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_{3\times2}\\ \mathbf{0}_{2\times3} & \mathbf{Q}_{2\times2}\end{bmatrix} = \begin{bmatrix}\mathbf{I}_3 & \mathbf{Q}_{3\times2}\\ \mathbf{0}_{2\times3} & \mathbf{Q}_{2\times2}\end{bmatrix}.$$

4.7.3 Euclidean Camera Reference Frame

No more information can be extracted from the motion matrix and thus, we turn our attention to the second factor $\tilde{\mathbf{A}}$ which after applying a proper transformation should have the following algebraic form

$$\mathbf{A} = \begin{bmatrix}\mathbf{I}_3 & \mathbf{Q}_{3\times2}\\ \mathbf{0}_{2\times3} & \mathbf{Q}_{2\times2}\end{bmatrix}\tilde{\mathbf{A}} = \mathcal{C}_{(f)}\left[\Rightarrow_k \mathbf{S}^k \otimes \mathbf{C}^{k^T}\right]. \tag{4.27}$$

This is a particularly tricky instance of a bilinear system of equations in $\mathbf{Q}_{3\times2}$, $\mathbf{Q}_{2\times2}$, $\mathbf{S}^k$, and $\mathbf{C}^k$. We have even tried algebraic computer software to compute a valid closed-form solution, unfortunately without success. Nevertheless, we succeeded in deriving manually a solution using geometric intuition and reasoning.

Projection onto Plane of Motion

Eq. (4.27) together with the known matrix $\mathcal{C}_{(f)}$ in Eq. (4.8) tells that $\tilde{\mathbf{A}}_{[4:5,:]} = \left[\Rightarrow_k \mathbf{1}_{1\times N_k} \otimes \left[\mathbf{C}^k_{[:,1:3]}\mathbf{V}\mathbf{Q}^{-T}_{2\times 2}\right]^T\right]$, which means that the columns of $\tilde{\mathbf{A}}_{[4:5,:]}$ contain the coordinates (w.r.t. the basis $\mathbf{V}$) of the projection of the rows of the camera matrices (barring the translational component) onto the plane of rotation. These coordinates however have been distorted with a common, but unknown transformation $\mathbf{Q}_{2\times 2}$. This observation motivates the fact to restrict the reconstruction first to the plane of rotation. Such a step requires a projection of the available data onto the plane of rotation. App. 4.A.4 shows that this can be done by subtracting the second from the first row and keeping the third row of Eq. (4.27)

$$\begin{aligned}
&\begin{bmatrix}1 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}\tilde{\mathbf{A}}_{[1:3,:]} + \underbrace{\begin{bmatrix}1 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}\mathbf{Q}_{3\times 2}}_{=\mathbf{T}_{2\times 2}}\tilde{\mathbf{A}}_{[4:5,:]}\\
&= \begin{bmatrix}\mathrm{vec}\left(\mathbb{P}_\mathbf{V}\right)^T\\ \mathrm{vec}\left([\mathbf{a}]_\times\right)^T\end{bmatrix}\left[\Rightarrow_k \left[\mathbb{P}_\mathbf{V}\mathbf{S}^k_{[1:3,:]}\right]\otimes\left[\mathbb{P}_\mathbf{V}\mathbf{C}^k_{[:,1:3]}{}^T\right]\right]\\
&= \begin{bmatrix}\mathrm{vec}\left(\mathbb{P}_\mathbf{V}\right)^T\\ \mathrm{vec}\left([\mathbf{a}]_\times\right)^T\end{bmatrix}\left[\Rightarrow_k \left[\mathbb{P}_\mathbf{V}\mathbf{S}^k_{[1:3,:]}\right]\otimes\left[\mathbf{V}\mathbf{Q}_{2\times 2}\right]\left[\mathbf{Q}^{-1}_{2\times 2}\mathbf{V}^T\mathbf{C}^k_{[:,1:3]}{}^T\right]\right].
\end{aligned} \tag{4.28}$$

In the last step we have used $\mathbb{P}_\mathbf{V} = \mathbf{V}\mathbf{Q}_{2\times 2}\mathbf{Q}^{-1}_{2\times 2}\mathbf{V}^T$ and the parenthesis in the last term should stress out that for all the cameras the term $\mathbf{Q}^{-1}_{2\times 2}\mathbf{V}^T\mathbf{C}^k_{[:,1:3]}{}^T$ can be read off from $\tilde{\mathbf{A}}_{[4:5,:]}$. The unknowns of this bilinear equation are the points and the 2-by-2 transformations $\mathbf{T}_{2\times 2}$ and $\mathbf{Q}_{2\times 2}$.

Per-Camera Reconstruction in the Plane of Rotation

Eq. (4.28) describes a reconstruction problem in a plane which is still bilinear. As with any rigid reconstruction, there are several gauge freedoms. Specifically, the origin and the orientation of the reference frame can be chosen arbitrarily[2]. In the planar case, this means a

[2] The first three columns of the motion matrix have already been fixed and the translation of the cameras has been lost by the projection step. Thus, there

2D offset and the orientation of one 2D vector can be chosen freely. In the following we will make use of the gauge freedoms in order to render this bilinear problem in multiple sequential linear problems. The reconstruction procedure described in the upcoming paragraphs could be applied to one single camera. This would provide $\mathbf{T}_{2\times 2}$ and $\mathbf{Q}_{2\times 2}$ which could then be used to solve for the points in the remaining cameras. However, increased robustness can be achieved by solving the sequential linear problems for each camera separately and aligning the results in a final step in a consistent coordinate frame. For each camera, the gauge freedoms will be fixed in a different way which enables the computation of a reconstruction for each camera. The reference frames of the reconstructions then differ only by similarity transformations. This fact will be used in the next section in order to register all the reconstructions in a globally consistent reference frame.

In single camera rigid factorizations, the translational gauge freedoms are usually chosen such that the centroid of the points matches the origin of the coordinate system, i.e. $\frac{1}{N}\mathbf{S}\mathbf{1}_{N\times 1} = \mathbf{0}$. We will make the same choice $\frac{1}{N_k}\mathbf{S}^k\mathbf{1}_{N_k\times 1} = \mathbf{0}$ on a per-camera basis. Let $\tilde{\mathbf{A}}^k$ denote the columns of $\tilde{\mathbf{A}}$ corresponding to camera k. By closer inspection of Eq. (4.28) and with the Kronecker product property of Eq. (2.4) we get

$$\begin{aligned}&\left[\begin{bmatrix}1 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}\tilde{\mathbf{A}}^k_{[1:3,:]} + \mathbf{T}_{2\times 2}\tilde{\mathbf{A}}^k_{[4:5,:]}\right]\left[\frac{1}{N_k}\mathbf{1}_{N_k\times 1}\otimes\mathbf{I}_2\right]\\ =&\begin{bmatrix}\operatorname{vec}\left(\mathbb{P}_{\mathbf{V}}\right)^T\\ \operatorname{vec}\left([\mathbf{a}]_\times\right)^T\end{bmatrix}\left(\mathbb{P}_{\mathbf{V}}\mathbf{S}^k_{[1:3,:]}\frac{1}{N_k}\mathbf{1}_{N_k\times 1}\right)\otimes\left(\mathbb{P}_{\mathbf{V}}{\mathbf{C}^k_{[:,1:3]}}^T\right) = \mathbf{0}_{2\times 2}.\end{aligned}\tag{4.29}$$

The last equation followed since the centroid has been chosen as the origin. The above linear system consists of four linearly independent equations which can readily be solved for the four unknowns in $\mathbf{T}_{2\times 2}$.

The remaining two gauge freedoms are due to the arbitrary choice of the orientation of the coordinate frame inside the plane of rotation. These gauge freedoms can be chosen s.t. the first row $\begin{pmatrix}1 & 0\end{pmatrix}\mathbf{C}^k_{[:,1:3]}\mathbf{V}$ of the k^{th} camera matrix equals the known row $\begin{pmatrix}1 & 0\end{pmatrix}\mathbf{C}^k_{[:,1:3]}\mathbf{V}\mathbf{Q}^{-T}_{2\times 2}$.

is only one planar similarity transformation left from the two mentioned in Sec. 4.4.

Such a choice poses two constraints on $\mathbf{Q}_{2\times2}$

$$\begin{aligned}\begin{pmatrix}1 & 0\end{pmatrix}\mathbf{C}^k_{[:,1:3]}\mathbf{V} &= \begin{pmatrix}1 & 0\end{pmatrix}\left[\mathbf{C}^k_{[:,1:3]}\mathbf{V}\mathbf{Q}^{-T}_{2\times2}\right] \\ &= \begin{pmatrix}1 & 0\end{pmatrix}\left[\mathbf{C}^k_{[:,1:3]}\mathbf{V}\mathbf{Q}^{-T}_{2\times2}\right]\mathbf{Q}^{T}_{2\times2}. \qquad (4.30)\end{aligned}$$

Knowing $\mathbf{T}_{2\times2}$ as well as the first row of $\mathbf{C}^k_{[:,1:3]}\mathbf{V}$ implies that the remaining unknowns in every second column of $\tilde{\mathbf{A}}^k$ (i.e. the columns which depend on the first row) are only the points. This results in $2N_k$ linear equations in the $2N_k$ unknowns of the projected point coordinates $\mathbb{P}_\mathbf{V}\mathbf{S}^k_{[1:3,:]}$. After solving this system, only the entries of $\mathbf{Q}_{2\times2}$ are not yet known. The two linear constraints of Eq. (4.30) enable a reparameterization with only two parameters $\mathbf{Q}_{2\times2} = \mathbf{Q}_0+\lambda_1\mathbf{Q}_1+\lambda_2\mathbf{Q}_2$. Inserting this parameterization into Eq. (4.28) and considering only every other second column (i.e. the columns corresponding to the second row of the camera) leads to a linear system in λ_1 and λ_2 with $2N_k$ linear equations. The linear least squares solution provides the values for λ_1 and λ_2.

The above procedure works fine as long as every camera tracks at least two points. Otherwise the computation of λ_1 and λ_2 in the final step will fail because of our choice to set the mean to the origin. The coordinates of the single point are then equal to the zero vector and hence, this single point does not provide any constraints on the two unknowns. In order to avoid this problem we use the following trick: instead of choosing the origin as the mean of the points which are tracked by the camera currently under investigation, the origin is rather fixed at the mean of the points of *another* camera. Such a choice is perfectly fine as the origin can be chosen arbitrarily. The computation of $\mathbf{T}_{2\times2}$ for camera k is therefore based on the data of another camera $k' \neq k$. This trick allows to compute a reconstruction even for cameras which only track one single point.

Registration in a Common Frame Inside the Plane of Motion

After the previous per-camera reconstruction, the camera matrix restricted to the plane of motion $\mathbf{C}^k_{[:,1:3]}\mathbb{P}_\mathbf{V}$ is known for each camera. Let $\tilde{\mathbf{C}}^k$ denotes its first three columns whose projection onto the plane of rotation is correct up to a registration with a 2-by-2 scaled rotation matrix $\lambda_k\mathbf{R}_k$. On the other hand, we also know the projections

$\mathbf{C}^k_{[:,1:3]}\mathbf{V}\mathbf{Q}^{-T}_{2\times 2}$ of the camera matrices onto the plane of rotation up to an unknown distortion transformation $\mathbf{Q}_{2\times 2}$ which is the same for all the cameras. This implies $\tilde{\mathbf{C}}^k\mathbf{V}\mathbf{R}_k\lambda_k = \mathbf{C}^k_{[:,1:3]}\mathbf{V}\mathbf{Q}^{-T}_{2\times 2}$ and thus

$$\tilde{\mathbf{C}}^k\mathbf{V}\mathbf{V}^T\tilde{\mathbf{C}}^{k,T}\lambda_k^2 = \left[\mathbf{C}^k_{[:,1:3]}\mathbf{V}\mathbf{Q}^{-T}_{2\times 2}\right]\mathbf{Q}^T_{2\times 2}\mathbf{Q}_{2\times 2}\left[\mathbf{Q}^{-1}_{2\times 2}\mathbf{V}^T\mathbf{C}^k_{[:,1:3]}{}^T\right].$$

This is a linear system in the three unknowns of symmetric $\mathbf{Q}^T_{2\times 2}\mathbf{Q}_{2\times 2}$ and K scale factors λ_k^2 which is again solved in the least squares sense. Doing so provides a least squares estimate of the three unknowns of $\mathbf{Q}^T_{2\times 2}\mathbf{Q}_{2\times 2}$. An eigenvalue decomposition $\mathbf{E}\mathbf{\Lambda}\mathbf{E}^T = \mathbf{Q}^T_{2\times 2}\mathbf{Q}_{2\times 2}$ provides a mean to recover $\mathbf{Q}_{2\times 2} = \mathbf{E}^T\mathbf{\Lambda}^{\frac{1}{2}}$ which allows to express the projections of the camera matrices

$$\mathbf{C}^k_{[:,1:3]}\mathbb{P}_{\mathbf{V}} = \left[\mathbf{C}^k_{[:,1:3]}\mathbf{V}\mathbf{Q}^{-T}_{2\times 2}\right]\mathbf{Q}^T_{2\times 2}\mathbf{V}^T$$

onto the plane in one single similarity frame.

Orthogonality and Equality of Norm Constraints

As has been previously mentioned in Sec. 4.4.2, the correct scaling along the rotation axis can only be recovered by using additional constraints, like the orthogonality and equal norm constraints on the two camera axes of a camera (which implicitly assumes a partially known intrinsic calibration matrix). These constraints will be used in the following to compute the remaining projection of the camera matrix onto the axis of rotation. Due to $\mathbf{C}^k_{[:,1:3]} = \mathbf{C}^k_{[:,1:3]}[\mathbb{P}_{\mathbf{V}} + \mathbb{P}_{\mathbf{a}}]$ and $\mathbb{P}_{\mathbf{V}}\mathbb{P}_{\mathbf{a}} = \mathbf{0}$ we get

$$\lambda_k^2\mathbf{I}_2 = \mathbf{C}^k_{[:,1:3]}\mathbf{C}^k{}^T_{[:,1:3]} = \mathbf{C}^k_{[:,1:3]}\mathbb{P}_{\mathbf{V}}\mathbf{C}^k_{[:,1:3]}{}^T + \mathbf{C}^k_{[:,1:3]}\mathbb{P}_{\mathbf{a}}\mathbf{C}^k_{[:,1:3]}{}^T.$$

Thanks to the previous registration step, the projections $\mathbf{C}^k_{[:,1:3]}\mathbb{P}_{\mathbf{V}}$ are known for all cameras. As

$$\mathbf{C}^k_{[:,1:3]}\mathbb{P}_{\mathbf{a}}\mathbf{C}^k_{[:,1:3]}{}^T = \mathbf{C}^k_{[:,1:3]}\mathbf{a}\mathbf{a}^T\mathbf{C}^k_{[:,1:3]}{}^T$$

and replacing $\mathbf{C}^k_{[:,1:3]}\mathbf{a}$ by $\mathbf{w}^k$, the unknowns of the above equation become λ_k and the two components of the vector $\mathbf{w}^k$. This results in K independent 2^{nd}-order polynomial system of equations with 3 independent equations in the three unknowns $\mathbf{w}^k$ and λ_k. Straightforward algebraic manipulation will reveal the closed-form solution to this system (see App. 4.A.5 for details). Once $\mathbf{w}^k$ is recovered, the

camera matrix is given by solving the linear system $\mathbf{C}^k_{[:,1:3]}\left[\mathbb{P}_{\mathbf{V}}, \mathbf{a}\right] = \left[\mathbf{C}^k_{[:,1:3]}\mathbb{P}_{\mathbf{V}}, \mathbf{w}^k\right]$. The solution of the polynomial equation is unique up to the sign. This means that there is a per-camera sign ambiguity along the axis of rotation. Note that this is not a shortcoming of our algorithm, but this ambiguity is rather inherent due to the planar motion setting. However, the qualitative orientations of the cameras w.r.t. the rotation axis are often known. For example, the cameras might be known to observe a motion on the ground plane. Then the axis of rotation should point upwards in the camera images, otherwise the camera is mounted upside-down. Using this additional assumption, the sign ambiguity can be resolved.

Using the orthogonality and equality of norm constraints, it is tempting to omit the registration step in the plane of rotation and to directly set up the system of equations

$$\begin{aligned}\lambda_k^2\mathbf{I}_2 =& \mathbf{C}^k_{[:,1:3]}{\mathbf{C}^k_{[:,1:3]}}^T \\ =& \mathbf{C}^k_{[:,1:3]}\mathbb{P}_{\mathbf{V}}{\mathbf{C}^k_{[:,1:3]}}^T + \mathbf{C}^k_{[:,1:3]}\mathbb{P}_{\mathbf{a}}{\mathbf{C}^k_{[:,1:3]}}^T \\ =& \left[\mathbf{C}^k_{[:,1:3]}\mathbf{V}\mathbf{Q}^{-T}_{2\times2}\right]\mathbf{Q}^T_{2\times2}\mathbf{Q}_{2\times2}\left[\mathbf{Q}^{-1}_{2\times2}\mathbf{V}^T{\mathbf{C}^k_{[:,1:3]}}^T\right] + \mathbf{w}^k{\mathbf{w}^k}^T\end{aligned}$$

in the three unknowns of $\mathbf{Q}^T_{2\times2}\mathbf{Q}_{2\times2}$, the $2K$ unknowns of $\mathbf{w}^k$, and the K unknowns λ_k^2. Interestingly, these constraints on the camera axes are insufficient to compute a valid matrix $\mathbf{Q}_{2\times2}$ and valid vectors $\mathbf{w}^k$, even using non-linear local optimization methods (there are solutions with residuum 0 which however turn out to be invalid solutions). Moreover, experiments showed that this nonlinear formulation suffers from many local minima. This observation justifies the need for the registration step in the plane of motion.

Final Step

Once the first three columns of the camera matrices are known in an Euclidean reference frame, the first three rows in Eq. (4.27) become linear in the unknowns $\mathbf{Q}_{3\times2}$, $\mathbf{S}$, and the camera translations. A least squares approach again provides the solutions to the unknowns of this overdetermined linear system. The linear system has a $4+K$-dimensional nullspace in the noisefree case: 4 degrees of freedom due to the planar translational ambiguities (planar translation of the points or the cameras

# points per camera	$\text{rank}(\mathbf{A}) \stackrel{?}{=} 13$	Sec. 4.6.2 applicable
$(3, N_2 \geq 4)$	$\text{rank}(\mathbf{A}) \leq 12$	×
$(4, 4)$	$13 = 13$	✓
$(1, 3, 3)$	$13 = 13$	×
$(2, 3, 3)$	$13 = 13$	✓
$(2, 2, N_3 \geq 4)$	$13 = 13$	×
$(2, 2, 2, 2)$	$13 = 13$	×
$(2, 2, 2, 3)$	$13 = 13$	✓
$(2, 2, 2, 2, 2)$	$13 = 13$	✓

Table 4.1: Minimal cases: This table lists the number of points per camera (for example, (N_1, N_2) means the first camera observes N_1 points whereas the second tracks N_2 points) and whether the linear algorithm of Sec. 4.6.2 succeeds in computing a valid factorization or not (summarized in the last row). The first condition states that the observed points should span the complete 13-dimensional mode-f subspace. The second condition ensures that valid affine camera matrices are computable (see Sec. 4.6.2). Note that any additional data can only support the algorithm (e.g. if (N_1, N_2) works then (N_1', N_2', N_3) with $N_1' \geq N_1$ and $N_2' \geq N_2$ works as well, even if $N_3 = 1$).

can be compensated by the planar motion) and K degrees of freedom for the per-camera translation ambiguities along the axis of rotation.

4.8 Minimal Configurations

Our two algorithms require the union of all the feature trajectories spanning the complete 13- resp. 5-dimensional motion space. This poses constraints on the minimal number of camera axes, feature points, and on the rigid motion. In typical situations, the number of frames F is much larger than 13 or 5 and we can assume the rigid motion being general enough such that the whole 13 resp. 5 dimensional motion subspace gets explored. On the other hand, the constraints on the minimal number of camera axes and feature points are more interesting.

As mentioned in Sec. 4.7, the algorithm for the rank-5 factorization

can handle even the minimal case of just two points being tracked, either by one single camera, or by two cameras each of them tracking one point. Thus, the discussion about minimal configurations for the rank-5 case is less interesting than for the rank-13 factorization. The derivation of Eq. (4.5) for the rank-13 factorization assumed a rank-4 structure matrix $\mathbf{S}$. This assumption is violated if the observed object is planar. Our algorithm currently can not handle such situations and thus planar objects represent degenerated cases. Note however that each camera is allowed to track feature points which lie in a plane, as long as they are not contained in a common plane and the combined structure matrix $\left[\Rightarrow_k \mathbf{S}^k\right]$ is thus non-planar (see also the evaluation of the real data sequence in Sec. 4.10.2). A more detailed look at the algorithm for the rank-13 factorization reveals surprising properties, but requires some considerable effort using facts from tensor product spaces. The reward is some deeper insight in linear independence relationships in tensor product spaces.

4.8.1 Digression to Linear Independence in Tensor Product Spaces

Let us assume we are given full column-rank matrices $\mathbf{A}_i \in \mathbb{R}^{m \times r_{A_i}}$ and $\mathbf{B}_i \in \mathbb{R}^{n \times r_{B_i}}$ for $i = 1, \cdots, n$. We compute the Kronecker products $\mathbf{C}_i = \mathbf{A}_i \otimes \mathbf{B}_i$ and ask ourselves how long the columns of the resulting matrices continue to be linearly independent, i.e. when does the matrix $\mathbf{C} = [\Rightarrow_i \mathbf{C}_i]$ become rank-deficient. As long as either the columns of $\mathbf{A}_i$ are linearly independent from all the columns of previous $\mathbf{A}_j$ with $j < i$ or the columns of $\mathbf{B}_i$ are linearly independent from all the columns of previous $\mathbf{B}_j$ with $j < i$, concatenating the Kronecker product $\mathbf{C}_i = \mathbf{A}_i \otimes \mathbf{B}_i$ to $\mathbf{C} = [\mathbf{C}_1, \cdots, \mathbf{C}_{i-1}]$ increases the latter's rank by $r_{A_i} r_{B_i}$ and hence the columns stay linearly independent. However, as soon as both the columns of $\mathbf{A}_k$ and $\mathbf{B}_k$ become linearly dependent w.r.t. the columns of the previous matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ with $i < k$, the resulting columns of the Kronecker product $\mathbf{A}_k \otimes \mathbf{B}_k$ *might* become linearly dependent on the columns of previous Kronecker products. In order to show that, the columns of $\mathbf{A}_k$ and $\mathbf{B}_k$ are expressed as a linear

combination of the columns of the previous matrices

$$\mathbf{A}_k = \left[\Rightarrow_{i<k} \mathbf{A}_i\right]\left[\Downarrow_{i<k} \mathbf{X}_i\right] = \sum_{i<k} \mathbf{A}_i\mathbf{X}_i \tag{4.31}$$

$$\mathbf{B}_k = \left[\Rightarrow_{i<k} \mathbf{B}_i\right]\left[\Downarrow_{i<k} \mathbf{Y}_i\right] = \sum_{i<k} \mathbf{B}_i\mathbf{Y}_i. \tag{4.32}$$

Due to the bilinearity and the product property of the Kronecker product, it holds

$$\begin{aligned}\mathbf{A}_k \otimes \mathbf{B}_k &= \left[\sum_{i<k} \mathbf{A}_i\mathbf{X}_i\right] \otimes \left[\sum_{j<k} \mathbf{B}_j\mathbf{Y}_j\right] \\ &= \sum_{i<k} \underbrace{\left[\mathbf{A}_i \otimes \mathbf{B}_i\right]}_{\text{previously existing vectors}} \left[\mathbf{X}_i \otimes \mathbf{Y}_i\right] + \\ &\quad \sum_{i<k, j<k, i\neq j} \underbrace{\left[\mathbf{A}_i \otimes \mathbf{B}_j\right]}_{\text{new linearly independent vectors}} \left[\mathbf{X}_i \otimes \mathbf{Y}_j\right].\end{aligned}$$

The matrix resulting from the first sum is for sure linearly dependent on the previous matrices, as $\mathbf{A}_i \otimes \mathbf{B}_i$ capture the previously already existing vectors. The second sum however can result in new potentially linearly independent vectors. We need to answer the question: under which circumstances can we be sure that no $\mathbf{A}_i \otimes \mathbf{B}_j$ with $i \neq j$ contributes to an increase of the rank? A case distinction is necessary in order to answer this question.

1. Assume all the columns of $\mathbf{A}_i$ with $i < k$ are linearly independent *and* also all the columns of $\mathbf{B}_i$ with $i < k$ are linearly independent. Then the representation in the coefficients $\mathbf{X}_i$ and $\mathbf{Y}_i$ in Eq. (4.31) and Eq. (4.32) is unique. The only way $\mathbf{A}_k \otimes \mathbf{B}_k$ not to increase the rank is if all the $\mathbf{X}_i \otimes \mathbf{Y}_j = \mathbf{0}$ with $i \neq j$. Ignoring trivial cases where either $\mathbf{X}_i = \mathbf{0}$ for all $i < k$ or $\mathbf{Y}_j = \mathbf{0}$ for all $j < k$, this in turn implies $\mathbf{X}_i = \mathbf{0}$ and $\mathbf{Y}_i = \mathbf{0}$ for all i except at most one, say for $i = 1$. Hence, only one single summand $\mathbf{X}_i \otimes \mathbf{Y}_i$ in the first sum can be non-zero. These cases are thus easy to spot.

2. Assume the columns of $\mathbf{A}_i$ with $i < k$ are already linearly dependent, whereas the columns of $\mathbf{B}_i$ with $i < k$ are linearly independent. This implies that the representation in Eq. (4.31) is no longer unique. This is important, as the reasoning in the

previous case for $\mathbf{A}_k \otimes \mathbf{B}_k$ not to increase the rank no longer applies. Consider for example the case where we have $k-1$ different representations of the form $\mathbf{A}_k = \mathbf{A}_i \mathbf{X}_i$. Then we can deduce

$$\begin{aligned}\mathbf{A}_k \otimes \mathbf{B}_k &= \sum_{j<k} \mathbf{A}_k \otimes \mathbf{B}_j \mathbf{Y}_j = \sum_{j<k} \mathbf{A}_j \mathbf{X}_j \otimes \mathbf{B}_j \mathbf{Y}_j \\ &= \sum_{j<k} [\mathbf{A}_j \otimes \mathbf{B}_j][\mathbf{X}_j \otimes \mathbf{Y}_j],\end{aligned}$$

which shows that the new columns of $\mathbf{A}_k \otimes \mathbf{B}_k$ are just a linear combination of previously existing $\mathbf{A}_j \otimes \mathbf{B}_j$ with $j < k$ and hence the rank does not increase. This example shows that these cases are no longer as easy to spot as the cases described previously. As we will see in the next section, this example exactly covers the situation when the first camera tracks at least 4 points.

4.8.2 Returning to Rank-13 Factorizations

We realize that especially the second cases are more difficult to discover and might require detailed knowledge of the specific problem at hand. Therefore, let us look at a specific example which also sheds some light on the relationship between the previous reasonings and our rank-13 factorization problem. Assume a full rank $\mathbf{S}_1 \in \mathbb{R}^{4\times4}$, two equal $\mathbf{s}_2 = \mathbf{s}_3 = \mathbf{S}_1\mathbf{x} \in \mathbb{R}^{4\times1}$, two linearly independent $\mathbf{C}_1 \in \mathbb{R}^{3\times2}$ and $\mathbf{c}_2 \in \mathbb{R}^{3\times1}$, and finally a linearly dependent $\mathbf{c}_3 = [\mathbf{C}_1, \mathbf{c}_2]\begin{pmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{pmatrix} \in \mathbb{R}^{3\times1}$. Of course, the matrix $\mathbf{S}_1$ can be interpreted as the points tracked by the first camera, the matrix $\mathbf{C}_1$ as the first camera matrix, $\mathbf{c}_2$ and $\mathbf{c}_3$ as the first and second camera axes of the second camera, and $\mathbf{s}_2 = \mathbf{s}_3$ as a point observed by the second camera. This setup leads to

$$\begin{aligned}&\mathbf{S}_1 \otimes \mathbf{C}_1 \rightarrow 4 \cdot 2 \text{ basis vectors} \\ &\mathbf{s}_2 \otimes \mathbf{c}_2 = \mathbf{S}_1\mathbf{x} \otimes \mathbf{c}_2 \rightarrow 1 \text{ additional basis vector} \\ &\mathbf{s}_3 \otimes \mathbf{c}_3 = \mathbf{s}_3 \otimes [\mathbf{C}_1, \mathbf{c}_2]\begin{pmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{pmatrix} \\ &\qquad = \mathbf{S}_1\mathbf{x} \otimes \mathbf{C}_1\mathbf{y}_1 + \mathbf{s}_2 \otimes \mathbf{c}_2\mathbf{y}_2 \\ &\qquad = [\mathbf{S}_1 \otimes \mathbf{C}_1][\mathbf{x} \otimes \mathbf{y}_1] + [\mathbf{s}_2 \otimes \mathbf{c}_2][1 \otimes \mathbf{y}_2] \\ &\qquad \rightarrow \text{no new basis vectors.}\end{aligned}$$

In the last step, we concluded that $\mathbf{s}_3 \otimes \mathbf{c}_3$ does not provide any new linearly independent vector since it is expressible as a linear combination of previously existing vectors. The important observation is the following: if the first camera tracks at least four points then the second camera axis of any additional camera does only provide linearly dependent data (the second camera axis is redundant so to speak). A detailed analysis for each possible minimal case similar to the one above leads to the results summarized in Tab. 4.1. Note that for some cases, even though the rank of their mode-f subspace is 13, the computation of the affine cameras (Sec. 4.6.2) still fails because the points do not provide enough linear independent constraints for solving the linear system of Eq. (4.17) due to reasons akin to the one shown above. Interestingly, experiments showed that direct nonlinear iterative minimization such as the ones presented in Sec. 4.9 sometimes succeeded in solving

$$\mathcal{S}_{(f)}[\Downarrow_k \tilde{\mathbf{S}}^{k\,T} \otimes \tilde{\mathbf{C}}^k]^T = \mathbf{Q}_{kron}\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}}$$

for $\tilde{\mathbf{C}}^k$, $\tilde{\mathbf{S}}^k$, and $\mathbf{Q}_{kron}$ in cases where the linear algorithm was not applicable.

4.9 Iterative Optimization

Even thought the algorithmic part of this chapter focuses on the closed-form factorization based solution, due to practical reasons there is without doubt a need for iterative optimization methods: Firstly, the solution given by the linear algorithm described in Sec. 4.6.2 is suboptimal w.r.t. the trilinear nature of the data because sequentially solving linear problems might transfer errors from a previous step to the current step. This is especially true for data which originates from projective cameras. However, as our experiments with synthetic and real world data showed, the above mentioned closed-form solution still provides an accurate initial guess for an iterative non-linear optimization scheme. Secondly, in real world examples, it is often difficult to track feature points over all the frames even for just one camera. Feature points are usually only trackable over a couple of frames. They disappear and new ones will emerge. Each trajectory then has some missing entries and the factorization approach using a singular value decomposition is thus no longer applicable. However, thanks to the tensor formulation in Eq. (4.5) we know how the underlying algebraic structure of our data

should look like and this still provides strong constraints on the known data entries. Thus, provided enough entries are known, these entries can be used to compute a valid Tucker tensor decomposition with an iterative algorithm.

As a starting point for further work in multilinear factorization methods with missing entries, this section extends the alternating least squares and the Wiberg algorithm for bilinear matrix factorizations as presented Chap. 3 to trilinear problems which proved to work very well in our application. Handling general multilinear functions (or in case of the Wiberg algorithm to general non-linear functions) with these algorithms should then be straight-forward. The extension of the ALS algorithm to our multilinear setting is apparent once we realize that the data can be modeled as a third order tensor. This tensor can then be flattened along its three modes in alternation. A linear closed form solution is found for the subspace which has been exposed by flattening the tensor while keeping the remaining two subspace estimates fixed and by only considering the known entries. The Wiberg algorithm, which was originally developed for bilinear problems, can be adapted to the matrix factorization of $\mathcal{W}_{(f)}$ where the gradient is taken with respect to the unknown camera and structure matrices $\mathbf{C}$ resp. $\mathbf{S}$. In all our experiments, both the ALS and Wiberg optimization methods converged after just very few iterations (roughly 5 to 10 iterations) in a minimum when initialized with the closed-form solution. The closed-form solution thus seems to suit perfectly as an initial guess for an iterative refinement.

The ALS method is a first-order gradient based scheme (not steepest descent, however!). It is well known that the convergence behavior of ALS methods for general tensor decompositions with missing entries is very instable: at the beginning of the iterations the convergence is quite fast. Multilinear factorization problems have many plateaus where the local gradients approach zero. It is in these areas where the ALS scheme often gets stuck and the convergence then flatlines. In these circumstances it is advantageous to switch to a second-order method like Newton's method or the Wiberg algorithm. Based on our experience, the combination of ALS with the Wiberg algorithm proves to be very suitable to general tensor factorizations with missing entries. The next two sections therefore shortly provide an introduction to these methods, shown at the example of rigid multi-camera factorization. With the tools presented in Sec. 2.2.3 and Chap. 3, it is possible to apply these

ideas to other multilinear matrix and tensor equations.

4.9.1 Alternating Least Squares

A Rank-13 factorization minimizes the Frobenius norm between the data matrix $\mathcal{W}_{(f)}$ and its best rank-13 approximation. This can can be rewritten as a sum of squares over all the elements of the matrix

$$\begin{aligned}\Phi &= \frac{1}{2}\left\|\mathbf{M}\mathcal{S}_{(f)}\left(\mathbf{S}^T \otimes \mathbf{C}\right)^T - \mathcal{W}_{(f)}\right\|_F^2 \\ &= \frac{1}{2}\sum_{f,k,n}\left(\mathbf{M}_{[f,:]}\mathcal{S}_{(f)}\left(\mathbf{S}_{[:,n]}^T \otimes \mathbf{C}_{[k,:]}\right)^T - \mathcal{W}_{[f,k,n]}\right)^2.\end{aligned} \tag{4.33}$$

We introduce $\mathbf{w}_{(f)} = \mathrm{vec}\left(\mathcal{W}_{(f)}\right)$, $\mathbf{w}_{(k)} = \mathrm{vec}\left(\mathcal{W}_{(k)}\right)$, $\mathbf{w}_{(n)} = \mathrm{vec}\left(\mathcal{W}_{(n)}\right)$ in addition to $\mathbf{m} = \mathrm{vec}\left(\mathbf{M}\right)$, $\mathbf{c} = \mathrm{vec}\left(\mathbf{C}\right)$, and $\mathbf{s} = \mathrm{vec}\left(\mathbf{S}\right)$. The sum of squares problem in Eq. (4.33) is then equivalent to $\Phi = \frac{1}{2}\left\|\mathbf{r}\right\|_2^2$ with the residuum vector

$$\begin{aligned}\mathbf{r} &= \left[\left(\mathbf{S}^T \otimes \mathbf{C}\right)\mathcal{S}_{(f)}^T \otimes \mathbf{I}_F\right]\mathbf{m} - \mathbf{w}_{(f)} \\ &= \mathbf{\Pi}_{k\to f}\left(\left[\left(\mathbf{S}^T \otimes \mathbf{M}\right)\mathcal{S}_{(k)}^T \otimes \mathbf{I}_{2K}\right]\mathbf{c} - \mathbf{w}_{(k)}\right) \\ &= \mathbf{\Pi}_{n\to f}\left(\left[\left(\mathbf{M}^T \otimes \mathbf{C}\right)\mathcal{S}_{(n)}^T \otimes \mathbf{I}_N\right]\mathbf{s} - \mathbf{w}_{(n)}\right),\end{aligned} \tag{4.34}$$

where Eq. (2.3) has been used to expose the unknown vectors $\mathbf{m}$, $\mathbf{c}$, and $\mathbf{s}$ and the row-permutation matrices $\mathbf{\Pi}_{k\to f}$ and $\mathbf{\Pi}_{n\to f}$ reorder the rows of the residuum to match those of Eq. (4.34). For later reference, we also note the partial Jacobians of the residuum vector

$$\begin{aligned}\partial_{\mathbf{m}}\mathbf{r} &= \left[\left(\mathbf{S}^T \otimes \mathbf{C}\right)\mathcal{S}_{(f)}^T \otimes \mathbf{I}_F\right] \\ \partial_{\mathbf{c}}\mathbf{r} &= \mathbf{\Pi}_{k\to f}\left[\left(\mathbf{S}^T \otimes \mathbf{M}\right)\mathcal{S}_{(k)}^T \otimes \mathbf{I}_{2K}\right] \\ \partial_{\mathbf{s}}\mathbf{r} &= \mathbf{\Pi}_{n\to f}\left[\left(\mathbf{M}^T \otimes \mathbf{C}\right)\mathcal{S}_{(n)}^T \otimes \mathbf{I}_N\right].\end{aligned}$$

With this notation in place, the alternating least squares algorithm (ALS) is easily explained as a cyclic block-coordinate gradient descent algorithm which alternates its descent directions according to the partial derivatives of Φ w.r.t. $\mathbf{m}$, $\mathbf{c}$, and $\mathbf{s}$ (see Alg. 4). If there are unknown entries in data tensor $\mathcal{W}$, then these entries are simply omitted in the sum of squares which corresponds to only considering the known rows of the residuum vector $\mathbf{r}$. The ALS algorithm can be slightly optimized by making use of the known constant one-vector of the motion matrix

$\mathbf{M}$ and the known homogeneous coordinate of the points $\mathbf{S}$. Moreover, the linear systems can be formulated without vectorizing the unknowns which leads to smaller linear systems.

Algorithm 4: ALS applied to our trilinear factorization problem

Input: Measurements $\mathbf{w}$ and initial guesses for rigid motion $\mathbf{m}$, camera matrix $\mathbf{c}$, and points $\mathbf{s}$

Output: Affine estimates for rigid motion $\mathbf{m}$, camera matrix $\mathbf{c}$, and points $\mathbf{s}$

while *not converged* **do**

$\quad \mathbf{m} = \arg\min_{\mathbf{m}} \left\| \left[\left(\mathbf{S}^T \otimes \mathbf{C} \right) \mathcal{S}_{(f)}^T \otimes \mathbf{I}_F \right] \mathbf{m} - \mathbf{w}_{(f)} \right\|_2^2$

$\quad \mathbf{c} = \arg\min_{\mathbf{c}} \left\| \left[\left(\mathbf{S}^T \otimes \mathbf{M} \right) \mathcal{S}_{(k)}^T \otimes \mathbf{I}_{2K} \right] \mathbf{c} - \mathbf{w}_{(k)} \right\|_2^2$

$\quad \mathbf{s} = \arg\min_{\mathbf{s}} \left\| \left[\left(\mathbf{M}^T \otimes \mathbf{C} \right) \mathcal{S}_{(n)}^T \otimes \mathbf{I}_N \right] \mathbf{s} - \mathbf{w}_{(n)} \right\|_2^2$

end

4.9.2 Wiberg Algorithm

Throughout this derivation we have to distinguish between the partial derivative $\partial_{\mathbf{x}}\mathbf{f}$ of a function $\mathbf{f}$ w.r.t. input argument $\mathbf{x}$ and the total derivative $d_{\mathbf{x}}\mathbf{f}$ of $\mathbf{f}$ w.r.t. input argument $\mathbf{x}$. The latter might require the application of the chain-rule, as we will see later.

As mentioned in Sec. 3.6, the Wiberg algorithm is a variation of the Gauss-Newton algorithm with an interleaved variable projection step. However, in that section, the derivation of the Wiberg algorithm was based on the Schur-complement trick. Here, we present the more traditional derivation based on the variable elimination and a small-residuum assumption. We again minimize the norm $\Phi = \frac{1}{2}\|\mathbf{r}\|_2^2$ of the residuum vector

$$\mathbf{r} = \left[\left(\mathbf{S}^T \otimes \mathbf{C} \right) \mathcal{S}_{(f)}^T \otimes \mathbf{I}_F \right] \mathbf{m} - \mathbf{w}_{(f)}, \tag{4.35}$$

but this time Eq. (4.35) is considered as a function of only $\mathbf{c}$ and $\mathbf{s}$ since a least-squares, closed-form solution for $\mathbf{m}$ is easily computable if $\mathbf{c}$ and $\mathbf{s}$ are fixed ($\mathbf{m}$ is therefore considered as a function of $\mathbf{c}$ and $\mathbf{s}$). In analogy to Gauss-Newton, a 2^{nd}-order Taylor expansion of the

objective function at the current i^{th} iteration is minimized

$$\begin{pmatrix}\mathbf{dc}\\ \mathbf{ds}\end{pmatrix} = \arg\min_{\mathbf{dc},\mathbf{ds}} \left(\Phi_{\mathbf{c}_i,\mathbf{s}_i} + \mathbf{r}^T d_{\mathbf{c}_i,\mathbf{s}_i}\mathbf{r} \begin{pmatrix}\mathbf{dc}\\ \mathbf{ds}\end{pmatrix} + \frac{1}{2}\begin{pmatrix}\mathbf{dc}\\ \mathbf{ds}\end{pmatrix}^T \mathbf{H}_{\mathbf{c}_i,\mathbf{s}_i} \begin{pmatrix}\mathbf{dc}\\ \mathbf{ds}\end{pmatrix} \right),$$

with approximative Hessian $\mathbf{H}_{\mathbf{c}_i,\mathbf{s}_i} \approx d_{\mathbf{c}_i,\mathbf{s}_i}\mathbf{r}^T d_{\mathbf{c}_i,\mathbf{s}_i}\mathbf{r}$. This minimization problem has the same minimum as the linear least squares problem

$$\begin{pmatrix}\mathbf{dc}\\ \mathbf{ds}\end{pmatrix} = \arg\min_{\mathbf{dc},\mathbf{ds}} \left(\left\| \mathbf{r} + d_{\mathbf{c}_i,\mathbf{s}_i} \begin{pmatrix}\mathbf{dc}\\ \mathbf{ds}\end{pmatrix} \right\|_2^2 \right)$$

which will be considered instead of the Taylor expansion. The method is complete, if we can find an expression for the total derivative $d_{\mathbf{c}_i,\mathbf{s}_i}\mathbf{r}$.

Since Φ is a sum of squares, a critical point of Φ must fulfill

$$\mathbf{0} = \partial_{\mathbf{m}}\Phi = (\partial_{\mathbf{m}}\mathbf{r})^T \mathbf{r} = \left[\left(\mathbf{S}^T \otimes \mathbf{C} \right) \mathcal{S}_{(f)}^T \otimes \mathbf{I}_F \right]^T \mathbf{r}. \tag{4.36}$$

As the left-hand side of Eq. (4.36) equals the constant zero-vector, its total derivative w.r.t. $\mathbf{c}$ and $\mathbf{s}$ is also zero and hence with the product rule

$$\mathbf{0} = d_{\mathbf{c},\mathbf{s}} \left((\partial_{\mathbf{m}}\mathbf{r})^T \mathbf{r} \right) = d_{\mathbf{c},\mathbf{s}} \left((\partial_{\mathbf{m}}\mathbf{r})^T \right) \mathbf{r} + (\partial_{\mathbf{m}}\mathbf{r})^T d_{\mathbf{c},\mathbf{s}}\mathbf{r}. \tag{4.37}$$

With a similar justification as in the Gauss-Newton algorithm, the multiplication between the residuum vector and its second order derivative is assumed to be negligible $d_{\mathbf{c},\mathbf{s}} \left((\partial_{\mathbf{m}}\mathbf{r})^T \right) \mathbf{r} \approx \mathbf{0}$. This approximation together with the chain-rule for total derivatives

$$d_{\mathbf{c},\mathbf{s}}\mathbf{r} = \partial_{\mathbf{m}}\mathbf{r}\partial_{\mathbf{c},\mathbf{s}}\mathbf{m} + \partial_{\mathbf{c},\mathbf{s}}\mathbf{r} \tag{4.38}$$

allows to rewrite Eq. (4.37) as

$$\mathbf{0} \approx (\partial_{\mathbf{m}}\mathbf{r})^T d_{\mathbf{c},\mathbf{s}}\mathbf{r} = (\partial_{\mathbf{m}}\mathbf{r})^T (\partial_{\mathbf{m}}\mathbf{r}\partial_{\mathbf{c},\mathbf{s}}\mathbf{m} + \partial_{\mathbf{c},\mathbf{s}}\mathbf{r}) \tag{4.39}$$

This last equation provides us with an estimate for the partial derivative of $\mathbf{m}$ considered as a function of $\mathbf{c}$ and $\mathbf{s}$

$$\partial_{\mathbf{c},\mathbf{s}}\mathbf{m} = -\left((\partial_{\mathbf{m}}\mathbf{r})^T \partial_{\mathbf{m}}\mathbf{r} \right)^{-1} (\partial_{\mathbf{m}}\mathbf{r})^T \partial_{\mathbf{c},\mathbf{s}}\mathbf{r} \tag{4.40}$$

Finally, an approximation of the sought after total derivative is given by (Eq. (4.38) and Eq. (4.40))

$$d_{\mathbf{c},\mathbf{s}}\mathbf{r} = \left[-\partial_{\mathbf{m}}\mathbf{r} \left((\partial_{\mathbf{m}}\mathbf{r})^T \partial_{\mathbf{m}}\mathbf{r} \right)^{-1} (\partial_{\mathbf{m}}\mathbf{r})^T + \mathbf{I} \right] \partial_{\mathbf{c},\mathbf{s}}\mathbf{r} = \mathbb{P}^{\perp}_{\partial_{\mathbf{m}}\mathbf{r}} \partial_{\mathbf{c},\mathbf{s}}\mathbf{r}.$$

In the last equation, the projection matrix $\mathbb{P}^{\perp}_{\partial_{\mathbf{mr}}}$ onto the orthogonal complement of the columns (i.e. the column nullspace) of $\partial_{\mathbf{mr}}$ has been introduced. Note that due to the multlilinearity of the objective function, the residuum vector is also expressible in terms of this projection matrix $\mathbf{r} = -\mathbb{P}^{\perp}_{\partial_{\mathbf{mr}}}\mathbf{w}$. Now, we have everything in place to state the Wiberg algorithm applied to our trilinear factorization problem (Alg. 5).

Algorithm 5: Wiberg algorithm applied to our trilinear factorization problem

Input: Measurements $\mathbf{w}$ and initial guesses for rigid motion $\mathbf{m}$, camera matrix $\mathbf{c}$, and points $\mathbf{s}$

Output: Affine estimates for rigid motion $\mathbf{m}$, camera matrix $\mathbf{c}$, and points $\mathbf{s}$

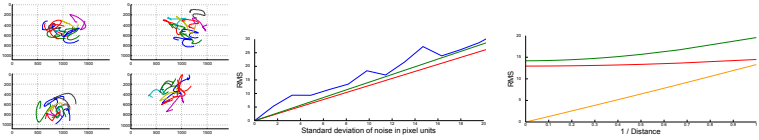
while *not converged* **do**

- $\mathbf{C} = \mathit{reshape}(\mathbf{c})$;
- $\mathbf{S} = \mathit{reshape}(\mathbf{s})$;
- $\mathbf{m} = \arg\min_{\mathbf{m}} \left\| \left[\left(\mathbf{S}^T \otimes \mathbf{C} \right) \mathcal{S}^T_{(f)} \otimes \mathbf{I}_F \right] \mathbf{m} - \mathbf{w} \right\|_2^2$;
- $\begin{bmatrix} \mathbf{dc} & \mathbf{ds} \end{bmatrix} = \arg\min_{\mathbf{dc},\mathbf{ds}} \left\| \mathbf{r} + d_{\mathbf{c},\mathbf{s}}\mathbf{r} \begin{pmatrix} \mathbf{dc} \\ \mathbf{ds} \end{pmatrix} \right\|_2^2$;
- $\mathbf{c} = \mathbf{c} + \mathbf{dc}$;
- $\mathbf{s} = \mathbf{s} + \mathbf{ds}$;

end

4.10 Evaluation: Rank-13 Factorization

The steps described in Sec. 4.6.2, Sec. 4.6.2, and Sec. 4.6.2 were applied sequentially to synthetically generated data and to a real data sequence in order to get an initial estimate for an iterative non-linear refinement. The ALS scheme was then iterated up to 10 times, with the previously computed solution as an initial guess. This already provided a very good reconstruction which could be even further improved by performing a couple of Wiberg iterations with the ALS solution as initialization. Finally, the metric upgrade was performed as described in Sec. 4.6.2. Because the metric upgrade step is based on a least squares formulation, orthogonality constraints are not enforced strictly on the



(a) Example 2D input trajectories for the synthetic experiments

(b) RMS as a function of the standard deviation of the noise

(c) RMS as a function of the inverse of the distance between the rigid object and projective cameras

Figure 4.4: Synthetic data experiments: The green line corresponds to the error between ground truth and noisy projections, the red line is the error of our algorithm where the orthogonality constraints on rotation matrices are not enforced strictly whereas the blue line shows the resulting error if exact rigid rotations are enforced with a polar decomposition. The orange line shows the error of the optimal affine camera approximation to the projective cameras in the absence of noise.

rotation matrices of the rigid motion and the resulting motion is thus not perfectly rigid. Perfectly rigid motions can be enforced if required in a supplemental step. We computed a polar decomposition of the matrix $\mathbf{R}_f = \mathbf{RP}$ at every frame, replaced $\mathbf{R}_f$ by $\mathbf{R}$ and ignored the non-rotational part $\mathbf{P}$ [3].

4.10.1 Synthetic Data

For the synthetic data experiments, the cameras were modeled as similar to the ones used for the real data experiments as possible (pixel density of $\frac{1080 pixels}{4.035 mm}$, image resolution of 1920×1080). The rigid motion was generated by specifying 5 keyframes and interpolating in between using third-order splines. We randomly picked 5 axes of rotation and rotation angles (which were limited to a maximum of 45 degrees). The translation vector of the rigid motions and the feature points were drawn from a

[3] The polar decomposition $\mathbf{A} = \mathbf{RP}$ provides the optimal approximation $\mathbf{R} \approx \mathbf{A}$ of a matrix $\mathbf{A}$ with an orthogonal matrix $\mathbf{R}$ w.r.t. the Frobenius-norm $\mathbf{R} = \arg\min_{\mathbf{Q}} \|\mathbf{A} - \mathbf{Q}\|_F$ subject to $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$.

normal distribution with a standard deviation of $5cm$. $K = 4$ cameras with focal length $f = 90mm$ were placed randomly $7.5m$ apart from the scene pointing toward the origin each of which tracked $N_k = 10$ feature points over 100 frames. 2D trajectories generated with this approach are visualized in Fig. 4.4a.

Firstly, the robustness with respect to isotropic Gaussian noise on the coordinates of the projected feature points was investigated. The synthetic data was generated with an affine camera model for this experiment. Inspecting Fig. 4.4b shows that our algorithm with non-strict orthogonality constraints slightly overfits the ground truth. This is mainly due to the fact that the rigidity of the motion was not strictly imposed. Enforcing truly rigid motions using polar decompositions increased the root mean squared (RMS) error

$$\frac{1}{\sqrt{F \sum_k N_k}} \left\| \mathbf{W} - \mathbf{M}\mathcal{S}_{(f)} \left[\Rightarrow_k \mathbf{S}^k \otimes \mathbf{C}^{k^T} \right] \right\|_F \tag{4.41}$$

of the reprojected moving points slightly.

Secondly, the influence of the distance between cameras and rigid object was investigated. The magnification factor was set to a constant of $m = \frac{61mm}{5m}$ which can be interpreted as choosing a focal length of $61mm$ with an average distance between camera and rigid object of $5m$. In order to keep the magnification factor constant, the focal length of the cameras was updated accordingly while changing the distance. In this second experiment the data was generated with projective camera models and noise with a standard deviation of $\sigma = 10$ pixels was added to the projections in order to make the experiment more realistic (Fig. 4.4c).

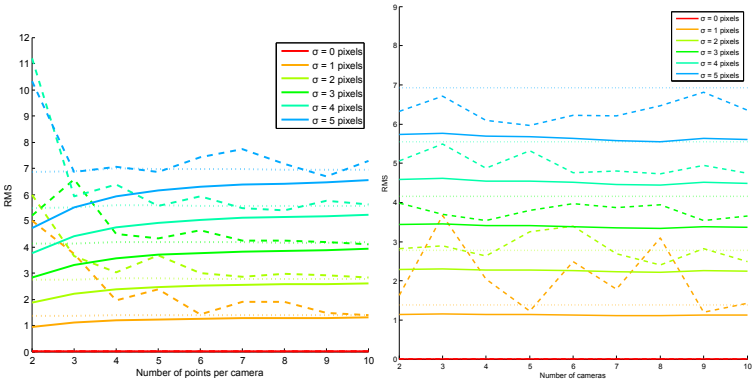
In a third synthetic experiment, the stability of the method w.r.t. the number of points observed per camera is investigated. The data has been generated in the same way as for the first experiment, but this time with $K = 6$ affine cameras. The method has been applied several times to an increasing number of points visible per camera by adding additional feature trajectories (each camera still tracked a different set of points). Fig. 4.5a shows the results of this experiment. From this figure, we can conclude that the method is stable to a fair amount of noise if at least 3 to 4 points are tracked per camera. The more extreme cases of just 2 points per camera seem to be substantially less robust w.r.t. noise.

The influence of an increasing number of cameras is shown in a last synthetic experiment. The total number of points $N = \sum_k N_k$ was held fixed but the number of affine cameras K varied. For each K, the N points were split into evenly sized disjoint subsets. Based on the results in Fig. 4.5b, our methods does not seem to depend strongly on the number of cameras.

Combining the results from Fig. 4.5b with Fig. 4.5a, we conclude that the method is considerably robust as soon as there is sufficient redundancy in the input data. For example, the accuracy decreases slightly in the $K = 6$ with $N_k = 2$ case as seen in Fig. 4.5a, however in the case $K = 10$ with $N_k = 2$ shown in Fig. 4.5b the accuracy stays more or less the same.

4.10.2 Real Data Sequence

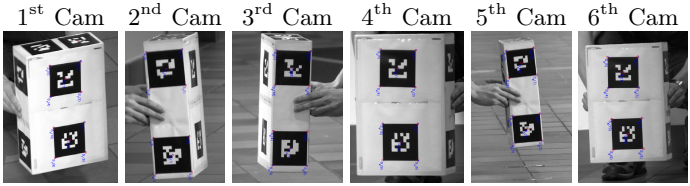
We evaluated our algorithm on a real sequence of a rotating rigid box. The cameras recorded in a resolution of 1920×1080 pixels. In order to ease the tracking we used a template based tracking algorithm [WS07] which provides 5 points per template (the 4 vertices and the middle point). The cameras were not aware of the fact that they might have tracked the very same feature points (i.e., no correspondences were established between different camera views). Each camera tracked 2 templates which were located on the same side of the box and hence, the structure of the points tracked by one single camera was actually planar. As the results show, our algorithm can handle this configuration. Fig. 4.6 shows the accuracy of the reconstruction. Cameras 1, 4, and 6 tracked the templates on the front facing plane of the box (these templates are drawn in cyan, magenta, and red color), cameras 2 and 5 tracked the templates on another side of the box (blue and yellow), whereas camera 3 was the only camera which tracked the templates on the opposite side (green). Note that a template which was tracked by more than two cameras gets reconstructed at almost the very same location in space, even though the algorithm is intentionally unaware of such correspondences. Since affine camera poses suffer from a depth ambiguity along the z-axis, all the cameras are drawn with the same distance to the scene. The size of the image plane however encodes the scaling factor of the cameras (the larger the image plane, the further away the camera) and together with a known focal length this would determine the distance along the z-axis. In our experiments, cameras 2



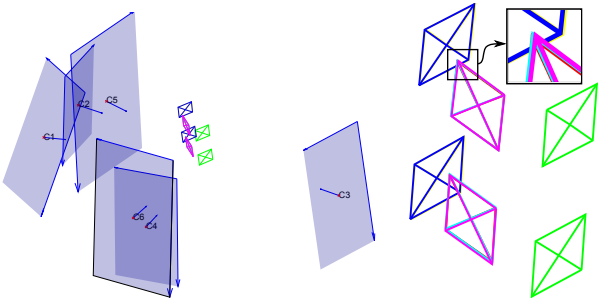
(a) Increasing number of points per camera

(b) Increasing number of cameras

Figure 4.5: Fig. 4.5a: Synthetic data experiment showing the influence of an increasing number of points observed per camera while holding the number of cameras fixed: The x-axis shows the number of points tracked in each of the $K = 6$ cameras whereas the y-axis shows the resulting RMS. The continuous lines show the error when the orthogonality constraints on rotation matrices are not strictly enforced and the dashed lines show the result after applying a subsequent polar decomposition in order to enforce exact rotation matrices. The dotted lines show the error of the ground truth reconstruction, i.e. they show the error due to the noise in the data. The amount of noise added to the images varied between 0 and 5 pixels.
Fig. 4.5b: Synthetic data experiment showing the influence of an increasing number of cameras while holding the total number of points fixed: The x-axis shows the number of cameras whereas the y-axis shows the resulting RMS. The total number of points was fixed to $N = \sum_k N_k = 20$ and the number of points per cameras was split evenly (e.g. for $K = 2$ each camera observed 10 points and for $K = 10$ cameras each camera observed 2 points). The line patterns encode the same semantics as as in Fig. 4.5a.



(a) $K = 6$ Camera Views



(b) 3D view of reconstruction (c) Closeup view

Figure 4.6: Resulting reconstruction of the real data sequence. Fig. 4.6a shows the reprojection of feature points (red circles) into the camera views along with the ground truth (blue crosses) for one specific frame (the frames are cropped in order to fit in the figure). Fig. 4.6b shows a 3D view of the reconstructed camera poses together with the points of the box at one specific frame. Fig. 4.6c shows a closeup view of the reconstructed points at this frame.

and 5 (4 and 6) have an almost parallel image plane, but camera 5 (6) was placed slightly further away from the box. A RMS of about 12.3 pixels resulted by using our linear algorithm to initialize 5 iterations of the ALS algorithm. Additional 5 iterations with the Wiberg optimization finally gave a RMS of about 2.6 pixels. Enforcing true rigid motions as a last step increased the RMS of the reconstruction to about 8.5 pixels. All the results shown in Fig. 4.7 and Fig. 4.6 are based on the reconstruction which enforces true rigid motions.

In a second experiment, we tried how robustly our algorithm can

handle a camera which only tracks one single feature point. We therefore excluded all but one feature trajectory in camera 3 and run the same algorithm again. The resulting reconstruction again had a RMS of about 2.6 pixels, respectively 8.5 pixels with enforced rotation matrices. Fig. 4.7 compares this reconstruction with the previous reconstruction which used all the 10 feature points per camera. This result shows that the new rank-13 constraint can be used to calibrate cameras which only track one single point which is not in correspondence with any other point tracked by the remaining cameras.

4.11 Evaluation: Rank-5 Factorization

If synthetic data is generated with affine cameras and without noise, the algorithm expectedly finds the exact solution in closed-form, even for the case of only two cameras each of them tracking one single point. Based on our experience with synthetic data according to a more realistic setting (i.e. projective camera models with realistic internal parameters, some noise and plausible planar motions) we concluded that the robustness of the algorithm strongly depends on the observed motion. This is actually an expected behavior. If the motion clearly spans the 5D motion subspace, the algorithm works robustly. However, if a dimension of this subspace is not explored sufficiently, noise will overrule this dimension and the reconstruction will deteriorate.

As a proof of concept the algorithm has been applied to a real data sequence. Fig. 4.8 shows the results of a real sequence with four cameras observing the planar motion of a rigid box. This time, no iterative refinement has been applied to the closed-form solution provided by the rank-5 factorization. The translation ambiguity along the rotation axis has been resolved such that the centroids of the front-facing tags share the same coordinate along the axis of rotation. A template based tracker [WS07] has been used to generate the feature trajectories. Each camera tracked between 10 to 20 points. Even though some cameras actually tracked the very same points, the algorithm was purposely not aware of these correspondences. Such hidden correspondences allow to evaluate the accuracy of the reconstruction. Based on the overlapping area of the 3D model of the tracked feature tags, we conclude that the algorithm succeeds in computing an accurate reconstruction given the fact that the reconstruction is based on the approximate affine camera

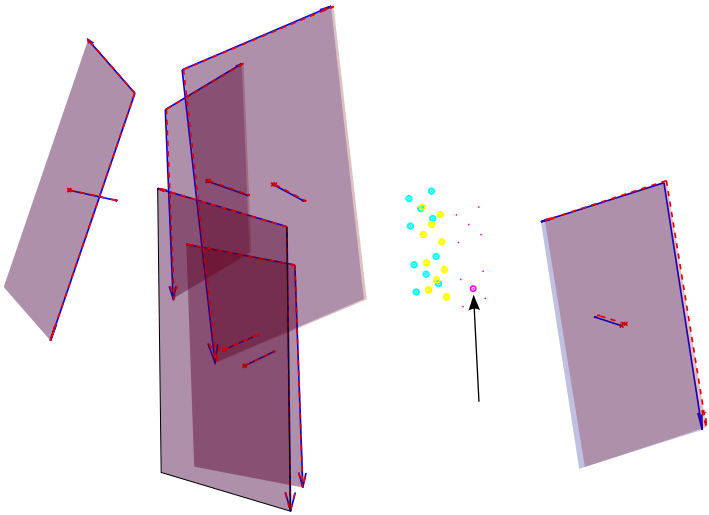


Figure 4.7: Comparison between two reconstructions of the real data sequence: All the 10 feature points per camera view are used for the first reconstruction (feature points are drawn as dots). In contrast, for the second reconstruction (feature points drawn as circles), the rightmost camera 3 only tracked one single feature point (black arrow). The pose and the tracked feature point of the third camera nonetheless got reconstructed very accurately. The cameras of the first (second) reconstruction are visualized semi-transparently in blue (red) color. The areas of overlap thus appear in violet color.

model and the solution is given in a non-iterative closed-form. The reprojection error of the closed-form solution is

$$\frac{1}{\sqrt{F \sum_k N_k}} \left\| \mathbf{W} - \mathbf{M}\mathcal{C}_{(f)} \left[\Rightarrow_k \mathbf{S}^k \otimes \mathbf{C}^{k^T} \right] \right\|_F = 7.5 \text{ pixels},$$

where the resolution of the cameras is 1920×1080. A successive non-linear refinement step still based on the affine camera model did not improve the reprojection error. This provides evidence that most of

Figure 4.8: Reconstruction of a planarly moving box: The right image shows a close-up view of the reconstructed structure (tags tracked by one specific camera share the same color). Two cameras have been positioned slightly below resp. above the remaining other cameras which is nicely captured by the reconstruction.

the error is due to the discrepancy between the employed affine camera approximation and the real projective cameras and not due to the sub-optimal sequential steps of the closed-form solution.

4.12 Conclusions and Future Work

This chapter brought together the ideas previously presented in [AP09] and [AP10]. Specifically, this chapter presented a unified analysis of rigidly moving objects, for general rigid motions as well as for the special case of planar rigid motions (Sec. 4.3). The key insight was that any trajectory of any point seen by any camera is restricted to a low-dimensional subspace, namely to a 13-dimensional subspace for general rigid motions and to a 5-dimensional subspace for planar rigid motions. The theoretical insights gained thereby enabled the development of two algorithms, which provide a closed-form solution to the structure from motion reconstruction problem where no feature point correspondences between the different camera views exist (Sec. 4.6 and Sec. 4.7). The cameras are only assumed to track feature points on a commonly observed moving rigid object. The motion correspondence, namely that all the cameras observe the same rigid motion, was captured by a 13D resp. by a 5D motion subspace. Tensorial notation provided us with the necessary tools and insights to derive two non-iterative algorithms which provide a closed-form solution. The first algorithm handles the case of general rigid motions and is based on a rank-13 factorization, whereas

the second algorithm is applicable when the observed rigid motion is planar and is based on a rank-5 factorization. Even though the setup for the two algorithms is almost the same, the steps required to compute a closed-form solution largely differ. These individual steps introduced several ideas and tricks which might prove useful for other factorization problems, as well. The algorithms were evaluated on synthetic data and have been shown to work on real data sequences (Sec. 4.10 and Sec. 4.11).

We hope the analysis and techniques presented in this chapter will be stimulating and boost potential future work, in factorization problems for SfM but also in other fields which deal with low-rank tensor models. We see several opportunities which build upon the work presented in this chapter. For example, one could think of adapting the rigid motion subspace constraints to a formulation with projective camera models. This probably asks for iterative solutions for which the closed-form algorithms might provide a good initialization. The low-rank constraint might also be used as a means to temporally synchronize multiple camera streams. A drawback of our current method is that the methods assume the feature tracks of each camera to be complete, i.e. the camera succeeds in tracking its feature points at every single frame of the sequence (see also Fig. 4.2). This prevents large rotations of the rigid object which cause eventual occlusions. This leads to a problem which currently enjoys interest from a wide variety of research communities, namely the previously mentioned matrix completion problem. The tensor notation introduced in this article is hopefully conducive to transferring ideas between different communities since the theory of matrix completion is currently rapidly evolving in parallel in different research areas.

4.A Detailed Derivations

The following appendices provide a detailed derivation of the linear systems of equations which need to be solved for the closed-form rank-13 or rank-5 factorization. Mostly based on algebraic manipulations, these derivations do not provide any additional intuition and can safely be skipped by the reader if not interested in implementation details.

4.A.1 Linear System for Affine Reconstruction of Camera matrices

The affine camera matrices must fulfill

$$\mathcal{S}_{(f),[10:13,:]}\left[\Downarrow_k \tilde{\mathbf{S}}^{k\,T} \otimes \tilde{\mathbf{C}}^k\right]^T = \mathbf{Q}_{kron,[10:13,:]}\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}} \tag{4.42}$$

$$=\left[\Rightarrow_k \mathbf{1}_{1\times N_k} \otimes \tilde{\mathbf{C}}^{k\,T}\right] = \mathbf{Q}_{kron,[10:13,:]}\mathbf{Q}_{aff}^{-1}\left[\Rightarrow_k \hat{\mathbf{A}}^k\right], \tag{4.43}$$

where we used $\hat{\mathbf{A}}^k$ to denote the submatrix of $\hat{\mathbf{A}}$ due to camera k. Let us first investigate only such a submatrix $\mathcal{S}_{(f),[10:13,:]}\left[\tilde{\mathbf{S}}^{k\,T} \otimes \tilde{\mathbf{C}}^k\right]^T$ due to one single camera k. Vectorization of this matrix equation using Eq. (2.7) and Eq. (2.4) gives

$$\mathbf{G}^k \operatorname{vec}\left(\tilde{\mathbf{C}}^{k\,T}\right) = \left[(\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}}^k)^T \otimes \mathbf{I}_4\right]\operatorname{vec}\left(\mathbf{Q}_{kron,[10:13,:]}\right) \tag{4.44}$$

$$=\mathbf{H}^k \operatorname{vec}\left(\mathbf{Q}_{kron,[10:13,1:12]}\right) + \mathbf{b}^k \tag{4.45}$$

where the following matrices were introduced for abbreviation

$$\mathbf{G}^k = [\mathbf{I}_{N_k} \otimes \mathbf{T}_{2,1} \otimes \mathbf{I}_4]\,[\mathbf{1}_{N_k\times 1} \otimes \mathbf{I}_{2\cdot 4}] \tag{4.46}$$

$$\mathbf{H}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[1:12,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_4\Big] \tag{4.47}$$

$$\mathbf{b}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[13,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_4\Big]\operatorname{vec}\left(\mathbf{Q}_{kron,[10:13,13]}\right). \tag{4.48}$$

Combining each of the linear systems due to a camera in one single linear system leads to

$$\left[\searrow_k \mathbf{G}^k \quad -\Downarrow_k \mathbf{H}^k\right]\begin{pmatrix}\Downarrow_k \operatorname{vec}\left(\tilde{\mathbf{C}}^{k\,T}\right)\\ \operatorname{vec}\left(\mathbf{Q}_{kron,[10:13,1:12]}\right)\end{pmatrix} = \left(\Downarrow_k \mathbf{b}^k\right). \tag{4.49}$$

However, closer inspection of Eq. (4.43) reveals that successive rows result in the very same linear constraints. To avoid an unnecessary increase in unknowns, we therefore only consider one row $i \in \{1,2,3,4\}$ for setting up the linear system which results in slightly changed matrices

$$\mathbf{G}^k = [\mathbf{I}_{N_k} \otimes \mathbf{T}_{2,1} \otimes \mathbf{I}_1]\,[\mathbf{1}_{N_k\times 1} \otimes \mathbf{I}_2] \tag{4.50}$$

$$\mathbf{H}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[1:12,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_1\Big] \tag{4.51}$$

$$\mathbf{b}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[13,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_1\Big]\operatorname{vec}\left(\mathbf{Q}_{kron,(9+i,13)}\right). \tag{4.52}$$

The resulting over-constrained linear system reads like

$$\left[\searrow_k \mathbf{G}^k \quad -\Downarrow_k \mathbf{H}^k\right]\begin{pmatrix}\Downarrow_k \operatorname{vec}\left(\tilde{\mathbf{C}}^k_{[:,i]}{}^T\right)\\ \operatorname{vec}\left(\mathbf{Q}_{kron,[9+i,1:12]}\right)\end{pmatrix} = \left(\Downarrow_k \mathbf{b}^k\right), \tag{4.53}$$

which consists of only $2K + 1 \cdot 12$ unknowns instead of $4 \cdot 2K + 4 \cdot 12$ unknowns. Note that $\mathbf{Q}_{kron,[10:13,13]} = (0,0,0,1)^T$ and therefore $\mathbf{b}^k$ is only non-zero for the last row which is associated with the camera translation. The system matrix in Eq. (4.53) however has a three-dimensional nullspace, and therefore provides four linear independent solutions for the four rows.

4.A.2 Linear System for Affine Reconstruction of Points

This derivation closely follows the one from Sec. 4.A.1. Let $\mathbf{X}^k$ denote the non-homogeneous part of the points $\mathbf{S}^{k\,T} = \begin{bmatrix}\mathbf{X}^{k\,T} & \mathbf{1}_{N_k \times 1}\end{bmatrix}$ and $\mathbf{P}^k$ stands for the non-translational columns of the camera matrix $\mathbf{C}^k = \begin{bmatrix}\mathbf{P}^k & \mathbf{t}^k\end{bmatrix}$. Using this notation, a valid affine reconstruction must fulfill

$$\mathcal{S}_{(f),[1:9,:]}[\Downarrow_k \tilde{\mathbf{S}}^{k\,T} \otimes \tilde{\mathbf{C}}^k]^T = \mathbf{Q}_{kron,[1:9,:]}\mathbf{Q}^{-1}_{aff}\hat{\mathbf{A}} \tag{4.54}$$

$$= \left[\Rightarrow_k \tilde{\mathbf{X}}^k \otimes \tilde{\mathbf{P}}^{k\,T}\right] = \mathbf{Q}_{kron,[1:9,:]}\mathbf{Q}^{-1}_{aff}\left[\Rightarrow_k \hat{\mathbf{A}}^k\right] \tag{4.55}$$

Vectorization of the submatrix equation due to camera k using Eq. (2.6) and Eq. (2.4) leads to

$$\mathbf{G}^k \operatorname{vec}\left(\tilde{\mathbf{X}}^k\right) = \left[[\mathbf{Q}^{-1}_{aff}\hat{\mathbf{A}}^k]^T \otimes \mathbf{I}_9\right]\operatorname{vec}\left(\mathbf{Q}_{kron,[1:9,:]}\right) \tag{4.56}$$

$$= \mathbf{H}^k \operatorname{vec}\left(\mathbf{Q}_{kron,[1:9,1:12]}\right) + \mathbf{b}^k \tag{4.57}$$

where the following matrices were introduced for abbreviation

$$\mathbf{G}^k = [\mathbf{I}_{N_k} \otimes \mathbf{T}_{2,3} \otimes \mathbf{I}_3][\mathbf{I}_{3N_k} \otimes \operatorname{vec}\left(\tilde{\mathbf{P}}^{k\,T}\right)] \tag{4.58}$$

$$\mathbf{H}^k = \left[\left[[\mathbf{Q}^{-1}_{aff}]_{[1:12,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_9\right] \tag{4.59}$$

$$\mathbf{b}^k = \left[\left[[\mathbf{Q}^{-1}_{aff}]_{[13,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_9\right]\operatorname{vec}\left(\mathbf{Q}_{kron,[1:9,13]}\right). \tag{4.60}$$

Combining again each of the linear systems due to a camera in one single linear system leads to

$$\left[\diagdown\!\!\!\downarrow_k \mathbf{G}^k \quad -\Downarrow_k \mathbf{H}^k \right] \begin{pmatrix} \Downarrow_k \operatorname{vec}\left(\tilde{\mathbf{X}}^k\right) \\ \operatorname{vec}\left(\mathbf{Q}_{kron,[1:9,1:12]}\right) \end{pmatrix} = \left(\Downarrow_k \mathbf{b}^k\right). \tag{4.61}$$

A similar observation as in Sec. 4.A.1 holds true for Eq. (4.61). More specifically, successive row-triples in Eq. (4.55) result in the very same linear constraints. To avoid an unnecessary increase in unknowns, we therefore only consider one row triple for setting up the linear system which results again in slightly changed matrices

$$\mathbf{G}^k = [\mathbf{I}_{N_k} \otimes \mathbf{T}_{2,1} \otimes \mathbf{I}_3][\mathbf{I}_{N_k} \otimes \operatorname{vec}\left(\tilde{\mathbf{P}}_k^T\right)] \tag{4.62}$$

$$\mathbf{H}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[1:12,:]} \hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_3 \tag{4.63}$$

$$\mathbf{b}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[13,:]} \hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_3 \operatorname{vec}\left(\mathbf{Q}_{kron,[3i-2:3i,13]}\right). \tag{4.64}$$

The resulting over-constrained linear system reads like

$$\left[\diagdown\!\!\!\downarrow_k \mathbf{G}^k \quad -\Downarrow_k \mathbf{H}^k \right] \begin{pmatrix} \Downarrow_k \operatorname{vec}\left(\tilde{\mathbf{X}}_{k,[i,:]}\right) \\ \operatorname{vec}\left(\mathbf{Q}_{kron,[3i-2:3i,1:12]}\right) \end{pmatrix} = \left(\Downarrow_k \mathbf{b}^k\right), \tag{4.65}$$

which consists of only $\sum_k N_k + 3 \cdot 12$ unknowns instead of $3\sum_k N_k + 9 \cdot 12$ unknowns. Note that $\mathbf{Q}_{kron,[1:9,13]} = \mathbf{0}_{9\times 1}$ and therefore $\mathbf{b}^k$ is zero for every row-triple. However, the system matrix has a four dimensional nullspace, which should not come as a surprise since each basis vector of this nullspace provides a solution to a different row triple (one solution corresponds to the homogeneous coordinate of the points, which we do not need to solve for).

4.A.3 Extracting Rank-Degenerate Solutions

The linear system in Eq. (4.26) is concisely formulated as

$$\left[\hat{\mathbf{M}}^T \odot \hat{\mathbf{M}}^T\right]^T \mathbf{D}_5 \operatorname{vecs}(\mathbf{Q}) = \mathbf{1}, \tag{4.66}$$

where $\odot$ denotes the Khatri-Rao product with column-wise block partitioning (i.e. column-wise Kronecker product), $\operatorname{vecs}()$ vectorizes the upper triangular part of a matrix, and $\mathbf{D}_5$ is the duplicity matrix s.t. $\operatorname{vec}(\mathbf{Q}) = \mathbf{D}_5 \operatorname{vecs}(\mathbf{Q})$ (we refer to Sec. 2.2.2 and reference [MN99] for

more details about these operators). Eq. (4.66) can be solved in the least squares sense. The solution will in general have rank 3. Let $\mathbf{Q}_0 \in \mathbb{R}^{5\times5}$ denote a particular solution and $\mathbf{N} \in \mathbb{R}^{5\times5}$ denote the nullspace of the linear system in Eq. (4.66). The particular solution and the solution of the nullspace will be of rank 3 and will have the following parameterization $b(\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T) + (1-b)(\mathbf{q}_1+\mathbf{q}_2)(\mathbf{q}_1+\mathbf{q}_2)^T$ with $b \in \mathbb{R}$ in the unknown $b_{\mathbf{Q}_0}$ resp. $b_{\mathbf{N}}$. In order to find the rank deficient solutions, a 3^{rd}-order polynomial constraint in x could be imposed on all the 3×3 subdeterminants of $\mathbf{Q}_0 + x\mathbf{N}$. However, it is difficult to robustly combine the constraints of all the 3-by-3 subdeterminants in one polynomial constraint. Another approach is based on the fact, that we can readily solve $\hat{\mathbf{M}}(\mathbf{q}_1+\mathbf{q}_2) = \mathbf{1}_{F\times1}$ for the vector $\mathbf{q}_1+\mathbf{q}_2$. Then we have

$$\begin{aligned}\mathbb{P}^{\perp}_{\mathbf{q}_1+\mathbf{q}_2}\left[\mathbf{Q}_0 + x\mathbf{N}\right] =& \mathbb{P}^{\perp}_{\mathbf{q}_1+\mathbf{q}_2}\Big[\left(b_{\mathbf{Q}_0} + xb_{\mathbf{N}}\right)\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right] + \\ & \left(1 - b_{\mathbf{Q}_0} + x\left(1-b_{\mathbf{N}}\right)\right)\left(\mathbf{q}_1+\mathbf{q}_2\right)\left(\mathbf{q}_1+\mathbf{q}_2\right)^T\Big] \\ =& \mathbb{P}^{\perp}_{\mathbf{q}_1+\mathbf{q}_2}\left[\left(b_{\mathbf{Q}_0} + xb_{\mathbf{N}}\right)\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right]\right].\end{aligned}$$

The row space of the resulting matrix reveals the span of the rank-2 matrix $\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T$. This allows us to compute

$$\begin{aligned}&\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\mathbf{Q}_0\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T} \\ &= \left(1-b_{\mathbf{Q}_0}\right)\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\left(\mathbf{q}_1+\mathbf{q}_2\right)\left(\mathbf{q}_1+\mathbf{q}_2\right)^T\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\end{aligned} \tag{4.67}$$

and

$$\begin{aligned}&\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\mathbf{N}\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T} \\ &= \left(1-b_{\mathbf{N}}\right)\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\left(\mathbf{q}_1+\mathbf{q}_2\right)\left(\mathbf{q}_1+\mathbf{q}_2\right)^T\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T},\end{aligned} \tag{4.68}$$

which in turn enables the computation of the fraction $\frac{1-b_{\mathbf{Q}_0}}{1-b_{\mathbf{N}}}$. Finally, this leads to a valid rank-2 solution

$$\begin{aligned}\mathbf{Q}_0 - \frac{1-b_{\mathbf{Q}_0}}{1-b_{\mathbf{N}}}\mathbf{N} = &\left(b_{\mathbf{Q}_0} - \frac{1-b_{\mathbf{Q}_0}}{1-b_{\mathbf{N}}}b_{\mathbf{N}}\right)\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right] + \\ &\underbrace{\left(1 - b_{\mathbf{Q}_0} - \frac{1-b_{\mathbf{Q}_0}}{1-b_{\mathbf{N}}}\left(1-b_{\mathbf{N}}\right)\right)}_{=0}\left(\mathbf{q}_1+\mathbf{q}_2\right)\left(\mathbf{q}_1+\mathbf{q}_2\right).\end{aligned} \tag{4.69}$$

The last step consists in decomposing the solution $\mathbf{Q}_2 = \mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3^T\mathbf{q}_3^T$ into the vectors $\mathbf{q}_1$ and $\mathbf{q}_2$. This can be done with an eigenvalue

decomposition of $\mathbf{Q}_2$ and assigning $\mathbf{q}_1$ and $\mathbf{q}_3$ the eigenvectors scaled by the square root of its corresponding eigenvalue.

A small detail needs to be mentioned. Because $\cos^2\alpha_f + (-1-\cos\alpha_f)^2 + 2\cos\alpha_f(1-\cos\alpha_f) = 1$ (compare with Eq. (4.25)) the second column of $\hat{\mathbf{M}}\mathbf{Q}_{trig}$ might correspond to $-1-\cos\alpha_f$ rather than $1-\cos\alpha_f$. However, if this happens (which is easy to check since $-1-\cos\alpha_f \leq 0 \leq 1-\cos\alpha_f$), $\mathbf{q}_2$ is replaced with $-\mathbf{q}_2 - 2\mathbf{q}_1$ (because $-(-1-\cos\alpha_f) - 2\cos\alpha_f = 1 - \cos\alpha_f$).

4.A.4 Projection onto Plane of Rotation

This section shows how feature trajectories of planar motions can be projected onto the plane of rotation knowing neither the camera matrices nor the 3D coordinates of the points. The derivation starts by subtracting the first row (the mean of the rows could be subtracted instead as well) from the data matrix

$$
\begin{aligned}
&\left[\mathbf{I}_F - \mathbf{1}_{F\times 1}\left[1, \mathbf{0}_{1\times F-1}\right]\right]\mathbf{W} \\
&= \left[\mathbf{I}_F - \mathbf{1}_{F\times 1}\left[1, \mathbf{0}_{1\times F-1}\right]\right]\mathbf{M}\mathcal{C}_{(f)}\mathbf{S}\otimes\mathbf{C}^T \\
&= \left[\Downarrow_f \cos\alpha_f - \cos\alpha_1, \sin\alpha_f - \sin\alpha_1, \mathbf{t}_f^T - \mathbf{t}_1^T\right]\cdot \\
&\quad \begin{bmatrix} 1 & -1 & 0 & \mathbf{0}_{1\times 2} \\ 0 & 0 & 1 & \mathbf{0}_{1\times 2} \\ \mathbf{0}_{2\times 1} & \mathbf{0}_{2\times 1} & \mathbf{0}_{2\times 1} & \mathbf{I}_2 \end{bmatrix}\mathcal{C}_{(f)}\mathbf{S}\otimes\mathbf{C}^T.
\end{aligned}
$$

The algebraic structure of $\mathbf{M} = \left[\Downarrow_f \cos\alpha_f, 1-\cos\alpha_f, \sin\alpha_f, \mathbf{t}_f^T\right]$ together with $(1-\cos\alpha_f) - (1-\cos\alpha_1) = -\cos\alpha_f + \cos\alpha_1$ has been used to replace the motion matrix $\mathbf{M}$ of rank 5 by a rank 4 matrix which is right multiplied with a suitable matrix in order to get the motion matrix with subtracted first row. It is interesting to see what happens if this matrix is left multiplied with the second factor $\mathbf{A} = \mathcal{C}_{(f)}\mathbf{S}\otimes\mathbf{C}^T$ of the

rank-5 decomposition

$$
\begin{aligned}
&\begin{bmatrix} 1 & -1 & 0 & \mathbf{0}_{1\times 2} \\ 0 & 0 & 1 & \mathbf{0}_{1\times 2} \\ \mathbf{0}_{2\times 1} & \mathbf{0}_{2\times 1} & \mathbf{0}_{2\times 1} & \mathbf{I}_2 \end{bmatrix} \mathcal{C}_{(f)} \mathbf{S} \otimes \mathbf{C}^T \\
&= \begin{bmatrix} \operatorname{vec}\left(\mathbf{I}_3 - \mathbf{a}\mathbf{a}^T\right) & \mathbf{0}_{1\times 3} & 0 \\ \operatorname{vec}\left([\mathbf{a}]_\times\right) & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{2\times 9} & \mathbf{V}^T & \mathbf{0}_{2\times 1} \end{bmatrix} \mathbf{S} \otimes \mathbf{C}^T \\
&= \begin{bmatrix} \operatorname{vec}\left(\mathbb{P}_{\mathbf{a}}^{\perp}\right) & \mathbf{0}_{1\times 3} & 0 \\ \operatorname{vec}\left([\mathbf{a}]_\times\right) & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{2\times 9} & \mathbf{V}^T\mathbb{P}_{\mathbf{V}} & \mathbf{0}_{2\times 1} \end{bmatrix} \mathbf{S} \otimes \mathbf{C}^T \\
&= \begin{bmatrix} \operatorname{vec}\left(\mathbb{P}_{\mathbf{V}}\right) & \mathbf{0}_{1\times 3} \\ \operatorname{vec}\left(\mathbb{P}_{\mathbf{V}}[\mathbf{a}]_\times \mathbb{P}_{\mathbf{V}}\right) & \mathbf{0}_{1\times 3} \\ \mathbf{0}_{2\times 9} & \mathbf{V}^T\mathbb{P}_{\mathbf{V}} \end{bmatrix} \left[\begin{bmatrix} \mathbb{P}_{\mathbf{V}} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \mathbf{S} \right] \otimes \left[\mathbb{P}_{\mathbf{V}} \mathbf{C}_{[:,1:3]}^T\right].
\end{aligned}
$$

The properties $\mathbf{I}_3 - \mathbf{a}\mathbf{a}^T = \mathbb{P}_{\mathbf{a}}^{\perp} = \mathbb{P}_{\mathbf{V}}$, $[\mathbf{a}]_\times = \mathbb{P}_{\mathbf{V}}[\mathbf{a}]_\times \mathbb{P}_{\mathbf{V}}$, $\mathbf{V}^T = \mathbf{V}^T\mathbb{P}_{\mathbf{V}}$, and the symmetry and idempotence of orthogonal projection matrices have been used. This final formula actually has a very intuitive explanation. By subtracting the first row (or the mean of all the rows) the non-dynamic aspect in the data is removed. The coordinates of the points along the rotation axis remain constant, so does the camera translation. Both the point coordinates along the rotation axis and the camera translation are thus removed by subtracting the first row.

4.A.5 Polynomial Solution to Orthogonality and Equality of Norm Constraints

For notational reasons, the symmetric 2-by-2 matrix $\mathbf{C}_{[:,1:3]}^k \mathbb{P}_{\mathbf{V}} {\mathbf{C}_{[:,1:3]}^k}^T$ in

$$
\begin{aligned}
\lambda_k^2 \mathbf{I}_2 &= \mathbf{C}_{[:,1:3]}^k {\mathbf{C}_{[:,1:3]}^k}^T \\
&= \mathbf{C}_{[:,1:3]}^k \mathbb{P}_{\mathbf{V}} {\mathbf{C}_{[:,1:3]}^k}^T + \mathbf{C}_{[:,1:3]}^k \mathbb{P}_{\mathbf{a}} {\mathbf{C}_{[:,1:3]}^k}^T
\end{aligned}
$$

is denoted as $\mathbf{G}^k$. Thus, it follows

$$
\begin{aligned}
\lambda_k^2 \mathbf{I}_2 &= \mathbf{G}^k + \mathbf{w}_k \mathbf{w}_k^T \\
&= \begin{bmatrix} \mathbf{G}_{[1,1]}^k & \mathbf{G}_{[1,2]}^k \\ \mathbf{G}_{[1,2]}^k & \mathbf{G}_{[2,2]}^k \end{bmatrix} + \begin{bmatrix} \mathbf{w}_{k,[1]}^2 & \mathbf{w}_{k,[1]}\mathbf{w}_{k,[2]} \\ \mathbf{w}_{k,[1]}\mathbf{w}_{k,[2]} & \mathbf{w}_{k,[2]}^2 \end{bmatrix}.
\end{aligned}
$$

The unknown scale factor λ_k^2 can be eliminated by subtracting the two equations on the diagonal from each other which leads to the system

$$\mathbf{G}^k_{[1,1]} - \mathbf{G}^k_{[2,2]} + \mathbf{w}^2_{k,[1]} - \mathbf{w}^2_{k,[2]} = 0 \tag{4.70}$$

$$\mathbf{G}^k_{[1,2]} + \mathbf{w}_{k,[1]}\mathbf{w}_{k,[2]} = 0. \tag{4.71}$$

The second equation Eq. (4.71) can be solved for $\mathbf{w}_{k,[1]} = -\frac{\mathbf{G}^k_{[1,2]}}{\mathbf{w}_{k,[2]}}$ (if either $\mathbf{w}_{k,[2]} = 0$ or $\mathbf{w}_{k,[1]} = 0$ the above system becomes a second-order polynomial in one unknown which is trivial to solve). Substituting $\mathbf{w}_{k,[1]}$ in Eq. (4.70) leads to a polynomial in the monomials $\mathbf{w}^2_{k,[2]}$ and $\mathbf{w}^4_{k,[2]}$. This polynomial can be solved for $\mathbf{w}^2_{k,[2]}$ which implicitly gives $\mathbf{w}_{k,[2]}$ and $\mathbf{w}_{k,[1]}$. This approach provides four solutions, two of them are conjugate complex and the remaining two are equal up to the sign. Hence, the solution is unique up to the sign.

5 Extension to Non-Rigid Structure-from-Motion

As we have seen in previous chapters and references therein, multiple-view geometry and structure-from-motion are well established techniques to compute the structure of a moving rigid object. These techniques are all based on strong algebraic constraints imposed by the rigidity of the object. Unfortunately, many scenes of interest, e.g. faces or cloths, are highly deformable and the rigidity constraint no longer holds. Hence, there is a need for non-rigid structure-from-motion (NRSfM) methods which can deal with dynamic scenes. A prominent framework to model deforming and moving non-rigid objects is the factorization technique where the measurements are assumed to lie in a low-dimensional subspace. Many different formulations and variations for factorization-based NRSfM have been proposed in recent years. However, due to the complex interactions between several subspaces, the distinguishing properties between two seemingly related approaches are often unclear. For example, do two approaches just vary in the optimization method used or is really a different model beneath?

In this chapter, which is based on our publication [AP12], we show that these NRSfM factorization approaches are most naturally modeled with tensor algebra. This results in a clear presentation which subsumes many previous techniques. In this regard, this chapter brings several strings of research together and provides a unified point of view. Thanks to the tensor formulation, the interplay between different subspaces is highlighted: ambiguities are easier to spot, algorithms can more easily be derived, and a potential choice of priors for some subspaces is straight-forward. Moreover, analogously to the previous chapter, the tensor formulation can be extended to the case of a camera network where multiple static affine cameras observe the same deforming and moving non-rigid object. Thanks to the insights gained through this tensor notation, the closed-form solution of the previous chapter only needs to be slightly adapted in order to be also applicable to the non-rigid case. Hence, non-rigid reconstruction become possible without feature point correspondences between different cameras.

5.1 Related Work

Since this chapter builds heavily upon previous factorization formulations, references to prior work in this area will often be given at the appropriate places throughout the text. Here, only a short overview of the most important developments in the area of factorizations for rigid and especially non-rigid structure-from-motion problems will be presented. For further references about factorizations for structure-from-motion we also refer to the previous chapter.

The previous chapter has already described Tomasi and Kanade's seminal work on a low-rank factorization for rigid structure-from-motion problems [TK92]. To recapitulate, in their work, 2D feature point tracks of a single rigidly moving object observed by an affine camera have been shown to be restricted to a 4-dimensional subspace. In a similar way, trajectories of multiple independently moving objects also give rise to a low-dimensional data matrix [CK95]. This time however, the trajectories originate from multiple independent subspaces and the SfM-problem gets combined with a motion segmentation problem [TV07]. Articulated objects can also be modeled with low-rank factorizations and multiple subspaces. In contrast to multiple independent rigid objects, the subspaces of articulated objects are not independent anymore and can intersect each other [YP08; TR05]. In this chapter, we focus on non-articulated deformable objects. In contrast to the previously mentioned approaches, non-rigid structure-from-motion (NRSfM) enjoys less strict algebraic constraints: the low-rank assumption only holds approximatively. The classical way to model these kind of deformations is the basis-shape model [BHB00] which will be presented in detail in Sec. 5.2.2. Basis-shape factorization approaches suffer from the fact that the initial factorization must be corrected with a so-called corrective transformation in order to account for the algebraic Kronecker-structure prescribed by the basis-shape model. Later work addressed this issue in detail [Bra01] and presented a closed-form linear solution [XCK04] or a more robust non-linear solution [Bra05]. Another line of research for NRSfM are piecewise approaches which depart from the classical factorization framework. With piecewise NRSfM we mean non-factorization based approaches which build a patch-based representation of non-rigid deformable shapes and glue these patches together using heuristics such as smoothness assumptions of motion and shape. Even though piecewise approaches for NRSfM present an interesting line

of future research, in this chapter we solely focus on factorization approaches for NRSfM and refer to [FRA11] and references therein for piece-wise NRSfM.

As already mentioned, this chapter presents a unified view of low-rank models for 3D point trajectories of non-rigidly moving 3D point clouds, such as basis-shape approaches [THB08], implicit low-rank shape models [PBA10], or such as the more recent representations using a Discrete Cosine Transform (DCT) basis [Akh+08; Akh+11]. A tensor-based formulation is derived which seamlessly handles the case of multiple cameras, subsuming earlier models for binocular NRSfM [WZ02].

Building upon insights from the rigid case in Chap. 4, we are able to give a clear and intuitive description of the algebraic constraints encapsulated in 2D non-rigid feature point trajectories seen in different cameras. Equipped with this deeper understanding, the previous closed-form factorization algorithm of Chap. 4 can be extend from the rigid to the non-rigid setting. Alternatively, building upon Chap. 3, an iterative multi-linear optimization can be used which can handle partial feature tracks. In contrast to recently presented NRSfM algorithms for multiple cameras [BA06; Lla+11; Zah+11], our algorithms do not require feature point correspondences between different cameras. In summary, the main contributions of this chapter are:

i) A unified formulation for low-rank nonrigid deforming shapes which clearly reveals the interactions of all the involved subspaces and enables an intuitive reasoning about these subspaces thereby avoiding getting lost in shuffling around indices. As we will see, this also facilitates the development of algorithms.

ii) A closed-form factorization algorithm or a non-linear iterative algorithm which compute the 3D reconstruction given 2D feature tracks in multiple cameras. No feature point correspondences between different cameras need to be known.

5.2 Low-Rank Non-Rigid Deformations

5.2.1 Redundancy in trajectories

Wolf and Zomet's work [WZ02] considers the case of two cameras observing a non-rigid object. They assume that every 3D point tracked in the second camera can be expressed as a linear combination of some

of the 3D points tracked by the first camera. This approach can be generalized by assuming that any point $\mathbf{x}_{n,f} \in \mathbb{R}^3$ in 3D-space can be expressed as a linear combination $\mathbf{x}_{n,f} = \mathbf{Y}_f \mathbf{s}_n$ of d_S time-varying basis points $\mathbf{Y}_f \in \mathbb{R}^{3 \times d_S}$. Stacking the data from multiple points over multiple frames into one matrix gives

$$\mathbf{X} = \begin{bmatrix} \Downarrow_f \Rightarrow_n \mathbf{x}_{f,n} \end{bmatrix} = \begin{bmatrix} \Downarrow_f \mathbf{Y}_f \end{bmatrix} \begin{bmatrix} \Rightarrow_n \mathbf{s}_n \end{bmatrix} = \mathbf{Y}\mathbf{S} \in \mathbb{R}^{3F \times N}. \tag{5.1}$$

This representation reveals two important facts: Firstly, the matrix $\mathbf{X}$ is highly redundant as it factorizes into two lower-rank matrices (given $d_S < \min(3F, N)$) and secondly, the temporally varying part $\mathbf{Y}$ is split from the temporally static part $\mathbf{S}$. This low-rank factorization due to redundancies in trajectories lies at the heart of all bilinear non-rigid shape models. This representation has also been suggested in [PBA10] where this low-rank model for 3D trajectories is called 3D-implicit low-rank shape model. As will be seen in Sec. 5.3, this low-rank model leads to severe ambiguities in the 3D structure for monocular image sequences: For any regular 3-by-3 matrices $\mathbf{H}_f$, $\mathbf{X}$ and $\begin{bmatrix} \searrow_f \mathbf{H}_f \end{bmatrix} \mathbf{X}$ will fulfill the same low-rank constraint leaving the dynamic 3D structure ambiguous. Hence, in monocular sequences there is a need for additional constraints, such as a Kronecker structure due to a basis-shape model (see Sec. 5.2.2) or smoothness priors on $\mathbf{Y}_f$ and as-rigid-as-possible assumption on $\mathbf{S}$ as done in [PBA10]. Note for multiple cameras however, the low-rank assumption itself is sufficient and no additional constraints are necessary (see also Sec. 5.5). Related to the above formulation is the implicit model of [OB08] where the low-rank model has not been applied directly to the 3D trajectory matrix $\mathbf{X}$ but rather to the observed 2D image trajectories. Specifically, the observed image trajectories were given by $\mathbf{W} = \mathbf{A}\mathbf{S}$ where $\mathbf{A} = \begin{bmatrix} \searrow_f \mathbf{C}_f \end{bmatrix}_{2F \times 3F} \mathbf{Y}$ is a combination between the time-varying basis points $\mathbf{Y}$ and the affine camera matrices $\mathbf{C}_f$ of a single moving camera. However, [OB08] did not enforce the correct algebraic structure on $\mathbf{A}$ and therefore this low-rank model regularizes the 2D feature tracks but does not provide any 3D reconstruction of the moving points.

5.2.2 Basis-Shapes

Traditionally, the 3D shape $\mathbf{X}_f \in \mathbb{R}^{3 \times N}$ of a deformable object at frame f is modeled as a linear combination $\mathbf{X}_f = \sum_{b=1}^{B} \mathbf{\Omega}_{f,b} \mathbf{S}_b = \begin{bmatrix} \mathbf{\Omega}_{f,:} \otimes \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \Downarrow_b \mathbf{S}_b \end{bmatrix}$ of B temporally static basis shapes $\mathbf{S}_b \in \mathbb{R}^{3 \times N}$ with $b = [B]$, weighted

by time-varying weights $\mathbf{\Omega} \in \mathbb{R}^{F \times B}$ [BHB00]. Collecting the data over all frames leads to

$$\mathbf{X} = \begin{bmatrix} \Downarrow_f \mathbf{X}_f \end{bmatrix} = \begin{bmatrix} \mathbf{\Omega} \otimes \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \Downarrow_b \mathbf{S}_b \end{bmatrix}. \tag{5.2}$$

Hence, the basis shape approach follows from Eq. (5.1) by choosing $\mathbf{Y} = \mathbf{\Omega} \otimes \mathbf{I}_3$.

5.2.3 Smooth Trajectories With Discrete Cosine Transform

Now, assume that the 3D shape $\mathbf{X}_f$ deforms smoothly over time, which implies that the x-, y-, and z-coordinates of the time-varying basis points can be represented in a truncated Discrete Cosine Transform (DCT) basis $\mathbf{Y} = [\mathbf{D} \otimes \mathbf{I}_3]$ where $\mathbf{D} \in \mathbb{R}^{F \times B}$ is the truncated DCT basis. We notice that this is of the same algebraic form as the previous basis shape formulation. The semantic connection is that a smooth trajectory motion in the basis shape model implies that the basis shape weights vary smoothly as well and can therefore be represented in a truncated DCT basis $\mathbf{\Omega} = \mathbf{DQ}$ where $\mathbf{Q} \in \mathbb{R}^{B \times B}$ is the change-of-basis matrix. Inserting this into Eq. (5.2) and using the Kronecker product property of Eq. (2.3) we get the chain of equations $\mathbf{X} = \begin{bmatrix} \mathbf{DQ} \otimes \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \Downarrow_b \mathbf{S}_b \end{bmatrix} = \begin{bmatrix} \mathbf{D} \otimes \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{Q} \otimes \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \Downarrow_f \mathbf{S}_b \end{bmatrix}$. The change of basis of the basis shapes weights thus leads to new basis shapes $[\mathbf{Q} \otimes \mathbf{I}_3][\Downarrow_b \mathbf{S}_b]$. Each column of this new basis shape matrix corresponds to one smoothly moving 3D point of the deformable object and captures the collection of coefficients for its three separate F-dimensional trajectories in x-, y, and z-direction expressed in a truncated DCT basis. This representation clearly reveals that a linear transformation of the basis shape coefficients implies a change of the basis shapes (and the other way around) due to the bilinearity of the shape representation in Eq. (5.2). A slightly different derivation of this observation has first appeared in [Akh+08; Akh+11][1] where it was called duality of the shape and trajectory basis. It is important to highlight that from an algebraic point of view, the trajectory space approach is completely equivalent to the basis shape approach.

[1] In [Akh+08], the basis $\mathbf{I}_3 \otimes \mathbf{D}$ has been used which is a column and row permutation of $\mathbf{D} \otimes \mathbf{I}_3$ (This has also been noted in [GM11]). As our derivation shows, the latter version is more natural and has also been used in [Akh+11].

5.3 Projecting Low-Rank Non-Rigid Deformations

Having established the low-dimensional structure of deforming 3D points in the previous section, this section presents an analysis of the resulting 2D trajectories observed in affine cameras. We will see that the image observations originate from three interacting subspaces which are most naturally modeled in a multilinear algebra framework.

5.3.1 General Low-Rank Non-Rigid Motion

In preparation for multiple cameras, the bilinear models for non-rigid trajectories of the previous section are slightly reformulated: the rigid component of the non-rigid motion is modeled explicitly with a temporally varying rotation $\mathbf{R}_f$ and translation $\mathbf{t}_f$. This has two advantages: firstly, the non-rigid deformation does not need to explain the rigid component of the motion which is advantageous especially for large rigid transformations, and secondly it facilitates the extension of the model with a camera rig observing the deformable object.

In the most general case, the non-rigid trajectories are given by

$$\begin{pmatrix}\mathbf{x}_{f,n}\\1\end{pmatrix} = \begin{bmatrix}\mathbf{R}_f & \mathbf{t}_f\\ \mathbf{0}_{1\times 3} & 1\end{bmatrix}\begin{bmatrix}\mathbf{Y}_f & \mathbf{0}_{3\times 1}\\ \mathbf{0}_{1\times d_S} & 1\end{bmatrix}\begin{bmatrix}\mathbf{s}_n\\1\end{bmatrix} = \begin{bmatrix}\mathbf{R}_f\mathbf{Y}_f & \mathbf{t}_f\\ \mathbf{0}_{1\times d_S} & 1\end{bmatrix}\begin{bmatrix}\mathbf{s}_n\\1\end{bmatrix}. \quad (5.3)$$

The following derivations mostly follow similar steps as performed in Chap. 4. By making use of the Kronecker-product property of Eq. (2.3), the projection of point n into affine camera axis $\mathbf{c}_k^T \in \mathbb{R}^{1\times 4}$ at frame f is given by

$$\mathcal{W}_{k,f,n} = \mathbf{c}_k^T\begin{pmatrix}\mathbf{x}_{f,n}\\1\end{pmatrix} = \mathbf{c}_k^T\begin{bmatrix}\mathbf{R}_f\mathbf{Y}_f & \mathbf{t}_f\\ \mathbf{0}_{1\times d_S} & 1\end{bmatrix}\begin{pmatrix}\mathbf{s}_n\\1\end{pmatrix} \quad (5.4)$$

$$= \operatorname{vec}\left(\begin{bmatrix}\mathbf{R}_f\mathbf{Y}_f & \mathbf{t}_f\\ \mathbf{0}_{1\times d_S} & 1\end{bmatrix}\right)^T\left[\begin{pmatrix}\mathbf{s}_n\\1\end{pmatrix}\otimes\mathbf{c}_k^T\right] \quad (5.5)$$

$$= \begin{bmatrix}\operatorname{vec}(\mathbf{R}_f\mathbf{Y}_f)^T & \mathbf{t}_f^T & 1\end{bmatrix}\mathcal{S}_{(f)}\left[\begin{pmatrix}\mathbf{s}_n\\1\end{pmatrix}\otimes\mathbf{c}_k^T\right] \quad (5.6)$$

with the flattened core tensor

$$\mathcal{S}_{(f)} = \begin{bmatrix}\mathbf{I}_{d_S}\otimes\begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_{3\times 1}\end{bmatrix} & \mathbf{0}_{3d_S\times 4}\\ \mathbf{0}_{4\times 4d_S} & \mathbf{I}_4\end{bmatrix} \in \mathbb{R}^{3d_S+4\times 4d_S+4}. \quad (5.7)$$

Stacking the dynamic part row-wise and the temporally non-varying part column-wise leads to a flattened data tensor along the temporal mode

$$\mathcal{W}_{(f)} = \left[\Downarrow_f \Rightarrow_{n,k} \mathcal{W}_{k,f,n}\right] = \mathbf{M}\mathcal{S}_{(f)}(\mathbf{S}^T \otimes \mathbf{C})^T \in \mathbb{R}^{F\times 2KN} \quad (5.8)$$

with
$$\mathbf{M} = \left[\Downarrow_f \left(\mathrm{vec}\left(\mathbf{R}_f \mathbf{Y}_f\right)^T, \mathbf{t}_f^T, 1\right)\right] \in \mathbb{R}^{F\times 3d_S+4} \quad (5.9)$$

$$\mathbf{S} = \left[\Rightarrow_n \begin{pmatrix}\mathbf{s}_n \\ 1\end{pmatrix}\right] \in \mathbb{R}^{d_S+1\times N} \quad (5.10)$$

$$\mathbf{C} = \left[\Downarrow_k \mathbf{c}_k^T\right] \in \mathbb{R}^{2K\times 4}. \quad (5.11)$$

$\mathbf{M}$, $\mathbf{S}$, and $\mathbf{C}$ capture the motion, structure, and camera subspaces respectively. The flattened data tensor along the temporal mode $\mathcal{W}_{(f)}$ must be of rank $3d_S + 4$ due to the revealed factorization. Reshaping this matrix into a third-order Tucker tensor gives $\mathcal{W} = \mathcal{S} \times_k \mathbf{C} \times_f \mathbf{M} \times_n \mathbf{S}$ with core tensor $\mathcal{S} \in \mathbb{R}^{4\times 3d_S+4\times d_S+1}$. The rigid motion case presented in Chap. 4 results by choosing $d_S = 3$ and every $\mathbf{Y}_f$ as the identity matrix.

5.3.2 Basis-Shape Model

In the more specific basis shape representation (or equivalently in the trajectory basis representation), a property of the Kronecker product (see Eq. (2.4)) leads to an interesting result if the model of Eq. (5.2) is again extended with a rigid transformation

$$\begin{bmatrix}\mathbf{X}_f \\ \mathbf{1}_{1\times N}\end{bmatrix} = \begin{bmatrix}\mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0}_{1\times 3} & 1\end{bmatrix}\begin{bmatrix}\mathbf{\Omega}_{f,:} \otimes \mathbf{I}_3 & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3B} & 1\end{bmatrix}\begin{bmatrix}\Downarrow_b \mathbf{S}_b \\ \mathbf{1}_{1\times N}\end{bmatrix} \quad (5.12)$$

$$= \begin{bmatrix}\mathbf{\Omega}_{f,:} \otimes \mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0}_{1\times 3B} & 1\end{bmatrix}\begin{bmatrix}\Downarrow_b \mathbf{S}_b \\ \mathbf{1}_{1\times N}\end{bmatrix} \in \mathbb{R}^{4\times N}. \quad (5.13)$$

The rigid rotation $\mathbf{R}_f$ interacts with the non-rigid dynamic part $\mathbf{\Omega}$ through a Kronecker product. Inserting Eq. (5.13) in Eq. (5.3), we immediately see that the basis shape approach follows from the previous formulation in Eq. (5.8) by setting $d_S = 3B$ and choosing

$$\mathbf{M} = \left[\Downarrow_f \left(\mathrm{vec}\left(\mathbf{\Omega}_{f,:} \otimes \mathbf{R}_f\right)^T, \mathbf{t}_f^T, 1\right)\right] \in \mathbb{R}^{F\times 9B+4}, \ \mathbf{S} = \begin{bmatrix}\Downarrow_b \mathbf{S}_b \\ \mathbf{1}_{1\times N}\end{bmatrix} \in \mathbb{R}^{3B+1\times N}. \quad (5.14)$$

This representation might seem rather unconventional, however it clearly reveals all the multilinear relationships encoded in the data. This

representation not only seamlessly handles the case of multiple cameras, but also facilitates the exposure of the unknown matrices for an iterative optimization algorithm. In order to exemplify this fact, we arrange the entries of the tensor in a better-known form revealing a matrix of rank $3B+1$ which exposes the basis shape matrix $\mathbf{S}$. This arrangement actually corresponds to the transpose of the flattening of the tensor along the mode of the points

$$\mathcal{W}_{(n)}^T = \mathcal{W}_{(f,k)} = \left[\mathbf{C} \otimes \mathbf{M}\right]^T \mathcal{S}_{(n)}^T \mathbf{S} = \left[\Downarrow_f \mathbf{C} \begin{bmatrix} \mathbf{\Omega}_{f,:} \otimes \mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0}_{1\times 3B} & 1 \end{bmatrix}\right] \mathbf{S}. \quad (5.15)$$

Note that flattening a tensor along a mode is a purely algebraic operation and can easily be done by strictly following some rules. In previous work, usually only one single camera with orthogonal camera axes with non-varying scale has been considered. This corresponds to choosing $\mathbf{C} = [\mathbf{I}_2, \mathbf{0}_{2\times 2}]$ in the above formulation which leads to the standard monocular NRSfM basis shape equation

$$\mathcal{W}_{(n)}^T = \left[\Downarrow_f \left[\mathbf{\Omega}_{f,:} \otimes \mathbf{R}_{f,1:2,:}, \mathbf{t}_{f,1:2}\right]\right] \begin{bmatrix} \Downarrow_b \mathbf{S}_b \\ \mathbf{1}_{1\times N} \end{bmatrix}. \quad (5.16)$$

At this point it is interesting to put this formulation in relation to Torresani et.al. 's work [Tor+01]. Based on the basis-shape model in Eq. (5.15), [Tor+01] proposes a low-rank constraint for optical flow of non-rigid shapes and a 3D reconstruction algorithm for basis-shapes (i.e. merging [BHB00] with Irani's rigid optical flow constraints [Ira99]). Even though presented originally in a completely different way, the optical flow constraint actually is a consequence of considering $\mathcal{W}_{(f)}$ whereas the 3D reconstruction is based on a block-coordinate descent algorithm derived from $\mathcal{W}_{(n)}^T$. [Tor+01] even formulated an extension of their 3D reconstruction algorithm for multiple cameras with the simplification that the data is centered, i.e. translations are not modeled. Based on the same formulation, Del Bue and Agapito showed experimentally [BA06] that a stereo setup indeed improves the reconstruction accuracy of a basis shape model. More recently, Lladó et.al. [Lla+11] drew the same conclusion in an iterative Ransac-framework for a binocular stereo setup with perspective cameras. However, similar to [Zah+11], all these approaches require feature point correspondences between different cameras to be known because the exact relation between $\mathcal{W}_{(f)}$ and $\mathcal{W}_{(n)}$ has not been established. By making use of this relation, Sec. 5.5

presents an algorithm which can handle cases where no correspondences between different cameras are available. The interested reader is also referred to Hartley and Vidal's work [HV08] which presents a solution for the monocular basis-shape model with perspective rather than affine cameras. We leave it as an open question whether there exists a similar solution to the perspective multi-camera basis-shape model[2].

5.4 A Detailed View on Basis-Shapes Models

In this section, existing results for the basis-shape representation are recompiled in our tensor notation. This section will therefore not present new results per se, but rather a new way of deriving and reasoning about them. We think that such a unified presentation of previously dispersed results in a consistent formulation will ultimately lead to a better and clearer understanding, thereby avoiding that the same results are derived multiple times because they were phrased in a slightly different notation or context in previous work.

In Sec. 5.4.1, the non-uniqueness of factorization approaches will be recapitulated shortly, while Sec. 5.4.2 presents a different view on the results of [Par+10]. Finally, Sec. 5.4.3 gives a generalized view on the widely-used orthogonality constraints for computing the corrective transformation after a low-rank factorization of the data matrix (establishing connections to [Par+10; XCK04; Bra05] amongst others).

5.4.1 Ambiguities

In this section, we will look at the ambiguities inherently contained in low-rank non-rigid trajectories. To start, let us define the two affine transformations for the camera and the structure

$$\mathbf{Q}_C = \begin{bmatrix} \mathbf{T}_C & \mathbf{t}_C \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \quad \text{and} \quad \mathbf{Q}_S = \begin{bmatrix} \mathbf{T}_S & \mathbf{t}_S \\ \mathbf{0}_{1\times d_S} & 1 \end{bmatrix} \in \mathbb{R}^{d_S+1\times d_S+1}.$$

These affine transformation are the source for the ambiguities. Applying these transformations to the camera and structure matrices should not

[2] Note however that even the much simpler extension of the multi-camera rigid model presented in Chap. 4 to perspective cameras proves to be rather challenging.

modify the core tensor. Hence, solving

$$\mathcal{S} = \mathcal{S} \times_f \mathbf{Q}_M \times_k \mathbf{Q}_C \times_n \mathbf{Q}_S^T \tag{5.17}$$

for $\mathbf{Q}_M$ gives the transformation for the motion space $\mathbf{M}\mathbf{Q}_M^{-1}$ in order to compensate for the affine transformation of the cameras and the structure. As an explicit form for $\mathbf{Q}_M$ reveals all the ambiguities in the NRSfM formulation, in particular for the basis shape approach, we will go through the necessary algebra. From Eq. (5.4) and the Kronecker product properties in Eq. (2.3) and Eq. (2.4) it follows

$$\begin{aligned}
\mathcal{W}_{:,f,n} =& \mathbf{C}\mathbf{Q}_C^{-1}\mathbf{Q}_C \begin{bmatrix} \mathbf{R}_f\mathbf{Y}_f & \mathbf{t}_f \\ \mathbf{0}_{1\times d_S} & 1 \end{bmatrix} \mathbf{Q}_S\mathbf{Q}_S^{-1} \begin{pmatrix} \mathbf{s}_n \\ 1 \end{pmatrix} && (5.18)\\
=& \left[\mathrm{vec}\left(\mathbf{T}_C\mathbf{R}_f\mathbf{Y}_f\mathbf{T}_S\right)^T, \left(\mathbf{T}_C\mathbf{R}_f\mathbf{Y}_f\mathbf{t}_S + \mathbf{T}_C\mathbf{t}_f + \mathbf{t}_C\right)^T, 1\right] \mathcal{S}_{(f)} \\
& \left[\mathbf{Q}_S^{-1}\begin{bmatrix} \mathbf{s}_n \\ 1 \end{bmatrix} \otimes \mathbf{Q}_C^{-T}\mathbf{C}^T\right] && (5.19)\\
=& \left[\mathrm{vec}\left(\mathbf{R}_f\mathbf{Y}_f\right)^T, \mathbf{t}_f^T, 1\right] \underbrace{\begin{bmatrix} \mathbf{T}_S \otimes \mathbf{T}_C^T & \mathbf{t}_S \otimes \mathbf{T}_C^T & \mathbf{0}_{3d_S\times 1} \\ \mathbf{0}_{3\times 3d_S} & \mathbf{T}_C^T & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3d_S} & \mathbf{t}_C^T & 1 \end{bmatrix}}_{=\mathbf{Q}_M^{-1}} \mathcal{S}_{(f)} \\
& \left[\mathbf{Q}_S^{-1} \otimes \mathbf{Q}_C^{-T}\right]\left[\begin{bmatrix} \mathbf{s}_n \\ 1 \end{bmatrix} \otimes \mathbf{C}^T\right]. && (5.20)
\end{aligned}$$

We recall how the basis shape approach models the motion, namely as $\mathbf{R}_f\mathbf{Y}_f = \mathbf{\Omega}_{f,:} \otimes \mathbf{R}_f$. As shown in Eq. (5.18) this motion is right-multiplied with $\mathbf{T}_S$, the non-translational component of the affine transformation of the structure. However, the algebraic structure in the motion must be preserved, hence $[\mathbf{\Omega}_{f,:} \otimes \mathbf{R}_f]\,\mathbf{T}_S$ must have Kronecker-structure as well. If the affine transformation looks like $\mathbf{T}_S = \mathbf{Q}_\Omega \otimes \mathbf{T}_B$ with $\mathbf{Q}_\Omega \in \mathbb{R}^{B\times B}$ and $\mathbf{T}_B \in \mathbb{R}^{3\times 3}$, then again due to the Kronecker product property it follows

$$\mathbf{M}_f\mathbf{T}_S = [\mathbf{\Omega}_{f,:} \otimes \mathbf{R}_f]\,[\mathbf{Q}_\Omega \otimes \mathbf{T}_B] = [\mathbf{\Omega}_{f,:}\mathbf{Q}_\Omega] \otimes [\mathbf{R}_f\mathbf{T}_B], \tag{5.21}$$

and the algebraic structure gets preserved. Hence, the allowable affine transformations for the basis shape structure belong to a restricted class, specifically the inverse $\mathbf{T}_S^{-1} = \mathbf{Q}_\Omega^{-1} \otimes \mathbf{T}_B^{-1}$ is applied to the structure

$[\Downarrow_b \mathbf{S}_b]$

$$\mathbf{T}_S^{-1}\,[\Downarrow_b \mathbf{S}_b] = \left[\mathbf{Q}_\Omega^{-1} \otimes \mathbf{T}_B^{-1}\right][\Downarrow_b \mathbf{S}_b] \tag{5.22}$$

$$= \left[\mathbf{Q}_\Omega^{-1} \otimes \mathbf{I}_3\right]\left[\mathbf{I}_B \otimes \mathbf{T}_B^{-1}\right][\Downarrow_b \mathbf{S}_b] \tag{5.23}$$

$$= \left[\mathbf{Q}_\Omega^{-1} \otimes \mathbf{I}_3\right]\left[\Downarrow_b \mathbf{T}_B^{-1}\mathbf{S}_b\right], \tag{5.24}$$

which shows that the transformation $\mathbf{T}_B$ transforms all the basis shape basis separately $\mathbf{T}_B^{-1}\mathbf{S}_b$. Since the matrices $\mathbf{R}_f$ represent rotation matrices, the transformations $\mathbf{T}_C$ and $\mathbf{T}_B$ are restricted to scaled rotation matrices, where the scaling factors are reciprocal such that they cancel each other. The transformation $\mathbf{Q}_\Omega$ is an arbitrary regular matrix representing an arbitrary choice for the basis of the basis shape weights $\mathbf{\Omega}$. As previously pointed out, if the basis shape weights vary smoothly, the transformation $\mathbf{Q}_\Omega$ can be chosen such that $\mathbf{\Omega}\mathbf{Q}_\Omega = \mathbf{D}$ represents a truncated DCT basis.

Let us summarize our findings: for the NRSfM basis shape approach, we can freely choose

- a similarity transformation $\mathbf{Q}_C$ for the cameras $\mathbf{P}$
- a scaled rotation matrix $\mathbf{T}_B$ (with inverse scaling w.r.t. the camera similarity transformation) representing the choice of basis for the basis shapes $\mathbf{S}_b$, and a translation vector $\mathbf{t}_S \in \mathbb{R}^{3B\times 1}$
- an arbitrary regular matrix $\mathbf{Q}_\Omega$ representing the arbitrary choice of basis for the basis shape weights.

For the monocular NRSfM basis shape formulation, Brand [Bra05] already verbally mentioned the structure ambiguity in the form of a Kronecker product $\mathbf{T}_S = \mathbf{Q}_\Omega \otimes \mathbf{T}_B$ in a short note. Unfortunately, maybe due to the lack of formally underpinning this insight, the deeper implications of this fact have mostly been overlooked in later work, resulting in lengthy, obscure proofs for the non-uniqueness of the basis shape formulation.

5.4.2 Stability of Monocular Non-Rigid Trajectory Factorization

Restricting ourselves to just one single camera, the original quadrilinear factorization problem in Eq. (5.15) reduces to a trilinear problem in

the rigid motions $\mathbf{R}_f$ and $\mathbf{t}_f$, the basis shape weights $\mathbf{\Omega}$, and the basis shapes $\mathbf{S}_b$. Furthermore, by precomputing the rigid motion (e.g. from the rigid background), the problem becomes a bilinear factorization problem. Finally, by making use of the smoothly varying basis shape weights prior (which means fixing $\mathbf{\Omega}$ to a truncated DCT basis $\mathbf{D}$), the only unknowns are the $3BN$ DCT coefficients of the N point trajectories which results in a linear system of equations. This is exactly the setup used in [Par+10] where it is shown that a unique solution to this linear problem only exists if the rigid translations are *not* smooth over time, i.e. can not be exactly expressed in the truncated DCT basis. Given the remaining unknowns, the uniqueness of the trajectory reconstruction depends solely on the system matrix

$$[\mathbf{C} \otimes \mathbf{M}]^T \mathcal{S}_{(n)}^T = \left[\Downarrow_f \mathbf{C} \begin{bmatrix} \mathbf{\Omega}_{f,:} \otimes \mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0}_{1\times 3B} & 1 \end{bmatrix} \right] \tag{5.25}$$

of the linear system in Eq. (5.15) for the structure $\mathbf{S}$. If the camera matrix $\mathbf{C}$ has full column-rank, i.e. $\operatorname{rank}(\mathbf{C}) = 4$, then the system matrix has full column-rank and there is a unique solution for the structure $\mathbf{S}$ independently whether the rigid translation is smooth or not. However, if the column-rank of the camera matrix is smaller than 4 (this is for example the case for one single extrinsically calibrated camera which has only three rows) then the uniqueness of the solution depends on whether or not the rigid translation is representable in the basis of the basis shape weights

$$-[\Downarrow_f \mathbf{R}_f \mathbf{t}_f] = [\mathbf{\Omega} \otimes \mathbf{I}_3]\, \mathbf{t}_\Omega + [\mathbf{\Omega} \otimes \mathbf{I}_3]_\perp\, \mathbf{t}_\Omega^\perp, \tag{5.26}$$

for some coefficients $\mathbf{t}_\Omega \in \mathbb{R}^{3B\times 1}$ and $\mathbf{t}_\Omega^\perp \in \mathbb{R}^{3F-3B\times 1}$. The system matrix has full column-rank only if $\mathbf{t}_\Omega^\perp \neq \mathbf{0}$ and then there is a unique solution. As derived in [Par+10], with noisy data, the robustness of the solution depends on how accurate the point trajectories can be represented in the basis $\mathbf{\Omega}$ and on how strong the component $\mathbf{t}_\Omega^\perp$ of the camera translation is. Asymptotic results for the accuracy have been derived, specifically the reconstruction is exact when the ratio between the norm of $\mathbf{t}_\Omega^\perp$ and the reconstruction error of the point trajectory goes to infinity.

5.4.3 Orthogonality Constraints for Corrective Transformation

Enforcing the Kronecker-Structure With Corrective Transformation

Low-rank factorization methods, may it be for the monocular, multi-camera, rigid or non-rigid basis shape formulation, generally run into the same problem: The low-rank factorization of a matrix returns two matrix factors, one of which should have a certain bilinear Kronecker-structure with one factor having orthogonality constraints imposed on columns or rows. Unfortunately, a general purpose factorization method does not provide factors with the correct algebraic structure and a *corrective transformation* is required to transform the factors into a valid form.

As a concrete example, let us look at the low-rank factorization for the monocular basis shape formulation. However a similar reasoning applies for example also to the Kronecker structure of the second factor in the low-rank factorization of $\mathcal{W}_{(f)}$ in the multi-camera rigid and non-rigid case. The flattened data tensor $\mathcal{W}_{(n)}$ along the mode of the structure must have rank $3B+1$ as can be seen in Eq. (5.16). Hence, factoring the flattened tensor $\mathcal{W}_{(n)}^T = \hat{\mathbf{M}}\hat{\mathbf{S}}$ (e.g. with the singular value decomposition) provides two matrix factors $\hat{\mathbf{M}} \in \mathbb{R}^{F\times 3B+1}$ and $\hat{\mathbf{S}} \in \mathbb{R}^{3B+1\times N}$ of the correct dimensions, missing however the correct algebraic Kronecker-structure. A corrective transformation $\mathbf{Q} \in \mathbb{R}^{3B+1\times 3B+1}$ is needed such that

$$\hat{\mathbf{M}}\mathbf{Q} = [\Downarrow_f [\mathbf{\Omega}_{f,:} \otimes \mathbf{R}_{f,1:2,:}, \mathbf{t}_{f,1:2}]] \quad \text{and} \quad \mathbf{Q}^{-1}\hat{\mathbf{S}} = \begin{bmatrix} \Downarrow_b \mathbf{S}_b \\ \mathbf{1}_{1\times N} \end{bmatrix}. \tag{5.27}$$

The two rows of the truncated rotation matrix $\mathbf{R}_{f,1:2,:}$ should of course be orthonormal.

Direct Approach

Abstracting from this specific problem instance, the general problem is given $\mathbf{A} \in \mathbb{R}^{mp\times nq}$ to find $\mathbf{Q} \in \mathbb{R}^{nq\times nq}$ such that

$$\mathbf{A}\mathbf{Q} = \mathbf{B} \otimes \mathbf{C}, \tag{5.28}$$

with $\mathbf{B} \in \mathbb{R}^{m\times n}$ and $\mathbf{C} \in \mathbb{R}^{p\times q}$ of which some of the row subsets $\mathcal{I}_i$ have to obey orthogonality constraints $\mathbf{C}_{\mathcal{I}_i,:}\mathbf{C}_{\mathcal{I}_j,:}^T = \alpha_{i,j}\mathbf{I}_p$ where the scaling factors $\alpha_{i,j}$ might be known or unknown depending on the specific

problem. We remark that due to the Kronecker product property there are multiple solutions as

$$[\mathbf{B}\mathbf{Q}_B \otimes \mathbf{C}\mathbf{Q}_C] = [\mathbf{B} \otimes \mathbf{C}]\,[\mathbf{Q}_B \otimes \mathbf{Q}_C] = \mathbf{A}\mathbf{Q}\,[\mathbf{Q}_B \otimes \mathbf{Q}_C]$$

with orthogonal $\mathbf{Q}_C$ and therefore $\mathbf{Q}\,[\mathbf{Q}_B \otimes \mathbf{Q}_C]$ is also a valid solution. Hence the corrective solution is unique up to a right multiplication with $[\mathbf{Q}_B \otimes \mathbf{Q}_C]$.

We can make use of this ambiguity, however. The transformations $\mathbf{Q}_B$ and $\mathbf{Q}_C$ can be chosen arbitrarily, for example we might impose that $\mathbf{B}_{1:n,:}\mathbf{Q}_B = \mathbf{I}_n$ and $\mathbf{C}_{1:q,:}\mathbf{Q}_C = \mathbf{I}_q$ and hence $\mathbf{A}_{1:nq,:}\mathbf{Q} = \mathbf{I}_n \otimes \mathbf{I}_q$ are $nq \times nq$ linear equations in the $nq \times nq$ unknowns of $\mathbf{Q}$. Unfortunately, this approach can not be applied to the basis shape formulation, as only data for $\mathbf{\Omega}_{c,:} \otimes \mathbf{R}_f$ with $c = f$ is observed, the remaining bilinear terms of $[\Downarrow_c \mathbf{\Omega}_{c,:}] \otimes [\Downarrow_f \mathbf{R}_f]$ for which $f \neq c$ are missing and therefore there is an insufficient number of equations. In such cases, other constraints must be considered such as the orthonormality constraints.

Orthonormality Constraints

The *orthonormality constraints* make use of the fact that some tuples of row subsets of the factor $\mathbf{C}$ are constrained by orthogonality constraints: for each such tuple $(\mathcal{I}_i, \mathcal{I}_j)$, a block $\alpha_{i,j}\mathbf{I}_{|\mathcal{I}_i|}$ in the symmetric matrix $\mathbf{C}\mathbf{C}^T$ is known (maybe only up to the scaling $\alpha_{i,j}$). These blocks provide linear constraints for the unknown $\mathbf{Q}\mathbf{Q}^T$

$$\mathbf{A}\mathbf{Q}\mathbf{Q}^T\mathbf{A}^T = [\mathbf{B} \otimes \mathbf{C}]\,[\mathbf{B} \otimes \mathbf{C}]^T = \left[\mathbf{B}\mathbf{B}^T \otimes \mathbf{C}\mathbf{C}^T\right]. \tag{5.29}$$

Given enough observations, Eq. (5.29) results in an overdetermined linear system[3] in the $\frac{nq(nq+1)}{2}$ unknowns of the symmetric matrix $\mathbf{Q}\mathbf{Q}^T$. Once this linear system is solved, a final eigenvalue decomposition of $\mathbf{Q}\mathbf{Q}^T$ gives the solution $\mathbf{Q}$. Note that from the previous discussion we know that $\mathbf{Q}$ is unique only up to $[\mathbf{Q}_B \otimes \mathbf{Q}_C]$ which leads to

$$\mathbf{A}\mathbf{Q}\mathbf{Q}^T\mathbf{A}^T = \mathbf{A}\mathbf{Q}\,[\mathbf{Q}_B \otimes \mathbf{Q}_C]\,[\mathbf{Q}_B \otimes \mathbf{Q}_C]^T\,\mathbf{Q}^T\mathbf{A}^T \tag{5.30}$$

$$= \mathbf{A}\mathbf{Q}\left[\mathbf{Q}_B\mathbf{Q}_B^T \otimes \mathbf{I}_q\right]\mathbf{Q}^T\mathbf{A}^T, \tag{5.31}$$

[3] Alternatively, the problem can also be formulated as a positive-semidefinite programming (SDP) problem. In our experiments, this sometimes proved to be more robust than the linear least squares approach.

and therefore the solution to the orthonormality constraints in Eq. (5.29) is unique up to $\left[\mathbf{Q}_B\mathbf{Q}_B^T \otimes \mathbf{I}_q\right]$, which has been called a *block-scaled identity matrix* by Xiao et.al. [XCK04]. Each block (i,j) consists of a identity matrix of dimension $q \times q$ scaled by an unknown factor $\mathbf{Q}_{B,i,:}\mathbf{Q}_{B,j,:}^T$. Block-scaled identity matrices of this size have $\frac{n(n+1)}{2}$ degrees of freedom. Therefore there is a $\frac{n(n+1)}{2}$ dimensional solution space for the orthonormality constraints and $\frac{nq(nq+1)}{2} - \frac{n(n+1)}{2}$ linearly independent equations are necessary. The immediate question is: under which circumstances do the orthonormality constraints result in linearly independent equations?

Block-Skew-Symmetric Matrices

Let us present a new class of matrices, the so called *block-skew-symmetric matrices*. These matrices have also been introduced in [XCK04] and are defined as symmetric matrices $\mathbf{Y} = \left[\Downarrow_{i=1,\cdots,n} \Rightarrow_{j=1,\cdots,n} \mathbf{Y}_{i,j}\right] \in \mathbb{R}^{nq \times nq}$ whose $\mathbf{Y}_{i,j} \in \mathbb{R}^{q \times q}$ subblocks are skew-symmetric. This definition implies $\mathbf{Y}_{i,i} = \mathbf{0}_{q \times q}$ and since $\mathbf{Y}$ is symmetric, it holds that $\mathbf{Y}_{i,j} = \mathbf{Y}_{j,i}^T = -\mathbf{Y}_{j,i}$. Block-skew-symmetric matrices have $\frac{q(q-1)}{2}\frac{n(n-1)}{2}$ degrees of freedom (the diagonal blocks are all zero, hence there are $\frac{1}{2}n(n-1)$ non-zero blocks with $\frac{q(q-1)}{2}$ degrees of freedom each). Block-skew-symmetric matrices $\mathbf{Y} \in \mathbb{R}^{nq \times nq}$ have a nasty property. Consider the quadrilinear form in $\mathbf{a}, \mathbf{c} \in \mathbb{R}^n$ and in $\mathbf{b}, \mathbf{d} \in \mathbb{R}^q$ given by the block-skew-symmetric matrix $\mathbf{Y}$

$$\left[\mathbf{a} \otimes \mathbf{b}\right] \mathbf{Y} \left[\mathbf{c} \otimes \mathbf{d}\right].$$

This quadrilinear form equals zero if $\mathbf{a} = \mathbf{c}$ or if $\mathbf{b} = \mathbf{d}$. Hence, we get for *any* block-skew symmetric $\mathbf{Y} \in \mathbb{R}^{nq \times nq}$

$$0 = \left[\mathbf{B}_{i,:} \otimes \mathbf{C}_{j,:}\right] \mathbf{Y} \left[\mathbf{B}_{k,:} \otimes \mathbf{C}_{l,:}\right]^T, \tag{5.32}$$

whenever $i = k$ or $j = l$. If there are only constraints for which $i = k$ or $j = l$ then the general solution to the orthonormality constraints is $\mathbf{Q}\left[\mathbf{Y} + \mathbf{Q}_B\mathbf{Q}_B^T \otimes \mathbf{I}_q\right]\mathbf{Q}^T$ with $\frac{n(n+1)}{2} + \frac{q(q-1)}{2}\frac{n(n-1)}{2}$ linear degrees of freedom. Even more importantly, block-skew symmetric matrices are indefinite and hence the solution space also contains indefinite solutions which can not be factored into real valued corrective transformation $\mathbf{Q}$. Semi-definite programming (SDP) could be used to find a valid positive definite solution in this solution space which then in turn can be factored into $\mathbf{Q}$ and $\mathbf{Q}^T$. However, SDP does not force the block-skew

symmetric matrix $\mathbf{Y}$ to the zero matrix, but rather makes sure that the guaranteed positive definite component $\mathbf{Q}_B\mathbf{Q}_B^T \otimes \mathbf{I}_q$ outweights the indefinite component $\mathbf{Y}$. The indefinite part $\mathbf{Y}$ still spoils the Kronecker-product structure of the solution to the orthonormality constraints. There are multiple ways to resolve this problem:

i. **Rank-Constraint:** Only consider a slice of q consecutive columns, as proposed in [Bra05]: This results in a non-linear rank-q constraint on $\mathbf{Q}\mathbf{Q}^T$. Formally, let us introduce the matrix $\mathbf{0}_{+r} \in \mathbb{R}^{nq \times nq}$ which is equal to the zero matrix except with $\mathbf{I}_q$ on its r^{th} diagonal $q \times q$ block. This matrix basically slices q consecutive columns (or rows) out of $\mathbf{Q}$. Then we are looking for a solution to the orthonormality constraints of the form

$$\mathbf{Q}\mathbf{0}_{+r}\left[\mathbf{Y} + \mathbf{Q}_B\mathbf{Q}_B^T \otimes \mathbf{I}_q\right]\mathbf{0}_{+r}^T\mathbf{Q}^T \tag{5.33}$$

$$=\mathbf{Q}_{:,(r-1)q+1:rq}\left[\mathbf{Y}_{r,r} + \mathbf{Q}_{B,:,r}\mathbf{Q}_{B,:,r}^T \otimes \mathbf{I}_q\right]\mathbf{Q}_{:,(r-1)q+1:rq}^T. \tag{5.34}$$

Because the diagonal block $\mathbf{Y}_{r,r}$ of the block skew-symmetric matrix is necessarily zero the problem has been resolved. There is an irrelevant ambiguity of a Kronecker-product between a symmetric rank-1 matrix $\mathbf{Q}_{B,:,r}\mathbf{Q}_{B,:,r}^T$ and $\mathbf{I}_q$. This again corresponds to the arbitrary choice of basis for the columns of matrix $\mathbf{B}$. The drawback of this rank constraint is that we have to resort to non-linear iterative optimization methods which might get stuck in a local minima.

ii. **Mixed Constraints:** As previously mentioned, mixed constraints for pairwise linear independent vectors $\mathbf{a}$ and $\mathbf{c}$ resp. $\mathbf{b}$ and $\mathbf{d}$ in Eq. (5.32) help to resolve the problem. Hence, observations $0 = [\mathbf{B}_{i,:} \otimes \mathbf{C}_{j,:}]\,\mathbf{Y}\,[\mathbf{B}_{k,:} \otimes \mathbf{C}_{l,:}]$ with $i \neq k$ *and* $j \neq l$ are necessary. However, most often such mixed constraints are not available. And even if, is is not easy to spot that such constraints are actually contained in the available observations. For example in rigid and non-rigid multi-camera factorization without correspondences between different cameras, such constraints are indeed present since multiple points are usually tracked in a camera: $\mathbf{B}_{i,:}$ and $\mathbf{B}_{k,:}$ correspond to the coordinates of points i and k whereas $\mathbf{C}_{j,:}$ and $\mathbf{C}_{l,:}$ correspond to the x- and y-axes of the same camera. Unfortunately, a large number of points need to be tracked in order to get sufficiently many linear equations which renders

this approach practically less useful and the closed-form solutions presented in this chapter and Chap. 4 are preferable.

iii. **Basis Constraints:** Due to the basis ambuiguity, we can choose $\mathbf{B} = \left[\mathbf{I}_n, \mathbf{D}^T\right]^T$. The orthonormality constraints then look like

$$\mathbf{A}\mathbf{Q}\mathbf{Q}^T\mathbf{A}^T = \begin{bmatrix}\mathbf{I}_n \\ \mathbf{D}\end{bmatrix}\begin{bmatrix}\mathbf{I}_n & \mathbf{D}^T\end{bmatrix} \otimes \mathbf{C}\mathbf{C}^T = \begin{bmatrix}\mathbf{I}_n & \mathbf{D}^T \\ \mathbf{D} & \mathbf{D}\mathbf{D}^T\end{bmatrix} \otimes \mathbf{C}\mathbf{C}^T. \tag{5.35}$$

This specific choice of basis zeroed out many entries in $\mathbf{B}\mathbf{B}^T \otimes \mathbf{C}\mathbf{C}^T$. Especially, some potentially previously unknown entries $[\mathbf{B}_{i,:} \otimes \mathbf{C}_{j,:}]\,[\mathbf{B}_{k,:} \otimes \mathbf{C}_{l,:}]^T = \mathbf{B}_{i,:}\mathbf{B}_{k,:}^T \otimes \mathbf{C}_{j,:}\mathbf{C}_{l,:}^T$ with $i \neq k$ or $j \neq l$ are forced to zero and hence, now there are mixed constraints which resolve the block skew-symmetric matrix ambiguity. Adding these constraints results in a *unique* full rank solution for $\mathbf{Q}\mathbf{Q}^T$: the block-skew symmetric matrix $\mathbf{Y}$ must be zero and the ambiguity $\mathbf{Q}_B\mathbf{Q}_B^T \otimes \mathbf{I}_q$ has been fixed through our choice of basis. Unfortunately, extracting the corrective transformation $\mathbf{Q}$ from $\mathbf{Q}\mathbf{Q}^T$ proves to be difficult: an arbitrary decomposition, like for example the eigendecomposition $\mathbf{Q}\mathbf{Q}^T = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ and setting $\mathbf{Q} = \mathbf{V}\mathbf{\Lambda}^{\frac{1}{2}}$, will destroy the Kronecker-structure in $\mathbf{A}\mathbf{Q} = \mathbf{B} \otimes \mathbf{C}$. Hence, even though the solution provided by the basis constraints is unique, the extraction of the corrective transformation poses difficulties.

iv. **Basis Constraints with Rank-Constraint:** A closed-form solution can be obtained by combining the basis constraints with the rank constraint [XCK04]. We again choose $\mathbf{B} = \left[\mathbf{I}_n, \mathbf{D}^T\right]^T$, but this time only a slice of q consecutive columns are extracted.

Using the same notation as before, we get

$$\mathbf{A}\mathbf{Q}\mathbf{0}_{+r}\mathbf{0}_{+r}^T\mathbf{Q}^T\mathbf{A}^T = \left[\begin{bmatrix}\mathbf{I}_n\\ \mathbf{D}\end{bmatrix}\otimes\mathbf{C}\right]\mathbf{0}_{+r}\mathbf{0}_{+r}^T\left[\begin{bmatrix}\mathbf{I}_n\\ \mathbf{D}\end{bmatrix}\otimes\mathbf{C}\right]^T \tag{5.36}$$

$$= \left[\begin{pmatrix}\mathbf{e}_r\\ \mathbf{D}_{:,r}\end{pmatrix}\otimes\mathbf{C}\right]\left[\begin{pmatrix}\mathbf{e}_r\\ \mathbf{D}_{:,r}\end{pmatrix}\otimes\mathbf{C}\right]^T = \left[\begin{pmatrix}\mathbf{e}_r\\ \mathbf{D}_{:,r}\end{pmatrix}\begin{pmatrix}\mathbf{e}_r\\ \mathbf{D}_{:,r}\end{pmatrix}^T\otimes\mathbf{C}\mathbf{C}^T\right] \tag{5.37}$$

$$= \left[\begin{bmatrix}\mathbf{0}_{r-1\times r-1} & \mathbf{0}_{r-1\times 1} & \mathbf{0}_{r-1\times n-r}\\ \mathbf{0}_{1\times r-1} & 1_{1\times 1} & \mathbf{0}_{1\times n-r}\\ \mathbf{0}_{n-r\times r-1} & \mathbf{0}_{n-r\times 1} & \mathbf{0}_{n-r\times n-r}\\ \begin{bmatrix}\mathbf{0}_{m-n\times r-1} & \mathbf{D}_{:,r} & \mathbf{0}_{m-n\times n-r}\end{bmatrix}\end{bmatrix}\begin{bmatrix}\mathbf{0}_{r-1\times m-n}\\ \mathbf{D}_{:,r}^T\\ \mathbf{0}_{n-r\times m-n}\\ \begin{bmatrix}\mathbf{D}_{:,r}\mathbf{D}_{:,r}^T\end{bmatrix}\end{bmatrix}\right]\otimes\mathbf{C}\mathbf{C}^T, \tag{5.38}$$

where $\mathbf{e}_r$ denotes the r-th canonical basis vector (equal to the zero vector except with a one at index r). In comparison with Eq. (5.35), these low-rank basis constraints drive even more entries to zero. These constraints provide a unique rank-q solution $\mathbf{Q}\mathbf{Q}^T$ from which a slice from the corrective transformation $\mathbf{Q}_r$ can be extracted (with the above mentioned eigendecomposition approach, for example). This then in turn allows to solve for $\mathbf{B}$ and $\mathbf{C}$ in $\mathbf{A}\mathbf{Q}_r = \mathbf{B}_{:,r}\otimes\mathbf{C}$.

5.5 Algorithms

From a practical point of view, the tensor framework also facilitates the development of algorithms. Here, we present a closed-form and an iterative algorithm.

5.5.1 Closed-From Factorization Algorithm

Analogously to the rigid motion setting in Chap. 4, let us consider a multi-camera setup with affine cameras where each camera tracks its own set of feature points: there are no feature point correspondences between different cameras. In order to apply matrix factorizations, the trajectories must be known completely, i.e. from the first to the last frame. Furthermore, a non-degenerate motion according to Sec. 5.3.1 is assumed. In this case, a closed-form factorization algorithm can be derived following along similar lines as in Sec. 4.6. The formulation and

derivations in Sec. 5.3 highlighted a close similarity between the rigid and non-rigid case: a rigid motion corresponds to choosing $d_S = 3$ for the dimensionality of the structure coefficients $\mathbf{s}_n \in \mathbb{R}^{d_S \times 1}$ in Eq. (5.3). It follows that for deformable objects where $d_S > 3$, the algorithm in Sec. 4.6 can be adapted by changing the dimensionality of the motion matrix and structure matrix accordingly. The intuition of the resulting algorithm is that instead of using point correspondences between different cameras, we make use of the motion correspondence: all the cameras observe the same non-rigid deforming object and this low-rank motion correspondence enables the registration of all the cameras in one consistent affine coordinate frame. Self-calibration can be applied to the camera matrices in order to get a representation of the non-rigid deformation in a similarity coordinate frame. According to our knowledge, this is the first closed-form reconstruction algorithm for multiple affine cameras which track feature points on a low-rank non-rigid object without correspondences between different cameras. The existence of such an algorithm is quite surprising since

i) without correspondences between different cameras, feature points cannot be triangulated using the rigidity of the object when observed from multiple cameras at the same point in time.

ii) there is no rigidity constraint between successive frames and hence, a feature point tracked in just one camera can not be directly triangulated by standard multiple-view techniques either.

iii) an independent reconstruction per camera is not possible without further assumptions, e.g. a basis shape assumption.

In the following sections, the algorithmic changes w.r.t. Sec. 4.6 will be explained in more detail. After an initial low-rank factorization, again a three step stratified approach will be used where the first step singles out the camera translations (and thus corresponds to an affine upgrade in standard multiple-view geometry), the second step computes the camera matrices, and the last step computes the structure. Optionally, a fourth step can subsequently be applied to upgrade the resulting reconstruction w.r.t. an affine coordinate frame to a similarity coordinate frame.

Low-Rank Factorization

The initial factorization is based on the flattened data tensor $\mathcal{W}_{(f)}$ along the temporal mode in Eq. (5.8). However, due to the non-existence of correspondences between different cameras, many columns of $\mathcal{W}_{(f)}$ will be unknwon: if point n is not observed in camera k, then the corresponding columns in $\mathcal{W}_{(f)}$ will be missing. Let us denote with $\mathbf{S}_k \in \mathbb{R}^{d_S+1 \times N_k}$ the points tracked by camera k and with $\mathbf{C}_k \in \mathbb{R}^{2\times 4}$ the camera matrix of camera k. Then the known columns of $\mathcal{W}_{(f)}$ can be stacked in a matrix $\mathbf{W}$ which leads to

$$\mathbf{W} = \mathbf{M}\mathcal{S}_{(f)} \left[\Rightarrow_k \mathbf{S}_k \otimes \mathbf{C}_k^T\right] \in \mathbb{R}^{F \times 2\sum_k N_k}, \tag{5.39}$$

which shows that even though no correspondences between the cameras are known, the motion matrix $\mathbf{M}$ is nonetheless completely defined by the given observations. A low-rank factorization of $\mathbf{W} = \hat{\mathbf{M}}\hat{\mathbf{A}}$ into rank-$(3d_S + 4)$ matrices $\hat{\mathbf{M}}$ and $\hat{\mathbf{A}}$ yields the correct column span $\text{span}\left(\hat{\mathbf{M}}\right) = \text{span}\left(\mathbf{M}\right)$ of the motion matrix. An arbitrary low-rank factorization, like e.g. given by a low-rank singular value decomposition, will not comply with the required Kronecker-structure in the second matrix factor as revealed in Eq. (5.8). Hence, a corrective transformation $\mathbf{Q} \in \mathbb{R}^{3d_S+4 \times 3d_S+4}$ is necessary to establish the correct structure in $\mathbf{M} = \hat{\mathbf{M}}\mathbf{Q}$ and in $\mathbf{Q}^{-1}\hat{\mathbf{A}} = \mathcal{S}_{(f)}\left[\Rightarrow_k \mathbf{S}_k \otimes \mathbf{C}_k^T\right]$. As mentioned previously, this corrective transformation will be computed in a stratified way.

Singling out the Camera Translations

Eq. (5.9) shows that the last column of the motion matrix equals the constant one vector. This is enforced by solving the overconstrained system $\hat{\mathbf{M}}\mathbf{q} = \mathbf{1}_{F\times 1}$ in the least-squares sense for $\mathbf{q} \in \mathbb{R}^{3d_S+4}$. Proceeding by defining $\mathbf{Q}_1 = \left[[\mathbf{q}]_\perp, \mathbf{q}\right]$ we get $\tilde{\mathbf{M}} = \hat{\mathbf{M}}\mathbf{Q}_1$ and $\tilde{\mathbf{A}} = \mathbf{Q}_1^{-1}\hat{\mathbf{A}}$. Note that since the last column of $\tilde{\mathbf{M}}$ should no longer get modified, any further corrective transformation should not change the last column of $\mathbf{Q}_1$. This implies that for any further corrective transformation $\mathbf{Q}$, the last column of $\mathbf{Q}$ and $\mathbf{Q}^{-1}$ must equal $\mathbf{e}_{3d_S+4} = (\mathbf{0}_{3d_S+3}, 1)^T$.

Extracting Camera Matrix

Combining the structure of the flattened core tensor $\mathcal{S}_{(f)}$ as shown in Eq. (5.7) with the homogeneous coordinate of the structure in Eq. (5.10)

shows that the last four rows of $\mathcal{S}_{(f)}\left[\mathbf{S}_k \otimes \mathbf{C}_k^T\right]$ equals a replication $\mathbf{1}_{1\times N_K} \otimes \mathbf{C}_k^T$ of the camera matrix $\mathbf{C}^T$. This observation yields an overconstrained linear system for the camera matrices $\mathbf{C}_k$ and the partial corrective transformation $\mathbf{Q}_2 \in \mathbb{R}^{4\times 3d_S+4}$

$$\mathbf{Q}_2\tilde{\mathbf{A}} = \left[\Rightarrow_k \mathbf{1}_{1\times N_k} \otimes \mathbf{C}_k^T\right]. \tag{5.40}$$

Note that due to the observation mentioned at the end of the previous subsection, the last column of $\mathbf{Q}_2$ must equal $(0,0,0,1)^T$. After solving this linear system, the camera matrices $\mathbf{C}_k$ are all known w.r.t. a consistent affine coordinate reference frame. Note that due to the gauge freedoms, the resulting system matrix for $\mathrm{vec}\,(\mathbf{Q}_2)$ and $(\Downarrow_k \mathrm{vec}\,(\mathbf{C}_k))$ will be rank deficient and will have a 3 dimensional nullspace (the camera translation has already been singled out and hence the nullspace is only 3 dimensional rather than 4 dimensional). This provides three different homogeneous solutions and one inhomogeneous solution (corresponding to last row of $\mathbf{Q}_2$) for the four columns of the camera matrices.

Computing the Structure Matrix

Once the camera matrices are known the original bilinear problem $\mathbf{Q}_3\tilde{\mathbf{A}} = \mathcal{S}_{(f)}\left[\Rightarrow_k \mathbf{S}_k \otimes \mathbf{C}_k^T\right]$ reduces to a linear problem in $\mathbf{Q}_3$ and in $\mathbf{S}_k$. This overconstrained system can again be solved in the least-squares sense. Note that due to the gauge freedoms, the resulting system matrix for $\mathrm{vec}\,(\mathbf{Q}_3)$ and $(\Downarrow_k \mathrm{vec}\,(\mathbf{S}_k))$ will be rank deficient and will have a d_S+1 dimensional nullspace. From this nullspace, d_S+1 linearly independent solutions can be extracted corresponding to the d_S+1 coordinates of the points. Applying the inverse of this corrective transformation to the motion matrix leads to the corrected motion matrix $\mathbf{M} = \tilde{\mathbf{M}}\mathbf{Q}_3^{-1}$.

Metric Upgrade

The previous steps give a reconstruction w.r.t. an affine coordinate frame: the camera matrices will not have orthogonal axes as is usually enforced in factorization methods. In Sec. 4.6, a similarity upgrade was computed by enforcing rigid motions in the motion matrix: the first 9 entries of each row of $\mathbf{M}$ must equal a vectorized rotation matrix. In the NRSfM case however, this is no longer the case (see Eq. (5.9)).

We propose to compute a similarity upgrade based on auto-calibration techniques. The plane at infinity is already known, since the camera

matrices are computed w.r.t. an affine reference frame. As mentioned in Sec. 19.5 of [HZ04], the computation of the plane at infitiny is usually the most difficult part of auto-calibration. Using constraints on the internal camera calibration matrices, the affine reconstruction can be upgraded to a similarity reconstruction. We refer to Sec. 19.5 in [HZ04] for details on auto-calibration with known plane at infinity. As an example, assuming square pixels, three cameras are sufficient to compute this upgrade. Note that in the non-rigid case with general low-rank assumption on the trajectories, two cameras can not be sufficient due to the inherent bas-relief ambiguity between two affine cameras (p.356 in [HZ04]).

5.5.2 Iterative Refinement

The previously described algorithm follows several sequential steps and is thus not optimal in the sense that errors in early steps are propagated and maybe even amplified in subsequent steps. However, this algorithm can serve as an initialization for an iterative optimization for

$$\min_{\mathbf{M},\mathbf{S},\mathbf{C}} \frac{1}{2} \left\| \mathbf{H} \odot \left[\mathbf{M}\mathcal{S}_{(f)} \left[\mathbf{S} \otimes \mathbf{C}^T \right] - \mathcal{W}_{(f)} \right] \right\|_F^2 , \tag{5.41}$$

where $\mathbf{H}$ masks the unobserved entries and $\odot$ denotes the Hadamard (elementwise) product. For simplicity, we implemented an alternating least squares (ALS) method. One has to keep in mind however, that the number of unknowns is quite large and the Kronecker-structure in the Jacobians must be used wisely otherwise performance suffers too much. We recall that ALS is known to flatline rather quickly when not properly initialized requiring lots of random multiple restarts (see also the results in Chap. 3). Based on our experiments with random initialization, this is indeed also the case in the above trilinear problem of Eq. (5.41). With the initialization provided by the closed-form algorithm however, ALS converged in very few iteration and we never had to randomly reinitialize ALS.

In the presence of incomplete trajectories, this iterative optimization can obviously also be used, even though the initialization is slightly more tricky since the closed-form algorithm can no longer be applied. As a simplification or initialization, the motion subspace can be fixed to a truncated DCT-basis in the case of smooth motions. For the sake

of completeness, the Jacobians required for ALS will be provided in the next section.

Jacobian Matrices

This section presents the Jacobian matrices required for an Alternating Least Squares (ALS) approach in order to solve the optimization problem in Eq. (5.41). This matrix valued least-squares problem can be formulated in a standard vector-valued least squares problem by vectorizing the residual matrix e.g. columnwise. This choice corresponds to a vectorization of the data tensor $\mathcal{W}$ along all three modes. The unknown entries, for which $\mathbf{H}_{i,j} = 0$, can be discarded from the residual vector $\mathbf{r} \in \mathbb{R}^{2FKN\times 1}$ by a left-multiplication $\mathbb{P}_{\mathbf{H}}\mathbf{r}$ with a row-amputed identity matrix $\mathbb{P}_{\mathbf{H}}$ slicing out the known entries from the residual vector. ALS proceeds by alternately solving a least-squares problem in each variable while holding the remaining ones fixed. Let $\mathbf{J}_{\mathbf{X}}$ denote the Jacobian of the objective function w.r.t. variable $\mathbf{X}$ and $\mathbf{b}_{\mathbf{X}}$ the corresponding right-hand side of the least-squares system. In each substep of ALS, a system of the form $\min_{\mathbf{x}} \|\mathbf{J}_{\mathbf{X}}\mathbf{x} - \mathbf{b}_{\mathbf{X}}\|_2^2$ with $\mathbf{x} = \mathrm{vec}(\mathbf{X})$ must be solved. Such a least-squares problem can for example efficiently be solved with a QR-decomposition of $\mathbf{J}_{\mathbf{X}}$.

In order to compute the Jacobians, let us introduce

$$\mathbf{M} = \left[\tilde{\mathbf{M}}, \mathbf{T}, \mathbf{1}_{F\times 1}\right] \in \mathbb{R}^{F\times 3d_S+4}, \mathbf{S} = \begin{bmatrix} \tilde{\mathbf{S}} \\ \mathbf{1}_{1\times N} \end{bmatrix} \in \mathbb{R}^{d_S+1\times N}, \mathbf{C} = [\mathbf{P}, \mathbf{t}] \in \mathbb{R}^{2K\times 4},$$

where $\tilde{\mathbf{M}} \in \mathbb{R}^{F\times 3d_S}$, $\mathbf{T} \in \mathbb{R}^{F\times 3}$, $\tilde{\mathbf{S}} \in \mathbb{R}^{3d_S\times N}$, $\mathbf{P} \in \mathbb{R}^{2K\times 3}$, and $\mathbf{t} \in \mathbb{R}^{2K\times 1}$. With this notation in place, the objective function can be formulated in a refined way

$$\min_{\mathbf{M},\mathbf{S},\mathbf{C}} \frac{1}{2} \left\| \tilde{\mathbf{M}}\left[\tilde{\mathbf{S}} \otimes \mathbf{P}^T\right] + \mathbf{T}\left[\mathbf{1}_{1\times N} \otimes \mathbf{P}^T\right] + \left[\mathbf{1}_{F\times N} \otimes \mathbf{t}^T\right] - \mathcal{W}_{(f)} \right\|_F^2 . \tag{5.42}$$

Computing Jacobian matrices of matrix valued functions w.r.t. matrix valued variables can be a tricky task. We suggest following the rules presented in [MN99], the most important of those are also summarized

in Sec. 2.2.3. This results in the following Jacobians

$$\mathbf{J}_{\tilde{\mathbf{M}}} = \mathbb{P}_{\mathbf{H}} \left[\tilde{\mathbf{S}}^T \otimes \mathbf{P} \otimes \mathbf{I}_F \right] \tag{5.43}$$

$$\mathbf{J}_{\mathbf{T}} = \mathbb{P}_{\mathbf{H}} \left[\mathbf{1}_{N\times 1} \otimes \mathbf{P} \otimes \mathbf{I}_F \right] \tag{5.44}$$

$$\mathbf{J}_{\tilde{\mathbf{S}}} = \mathbb{P}_{\mathbf{H}} \left[\mathbf{I}_{2KN} \otimes \tilde{\mathbf{M}} \right] \left[\mathbf{I}_N \otimes \mathbf{T}_{2K,d_S} \otimes \mathbf{I}_3 \right] \left[\mathbf{I}_{d_S N} \otimes \operatorname{vec}\left(\mathbf{P}^T\right) \right] \tag{5.45}$$

$$\mathbf{J}_{\mathbf{P}^T} = \mathbb{P}_{\mathbf{H}} \left[\mathbf{I}_{2KN} \otimes \tilde{\mathbf{M}} \right] \left[\mathbf{I}_N \otimes \mathbf{T}_{2K,d_S} \otimes \mathbf{I}_3 \right] \left[\operatorname{vec}\left(\tilde{\mathbf{S}}\right) \otimes \mathbf{I}_{3\cdot 2K} \right] \tag{5.46}$$

$$\mathbf{J}_{\mathbf{t}^T} = \mathbb{P}_{\mathbf{H}} \left[\mathbf{I}_N \otimes \mathbf{T}_{2K,F} \otimes \mathbf{I}_1 \right] \left[\operatorname{vec}\left(\mathbf{1}_{F\times N}\right) \otimes \mathbf{I}_{2K} \right]. \tag{5.47}$$

The matrix $\mathbf{T}_{m,n}$ is the so-called commutator matrix (see Sec. 2.2.2). The formulas above can be slightly simplified which we omit here. Note that the resulting matrices are highly sparse and it is important to use sparse matrices for the computations. The corresponding right-hand sides for the least-squares system are straight-forward to compute and we omit explicit formulas here.

5.5.3 Comparison to [Zah+11]

At this point, a comparison with Zaheer et.al. 's recently proposed algorithm [Zah+11] is suitable. Their work addresses the same setup of multiple affine cameras observing a non-rigid scene. In a nutshell, the algorithm proposed in [Zah+11] reads in our tensor formulation like:

1. Impute missing 2D trajectory entries with a truncated DCT interpolation of the known entries.

2. Compute an orthogonal basis $\mathbf{\Omega}$ for the dominant subspace of the completed trajectories.

3. Factorize $\mathcal{W}_{(k)} \left[\mathbf{I}_N \otimes \mathbf{\Omega} \right]$ in a rank-3 matrix in order to extract the first three columns of the camera matrices.

The first step can be considered as a preprocessing step and is not explained further here. Zaheer et.al. 's algorithm assumes *all* the correspondences between cameras to be known, or if some entries are missing, these entries must be interpolated in the first step. Otherwise, the factorization in the third step can not be performed. The last two steps actually correspond to a partial first iteration of the Higher-Order SVD algorithm [LMV00] where the dominant subspaces of a tensor are alternately exposed by reshaping the tensor in matrix form and

performing PCA to extract the dominant subspaces. This is very similar to ALS with the distinction that the exposed matrices, e.g. the motion matrix $\mathbf{M}$, are forced to have orthogonal columns leading to $\mathbf{\Omega Q} = \mathbf{M}$. The Higher-Order SVD absorbs the factor $\mathbf{Q}$ in the core tensor. A full iteration of the Higher-Order SVD would expose the structure subspace by forming $\mathcal{W}_{(n)}$ after step 3 above before subsequently computing the core tensor given fixed motion, camera, and structure subspaces. [Zah+11] also assumes centered input data, hence the translations are not modeled and the camera matrices are of rank 3. Subtracting the column means in a centering step factors out the plane at infinity and results automatically in a reconstruction w.r.t. an affine coordinate frame. A standard rank 4 factorization including translations would result in a projective reconstruction asking for more complex auto-calibration techniques. In contrast, even though our formulation explicitly models the translations, the resulting reconstruction is still in an affine frame.

Our work is different in many aspects and refines the results reported in [Zah+11] considerably. The derivation of [Zah+11] is missing an important algebraic relation (specifically the algebraic relation between Eq. (2) and (9) in their paper). This can be seen by using our tensor formulation. Specifically, the second step starts by exposing the trajectory subspace by flattening the data tensor along the temporal mode

$$\mathcal{W}_{(f)} = \mathbf{M}\mathcal{S}_{(f)}\left[\mathbf{S} \otimes \mathbf{C}^T\right] = \mathbf{M}\mathcal{S}_{(f)}\left[\mathbf{S} \otimes \mathbf{I}_4\right]\left[\mathbf{I}_N \otimes \mathbf{C}^T\right], \tag{5.48}$$

where the reformulation in the last equation is not strictly necessary but shows the exact equivalence between Eq. (5.48) and Eq. (9) in [Zah+11]. The trajectory subspace can be computed from the column span of $\mathbf{M}$ which is equal to $\operatorname{span}\left(\mathcal{W}_{(f)}\right)$. Let $\mathbf{\Omega} \in \mathbb{R}^{F \times d_M}$ be a matrix with orthonormal columns spanning this subspace. The final step in [Zah+11] is the computation of the camera matrices. In the tensor framework, this is done by flattening the data tensor along the camera mode

$$\mathcal{W}_{(k)} = \mathbf{C}\mathcal{S}_{(k)}\left[\mathbf{S} \otimes \mathbf{M}^T\right], \tag{5.49}$$

where the last equation corresponds to Eq. (2) in [Zah+11]. [Zah+11] then continues by a right-multiplication

$$\mathcal{W}_{(k)}\left[\mathbf{I}_N \otimes \mathbf{\Omega}\right] = \mathbf{C}\mathcal{S}_{(k)}\left[\mathbf{S} \otimes \mathbf{M}^T\mathbf{\Omega}\right] = \mathbf{C}\underbrace{\mathcal{S}_{(k)}\left[\mathbf{S} \otimes \left[\mathbf{I}_{d_S}, \mathbf{0}_{d \times d_S}\right]^T\right]}_{=\mathbf{A} \in \mathbb{R}^{4 \times N d_S}}, \tag{5.50}$$

where in the last equation we have assumed w.l.o.g. that the motion matrix $\mathbf{M}$ equals $[\mathbf{\Omega}, \mathbf{\Omega}_\perp]$ for some $\mathbf{\Omega}_\perp \in \mathbb{R}^{F \times F - d_M}$ (since $\mathbf{M}$ can always be orthogonalized by absorbing the required factor in $\mathcal{S}_{(f)}$). Hence, this step amounts to projecting the motion onto a low-rank subspace of dimension d_M spanned by $\mathbf{\Omega}$. For exact low-rank and noise-free data (i.e. then $\mathbf{\Omega}_\perp = \mathbf{0}$), this projection step is not necessary and the same result is given by factorizing the data according to Eq. (5.49). In such a situation, [Zah+11] boils down to i) fitting known entries in the least-squares sense with $\mathbf{\Omega}$ in order to estimate missing entries and ii) factorizing $\mathcal{W}_{(k)}$.

Note that if there are no point correspondences between different cameras, then the pattern of known entries in $\mathcal{W}_{(k)}$ follows a block-diagonal matrix (see also Fig. 4.2) and Eq. (5.49) can not solely be used to align all the cameras in a consistent coordinate frame. That is the reason why [Zah+11] requires all feature point correspondences to be known. Furthermore by comparing Eq. (5.49) and Eq. (5.48) with eq. (2) and eq. (9) in [Zah+11], we see that [Zah+11] misses the exact algebraic interplay between $\mathbf{C}$, $\mathbf{M}$, and $\mathbf{S}$ (resp. $\mathbf{R}$, $\mathbf{\Theta}$, and $\mathbf{A}$ in their formulation) which provides valuable constraints. As described in the previous sections, by using these constraints it is possible to compute a valid reconstruction in closed-form even if there are no correspondences between different cameras. This is obviously a considerably stronger result. Of course, if correspondences between cameras are available, then our algorithms derived from the tensor formulation can and should make use of them.

Furthermore, our presentation assumes no centered data which corrects for the fact that the translational part can *not* be eliminated by subtracting the column mean in the presence of missing data. This has been overseen in [Zah+11] and hence the derivation and their algorithm only hold if *all* the data is known in the centering step and entries are only deleted afterward. Our algorithms are therefore applicable whenever [Zah+11] is but cover additional settings as well (e.g. missing correspondences between cameras and correct handling of translation in case of incomplete trajectories).

5.6 Results

Since the unified tensor formulation is one of the main contributions, practical as well as theoretical results of this formulation will be presented in the next sections.

5.6.1 Theoretical Result: Degenerate Low-Rank Non-Rigid Motion

As a theoretical result of the proposed tensor formulation, degenerate motions will be investigated more closely in this section. The previous derivations implicitly assumed that if $\operatorname{rank}\left(\left[\Downarrow_f \mathbf{R}_f \mathbf{Y}_f\right] \in \mathbb{R}^{3F \times d_S}\right) = d_S$ then $r = \operatorname{rank}\left(\left[\Downarrow_f \operatorname{vec}\left(\mathbf{R}_f \mathbf{Y}_f\right)^T\right] \in \mathbb{R}^{F \times 3d_S}\right) = 3d_S$. For general matrices, the algebraic rank of such a reshaped matrix indeed fulfills this equality. However, in special cases or in practical cases where the rank is estimated based on a robust rank estimator it might be that $\left[\Downarrow_f \operatorname{vec}\left(\mathbf{R}_f \mathbf{Y}_f\right)^T\right]$ is again redundant and can be factorized even further into two lower-rank matrices $\left[\Downarrow_f \operatorname{vec}\left(\mathbf{R}_f \mathbf{Y}_f\right)^T\right] = \tilde{\mathbf{M}}\mathbf{Q}$ and we get $r < 3d_S$. Such a rank reduction happens for example if the trajectories in x-direction are highly correlated with the trajectories in y- and z-direction which is actually not that uncommon. In such cases, the matrix $\mathbf{Q} = \left[\mathbf{Q}_1, \ldots, \mathbf{Q}_{d_S}\right] \in \mathbb{R}^{r \times 3d_S}$ can be absorbed by $\mathcal{S}_{(f)}$

$$\begin{aligned}\mathbf{M}\mathcal{S}_{(f)} &= \left[\tilde{\mathbf{M}}\mathbf{Q}, \left[\Downarrow_f \mathbf{t}_f^T\right], \mathbf{1}_{F\times 1}\right]\mathcal{S}_{(f)} \\ &= \left[\tilde{\mathbf{M}}, \left[\Downarrow_f \mathbf{t}_f^T\right], \mathbf{1}_{F\times 1}\right]\underbrace{\begin{bmatrix}\left[\Rightarrow_{b=1}^{d_S}\left[\mathbf{Q}_b, \mathbf{0}_{r\times 1}\right]\right] & \mathbf{0}_{r\times 4} \\ \mathbf{0}_{4\times 4d_S} & \mathbf{I}_4\end{bmatrix}}_{=\tilde{\mathcal{S}}_{(f)} \in \mathbb{R}^{r+4\times 4d_S+4}},\end{aligned} \tag{5.51}$$

which increases the complexity of the factorization since the core tensor is then also (partially) unknown. This complexity on the other hand might lead to greater robustness since the motion subspace can be of any rank r not necessarily equal to $3d_S$. Fig. 5.1 exemplifies this observation further.

Aware of these degenerate situations, we leave it as future work how to deal with such degenerate motions algorithmically. In the next section, we will assume general non-degenerate motions such as presented in Sec. 5.3.1.

5.6.2 Experiments

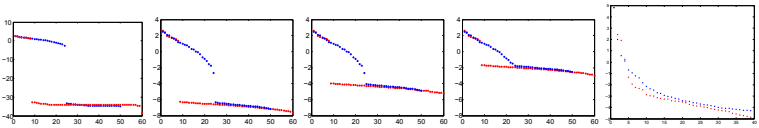
The CMU facial motion capture sequence, which has also been used in [THB08], is used for practical evaluation because this is a widely used dataset for NRSfM. The facial mocap sequence is projected into $K = 3$ affine cameras spaced 45 degrees apart from each other where the middle one is facing the face directly from the front and is equal to the camera used by [THB08]. Each camera observed all of the 40 points, however no correspondences between the different cameras have been enforced. The closed-form algorithm has been used for initializing 20 iterations of ALS. The results are summarized in Fig. 5.2. An interesting observation we made was that the reprojection error directly after the closed-form algorithm was sometimes quite high (up to roughly 10% of the camera resolution). However, after one single iteration of ALS, the error often dropped below 1%. We explain this observation by referring to Fig. 5.1: the closed-form algorithm is based on several sequential steps. The initial step is a low-rank factorization of the data tensor flattened along the temporal mode $\mathcal{W}_{(f)}$. This corresponds to the blue points in these figures. Since the low-rank model holds only approximatively, a considerable amount of noise might mix in which then propagates to the subsequent steps. This sometimes leads to inaccurate structure estimates $\mathbf{S}$, the cameras and the majority of the motion matrix were usually estimated quite accurately. A few ALS iterations can correct for this error propagation, as shown in the results.

Similarly to the rigid case in Chap. 4, the algorithm can also handle the extreme minimal case where a camera only tracks one single feature point which is not in correspondence with any of the other feature points tracked by the other cameras. As long as all the cameras together provide sufficient data to estimate the motion space $\mathbf{M}$, one trajectory is sufficient to reconstruct the camera pose and the 3D motion of this trajectory. In order to validate this, a single point was selected from camera 3 and all the remaining ones of camera 3 were omitted in the algorithm. With $d_S = 10$, for an almost rigidly moving point an overall relative 3D error of 0.021 resulted and restricting to only the selected point, the relative 3D error of this trajectory was 0.044. For a largely non-rigidly moving point on the mouth, the overall relative 3D error was 0.025 and the relative 3D error of the trajectory was 0.061. We refer to Fig. 5.2 for the reprojection error of this non-rigid point in camera 3.

5.7 Conclusion and Future Work

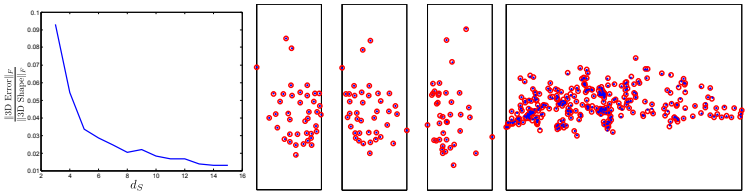
This chapter presented a unified formulation based on tensor algebra for factorization-based NRSfM approaches thereby expressing monocular, binocular, and camera network approaches for NRSfM in a common framework. We have shown that the natural way to capture such multilinear interactions between a motion subspace, camera subspace, and a structure subspace is a Tucker-tensor decomposition. This new understanding of how the subspaces interact with each other enabled us to come up with a closed-form and an iterative algorithm which can handle the case where no feature point correspondences between different cameras are available. Experiments validated the presented algorithms on motion capture data.

As future work, we plan to address the dimensionality selection problem for the motion and structure subspaces. Sec. 5.6.1 already presented some first theoretical insights but we are currently lacking an algorithm to handle these situations in a principled manner. A first idea might be to impose trace norm regularization terms on the reshaped tensors $\mathcal{W}_{(f)}$ and $\mathcal{W}_{(n)}$ thereby automatically favoring low-dimensional motion and structure subspaces. Another thrust for future research is the application of the low-rank constraints amongst different cameras for optical flow: the flow field of each camera is constrained by one joint motion subspace. From an optimization point of view, the robust handling of outliers in the measurement remains an open issue.



(a) No Noise (b) 0.01% Noise (c) 0.1% Noise (d) 1% Noise (e) Facial Mocap

Figure 5.1: This figure shows the problem of implicitly choosing the dimensionality of the motion too high. For Fig. 5.1a-Fig. 5.1d a sequence of a non-rigidly deforming structure $\mathbf{x}_{f,n}$ with $d_S = 7$ was generated. The motion actually corresponds to two realistically independently moving rigid objects and hence the overall structure is perceived as non-rigid. The logarithms of the resulting singular values of $\mathcal{X}_{(f)} = \left[\Downarrow_f \Rightarrow_n \mathbf{x}_{f,n}^T\right] = \mathbf{M}\mathcal{S}_{(f)}\left[\mathbf{S} \otimes [\mathbf{I}_3, \mathbf{0}_{3\times 1}]\right]^T \in \mathbb{R}^{F\times 3N}$ and $\mathcal{X}_{(n)} = [\Downarrow_f \Rightarrow_n \mathbf{x}_{f,n}]^T = \mathbf{S}\mathcal{S}_{(n)}\left[\mathbf{M} \otimes [\mathbf{I}_3, \mathbf{0}_{3\times 1}]\right]^T \in \mathbb{R}^{N\times 3F}$ are visualized in blue resp. red. Fig. 5.1a with no noise shows the true underlying ranks $\operatorname{rank}\left(\mathcal{X}_{(n)}\right) = d_S + 1 = 8$ and $\operatorname{rank}\left(\mathcal{X}_{(f)}\right) = 3d_S + 3 = 24$. It can be clearly seen that only a minor increase of Gaussian distributed noise already corrupts the rank-24 approximation of $\mathcal{X}_{(f)}$ considerably. However, the numerical rank of $\mathcal{X}_{(n)}$ is considerably more robust w.r.t. noise. This is a clear indication that a tensor-formulation taking the multi-rank of the tensor into account provides increased robustness. Fig. 5.1e shows the same analysis for the CMU facial motion capture sequence used in [THB08].



(a) Relative 3D reconstruction error

(b) Reprojection error at frame f=132

(c) Reprojection error in single-point experiment

Figure 5.2: Results of facial mocap sequence: Fig. 5.2a shows the relative 3D error of the reconstructed 3D trajectories as a function of the structure dimensionality. Compared to [THB08], our approach achieves only slightly better results: an indicator that both Torresani et.al. 's and our approach succeed in an accurate reconstruction, even though using completely different presuppositions ([THB08] works in a monocular blend-shape setting with strong smoothness prior on shape and motion whereas our approach uses multiple-cameras but imposes no priors besides a low-rank assumption.). Fig. 5.2b shows the ground truth in blue and reprojections in red for the three cameras at frame $f = 132$ for $d_S = 15$. Fig. 5.2c shows the reprojection error in camera 3 of the single-point reconstruction experiment. The reprojection error for this camera is roughly 3 pixels for an image of resolution 1200×1200.

6 Convex Relaxations

Being faced with a difficult optimization problem, there are basically two alternatives. Either the original problem is addressed with sophisticated optimization methods such as non-linear local optimization methods with multiple random restarts and randomized algorithms (e.g. simulated annealing, RANSAC) or the problem is replaced with a simpler surrogate. Methods belonging to the former class have been presented in Chap. 3 and will be discussed in Chap. 7. This chapter is concerned with convex relaxations of the original non-convex problem. Convex problems are attractive since they guarantee that a global optima can be found. Convex optimization and relaxations are a well-established field, Rockafellar's work [Roc70] representing one of the early cornerstones. We refer to his or Boyd and Vandenberghe's more recent book [BV04] for a general introduction to convex optimization.

Unfortunately, a global solution to the convex relaxation of a problem is not necessarily also a global solution to the original problem. There are pros- and cons to either approach (non-linear local optimization vs. convex relaxations) and we do not claim one to be superior to the other for general problems. However, there exist problem instances, where a solution of the convex relaxation

- is guaranteed to yield also a global solution to the original problem (e.g. max-flow problems).
- is guaranteed to be ϵ-optimal, i.e. not worse than $1 + \epsilon$ w.r.t. a true global optimum of the unrelaxed problem [GW95].
- is also a global solution of the original problem with very high probability (e.g. compressive sensing).

The material presented in this chapter are motivated by results in compressive sensing. For specific problems such as matrix completion problems the convex relaxation has been shown to work surprisingly well [RFP10; CT10], in theory as well as in practice. In these problems, the trace norm relaxation of the rank acts as a convex proxy for low

rank constraints. Recent results [Gro11; Neg+10] provide strong guarantees under which conditions (originally called the *restricted isometry property*) the solutions of the convex relaxation with trace norm are equivalent to the original non-convex problem. Surprisingly, several random measurement matrices (e.g. Gaussian ensembles) fulfill this restricted isometry condition with very high probability.

Motivated by problems in geometric computer vision, in this chapter, we introduce a structured trace norm, called the generalized trace norm. This enables the introduction of a non-uniform penalization of row- and column-subspaces of a matrix. In geometric computer vision, the structure from motion (SfM) problem can be formulated as a optimization problem with a rank constraint, as seen in the preceding chapters. In recent work [OO09; DLH10], the trace-norm relaxation has been applied to the SfM problem. However, SfM problems often exhibit a certain structure, for example a smooth camera path. Unfortunately, the standard trace norm relaxation can not make use of this additional structure. This observation motivates the main contribution of this chapter which is based on our publication [AZP11]: The generalized trace norm allows to encode prior knowledge about a specific problem into a convex regularization term which enforces a low rank solution while at the same time taking the problem structure into account.

While deriving the generalized trace norm and stating its different formulations, this chapter draws interesting connections to other fields, most importantly to the field of compressive sensing. Even though the generalized trace norm is a very general concept with a wide area of potential applications we are ultimately interested in applying it to SfM problems. Therefore, this chapter also presents an efficient algorithm to optimize the resulting generalized trace norm regularized optimization problems.

6.1 Trace-Norm

The trace-norm of a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is defined as the sum of the singular values of the matrix $\mathbf{X}$

$$\|\mathbf{X}\|_* = \sum_{i=1}^{\min(m,n)} \sigma_i(\mathbf{X}),$$

where $\sigma_i(\mathbf{X})$ denotes the i^{th} singular value of $\mathbf{X}$. What the $L1$-norm is for vectors, the trace norm is for matrices. The trace norm is also known as the $\min(m,n)$-Ky-Fan norm or the nuclear norm [HJ94]. We will stick with the name trace norm for reasons which will become clear shortly. The trace norm is proportional to the the tightest convex lower bound of the rank function over the set of matrices with spectral norm smaller than β, i.e. the convex envelope of the rank function $\text{rank}(\mathbf{X})$ over the set $\{\mathbf{X} \in \mathbb{R}^{m\times n} | \, \|\mathbf{X}\|_2 \leq \beta\}$ is $\frac{1}{\beta}\|\mathbf{X}\|_*$. The trace norm of a matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$ can be computed with a Semi-Definite Programming (SDP) formulation (e.g. see [RFP10])

$$\|\mathbf{X}\|_* = \frac{1}{2} \min_{\mathbf{X}_1,\mathbf{X}_2} \left(\text{trace}(\mathbf{X}_1) + \text{trace}(\mathbf{X}_2)\right) \tag{6.1}$$

$$\text{s.t.} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X} \\ \mathbf{X}^T & \mathbf{X}_2 \end{bmatrix} \succeq \mathbf{0}_{m+n\times m+n}, \tag{6.2}$$

where $\succeq$ denotes a generalized inequality constraint on the cone of positive semidefinite matrices $\mathbb{S}_+^{m+n}$. SDP problems are convex. As for example stated in [BMP08] there are three other formulations of the trace norm based on a factorization of $\mathbf{X}$ into two matrices $\mathbf{U}$ and $\mathbf{V}$

$$\begin{aligned} \|\mathbf{X}\|_* &= \min_{\mathbf{X}=\mathbf{U}\mathbf{V}^T} \|\mathbf{U}\|_F \|\mathbf{V}\|_F = \min_{\mathbf{X}=\mathbf{U}\mathbf{V}^T} \sum_i \|\mathbf{U}_{:,i}\|_2 \|\mathbf{V}_{:,i}\|_2 \\ &= \frac{1}{2} \min_{\mathbf{X}=\mathbf{U}\mathbf{V}^T} \left(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2\right). \end{aligned} \tag{6.3}$$

6.2 Generalized Trace-Norm

The starting point for the generalized trace norm is the following: the objective of the SDP-formulation of the trace norm in Eq. (6.1) actually corresponds to an inner product between an identity matrix an the matrix appearing in Eq. (6.2)

$$\|\mathbf{X}\|_* = \frac{1}{2} \min_{\mathbf{X}_1,\mathbf{X}_2} \langle \mathbf{I}_{m+n}, \begin{bmatrix} \mathbf{X}_1 & \mathbf{X} \\ \mathbf{X}^T & \mathbf{X}_2 \end{bmatrix} \rangle \tag{6.4}$$

$$\text{s.t.} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X} \\ \mathbf{X}^T & \mathbf{X}_2 \end{bmatrix} \succeq \mathbf{0}_{m+n\times m+n}. \tag{6.5}$$

Here $\langle \mathbf{A}, \mathbf{B} \rangle = \text{trace}\left(\mathbf{A}^T \mathbf{B}\right)$ denotes the Frobenius inner product. We define for any matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ the generalized trace norm as

$$\|\mathbf{X}\|_{*\mathbf{D}} = \frac{1}{2} \min_{\mathbf{X}_1, \mathbf{X}_2} \langle \mathbf{D}, \begin{bmatrix} \mathbf{X}_1 & \mathbf{X} \\ \mathbf{X}^T & \mathbf{X}_2 \end{bmatrix} \rangle \tag{6.6}$$

$$= \frac{1}{2} \min_{\mathbf{X}_1, \mathbf{X}_2} \text{trace}\left(\mathbf{D}^T \begin{bmatrix} \mathbf{X}_1 & \mathbf{X} \\ \mathbf{X}^T & \mathbf{X}_2 \end{bmatrix} \right) \tag{6.7}$$

$$\text{s.t. } \begin{bmatrix} \mathbf{X}_1 & \mathbf{X} \\ \mathbf{X}^T & \mathbf{X}_2 \end{bmatrix} \succeq \mathbf{0}_{m+n \times m+n}, \tag{6.8}$$

where $\mathbf{D} \in \mathbb{S}_+^{m+n}$ is a positive definite matrix. Hence, compared to the standard trace norm, the generalized trace norm replaces the identity with a positive semi-definite matrix and thereby introduces a non-uniform weighting, similar to a weighted $L1$-vector norm. It remains to be shown, that this definition fulfills the axioms of a norm over the vector space of $\mathbb{R}^{m \times n}$.

6.2.1 Proof: $\|\mathbf{X}\|_{*\mathbf{D}}$ is a Norm

If the matrix $\mathbf{D}$ is not block-diagonal, then the formulation in Eq. (6.6) is not a norm[1]. For the case of positive-definite block-diagonal

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c \end{bmatrix} \in \mathbb{S}_+^{m+n}$$

blocks $\mathbf{D}_r$ and $\mathbf{D}_c$ of sizes $m \times m$ and $n \times n$, respectively the proof starts with the Eigenvalue-decomposition of $\mathbf{D}_r = \mathbf{V}_r \mathbf{\Lambda}_r \mathbf{V}_r^T$ and $\mathbf{D}_c = \mathbf{V}_c \mathbf{\Lambda}_c \mathbf{V}_c^T$. Note that since $\mathbf{D} \in \mathbb{S}_+^{m+n}$ the diagonal blocks must also belong to the positive semi-definite cones $\mathbf{D}_r \in \mathbb{S}_+^m$ and $\mathbf{D}_c \in \mathbb{S}_+^n$. These eigenvalue decompositions can be assembled into an eigenvalue decomposition for $\mathbf{D} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T$ with

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_c \end{bmatrix} \text{ and } \mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_c \end{bmatrix}. \tag{6.9}$$

For later reference we introduce

$$\mathbf{\Lambda}^{\frac{1}{2}} \mathbf{V}^T = \begin{bmatrix} \mathbf{\Lambda}_r^{\frac{1}{2}} \mathbf{V}_r^T & \mathbf{0} \\ \mathbf{0} & \mathbf{\Lambda}_c^{\frac{1}{2}} \mathbf{V}_c^T \end{bmatrix} = \begin{bmatrix} \mathbf{C}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_c \end{bmatrix}.$$

[1] This can be seen by a numerical experiment: choosing a random matrix $\mathbf{A}$, then for a norm it should hold $\|\mathbf{A}\| = \|-\mathbf{A}\|$, however this is not true for random positive definite $\mathbf{D}$.

Then due to the cyclic property of the trace we get

$$\begin{aligned}\|\mathbf{X}\|_{*\mathbf{D}} &= \frac{1}{2}\min_{\mathbf{X}_1,\mathbf{X}_2} \operatorname{trace}\left(\mathbf{D}^T\begin{bmatrix}\mathbf{X}_1 & \mathbf{X}\\ \mathbf{X}^T & \mathbf{X}_2\end{bmatrix}\right)\\ &= \frac{1}{2}\min_{\mathbf{X}_1,\mathbf{X}_2} \operatorname{trace}\left(\mathbf{\Lambda}^{\frac{1}{2}}\mathbf{V}^T\begin{bmatrix}\mathbf{X}_1 & \mathbf{X}\\ \mathbf{X}^T & \mathbf{X}_2\end{bmatrix}\mathbf{V}\mathbf{\Lambda}^{\frac{1}{2}}\right)\\ &= \frac{1}{2}\min_{\mathbf{X}_1,\mathbf{X}_2} \operatorname{trace}\left(\begin{bmatrix}\mathbf{C}_r\mathbf{X}_1\mathbf{C}_r^T & \mathbf{C}_r\mathbf{X}\mathbf{C}_c^T\\ \mathbf{C}_c\mathbf{X}^T\mathbf{C}_r^T & \mathbf{C}_c\mathbf{X}_2\mathbf{C}_c^T\end{bmatrix}\right).\end{aligned}$$

Since $\mathbf{C}_r \in \mathbb{R}^{m\times m}$ and $\mathbf{C}_c \in \mathbb{R}^{n\times n}$ are both regular matrices, we can absorb them and change the variables of the minimization to $\mathbf{Y}_1 = \mathbf{C}_r\mathbf{X}_1\mathbf{C}_r^T$ and $\mathbf{Y}_2 = \mathbf{C}_c\mathbf{X}_2\mathbf{C}_c^T$ and get an equivalent problem which looks like

$$\begin{aligned}\|\mathbf{X}\|_{*\mathbf{D}} &= \frac{1}{2}\min_{\mathbf{Y}_1,\mathbf{Y}_2} \operatorname{trace}\left(\begin{bmatrix}\mathbf{Y}_1 & \mathbf{C}_r\mathbf{X}\mathbf{C}_c^T\\ \mathbf{C}_c\mathbf{X}^T\mathbf{C}_r^T & \mathbf{Y}_2\end{bmatrix}\right)\\ &= \left\|\mathbf{C}_r^T\mathbf{X}\mathbf{C}_c\right\|_*. \end{aligned} \tag{6.10}$$

The generalized trace-norm $\|\mathbf{X}\|_{*\mathbf{D}}$ with a block-diagonal matrix $\mathbf{D}$ reduces to the standard trace norm $\left\|\mathbf{C}_r\mathbf{X}\mathbf{C}_c^T\right\|_*$ of the matrix $\mathbf{C}_r\mathbf{X}\mathbf{C}_c^T$. Equipped with this insight, it is straight-forward to verify that the three axioms for a norm indeed hold for our definition of the generalized trace-norm. In the following we thus restrict ourselves to such block-diagonal choices.

This representation allows a first intuitive interpretation of the generalized trace norm. The row and column spaces of the matrix $\mathbf{X}$ are projected onto the eigenspaces of $\mathbf{D}_r$ and $\mathbf{D}_c$ and scaled according to the square roots of the eigenvalues of $\mathbf{D}_r$ and $\mathbf{D}_c$. By designing the norm on purpose in this way, the generalized trace-norm is *not* a unitarily invariant norm and hence does not reduce to a symmetric gauge function [HJ94] applied to the singular values of $\mathbf{X}$. The generalized trace norm can rather be thought of as the analog for matrices of what the weighted $L1$-vector norm is for vectors.

6.2.2 Factorized Formulation

The generalized trace-norm also allows a factorized interpretation instead of the SDP interpretation given first. Indeed, the SDP formulation

in Eq. (6.6) is equivalent to the factorized interpretation

$$\|\mathbf{X}\|_{*\mathbf{D}} = \min_{\mathbf{X}=\mathbf{U}\mathbf{V}^T} \frac{1}{2} \operatorname{trace}\left(\begin{bmatrix}\mathbf{U}^T & \mathbf{V}^T\end{bmatrix}\mathbf{D}\begin{bmatrix}\mathbf{U}\\ \mathbf{V}\end{bmatrix}\right). \tag{6.11}$$

If $\mathbf{D} = \mathbf{I}_{m+n}$, then Eq. (6.11) reduces to Eq. (6.3), i.e. a sum over the squared Frobenius norms of the two factors $\mathbf{U}$ and $\mathbf{V}$. The generalized trace norm thus replaces the standard inner product in the space of $m+n \times m+n$ matrices by the weighted inner product $\langle ., . \rangle_{\mathbf{D}}$. If the matrix $\mathbf{D}$ is chosen block-diagonal with blocks $\mathbf{D}_r \in \mathbb{S}_+^m$ and $\mathbf{D}_c \in \mathbb{S}_+^n$, then the generalized trace norm reduces to

$$\|\mathbf{X}\|_{*\mathbf{D}} = \min_{\mathbf{U},\mathbf{V}} \frac{1}{2} \left(\langle \mathbf{U}, \mathbf{U} \rangle_{\mathbf{D}_r} + \langle \mathbf{V}, \mathbf{V} \rangle_{\mathbf{D}_c}\right) \tag{6.12}$$

This representation shows that the generalized trace-norm with block-diagonal matrix $\mathbf{D}$ has an interpretation as working in two inner-product spaces with inner products $\langle \mathbf{a}, \mathbf{b} \rangle_{\mathbf{D}_r} = \mathbf{a}^T \mathbf{D}_r \mathbf{b}$ and $\langle \mathbf{c}, \mathbf{d} \rangle_{\mathbf{D}_c} = \mathbf{c}^T \mathbf{D}_c \mathbf{d}$ in the range and domain of the linear mapping induced by $\mathbf{X}$, respectively. From an algorithmic point of view, such a representation is useful for stochastic gradient based optimization techniques which represent the unknown matrix explicitly in a low-rank factorized form.

6.2.3 Relationship to Previous Work

According to our knowledge, concepts related to the generalized trace norm have appeared only sparsely in previous work. For example, the consistency of the trace norm regularization with an L2-loss has been investigated by Bach in [Bac08]. That paper however addressed a different question than we do. Specifically, it addresses the question of how to balance a trace-norm regularization term and a L2-loss term between measurements and an unknown matrix when the number of measurements goes to infinity (the size of the matrix stays constant). Hence, in contrast to the situation we are interested in, there are many more measurements available than linear degrees of freedom. Bach also showed in his paper that a least squares fit solely based on the L2-loss term in order to derive weighting factors can be used to derive stronger consistency results. To the best of our knowledge, this is the only paper where a trace-norm regularization term of the form of Eq. (6.10) has appeared. However, in strong contrast to our contribution, this

weighting scheme has not been used as a general means to include prior knowledge about the specific problem at hand but rather as a means to prove stronger consistency results in the setting when the number of measurements approaches infinity. The decomposition norms introduced in [BMP08] are also related to the generalized trace norm. The authors define a regularization term w.r.t. any norm of the column vectors $\mathbf{U}_{:,i}$ and $\mathbf{V}_{j,:}$ of the factorization $\mathbf{X} = \mathbf{U}\mathbf{V}^T$. However the case of arbitrary norms induced by an inner product $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{D}} = \mathbf{x}^T \mathbf{D} \mathbf{y}$ has not been investigated, the authors have rather focused their attention to the case of choosing the L2 norm in one vector space and the L1-norm in the other. A recent paper about matrix completion [SS10] generalized the trace norm by introducing for each of the terms $\|\mathbf{U}_{i,:}\|_F^2$ and $\|\mathbf{V}_{j,:}\|_F^2$ in the factorized trace-norm formulation of Eq. (6.3) an individual scaling. This actually corresponds to a diagonal choice for $\mathbf{D}$. Another distinction of our work is that the way in which the generalized trace norm is derived and presented, allows to draw interesting connections to related work such as matrix completion and compressive sensing.

6.3 Trace-Norm Regularized M-Estimators

Let us assume we are given the measurements $\mathbf{z} \in \mathbb{R}^{m \times 1}$ and we are looking for a model $\mathbf{W} \in \mathbb{R}^{p \times q}$ with $m << pq$ which explains the measurements as accurately as possible. The accuracy is measured with a data cost function $\mathcal{L}(\mathbf{W}, \mathbf{z})$. The situation $m << pq$ means that there are far fewer measurements than actual degrees of freedom in our model. Regularized M-estimators add a regularization term $\|\mathbf{W}\|$ (usually a norm) in order to handle this situation

$$\mathbf{W}_* \in \arg\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}, \mathbf{z}) + \mu \|\mathbf{W}\| . \tag{6.13}$$

For concreteness, the data cost function is assumed to be a function of $\mathcal{A}(\mathbf{W}) - \mathbf{z}$ where $\mathcal{A} \in \mathbb{R}^{m \times pq}$ is a linear operator relating the matrix valued unknown $\mathbf{W}$ to the measurements $\mathbf{z}$. Recent work has analyzed the situation when the regularization is chosen to be the trace norm $\|\mathbf{W}\|_*$ [Neg+10; RFP10]. In this chapter we are interested in the case where the regularizer corresponds to the generalized trace norm

$$\mathbf{W}_* \in \arg\min_{\mathbf{W}} \mathcal{L}(\mathcal{A}(\mathbf{W}) - \mathbf{z}) + \mu \|\mathbf{W}\|_{*\mathbf{D}} . \tag{6.14}$$

The immediate question is: what is the influence of the block-diagonal matrix $\mathbf{D}$ and how should it be chosen? In order to answer these questions we appeal to a Bayesian interpretation of the generalized trace norm.

6.3.1 Bayesian Interpretation

The representation in Eq. (6.12) is related to a generalized version of the Bayesian probabilistic matrix factorization [SM08]. There, a Bayesian hierarchical model is introduced which represents the unknown matrix $\mathbf{W} = \mathbf{U}\mathbf{V}^T$ in a factorized form. Zero-mean spherical Gaussian priors are put on the unknown factors $\mathbf{U}$ and $\mathbf{V}$. It can be shown that the negative log-posterior distribution exactly corresponds to an M-estimator with a L2-loss function and a standard trace norm regularizer. Instead of putting a $\mathcal{N}(0,1)$ iid. prior on each element of the matrix factors $\mathbf{U}$ and $\mathbf{V}$, we allow for correlations between the entries. Thus the matrix $\mathbf{D}$ can be thought of as a precision matrix for a Gaussian prior on the space of $m+n \times m+n$ matrices. If the matrix $\mathbf{D}$ is chosen block-diagonal, then the row and column subspaces are uncorrelated and hence due to the Gaussianity independent.

This representation is very flexible, as it allows to build in prior knowledge about the low rank matrix. If we have prior knowledge about the row- and columns subspaces of the unknown matrix then this prior knowledge can be used to define the block-diagonal weight matrices $\mathbf{D}_r$ and $\mathbf{D}_c$. Since $\mathbf{D}_r^{-1}$ and $\mathbf{D}_c^{-1}$ can be thought of as covariance matrices the generalized trace norm allows to penalize variations along certain directions less than along other directions. As we will see in the experimental section, we can for example impose smooth subspaces while at the same time favour low-rank matrices. In contrast to the standard trace norm which penalizes *all* the directions in the vector space of matrices $\{\mathbf{X} \in \mathbb{R}^{m \times n}\}$ uniformly, the generalized trace norm enables non-uniform regularization. It can be expected that with this non-isotropic penalization taking prior knowledge about the problem at hand into account the convex relaxation of the rank function will be stronger.

In summary, the generalized trace norm allows to combine more flexible subspace priors with a low-rank prior. The formulation still results in a convex problem which can be solved efficiently and globally optimal by any method of choice.

6.3.2 Change of Variable

If the weight matrices $\mathbf{D}_r$ and $\mathbf{D}_c$ are non-singular then the generalized trace norm is a true norm (otherwise if at least one of them is singular the generalized trace norm is only a semi-norm). In this case, the problem in Eq. (6.14) can be rewritten in the following way. Due to Eq. (6.10) we know that $\|\mathbf{W}\|_{*\mathbf{D}} = \left\|\mathbf{C}_r\mathbf{W}\mathbf{C}_c^T\right\|_*$. Let us introduce the new variable $\tilde{\mathbf{W}} = \mathbf{C}_r\mathbf{W}\mathbf{C}_c^T$ and reparameterize Eq. (6.14) w.r.t. this variable

$$\tilde{\mathbf{W}}_* \in \arg\min_{\tilde{\mathbf{W}}} \mathcal{L}(\mathcal{A}(\mathbf{C}_r^{-1}\tilde{\mathbf{W}}\mathbf{C}_c^{-T}) - \mathbf{z}) + \mu\left\|\tilde{\mathbf{W}}\right\|_*. \tag{6.15}$$

The variable we are actually interested in can be recovered by computing $\mathbf{W} = \mathbf{C}_r^{-1}\tilde{\mathbf{W}}_*\mathbf{C}_c^{-T}$. This problem formulation replaced the generalized trace norm with an ordinary trace-norm regularization. However, the data term now depends on the matrices $\mathbf{C}_r$ and $\mathbf{C}_c$. The formulation in Eq. (6.15) is equivalent to Eq. (6.14). However, the reformulation has algorithmic consequences as we will see shortly. This insight of the equivalence between the generalized trace-norm and the standard trace-norm with a reweighted data term is quite interesting in its own right. The matrix representation of the linear operator can absorb the weight matrices $\tilde{\mathcal{A}} = \mathcal{A}\left[\mathbf{C}_c^{-1} \otimes \mathbf{C}_r^{-1}\right]$ and hence we are left with the data cost $\mathcal{L}(\tilde{\mathcal{A}}(\tilde{\mathbf{W}}) - \mathbf{z})$. Recent results in compressive sensing [Gro11; Neg+10] are based on conditions which the linear operator appearing in the trace-norm regularized M-estimator framework has to meet in order to be able to recover the true low-rank solution with high probability. The previous derivation thus shows that the generalized trace norm can be used to modify this linear operator and hence we might expect the probability of successful low-rank matrix recovery to increase. As potential future work it might be interesting to follow this line of reasoning in order to establish quantitative statements for the generalized trace norm.

6.4 Application to Projective SfM

Even though the generalized trace norm is a very general concept with connections to compressive sensing, matrix completion, collaborative filtering, and general low-rank problems, we exemplify its power on

a very specific computer vision problem, namely the Structure-from-Motion (SfM) problem. Let the camera matrix at frame f be $\mathbf{P}_f \in \mathbb{R}^{3\times4}$ and the projective coordinates of the n^{th} feature point be $\mathbf{s}_n \in \mathbb{R}^{4\times1}$. Then the projection into the camera image plane is given by the projective equation $\lambda_{f,n}\mathbf{x}_{f,n} = \mathbf{P}_f\mathbf{s}_n$, where $\mathbf{x}_{f,n} = (u_{f,n}, v_{f,n}, 1)^T$ denotes the homogeneous coordinates of the given image observation. The goal of SfM is to recover the projective depths $\lambda_{f,n}$ from several image observations of the same 3D points at different points in time. Let F denote the total number of frames and N the total number of points. All the observations from all the points at all the frames are concisely described by introducing the stacked camera matrix $\mathbf{P} = [\Downarrow_f \ \mathbf{P}_f] \in \mathbb{R}^{3F\times4}$, the structure matrix $\mathbf{X} = [\Rightarrow_n \ \mathbf{s}_n]^T \in \mathbb{R}^{N\times4}$, the combined observation matrix $\mathbf{x} = [\Downarrow_f \Rightarrow_n (u_{f,n}, v_{f,n}, 1)^T] \in \mathbb{R}^{3F\times N}$ and the combined projective depth matrix $\lambda = [\Downarrow_f \Rightarrow_n \lambda_{f,n}] \in \mathbb{R}^{F\times N}$. Then we get the matrix equation

$$[\lambda \otimes \mathbf{1}_{3\times1}] \odot \mathbf{x} = \mathbf{P}\mathbf{X}^T = \mathbf{W} \in \mathbb{R}^{3F\times N}, \tag{6.16}$$

where $\otimes$ denotes the Kronecker product and $\odot$ denotes the Hadamard product. This problem is an instance of an affine rank minimization problem

$$\min_{\mathbf{W},\lambda} \operatorname{rank}(\mathbf{W}) \ \text{ s.t. } \ \mathbf{W} = [\lambda \otimes \mathbf{1}_{3\times1}] \odot \mathbf{x}. \tag{6.17}$$

Surprisingly, the convex relaxation of this projective rigid SfM problem with the trace-norm has only recently been introduced [DLH10], shortly after this relaxation has been applied to non-rigid affine SfM [OO09]. We refer to [Aan+02; OH07; ZH09; EH10; DLH10] and references therein for other, mostly iterative approaches to solve this low-rank SfM problem. Here, we are interested in the behavior of the convex relaxation using the generalized trace norm. But firstly, we have to note that the formulation in Eq. (6.17) is ambiguous. Specifically there is a $F + N$-dimensional solution space since the camera matrices *and* the points are projective entities and hence only defined up to scale. This can easily be seen by recalling Lemma 5.1.2 in [HJ94] which establishes the commutativity of the Hadamard product with diagonal matrices. Applied to our problem this looks like

$$[\mathbf{D}_P \otimes \mathbf{I}_3]\,\mathbf{P}\mathbf{X}^T\mathbf{D}_X = [[[\mathbf{D}_P \otimes \mathbf{I}_3]\,[\lambda \otimes \mathbf{1}_{3\times1}]\,\mathbf{D}_X] \odot \mathbf{x}],$$

with arbitrary diagonal matrices $\mathbf{D}_P \in \mathbb{R}^{F\times F}$ and $\mathbf{D}_X \in \mathbb{R}^{N\times N}$. Applying the Kronecker product property on the right hand side we get

$$[\mathbf{D}_P \otimes \mathbf{I}_3]\,\mathbf{P}\mathbf{X}^T\mathbf{D}_X = [\mathbf{D}_P\lambda\mathbf{D}_X \otimes \mathbf{1}_{3\times 1}] \odot \mathbf{x}, \tag{6.18}$$

Therefore λ is only unique up to a left- and right-multiplication with a diagonal matrix.

6.4.1 Avoiding Trivial Solutions

Without any further constraints, the optimal solution will lead to the trivial solution $\lambda = \mathbf{0}_{F\times N}$. In order to prevent this trivial solution, additional constraints on λ are required. In the past, the row- and column sum have been constrained while at the same time enforcing non-negative entries

$$\lambda \geq \mathbf{0}_{F\times N}, \lambda\mathbf{1}_{N\times 1} = \mathbf{c} \in \mathbb{R}^{F\times 1}, \lambda^T\mathbf{1}_{F\times 1} = \mathbf{r} \in \mathbb{R}^{N\times 1}. \tag{6.19}$$

Such a constraint set is known as the transportation polytope in the operations research community. The orientation of this transportation polytope depends upon the choice of $\mathbf{c}$ and $\mathbf{r}$. Note that the equality constraints are not independent, since the sum over the column sums has to equal the sum over the row sums and hence, the equality constraints fix $F+N-1$ degrees of freedom. Remember that there is a $F+N$ dimensional variety of possible rank-4 solutions. The affine equality constraints of the transportation polytope slice through this $F+N$ dimensional variety and cut out a 1D variety of possible rank-4 factorizations. A proper choice of $\mathbf{c}$ and $\mathbf{r}$ is crucial for the success of the trace-norm relaxation. Usually the choice $\mathbf{c} = N\mathbf{1}_{F\times 1}$ and $\mathbf{r} = F\mathbf{1}_{N\times 1}$ is made. However, based on several experiments with synthetic data we concluded that such a choice can often result in a bad convex relaxation. Instead, we propose to replace the transportation polytope constraints Eq. (6.19) by the single scalar equality constraint $\mathbf{1}_{F\times 1}^T\lambda\mathbf{1}_{N\times 1} = FN \in \mathbb{R}$, which can be thought of as fixing the scale of the overall reconstruction. The intuition is that the algorithm should figure out the optimal scaling on its own. Our experiments showed that this single equality constraint does only lead to the trivial or to an almost trivial solution with a large fraction of the entries in λ set to zero for really challenging data. If such a situation is encountered additional constraints (e.g. $\lambda \geq 0.1$) can still be introduced in our formulation.

6.4.2 SDP Formulation for Inexact Measurements

The generalized trace norm minimization problem can be formulated as a standard SDP problem. Hence, any off-the-shelf SDP solver could be used to find the minima of the convex problem. Standard SDP solvers are usually based on second-order interior point methods which enjoy quadratic convergence near the minimum. However the cost per iteration of these methods is prohibitively high for large matrices and hence standard interior-point solvers can only be applied to small-sized problems. This is why only small-sized problems could have been solved in [DLH10]. Due to the famous Netflix challenge, an increased interest in solvers for matrix completion problems has been observed (see [LCM10; MHT10] and references therein). These solvers are based on first-order methods and do not require solving a costly linear system involving the Hessian matrix in each iteration. These solvers have solved matrix completion problems with matrix sizes up to 10000×10000 and 10^5 observed entries in about 2.5 hours [MHT10]. The algorithm we are going to present makes use of the same underlying building blocks (especially the soft-thresholding operation) and hence also scales to thousands of image measurements. Unfortunately, the matrix completion solvers are tailored to the very specific linear equality constraints of the matrix completion problem and our problem formulation has more general linear equality constraints. Moreover, existing solvers for matrix completion problems or robust PCA are either based on a pure L1-noise model or on a pure L2-noise model. A more suitable choice for our purposes is a robust cost function, like the Huber cost function

$$F(\mathbf{E}) = \|\mathbf{E}\|_\epsilon = \sum_{i,j} |\mathbf{E}_{i,j}|_\epsilon \text{ where } |e|_\epsilon = \begin{cases} \frac{|e|^2}{2\epsilon} & |e| < \epsilon \\ |e| - \frac{\epsilon}{2} & |e| \geq \epsilon \end{cases}.$$

Therefore, we propose to minimize the following cost function with a first-order primal-dual algorithm (see Alg. 6)

$$\min_{\mathbf{W},\lambda} \|\mathbf{W}\|_{*\mathbf{C}_r,\mathbf{C}_c} + \alpha \|\mathbf{E}\|_\epsilon \tag{6.20}$$

$$\text{s.t. } \mathbf{E} = \mathbf{W} - [\lambda \otimes \mathbf{1}_{d\times 1}] \odot \mathbf{x} \tag{6.21}$$

$$\mathbf{1}_{F\times 1}^T \lambda \mathbf{1}_{N\times 1} = FN. \tag{6.22}$$

The matrix-size normalized trace complexity measure presented in [SS10] suggests choosing α proportional to $\sqrt{dFN}$. Such a choice is also

according to the consistency conditions derived in [Bac08] and hence α is always chosen proportional to $\sqrt{dFN}$ in our formulation.

6.5 Primal-Dual Proximal Optimization

Proximal point optimization methods are based on applying proximal operators to sub-problems of the original problem at each iteration. Applied to the problem in Eq. (6.20) this requires the proximal operator of the generalized trace-norm. It is non trivial to derive this operator. However, we can side-step this issue by appealing to the equivalent formulation based on Eq. (6.15) and rather solve

$$\min_{\tilde{\mathbf{W}},\lambda} \left\|\tilde{\mathbf{W}}\right\|_* + \alpha \left\|\mathbf{E}\right\|_\epsilon \tag{6.23}$$

$$\text{s.t. } \mathbf{E} = \mathbf{C}_r^{-1}\tilde{\mathbf{W}}\mathbf{C}_c^{-T} - \left[\lambda \otimes \mathbf{1}_{d\times 1}\right] \odot \mathbf{x} \tag{6.24}$$

$$\mathbf{1}_{F\times 1}^T \lambda \mathbf{1}_{N\times 1} = FN. \tag{6.25}$$

The equality constraints in Eq. (6.24) are linear in $\tilde{\mathbf{W}}$ and λ and hence can be written in the form $\mathrm{vec}(\mathbf{E}) = \mathbf{e} = \mathbf{L}\left(\tilde{\mathbf{w}}^T, \mathrm{vec}(\lambda)^T\right)^T$, where $\tilde{\mathbf{w}} = \mathrm{vec}\left(\tilde{\mathbf{W}}\right)$. For our problem, the matrix $\mathbf{L} \in \mathbb{R}^{dFN\times dFN+FN}$ looks like $\mathbf{L} = \left[\mathbf{C}_c^{-1} \otimes \mathbf{C}_r^{-1}, -\mathrm{diag}(\mathbf{x})\left[\mathbf{I}_{FN} \otimes \mathbf{1}_{d\times 1}\right]\right]$. This convex problem is not yet in a standard form such that a primal-dual proximal method can be applied. The first step in order to get such a standard form is to consider the Lagrangian

$$L(\tilde{\mathbf{w}}, \lambda, \mathbf{e}; \mathbf{q}) = \left\|\tilde{\mathbf{W}}\right\|_* + \mathbb{I}_{\mathbf{1}^T\lambda\mathbf{1}-FN} + \langle \mathbf{q}, \mathbf{L}\begin{pmatrix}\tilde{\mathbf{w}}\\ \lambda\end{pmatrix} - \mathbf{e}\rangle + \alpha\left\|\mathbf{e}\right\|_\epsilon \tag{6.26}$$

$$= \left\|\tilde{\mathbf{W}}\right\|_* + \mathbb{I}_{\mathbf{1}^T\lambda\mathbf{1}-FN} + \langle \mathbf{q}, \mathbf{L}\begin{pmatrix}\tilde{\mathbf{w}}\\ \lambda\end{pmatrix}\rangle - \langle \mathbf{q}, \mathbf{e}\rangle + \alpha\left\|\mathbf{e}\right\|_\epsilon, \tag{6.27}$$

with dual variables $\mathbf{q}$ and indicator function $\mathbb{I}_.$. The resulting Lagrangian dual equals

$$L(\mathbf{q}) = \inf_{\tilde{\mathbf{w}},\lambda,\mathbf{e}} \left\|\tilde{\mathbf{W}}\right\|_* + \mathbb{I}_{\mathbf{1}^T\lambda\mathbf{1}-FN} + \langle \mathbf{q}, \mathbf{L}\begin{pmatrix}\tilde{\mathbf{w}}\\ \lambda\end{pmatrix}\rangle - \langle \mathbf{q}, \mathbf{e}\rangle + \alpha\left\|\mathbf{e}\right\|_\epsilon \tag{6.28}$$

$$= \inf_{\tilde{\mathbf{w}},\lambda} \left\|\tilde{\mathbf{W}}\right\|_* + \mathbb{I}_{\mathbf{1}^T\lambda\mathbf{1}-FN} + \langle \mathbf{q}, \mathbf{L}\begin{pmatrix}\tilde{\mathbf{w}}\\ \lambda\end{pmatrix}\rangle - \sup_{\mathbf{e}} \langle \mathbf{q}, \mathbf{e}\rangle - \alpha\left\|\mathbf{e}\right\|_\epsilon \tag{6.29}$$

$$= \inf_{\tilde{\mathbf{w}},\lambda} G(\tilde{\mathbf{w}}, \lambda) + \langle \mathbf{q}, \mathbf{L}\begin{pmatrix}\tilde{\mathbf{w}}\\ \lambda\end{pmatrix}\rangle - \left(\alpha\left\|.\right\|_\epsilon\right)^*(\mathbf{q}), \tag{6.30}$$

where we have introduced the function $G(\tilde{\mathbf{w}}, \lambda) = \left\|\tilde{\mathbf{W}}\right\|_* + \mathbb{I}_{\mathbf{1}^T\lambda\mathbf{1}-FN}$ and the convex conjugate $\left(\alpha\left\|.\right\|_\epsilon\right)^*(\mathbf{q}) = \sup_{\mathbf{e}}\langle\mathbf{q},\mathbf{e}\rangle - \alpha\left\|\mathbf{e}\right\|_\epsilon$ of the scaled Huber error function $\alpha\left\|.\right\|_\epsilon$. If this convex conjugate is denoted with $F^*(\mathbf{q})$, exactly the primal-dual setting $\inf_{\mathbf{x}=(\tilde{\mathbf{w}}^T,\lambda^T)^T}\sup_{\mathbf{y}=\mathbf{q}}\langle\mathbf{L}\mathbf{x},\mathbf{y}\rangle + G(\mathbf{x}) - F^*(\mathbf{y})$ considered by Chambolle and Pock in [CP11] is recovered. Their algorithm requires the proximal operators $\text{prox}_{\tau G}$ and $\text{prox}_{\sigma F^*}$ of the functions $G(.)$ and $F^*(.)$. The former is simple as G is the sum of two completely decoupled functions which also means its proximal operator decouples into two independent proximal operators of the trace-norm and of the indicator function which are both well-known. The proximal operator for the standard trace norm $\text{prox}_{\tau\|.\|_*}(\mathbf{A})$ is defined as the solution to the problem (see [MHT10])

$$\mathbf{X}_* = \arg\min_X \tau\left\|\mathbf{X}\right\|_* + \frac{1}{2}\left\|\mathbf{X}-\mathbf{A}\right\|_F^2, \tag{6.31}$$

and the well-known solution to this problem is the soft-thresholding given in Alg. 7. The proximal operator the indicator function, which projects λ into the feasible region defined in Eq. (6.25), can be derived with Lagrange multipliers and is given in Alg. 8 (the Lagrange multipliers are denoted $\mathbf{y}$ in this algorithm). The proximal operator for the dual variables $\mathbf{q}$ due to the convex conjugate of the scaled robust Huber cost function is slightly more complex. The convex conjugate of the unscaled robust Huber cost has a closed-form solution $\left(\left\|.\right\|_\epsilon\right)^*(q) = \mathbb{I}_{|\mathbf{q}|\leq 1} + \frac{\epsilon}{2}|q|^2$. Its proximal operator equals

$$\text{prox}_{\sigma(\|.\|_\epsilon)^*}(p) = \arg\min_q \mathbb{I}_{|\mathbf{q}|\leq 1} + \frac{\sigma\epsilon}{2}|q|^2 + \frac{1}{2}\left\|q-p\right\|_2^2 = \frac{\frac{p}{1+\sigma\epsilon}}{\max\left(1,\left|\frac{p}{1+\sigma\epsilon}\right|\right)}.$$

The scaling property of convex conjugates

$$F^*(q) = \left(\alpha\left\|.\right\|_\epsilon\right)^*(q) = \alpha\left(\left\|.\right\|_\epsilon\right)^*\left(\frac{q}{\alpha}\right) = \alpha\mathbb{I}_{\left|\frac{q}{\alpha}\right|\leq 1} + \frac{\epsilon\alpha}{2}\left|\frac{q}{\alpha}\right|^2 = \mathbb{I}_{|q|\leq\alpha} + \frac{\epsilon}{2\alpha}|q|^2$$

leads to the following proximal operator for the scaled Huber cost

$$\text{prox}_{\sigma F^* = \sigma\left(\alpha\|.\|_\epsilon\right)^* = \sigma\left(\mathbb{I}_{|.|\leq\alpha+\frac{\epsilon}{2\alpha}|.|^2}\right)} \tag{6.32}$$

$$= \arg\min_q \mathbb{I}_{|q|\leq\alpha} + \frac{\sigma\epsilon}{2\alpha}|q|^2 + \frac{1}{2}\left\|q-p\right\|_2^2 = \frac{\frac{p}{1+\frac{\sigma\epsilon}{\alpha}}}{\max\left(1,\left|\frac{p}{\alpha+\sigma\epsilon}\right|\right)}. \tag{6.33}$$

Algorithm 6: First-Order Primal-Dual Algorithm

Data: $\mathbf{L}, \alpha, \epsilon$
input : Linear Equality Constraint Matrix: $\mathbf{L} \in \mathbb{R}^{m \times dFN+m}$; Weight parameter: $\alpha \in \mathbb{R}$; Robust Huber cost function parameter: $\epsilon \in \mathbb{R}$; Time step sizes $\tau\sigma < \frac{1}{\|\mathbf{L}\|_{op}^2}$
output : Minimizer $\mathbf{W}$ and λ for problem 6.23

// Initialization:
$\mathbf{q}_0 = \mathbf{0}_{m\times 1}, \lambda_0 = \mathbf{1}_{m\times 1}, \mathbf{W}_0 = \mathbf{0}_{dF\times N}$;
while *not converged* **do**
  // Primal Variable Update:
  $\begin{pmatrix} \mathrm{vec}\left(\tilde{\mathbf{W}}\right) \\ \mathrm{vec}\left(\tilde{\lambda}\right) \end{pmatrix} = \begin{pmatrix} \mathrm{vec}\left(\mathbf{W}_t\right) \\ \mathrm{vec}\left(\lambda_t\right) \end{pmatrix} - \tau \mathbf{L}^T \mathbf{q}_t$;
  // Primal Variable Update:
  $\mathbf{W}_{t+1}$ = ProxTrace($\tilde{\mathbf{W}}$, τ) // Alg. 7 ;
  λ_{t+1} = ProxFeasible($\tilde{\lambda}$) // Alg. 8 ;
  // Reflection step:
  $\hat{\mathbf{W}} = 2\mathbf{W}_{t+1} - \mathbf{W}_t$;
  $\hat{\lambda} = 2\lambda_{t+1} - \lambda_t$;
  // Dual Variable Update:
  $\tilde{\mathbf{q}} = \frac{1}{1+\frac{\sigma\epsilon}{\alpha}} \left(\mathbf{q}_t + \sigma \mathbf{L} \begin{pmatrix} \mathrm{vec}\left(\hat{\mathbf{W}}\right) \\ \mathrm{vec}\left(\hat{\lambda}\right) \end{pmatrix} \right)$;
  $\mathbf{q}_{t+1}$ = ProxBox($\tilde{\mathbf{q}}$,α) ;
  $t = t + 1$;
end

The complete algorithm is summarized in Alg. 6 and we refer the interested reader to [CP11] for more details about proximal splitting techniques.

Performance Improvements: The pseudo-inverse of the projection operator into the feasible region stays constant throughout all the iterations and could be pre-computed once and for all. However, even though the matrix is highly sparse, its pseudo-inverse becomes dense which poses a problem for large problem sets. Hence, instead of pre-computing the pseudo-inverse we rather pre-compute a sparse LU-decomposition which is guaranteed to stay sparse. Then in each iteration, a sparse LU-problem needs to be solved which can be done very efficiently. If the data is incomplete, i.e. if there are missing entries in our input data matrix $\mathbf{x}$ (maybe due to occlusions or tracking failure), there is no need to introduce a full F-by-N matrix for the unknown projective depths. We only need to keep track of those

Algorithm 7: Soft-Thresholding for Trace-Norm

Data: $\mathbf{A}, \tau$
input : Matrix to approximate: $\mathbf{A} \in \mathbb{R}^{dF \times N}$; soft-thresholding parameter: τ
output : Soft-thresholded solution for Eq. (6.31): $\mathbf{W}_*$

1 $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ = `svd( A)` ;
2 $\tilde{\mathbf{\Sigma}}$ = `max(`$\mathbf{\Sigma} - \tau, 0$`)` ;
3 $\mathbf{X}_* = \mathbf{U}\tilde{\mathbf{\Sigma}}\mathbf{V}^T$;

Algorithm 8: Projection Operator into Feasible Region

input : Vector to approximate: $\mathbf{a}$; Linear equality constraint matrix: $\mathbf{M}$; Right-hand side of equality constraints: $\mathbf{b}$
output : Closest point to $\mathbf{a}$ in feasible region: $\mathbf{x}_* = \arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{x} - \mathbf{a}\|_2^2$ s.t. $\mathbf{A}\mathbf{x} = \mathbf{b}$.

1 $\begin{pmatrix} \mathbf{x}_* \\ \mathbf{y}_* \end{pmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{M}^T \\ \mathbf{M} & \mathbf{0} \end{bmatrix}^\dagger \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix}$;

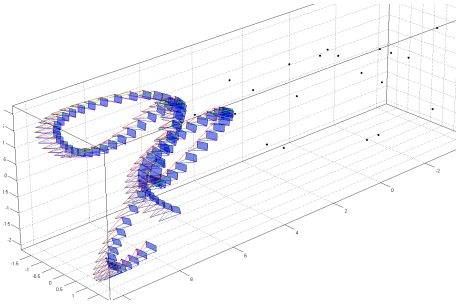
projective depths $\lambda_{f,n}$ which corresponds to visible points. Therefore we put all these entries in one single m-vector where m denotes the number of observations. For simplicity, this vector is still called λ in Alg. 6. In this case, the linear equality constraint matrix $\mathbf{L}$ of course only consists of those rows corresponding to observed points. There is one drawback of the formulation in Eq. (6.23) compared to Eq. (6.20). The equality constraints in Eq. (6.21) would lead to a highly sparse matrix $\mathbf{L}$, the equality constraint in Eq. (6.24) however lead to a dense first block in $\mathbf{L}$ due to the Kronecker-product $\mathbf{C}_c^{-1} \otimes \mathbf{C}_r^{-1}$. For large datasets this matrix can become huge. Luckily, there is no need to store the Kronecker-product: instead the Kronecker-product property $\text{vec}\left(\mathbf{C}_r^{-1}\tilde{\mathbf{W}}\mathbf{C}_c^{-T}\right) = \left[\mathbf{C}_c^{-1} \otimes \mathbf{C}_r^{-1}\right]\tilde{\mathbf{w}}$ is used in step 5 and 13 of Alg. 6.

Additional Insights: First order proximal methods are known to converge quickly at the beginning. When using L1-error terms however, these methods often oscillate around the true optimum and only slowly converge towards this optimum due to the non-differentiability of the L1-norm at 0. The robust Huber cost function replaces the L1-error around the origin with a L2-penalty which is differentiable. Hence, the Huber cost function is not only a more appropriate model for outliers and inliers contaminated by Gaussian noise it also leads to less oscillations and hence faster convergence.

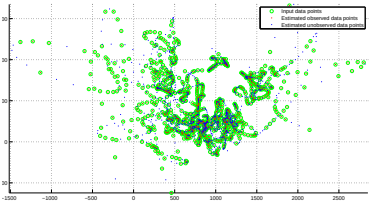
6.6 Experiments

6.6.1 Synthetic Experiments

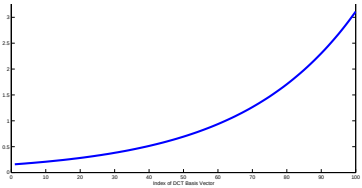
The robustness of factorization approaches, and actually of any SfM approach, is known to largely depend on the motion. With a larger variation in camera poses, the robustness of SfM method increases at the price of a more difficult feature point matching stage. In our synthetic experiments, we have generated realistic projections of feature points based on parameters of a real camera. The camera internal calibration matrix is generated with zero skew, square pixels, principal point in the center of the image, a focal length of 10mm, and a resolution of 1920-by-1080. In order to generate smooth camera motions, the external calibration matrices are based on spline interpolated camera rotations and translations. The mean of the camera centers is chosen at $(8, 0, 0)$ and varies smoothly along a spline curve with 10 uniformly sampled control points in $[6, 10] \times [-2, 2] \times [-2, 2]$ throughout the motion. The cameras are oriented such that the principal axis always points to the origin and rotations around the principal axis are limited to ± 25 degrees again smoothed with a spline interpolation. 100 camera poses have been sampled from the resulting spline interpolation curves. 20 points with a uniform distribution in a cube of size $[-5, 5] \times [2, 2] \times [2, 2]$ are generated in order to simulate realistic variations in depth. Fig. 6.1a shows the resulting spatial setup. Fifty percent of the entries in the resulting data matrix are marked uniformly at random as missing. No noise is added in order to facilitate the search for optimal parameters of the Huber cost function ϵ (ϵ is fixed to zero) and of the data fidelity trade-off α. The standard projective factorization approach with ordinary trace norm regularization (which has also been presented in [DLH10]) failed to recover the underlying structure for whatever choice of parameter value α. The smallest reprojection error we could achieve was 27.5 pixels. Note that the parameter α can always be chosen such that the resulting SDP formulation will indeed provide a rank-4 solution. Our experiments however show that such a low-rank solution is not equal to the true underlying low rank matrix: in order to get a rank-4 solution the trace norm regularization term has to be large enough compared to the data term which on the other hand implies a stronger regularization of the unknown matrix *uniformly* in all directions. This uniform penalization property is exactly the drawback of the standard trace norm. Either



(a) Spatial Setup



(b) Projections onto camera planes



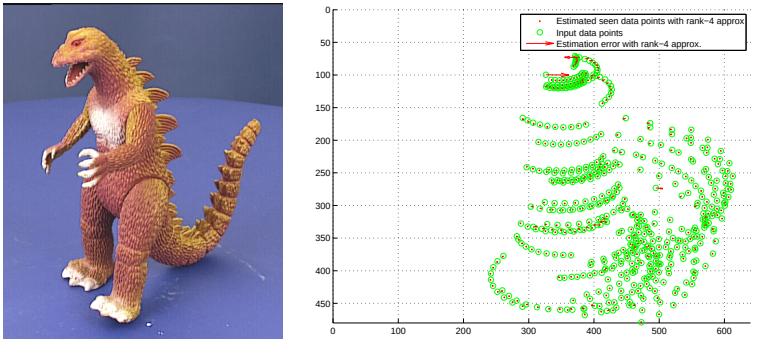
(c) DCT Basis Weights

Figure 6.1: Synthetic experiment: In Fig. 6.1a the spatial configuration of the cameras and points is shown whereas Fig. 6.1b shows the observed input data (green) and the reprojection of the estimated points (in red (blue) projections corresponding to observed (missing) entries) for all the frames. The smooth trajectories are clearly visible. Our approach achieves an error with a sub-pixel accuracy of 0.2 pixels. Fig. 6.1c shows the weighting of the DCT basis components. The matrix $\mathbf{C}_r$ is chosen such that the columns of $\mathbf{W}$ (which correspond to 3D point trajectories) are transformed into a three-fold Direct Cosine Transform (DCT) basis (one DCT basis for each (x,y,z) component). Motivated by the fact that natural data follows a power law decay rate in a DCT basis (e.g. also used for jpg-compression) we have chosen an exponential weighting scheme for the DCT basis vectors. As shown in the image, smooth directions in the DCT are penalized less heavily than non-smooth ones.

we choose a small α and get a highly regularized rank-4 matrix or we choose a larger α which leads to a full rank solution to the SDP problem. In either way, the solution of the convex relaxation is not the one we are looking for. Adding weight matrices $\mathbf{C}_r$ and $\mathbf{C}_c$ introduces *non-uniform* regularization, i.e. not all directions are penalized in the same way. We have chosen the matrix $\mathbf{C}_r$ such that smooth directions are penalized less heavily than non-smooth ones (see Fig. 6.1c). The matrix $\mathbf{C}_c$ down-weights the direction $\mathbf{1}_{1\times N}$ by a factor of roughly 4 since we already know that this direction should be present in our solution (this direction corresponds to the properly scaled homogeneous coordinates of the points, i.e. the last row of $\mathbf{X}^T$). With such a choice the generalized trace norm formulation successfully recovered the structure up to a sub-pixel reprojection error of 0.2 pixels (see Fig. 6.1b for a visual depiction of the measured input points and the recovered estimates). Note that for a fair comparison, all the reprojection errors we state are indeed based on the best rank-4 factorization in the Frobenius-norm sense of the SDP solution and *not* directly on the potentially full-rank SDP solution. This is important since especially for the standard trace norm regularized solutions, there might be quite a difference between the SDP solution and its best rank-4 approximation. Running the very same experiment with Gaussian noise with a standard deviation of 0.5 (2) pixels changed the average reprojection error of our method with the choice $\epsilon = 0.5$ ($\epsilon = 2$) for the Huber cost threshold to 0.6 (2.5) pixels. The algorithm is implemented in Matlab (single-threaded) on a Core i7 740QM and needs $0.023s$ per iteration, the algorithm converged after 1100 iterations for a 600×50 matrix with 50% missing entries. This is orders of magnitude faster than interior point solvers.

6.6.2 Real World Data

As in [DLH10], the Dinosaur Oxford data sequence has been used for verification purposes on a real-world data sequence. Only those points which were visible in at least 10 frames have been considered. This resulted in a sequence with $F = 17$ frames and $N = 47$ points with roughly 30 percent of missing entries. Exactly the same weight matrices as for the synthetic data experiment have been chosen. The robust Huber parameter has been set to a 2 pixel threshold. Our method achieves an average reprojection error of 0.95 pixels whereas the standard trace norm regularization achieves a reprojection error



(a) Dino Sequence (b) Reprojection Error

Figure 6.2: For the dinosaur sequence, our approach leads to an average reprojection error of 0.95 pixels which is a reduction by a factor of two compared to the standard trace-norm regularized formulation.

of 1.84 pixels. Hence, we get a reduction by a factor of two (see also Fig. 6.2). Note that the full dinosaur sequence has a very strong band diagonal pattern of known entries. Such a pattern proved to be very challenging for the SDP relaxations we considered in this chapter.

6.6.3 Conclusion

As the experiments with a projective SfM factorization problem show, the standard trace norm regularization can fail whereas the generalized trace norm still succeeds in finding the correct rank-4 solution. Moreover, as expected, the generalized trace norm with its built-in prior knowledge can indeed handle more missing entries. From an algorithmic point of view, the first-order primal-dual algorithm for generalized trace-norm regularized problems can handle considerably larger problems than previous algorithms for trace norm relaxations for SfM factorizations.

Unfortunately, there are no guarantees that the generalized trace norm relaxation gives the correct rank-4 solution. Based on our experience, this again depends largely on the motion, structure, and the pattern of known entries. Especially the band-diagonal structure proved to be

challenging for convex relaxations. Nevertheless, we see the generalized trace norm as a flexible tool and applicable to other problems, such as structured multi-class prediction problems.

6.7 Future Work: Tensor Trace Norm

Recently, new results for M-estimators of the form in Eq. (6.13) with trace-norm regularization have been derived [Neg+10; Gro11] which tell that if the operator $\mathcal{A}$ fulfills certain conditions then the trace-norm regularized convex problem will provide the correct low-rank solution with high probability. Results along these lines actually motivated our initial interest for convex trace norm relaxations. As we have seen in Chap. 4, tensors with a low multi-rank can be used to model complex interactions between multiple subspaces. Unfortunately, in the presence of missing entries, the involved subspaces can no longer easily be computed and we have to fall back to iterative local optimization, e.g. such as the ones presented in Chap. 3. An alternative is to define a convex relaxation of the multi-rank function. A straight-forward approach is to sum the trace norms of the individual tensor flattenings (maybe weighted appropriately by the number of entries)

$$\|\mathcal{W}\|_* = \sum_i \left\|\mathcal{W}_{(i)}\right\|_*, \tag{6.34}$$

where the summation goes over all the modes of the tensor. We have run several numerical experiments which showed that regularization with the trace norm $\|\mathcal{W}\|_*$ seems to be considerably stronger than regularization with a flattening just along one mode, i.e. regularization with $\|\mathcal{W}\|_*$ gave a low multi-rank tensor whereas $\left\|\mathcal{W}_{(1)}\right\|_*$ usually did not. Simultaneously while we did our experiments, Tomioka et.al. [THK10] found results which were consistent with our findings.

The Higher-Order SVD [LMV00] of the mode-d tensor $\mathcal{W} = \mathcal{S} \times_1 \mathbf{U}_1 \times_2 \ldots \times_d \mathbf{U}_d$ where each $\mathbf{U}_i$ has orthogonal columns leads to an interesting interpretation of the tensor-trace norm in Eq. (6.34). In contrast to the generalized trace norm, the standard trace norm is unitarily invariant, i.e. left or right multiplication with an orthogonal matrix does not

change the trace norm of a matrix. Therefore it holds

$$\begin{aligned}\|\mathcal{W}\|_* &= \sum_i \left\|\mathcal{W}_{(i)}\right\|_* = \sum_i \left\|\mathbf{U}_i \mathcal{S}_{(i)} \left[\mathbf{U}_1 \otimes \ldots \mathbf{U}_{i-1} \otimes \mathbf{U}_{i+1} \otimes \ldots \otimes \mathbf{U}_d\right]^T\right\|_* \\ &= \sum_i \left\|\mathcal{S}_{(i)}\right\|_* = \sum_{i,j} \sigma_{i,j}, \end{aligned} \tag{6.35}$$

where the singular values of $\mathcal{S}_{(i)}$ have been denoted as $\sigma_{i,j}$. [LMV00] shows that $\sigma_{i,j}$ equals the Frobenius norm of the slice of the core tensor given by holding the index j in mode i fixed[2]. Therefore, we conclude that the tensor trace norm in Eq. (6.34) can be interpreted as a group trace norm with overlapping groups given by slices through the core tensor along each mode. As such, the tensor trace norm favours compact core tensors $\mathcal{S} \in \mathbb{R}^{r_1 \times \ldots \times r_d}$ with small r_i. Due to the group trace norm however, the core tensor can be fully dense. As far as we know, this is a novel interpretation.

However, our original motivation was to extend the results for trace-norm for matrices [Neg+10; Gro11] to tensors. Our empirical results show that similar results indeed also hold for multi-rank tensors. Unfortunately, the analysis of the resulting convex problem becomes considerably more complex and we have to leave the proof of such an extension as future work.

[2] Or equivalently, $\sigma_{i,j}$ equals $\left\|\mathcal{S}_{(i),[j,:]}\right\|_F$

7 Sampling Approaches

This chapter presents an entirely probabilistic view on rigid structure-from-motion. A posterior is defined whose MAP defines the solution to the SfM problem. In order to find the MAP, a Markov chain is set up with Gibbs and Metropolis-Hastings samplers whose invariant distribution equals the posterior. This chapter presents some background information about sampling procedures, derives the sampling distributions for structure-from-motion, and then presents some preliminary tests. Unfortunately, it turns out that the sampling scheme does not seem to work well for some problem instances. The chapter therefore concludes by drawing several conclusions based on our experience and suggests improvements and alternatives for future work in sampling-based low-rank factorizations.

7.1 Motivation

From a conceptual point of view, the SfM problem has been shown in Chap. 6 to correspond to an affine rank minimization problem and in the special case of affine cameras, SfM reduces to a low-rank matrix completion problem (which is a special instance of an affine rank minimization problem). Local optimization and convex relaxations have been applied in previous chapter in order to address such affine rank minimization problems. Unfortunately, these approaches suffer from some short-comings which will be highlighted in the following.

The Wiberg algorithm has been shown in Chap. 3 to work especially well for such low-rank matrix completion problems. Unfortunately, the Wiberg algorithm is inherently based on a L2-error and outlier contaminated data will lead to bad predictions with the Wiberg algorithm. Motivated by the success of the Wiberg algorithm, Eriksson and van den Hengel [EH10] applied the derivation of the Wiberg algorithm in a L1-error framework which is known to be more robust to outliers. A concept known as convex iterations is used for optimization: a sequence of slightly modified linear programming problems is solved. Eriksson and van den Hengel report a sub-pixel mean reprojection error for

a 319 points and 36 frames subsequence of the Oxford dinosaur sequence. Unfortunately, as the authors state, a proper initialization is crucial, otherwise the algorithm falls into a local minima. Moreover, their algorithm runs for almost 18 minutes for the reported dinosaur subsequence. The solution based on convex relaxations as proposed in Chap. 6 or [DLH10] are independent of the initialization. Unfortunately, the convex relaxation of the rank can be rather loose which means that the convex optimization returns a solution which is not of rank 4 and hence is an invalid reconstruction. Moreover, semi-definite programming problems involving trace norms are computationally expensive, and hence the work [DLH10] only shows examples of size 20-by-20, for which a running time of approximately 10s is reported. For that reason, Chap. 6 introduced a more efficient first-order optimization method specifically tailored for trace-norm terms in the objective function. A further drawback of all these methods is that they do not minimize the true reprojection error: either an affine camera approximation is made or the projective depths are multiplied with the image observations such that the problem stays bilinear at the cost of minimizing an algebraic error.

In order to address these issues, this chapter presents a sampling based approach which targets the fully general case, meaning projective cameras, missing entries, and outliers. We cast the SfM problem into a probabilistic framework which establishes a close connection to recent results in collaborative filtering where Markov Chain Monte Carlo (MCMC) methods are used to find the mode of the posterior distribution. This posterior follows a non-standard distribution because the data likelihood will be based on the reprojection error. We therefore propose to use Gibbs and Metropolis-Hastings samplers. Metropolis-Hastings samplers can be seen as probabilistic versions of hypothesize-and-verify approaches and hence enjoy close connection to RANSAC [FB81] which is generally used in the first stage of state-of-the-art structure-from-motion pipelines. Metropolis-Hastings methods are very flexible in the way they generate hypothesis. As we will see in Sec. 7.5, the Metropolis-Hastings algorithm allows to generate hypothesis using information from many views or many points which is in contrast to the algebraic multiple-view constraints used in RANSAC. Multiple-view constraints are limited to at most 4 views since the weakest constraint is given by reprojecting lines from four views: the constraint then captures the condition that the resulting four planes need to intersect in one single point. In practice,

only two- or three-view constraints are used in RANSAC, however. The sampling approach presented in this chapter can therefore be seen as bridging the gap between highly robust RANSAC-based hypothesize-and-verify frameworks and non-linear optimization for low-rank matrix factorizations which have a more global view on the problem.

7.2 Previous Work

The work in this chapter builds upon many different fields, such as multiple-view geometry, probabilistic and deterministic matrix factorizations, and sampling methods. Probabilistic matrix factorizations for collaborative filtering and sampling methods for SfM problems will be introduced in the next two sections. The essentials of multiple-view geometry have already been discussed in previous chapters and deterministic matrix factorization has been covered in Chap. 3. The basics of sampling methods required in this chapter, more specifically Markov chains, Gibbs, and Metropolis-Hastings samplers, will be covered in Sec. 7.3.

7.2.1 Probabilistic Matrix Factorizations for Collaborative Filtering

Probabilistic matrix factorization algorithms have been developed for collaborative filtering applications where users rate certain items (e.g. users give a score to films they have watched). These ratings can then be arranged in a matrix where the rows represent the items and the columns the users. Of course, not every user watches and rates every single movie which means that the rating matrix is only partially observed. The goal of collaborative filtering is to complete the unknown entries using the known ones in order to suggest a new movie to a user. As already shortly mentioned in Sec. 6.3.1, Salakhutdinov and Mnih [SM08] have proposed to model a rating $\mathbf{R}_{i,j}$ by user i for item j as the inner product between a latent user- $\mathbf{u}_i$ and a latent item-vector $\mathbf{v}_j$, corrupted by additive Gaussian noise $n \sim \text{Normal}\left(0, \sigma^2\right)$. Thus, the probability of the ratings given the user and item vectors equals

$$p(\mathbf{R}|\mathbf{U}, \mathbf{V}) = \prod_{i,j} \text{Normal}\left(\mathbf{R}_{i,j}|\mathbf{u}_i^T \mathbf{v}_j, \sigma^2\right), \tag{7.1}$$

which means that, barring noise, the rating matrix factorizes into two matrices $\mathbf{R} = [\Downarrow_i \mathbf{u}_i^T][\Rightarrow_j \mathbf{v}_j] = \mathbf{UV}$. If there are missing observations then the product in Eq. (7.1) only considers known entries. The problem is then regularized by imposing Gaussian priors on the latent user vectors $\mathbf{u}_i \sim \text{Normal}(\mu_u, \mathbf{\Sigma}_u)$ and similarly on the item vectors. Finally conjugate Gaussian-Wishart priors are put on μ_u and $\mathbf{\Sigma}_u$. This concludes the specification of the full joint probability. The predictive distribution for a new rating is then computed by drawing samples from the posterior of the parameters given the observations $p(\mathbf{U}, \mathbf{V}, \text{Hyperparameters}|\mathbf{R})$. Sampling from the joint posterior is difficult, however blocked Gibbs sampling works very well for this problem. Blocked Gibbs sampling samples from conditional distributions (see Sec. 7.3). Due to the choice of conjugate priors w.r.t. the Gaussian likelihood in Eq. (7.1) (i.e. Gauss-Wishart priors, see [Bis06]), all the conditional probabilities will be either Gaussian or Wishart densities, both of which can be sampled directly and hence the sampling is efficient. We will see later that for this model, these conditional densities correspond to a Hessian based proposal with a full Newton step. The Bayesian probabilistic matrix factorization model could directly be applied to the affine SfM problem in order to handle missing entries. However, the presence of outliers will have large influence on the Gaussian data likelihood. Moreover projective cameras can not be handled within this framework. We therefore propose to use a more general Metropolis-Hastings-within-Gibbs framework which can handle arbitrary distributions.

Also note that an extension of probabilistic matrix factorization to low multi-rank tensors is relatively straight-forward and opens up the possibility to handle more general missing patterns in the affine multi-camera setup presented in Chap. 4. We refer to Hoff's work [Hof11; Hof10] as a starting point for probabilistic tensor factorizations.

7.2.2 Sampling Approaches for SfM

Even though randomized algorithms like RANSAC [FB81] are quite popular for SfM, Monte Carlo sampling methods have only rarely been proposed. We here focus on three representative cases. Kaess and Dellaert [KD10] propose a visual SLAM system with odometry data based on a probabilistic formulation for a calibrated camera rig. However, they target a different scenario than we do. Specifically, [KD10] presents a variation of a sequential filtering method which only updates those

parameters which directly depend on newly established correspondences in a new frame. The parameter update is formulated in an Expectation Maximization framework, where the E-step computes expected correspondences. This is computationally intractable and hence Kaess and Dellaert resort to a MCMC approximation of the E-step. Hence the only randomness in their approach is due to the approximative E-step, otherwise the system would be deterministic. Moreover, given fixed expected correspondences, a non-linear maximization has to be solved in the M-step which requires a close initialization to the optimal value. In contrast, we assume fixed correspondences (potentially contaminated by outliers) but an unknown camera calibration matrix. Our approach optimizes over all the sequence and hence resembles more closely a smoothing method. Additionally, our method is a truly randomized algorithm since random samples are drawn from proposal distributions during the optimization. Torr et.al. [TD03] presented an approach where the feature correspondences are unknown and a coarse-to-fine sampling strategy on an image pyramid is used to propagate feature matching information downwards. However, [TD03] only considers two view relations and hence does not make use of global information from a factorization point of view. Forthsyth et.al. [FHI01] applied Hamiltonian Monte Carlo (HMC, also known as Hybrid Monte Carlo) [Nea93] to the affine SfM formulation. The HMC approach augments each variable with a momentum variable and in order to generate a proposal, HMC performs a symplectic time integration in this augmented space. This symplectic integration basically allows to include deterministic energy minimization methods in the proposal generation, quite an attractive feature in order to compensate the random walk behavior of standard MCMC methods. We have also experimented with the HMC approach for affine and projective factorizations. However, we drew the conclusion that it is virtually impossible to tune the so-called mass matrix parameter in HMC to get adequate acceptance ratios while still exploring the state space sufficiently. This is on par with [FHI01], where the authors state that generating 2000 samples for a problem of size $F = 40$ and $N = 80$ took about a day, even when initialized close to the solution. Recent research in the statistics community tries to choose this mass matrix adaptively, which is possible at the price of significantly increased computational cost [GC11].

7.3 Markov Chain Monte Carlo

The next sections provide a high-level introduction to Markov Chains, Metropolis-Hasting samplers, and Gibbs samplers. For a detailed introduction we refer to Neal's [Nea93] or Tierney's [Tie94] excellent introduction or to the standard textbook [RC04].

7.3.1 Markov Chains

The transition kernel $P : \mathbb{X} \times \mathcal{X} \to [0, 1]$ on a measurable space $(\mathbb{X}, \mathcal{X})$[1] is a central piece for Markov chains. We refer to p.458 in [AL06] for a detailed definition and introduction. Here, we use a slightly simplified derivation along the lines of Chap. 1 and 3 in [LLC10] where the transition kernel $P_t(x, dy)$ is defined as the conditional probability measure for the random variable X_{t+1} given $X_t = x$. A Markov Chain at time t then evolves according to

$$P_{t+1}(dy) = \int_{\mathcal{X}} P_t(dx) P_t(x, dy), \tag{7.2}$$

where $P_t()$ denotes the marginal distribution of X_t at time t. We restrict ourselves to time-homogeneous Markov chains where $\forall t : P_t(x, dy) = P(x, dy)$. The basic goal of Markov chains is to construct a transition kernel with a certain target distribution $\pi(dx)$ as invariant (also called stationary) distribution. A distribution $\pi(dx)$ is invariant for the transition kernel $P(x, dy)$ if the balance condition

$$\pi(dy) = \int_{\mathbb{X}} \pi(dx) P(x, dy) \tag{7.3}$$

is fulfilled. In terms of probability densities $f(x)$, and $p(y|x)$, where $f(x)$ is the derivative of the probability measure $\pi(dx)$ w.r.t. the measure dx, i.e. $\pi(dx) = f(x)dx$ (and similarly for the transition density, i.e. $P(x, dy) = p(y|x)dy$) this balance condition reads

$$f(y) = \int_{\mathbb{X}} p(y|x) f(x) dx.$$

If $\lim_{n\to\infty} P^n(x, A) = \pi(A)$ for π-almost all x and for all measurable A, then the invariant distribution π is called an equilibrium distribution for this Markov chain. $P^n(x, A)$ in this formula denotes repeated

[1] $\mathbb{X}$ is the sample space and $\mathcal{X}$ is a σ-algebra over this sample space.

application of the transition rule in Eq. (7.2). Hence, a Markov chain with equilibrium distribution π eventually converges to π no matter in which state the chain has been started.

The immediate question which arises is: under which conditions does a repeated application of Eq. (7.2) lead to an π-invariant distribution? The answer is given by Theorem 1.5.1 in [LLC10] or equivalently by Theorem 1 in [Tie94] which tells that if a Markov chain with invariant distribution $\pi(dx)$ is both irreducible and aperiodic, then $\pi(dx)$ is the unique invariant distribution and $P_t(dx)$ converges toward $\pi(dx)$ for almost any initial $P_0(dx)$ and $\pi(dx)$ is an equilibrium distribution for this chain. A Markov chain is irreducible if any set $A \in \mathcal{X}$ with $\pi(A) > 0$ can be reached with positive probability from any initial state. A Markov chain is periodic if there exist two states such that the required steps to move between these two states must be a multiple of some integer. Otherwise, the chain is called aperiodic. Aperiodicity and irreducibility are very mild conditions and are usually easily enforced by choosing appropriate transition kernels. It is however trickier to design a transition kernel such that $\pi(dx)$ is an invariant distribution. In this chapter, we will make use of two standard approaches to design such transition kernels, namely the Metropolis-Hasting sampler and the Gibbs sampler.

Interestingly, given several valid transition kernels with the same invariant distribution, a combination of these kernels will result again in a valid kernel with the same invariant distribution. For example, at each sampling iteration, one transition kernel can be selected randomly from a set of predefined kernels which will then be applied at this iteration (see Sec. 2.4 in [Tie94]).

7.3.2 Metropolis-Hastings Sampler

The Metropolis-Hastings kernel is based on a reversible transition kernel. A transition kernel $P(x, dy)$ with invariant distribution $\pi(dx)$ is called reversible if

$$\forall A, B \in \mathcal{X} : \int_B \int_A \pi(dx)P(x, dy) = \int_A \int_B \pi(dy)P(y, dx). \tag{7.4}$$

This equation is known as detailed balance condition which is slightly stronger than the balance condition in Eq. (7.3), i.e. Eq. (7.4) implies Eq. (7.3).

Let the current state of the chain be x_t. A proposal distribution $Q(dy|x_t)$ (with density $q(y|x_t)$, i.e. $Q(dy|x_t) = q(y|x_t)dy$) is used to sample a new proposal state $y \sim Q(dy|x_t)$. This new proposal is accepted as the new state with probability $\alpha(x, y)$ and rejected with probability $1 - \alpha(x, y)$. In the latter case, the chain stays at the current state, i.e. $x_{t+1} = x_t$. The probability $\alpha(x, y)$ is known as the Metropolis-Hastings ratio for reasons which will become clear shortly. This Metropolis-Hasting sampling procedure defines a transition kernel

$$P(x, A) = \int_A Q(dy|x)\alpha(x, y) + \mathbb{I}_{x\in A} \int_{\mathbb{X}} Q(dy|x)(1 - \alpha(x, y)) \tag{7.5}$$

$$= \int_A Q(dy|x)\alpha(x, y) + \mathbb{I}_{x\in A} \left[1 - \int_{\mathbb{X}} Q(dy|x)\alpha(x, y)\right], \tag{7.6}$$

where the first summand corresponds to the acceptance case in which the chain jumps from x to a state $y \in A$ whereas the second summand corresponds to a rejection case in which no jump is performed but where the chain already is in a state $x \in A$. What remains to be shown now is that this kernel has indeed $\pi(dx)$ as its invariant distribution.

Metropolis-Hastings ensures that $\pi(dx)$ is the invariant distribution of $P(x, A)$ by choosing the ratio $\alpha(x, y)$ such that the reversibility condition

$$f(x)q(y|x)\alpha(x, y) = f(y)q(y|x)\alpha(y, x) \tag{7.7}$$

holds. In this case, the detailed balance condition in Eq. (7.4) holds true as the following sequence of equations with $r(x) = 1 - \int_{\mathbb{X}} Q(dy|x)\alpha(x, y)$, $\pi(dx) = f(x)dx$, and $Q(dy|x) = q(y|x)dy$ show

$$\int_B \int_A \pi(dx)P(x, dy) \tag{7.8}$$

$$= \int_B \int_A f(x)dxq(y|x)dy\alpha(x, y) + \int_{A\cap B} r(x)f(x)dx \tag{7.9}$$

$$= \int_B \int_A f(y)q(y|x)\alpha(y, x)dxdy + \int_{A\cap B} r(x)f(x)dx \tag{7.10}$$

$$= \int_A \int_B f(y)q(y|x)\alpha(y, x)dydx + \int_{B\cap A} r(y)f(y)dy \tag{7.11}$$

$$= \int_A \int_B \pi(dy)P(y, dx), \tag{7.12}$$

where the reversibility condition of Eq. (7.7) has been used in Eq. (7.10). If the kernel is additionally also irreducible and aperiodic (which is

generally straight-forward to ensure), then $\pi(dx)$ is the unique equilibrium distribution and a repeated application of the Metropolis-Hastings sampling procedure eventually converges to $\pi(dx)$.

The most-widely used form for the acceptance ratio is

$$\alpha = \min\left(1, \frac{f(x^*)q(x^t|x^*)}{f(x^t)q(x^*|x^t)}\right), \tag{7.13}$$

where x^t denotes the current state of the Markov chain and x^* is the newly proposed state drawn according to $x^* \sim q(x^*|q^t)$. Note that $q(x^*|x^t)$ can be seen as the forward-jumping probability whereas $q(x^t|x^*)$ is the backward-jumping probability. The dependency of the proposal distribution on the current state leads to a high correlation between successive samples. In order to achieve a sufficiently high acceptance rate while still exploring the state space quickly enough, the scale parameters of the proposal distribution must be set correctly. The art of developing efficient sampling schemes usually boils down to choosing suitable scale parameters.

7.3.3 Gibbs Sampler

Assume samples from a joint distribution over several variables are required. If the joint distribution is too complex to sample directly from, Gibbs sampling can be used instead. A Gibbs sampler alternately samples from the conditional probabilities of one variable given all the other ones. It is often much simpler to sample from conditional, lower dimensional probabilities than from the joint distribution. This approach rises the question whether the set of conditional distributions uniquely determine the full joint distribution. The answer is given by the Hammersley-Clifford theorem: Hammersley and Clifford showed that the set of all full conditionals completely determine the joint probability, given that the joint distribution is positive. Note that this is in contrast to the set of marginals which does not possess this property. A distribution with density $f(x_1, \ldots, x_n)$ satisfies the positivity condition if $\forall x_1, \ldots, x_n$ with $f_{X_i}(x_i) > 0$ imply $f(x_1, \ldots, x_n) > 0$, where $f_{X_i}()$ denotes the marginal density w.r.t. random variable X_i. Specifically, the Hammersley-Clifford theorem (e.g. see Theorem 10.5 in [RC04] or Theorem 2.1.1 in [LLC10]) tells that if $f(x_1, \ldots, x_n)$ satisfies this

positivity condition, then for all $y_1, \ldots, y_n$ in the support of f

$$f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) \prod_{i=1}^{n} \frac{f_{X_i|X_{-i}}(x_i|x_1, \ldots, x_{i-1}, y_{i+1}, \ldots, y_n)}{f_{X_i|X_{-i}}(y_i|x_1, \ldots, x_{i-1}, y_{i+1}, \ldots, y_n)}, \tag{7.14}$$

which means that the joint density $f(x_1, \ldots, x_n)$ is proportional to a product of ratios between conditional probabilities. The proportionality constant $f(y_1, \ldots, y_n)$ can be computed from the conditional probabilities by integration over the second factor on the right-hand side in Eq. (7.14), i.e. $\int_{\mathbf{x}} \prod_{i=1}^{n} \frac{f_{X_i|X_{-i}}(x_i|x_1,\ldots,x_{i-1},y_{i+1},\ldots,y_n)}{f_{X_i|X_{-i}}(y_i|x_1,\ldots,x_{i-1},y_{i+1},\ldots,y_n)} d\mathbf{x} = f(y_1, \ldots, y_n)^{-1}$. Note that the existence of a well-defined joint distribution just by the specification of all the conditionals is *not* guaranteed by the Hammersley-Clifford theorem.

The second immediate question is whether the Markov chain created by the Gibbs sampler has an invariant distribution and if so which one and whether the chain will converge to it. This again asks for the transition kernel. For the Gibbs sampler, the density of the transition kernel is

$$p(\mathbf{y}|\mathbf{x}) = f_{X_1|X_{-1}}(\mathbf{y}_1|\mathbf{x}_2, \ldots, \mathbf{x}_n) f_{X_2|X_{-2}}(\mathbf{y}_2|\mathbf{y}_1, \mathbf{x}_3, \ldots, \mathbf{x}_n) \ldots \cdot \tag{7.15}$$

$$f_{X_n|X_{-n}}(\mathbf{y}_n|\mathbf{y}_1, \ldots, \mathbf{y}_{n-1}). \tag{7.16}$$

It can be shown that under mild conditions, $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is indeed the unique equilibrium distribution of the Markov chain generated by this Gibbs sampling kernel. We refer to Sec. 10.2 in [RC04] for all the technical details.

In blocked Gibbs sampling, the variables are split into groups (or blocks), and samples are drawn in turn from the conditional distribution of the variables in one group given all the remaining variables. For example, a natural grouping for a SfM problem is to use two groups: one for all the cameras and one for all the points.

7.3.4 Simulated Annealing

Let us assume that a Markov chain is initialized at a state far away from the modes of the distribution. Due to the correlation between subsequent samples of the Markov chain it might take many samples to move from the low probability region to high probability regions.

Simulated annealing [KGV83] is method to allow larger jumps early on in the sampling. In its simplest form, a starting temperature T_0 and an annealing schedule $T_t = f(t, T_0)$ is defined. The annealing schedule reduces the temperate at each sampling iteration. The Metropolis-Hastings ratios are modified such that the acceptance probabilities are increased

$$\alpha = \min\left(1, \left(\frac{p(x^*)q(x^t|x^*)}{p(x^t)q(x^*|x^t)}\right)^{T_t^{-1}}\right). \tag{7.17}$$

Our current implementation uses the widely-used geometric annealing schedule $T_{t+1} \leftarrow T_t\beta$ for some value $\beta \in (0, 1)$. As future work, we plan to look into other ways of choosing the annealing schedule. When the temperature approaches 0, the probability mass gets concentrated more and more at the global maxima of the distribution. This is an attractive feature for global optimization.

7.4 Probabilistic Formulation for SfM

In this section, the probabilistic formulation for the projective SfM will be presented. The data likelihood is modeled as a function of the reprojection error. Specifically, we assume the reprojection errors to be normally distributed

$$p(\mathbf{U}|\lambda, \mathbf{P}, \mathbf{X}) = \tag{7.18}$$
$$\prod_{f,n}\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^2 \exp\left(-\frac{1}{2\sigma^2}\left\|\lambda_{f,n}^{-1}\mathbf{P}_{f,[1:2,:]}\mathbf{X}_n - \mathbf{u}_{f,n}\right\|_2^2\right),$$

subject to the constraint on the projective depths

$$\lambda_{f,n} = \mathbf{P}_{f,[3,:]}\mathbf{X}_n, \tag{7.19}$$

where $\mathbf{P}_{f,[1:2,:]}$ denotes the first two rows of the camera matrix for frame f, $\mathbf{P}_{f,[3,:]}$ denotes the last row of this camera matrix, $\mathbf{U}$ represents all the image observations $\mathbf{u}_{f,n} \in \mathbb{R}^2$ which are now represented in inhomogeneous coordinates, and $\mathbf{X}_n \in \mathbb{R}^4$ denotes the n-th point in homogeneous coordinates. Obviously, if a point n is not observed in frame f, then the corresponding factor in the above likelihood is left out. In order to get rid of the constraints in Eq. (7.19), the projective

depths in the likelihood could directly be replaced with $\mathbf{P}_{f,[3,:]}\mathbf{X}_n$. However, this would increase the correlation between $\mathbf{P}_f$ and $\mathbf{X}_n$ in the joint posterior considerably, which is problematic for the blocked Gibbs sampling which we are going to use. Moreover, coming up with suitable proposal distributions is more challenging when the constraints are eliminated in this way. We rather propose to use ideas from optimization techniques in order to handle the equality constraint. Note that similar ideas are known as auxiliary variables or data augmentation techniques in the sampling community [DM01]. The likelihood is augmented with hidden variables $\lambda_{f,n}$ which follow some distribution $p(\lambda_{f,n}|\mathbf{P}_f, \mathbf{X}_n)$

$$p(\mathbf{U}, \lambda|\mathbf{P}, \mathbf{X}) = p(\mathbf{U}|\lambda, \mathbf{P}, \mathbf{X})p(\lambda_{f,n}|\mathbf{P}_f, \mathbf{X}_n). \tag{7.20}$$

If $p(\lambda_{f,n}|\mathbf{P}_f, \mathbf{X}_n)$ is a normal distribution, the constraints are penalized by a quadratic loss function, whereas if $p(\lambda_{f,n}|\mathbf{P}_f, \mathbf{X}_n)$ is a Laplace distribution, the constraints are penalized with a L1-norm which is known to be an *exact* penalty function: choosing the standard deviation σ_λ of the Laplace distribution small enough (but not equal to zero), the constraint will be fulfilled with equality [NW06]. In our implementation, we have chosen a Gaussian penalty term for simplicity. By allowing a larger standard deviation σ_λ during early sampling iterations, the correlation between cameras and points is essentially decreased, meaning that Gibbs sampling is more efficient. Analogously to augmented Lagrangian methods, σ_λ is gradually decreased meaning that the constraints are met more precisely. Note that there is one catch with this approach. The penalty of the difference between $\lambda_{f,n}$ and $\mathbf{P}_{f,[3,:]}\mathbf{X}_n$ must depend on the scale of $\mathbf{P}_{f,[3,:]}\mathbf{X}_n$, e.g. for the quadratic case

$$\lambda_{f,n}|\mathbf{P}, \mathbf{X} \sim \text{Normal}\left(\mathbf{P}_{f,[3,:]}\mathbf{X}_n, (\mathbf{P}_{f,[3,:]}\mathbf{X}_n\sigma_\lambda)^2\right).$$

This is because camera matrices and points are projective entities which are only defined up to scale and hence rescaling the camera matrix $\mathbf{P}_f$, say, should not change the likelihood in Eq. (7.20).

Priors on the structure $p(\mathbf{X})$ or on the camera matrices $p(\mathbf{P})$ (such as auto-calibration constraints, for example) can easily be integrated leading to the joint probability

$$p(\mathbf{U}, \lambda, \mathbf{P}, \mathbf{X}) = p(\mathbf{U}|\lambda, \mathbf{P}, \mathbf{X})p(\lambda|\mathbf{P}, \mathbf{X})p(\mathbf{X})p(\mathbf{P}).$$

As the goal of this chapter is to present an as generic as possible sampling strategy, which can serve as a starting guide to more specific SfM instances, no auto-calibration constraints on the cameras are enforced.

Robustness to outliers can be achieved by replacing the Gaussian likelihood in Eq. (7.18) by a more heavy-tailed distribution, e.g. such as a student-t distribution [Bis06]

$$St(\mathbf{U}|\lambda,\mathbf{P},\mathbf{X}) \propto \prod_{f,n}\left[1+\frac{\frac{1}{\sigma^2}\left\|\lambda_{f,n}^{-1}\mathbf{P}_{f,[1:2,:]}\mathbf{X}_n-\mathbf{u}_{f,n}\right\|_2^2}{\nu}\right]^{-\frac{D+\nu}{2}},$$

where the dimensionality $D = 2$ and ν equals the degrees of freedom of the student-t distribution. Other choices are obviously also possible, like e.g. a mixture between a Gaussian and a uniform distribution.

Our goal is now to sample from the posterior

$$p(\mathbf{P},\mathbf{X},\lambda|\mathbf{U}) = \frac{p(\mathbf{U},\lambda,\mathbf{P},\mathbf{X})}{p(\mathbf{U})},$$

which follows a highly non-standard distribution. Hence, we resort to Metropolis-Hastings within blocked Gibbs sampling in order to draw samples. The next section presents this sampling procedure in detail.

7.5 Metropolis-Hastings within Blocked-Gibbs for SfM

As previously mentioned, we will use blocked Gibbs sampling to sample from the posterior $p(\mathbf{P},\mathbf{X},\lambda|\mathbf{U})$. A natural partition of the variables into three blocks is given by sampling from $p(\mathbf{P}|\mathbf{U},\mathbf{X},\lambda)$, $p(\mathbf{X}|\mathbf{U},\mathbf{P},\lambda)$, and $p(\lambda|\mathbf{U},\mathbf{P},\mathbf{X})$. Similar to the Bayesian probabilistic matrix factorization [SM08], each of these conditional probabilities factorizes into probabilities of separate cameras, points, and projective depths, respectively, e.g. for the cameras

$$p(\mathbf{P}|\mathbf{U},\mathbf{X},\lambda) = \prod_f p(\mathbf{P}_f|\mathbf{U},\mathbf{X},\lambda). \tag{7.21}$$

This is highly desirable for nowadays computing platforms since this factorization enables straight-forward parallelization: all the cameras can be sampled independently from each other in parallel, and the same holds true for the points and the projective depths. In contrast to the Bayesian probabilistic matrix factorization, the conditional distributions in our SfM are *not* Gaussian. Indeed, they have a completely non-standard form. Therefore in order to sample from those conditional distributions, we have to resort to some other sampling strategy. We

propose to use a Metropolis-within-Gibbs approach where the samples in a blocked Gibbs sampler are drawn with a Metropolis-Hastings sampling procedure.

The next sections present the proposal distributions which are used for the Metropolis-Hastings samplers. We have made several design decisions to trade of between speed of computing new proposal distributions and their approximation accuracy to the underlying conditional distribution. One could certainly come up with different, maybe more efficient proposals. This is the nice feature about the Metropolis-Hastings based sampling framework: it is highly modular and one can replace and combine several different sampling strategies.

7.5.1 Sampling λ

In order to sample from

$$p(\lambda_{f,n}|\mathbf{u}_{f,n}, \mathbf{P}_f, \mathbf{X}_n) \propto p(\mathbf{u}_{f,n}|\lambda_{f,n}, \mathbf{P}_f, \mathbf{X}_n) p(\lambda_{f,n}|\mathbf{P}_f, \mathbf{X}_n)$$

we use the fact that the penalty term $p(\lambda_{f,n}|\mathbf{P}_f, \mathbf{X}_n)$ can directly be sampled from since it is either a Laplacian or Gaussian distribution. Hence, the proposal distribution is chosen as

$$q(\lambda^*_{f,n}|\lambda^t_{f,n}) = q(\lambda^*_{f,n}) = p(\lambda^*_{f,n}|\mathbf{P}_f, \mathbf{X}_n). \tag{7.22}$$

In the Metropolis-Hastings ratio, this leads to a cancellation of the penalty terms and the ratio becomes a simple likelihood ratio

$$\alpha = \min\left(1, \frac{p(\mathbf{u}_{f,n}|\mathbf{P}^t_f, \mathbf{X}^t_n, \lambda^*_{f,n}) p(\lambda^*_{f,n}|\mathbf{P}_f, \mathbf{X}_n) q(\lambda^t_{f,n})}{p(\mathbf{u}_{f,n}|\mathbf{P}^t_f, \mathbf{X}^t_n, \lambda^t_{f,n}) p(\lambda^t_{f,n}|\mathbf{P}_f, \mathbf{X}_n) q(\lambda^*_{f,n})}\right) \tag{7.23}$$

$$= \min\left(1, \frac{p(\mathbf{u}_{f,n}|\mathbf{P}^t_f, \mathbf{X}^t_n, \lambda^*_{f,n})}{p(\mathbf{u}_{f,n}|\mathbf{P}^t_f, \mathbf{X}^t_n, \lambda^t_{f,n})}\right). \tag{7.24}$$

7.5.2 Sampling P and X

Sampling the cameras or the points is more involved than sampling the projective depths. For both the camera and point proposal distributions, the penalty terms $p(\lambda_{f,n}|\mathbf{P}_f, \mathbf{X}_n)$ are approximated with a Gaussian

$$\tilde{p}(\lambda_{f,n}|\mathbf{P}_f, \mathbf{X}_n) = \text{Normal}\left(\lambda_{f,n}|\mathbf{P}_f\mathbf{X}_n, (\lambda_{f,n}\sigma_\lambda)^2\right).$$

Note that the variance now depends on $\lambda_{f,n}$ rather than on the point and the camera. Furthermore, the likelihood is also approximated with a Gaussian

$$\tilde{p}(\mathbf{U}|\lambda,\mathbf{P},\mathbf{X}) \propto \prod_{f,n} \exp\left(-\frac{1}{2\sigma^2}\left\|\lambda_{f,n}^{-1}\mathbf{P}_{f,[1:2,:]}\mathbf{X}_n - \mathbf{u}_{f,n}\right\|_2^2\right),$$

which might or might not be exact depending on whether Gaussian or student-t distributed reprojection errors are used. For simplicity the priors on the points and cameras are assumed to be Gaussian with mean zero and precision $\mathbf{\Lambda}_{\mathbf{X}_0}$ and $\mathbf{\Lambda}_{\mathbf{P}_0}$. With these approximations, we get

$$p(\mathbf{X}_n|\mathbf{U}_{:,n},\mathbf{P}_{:},\lambda) \tag{7.25}$$

$$\propto p(\mathbf{U}_{:,n}|\lambda_{:,n},\mathbf{P}_{:},\mathbf{X}_n)p(\lambda_{:,n}|\mathbf{P}_{:},\mathbf{X}_n)p(\mathbf{X}_n) \tag{7.26}$$

$$\approx \tilde{p}(\mathbf{U}_{:,n}|\lambda_{:,n},\mathbf{P}_{:},\mathbf{X}_n)\tilde{p}(\lambda_{:,n}|\mathbf{P}_{:},\mathbf{X}_n)p(\mathbf{X}_n). \tag{7.27}$$

Note that Eq. (7.27) defines a Gaussian distribution in $\mathbf{X}_n$ because the logarithm of Eq. (7.27) is a quadratic function of $\mathbf{X}_n$. We could directly sample from this Gaussian. However, we observed that this can result in rather low acceptance rates. The reason is that the Gaussian in Eq. (7.27) defines an approximation to the true conditional distribution. Sampling around the mode of this Gaussian corresponds to computing the 2nd order Taylor expansion at $\mathbf{X}_n^t$ of Eq. (7.27) and taking a full Newton step, and then sampling from this location with the covariance matrix determined by the Hessian of the Taylor expansion. If the Gaussian approximation is not accurate enough, the Metropolis-Hastings acceptance ratio can become really small. We observed that experimentally: the ratio between the proposal distributions $\frac{q(x^t|x^*)}{q(x^*|x^t)}$ sometimes dropped almost to zero because the backward jumping probability $q(x^t|x^*)$ becomes really small. Note that the Gibbs sampling in Bayesian probabilistic matrix factorization corresponds to performing a full Newton step. In contrast to our model however, in Bayesian probabilistic matrix factorization, all conditional probabilities are indeed Gaussians and hence no Metropolis-Hastings step is required as sampling from multivariate Gaussians is straight-forward.

However, the interpretation of taking a Newton step provided us with another idea: instead of taking a full Newton step, rather do a partial step. This is the same idea as the Hessian-based MCMC algorithm by Qi and Minka [QM02]. However in contrast to their approach, we compute the 2nd order Taylor expansion at $\mathbf{X}_n^t$ of Eq. (7.27) rather

than of the original conditional distribution. This slightly simplifies the resulting proposal distributions since the Hessian matrices become independent of the current state $\mathbf{X}_n^t$ which makes the evaluation of the Metropolis-Hastings acceptance ratio slightly faster. Hessian-based MCMC computes the mode of a Gaussian proposal distribution by starting from the current state $\mathbf{X}_n^t$ and then performing a partial Newton step, i.e.

$$\mu_{\mathbf{X}_n} = \mathbf{X}_n^t - \gamma \mathbf{H}^{-1}\mathbf{g}, \tag{7.28}$$

where $\mathbf{H}$ and $\mathbf{g}$ denote the Hessian and the gradient of Eq. (7.27) at $\mathbf{X}_n^t$. The variable γ is the learning rate and controls how much we trust in the local quadratic approximation. As suggested by Qui and Minka, γ is sampled uniformly at random from $[0, 1]$. Algebraic manipulation then leads to the following formulas for precision matrix and mean vector of the Gaussian proposal

$$\mathbf{\Lambda}_{\mathbf{X}_n} = \sum_i \mathbf{P}_i^T \begin{bmatrix} \sigma^2\lambda_{i,n}^2\mathbf{I}_2 & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times 2} & \sigma_\lambda^2\lambda_{i,n}^2 \end{bmatrix}^{-1} \mathbf{P}_i + \mathbf{\Lambda}_{\mathbf{X}_0} \tag{7.29}$$

$$\mu_{\mathbf{X}_n} = (1-\gamma)\mathbf{X}_n^t + \tag{7.30}$$
$$\gamma\mathbf{\Lambda}_{\mathbf{X}_n}^{-1} \sum_i \mathbf{P}_i \begin{bmatrix} \sigma^2\lambda_{i,n}^2\mathbf{I}_2 & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times 2} & \sigma_\lambda^2\lambda_{i,n}^2 \end{bmatrix}^{-1} \lambda_{i,n} \begin{pmatrix} \mathbf{u}_{i,n} \\ 1 \end{pmatrix},$$

where the index i runs over those frames where point n has been observed. The proposal distribution for the Metropolis-Hastings is thus $q(\mathbf{X}_n^*|\mathbf{X}_n^t) = \text{Normal}\left(\mu_{\mathbf{X}_n}, \mathbf{\Lambda}_{\mathbf{X}_n}^{-1}\right)$. For the Metropolis-Hastings ratio, $q(\mathbf{X}_n^t|\mathbf{X}_n^*)$ must be evaluated which can be done efficiently since the precision is independent of the current state and the mean only depends on the state through $(1-\gamma)\mathbf{X}_n^t$. Note that for $\gamma = 1$, the mean $\mu_{\mathbf{X}_n}$ is not depending no the current state $\mathbf{X}_n^t$ and hence we get independent proposals. Furthermore, if $\gamma = 0$ the proposal will be symmetric and the ratio between the proposals hence cancels in the Metropolis-Hastings ratio.

A similar derivation for the cameras leads to the following values for the precision and mean of the first two rows of the camera matrix

$$\mathbf{\Lambda}_{\mathbf{P}_{f[1:2,:]}} = \sum_i \mathbf{X}_i^T (\sigma^2\lambda_{i,n}^2)^{-1}\mathbf{X}_i + \mathbf{\Lambda}_{\mathbf{P}_0} \tag{7.31}$$

$$\mu_{\mathbf{P}_{f,[1:2,:]}} = (1-\gamma)\mathbf{P}_{f,[1:2,:]}^t + \gamma\mathbf{\Lambda}_{\mathbf{P}_{f,[1:2,:]}}^{-1} \sum_i \mathbf{X}_i\sigma^{-2}\lambda_{f,i}^{-1}\mathbf{u}_{f,i}, \tag{7.32}$$

and for the last row

$$\mathbf{\Lambda}_{\mathbf{P}_{f[3,:]}} = \sum_i \mathbf{X}_i^T (\sigma_\lambda^2 \lambda_{i,n}^2)^{-1} \mathbf{X}_i + \mathbf{\Lambda}_{\mathbf{P}_0} \tag{7.33}$$

$$\mu_{\mathbf{P}_{f,[3,:]}} = (1-\gamma)\mathbf{P}_{f,[3,:]}^t + \gamma \mathbf{\Lambda}_{\mathbf{P}_{f,[3,:]}}^{-1} \sum_i \mathbf{X}_i \sigma_\lambda^{-2} \lambda_{f,i}^{-1}.$$

Note that all the precision matrices and mean vectors are 4-by-4 and 4-by-1 respectively. Matrices and vectors of this size can be handled especially efficient by the CPU or GPU.

As mentioned in [QM02], the learning rate could also be included in the Hessian as $\mathbf{H} + \lambda\mathbf{I}$ which obviously is related to the Levenberg-Marquardt method. This has the advantage that by choosing λ sufficiently large, the resulting precision matrix is guaranteed to be positive semi-definite. In our case however, the precision matrices are always positive semi-definite, as can be seen inEq. (7.29) and Eq. (7.31). Therefore along the lines of [QM02], we stick with the simpler learning rate approach according to Eq. (7.28).

These proposal distributions are interesting in so far as they bias the sampling towards directions pointed to by local optimization methods (i.e. Newton-based), while still preserving the randomized properties of a Metropolis-Hastings based sampling scheme and keeping the acceptance ratio sufficiently high.

7.5.3 Robust Proposals

The Gaussian proposal in Eq. (7.29) and Eq. (7.30) is based on all frames in which point n has been seen (similarly the Gaussian proposal in Eq. (7.31) and Eq. (7.32) is based on all points seen at frame f). Therefore, a single outlier can lead to an inappropriate proposal distribution. Similarly as in RANSAC, we propose to avoid this problem by sampling a random subset of all the frames where point n has been seen (for cameras, a random subset is sampled from all the points seen at frame f). Note that this can be interpreted as working with multiple Metropolis-Hastings proposal kernels[2] from which one is randomly selected in each iteration. This approach does not only speed up the

[2] Let F_n denote the number of frames where point n is visible and let k be the cardinality of the random subset. Then there are $\binom{F_n}{k}$ transition kernels for point n.

proposal distribution computation, but is also more robust since reconstructed outlying feature points will not interfere with the proposal distribution computation.

7.6 Evaluation

7.6.1 Experiments

We evaluated our algorithm on the well-known Dinosaur sequence which consists of 36 frames of a turntable sequence of a toy dinosaur, see Chap. 3 for some example images. We completed the sequence such that a ground truth data set with no missing entries resulted. The Markov Chains were always initialized with random values for the cameras and the points, whereas the projective depths were initialized with 1.

In our first experiment we used $N = 524$ points of this sequence. With no noise and no missing entries, an average reprojection error of 0.23 pixels resulted, where the resolution of the input images is 720-by-576. This error is probably due to the penalty term, which does not strictly enforce the projective depths constraints of Eq. (7.19). In a next test, we deleted 50 percent of the observations completely at random. Still, we got an average reprojection error of 0.3. Even with 80 percent missing entries, an average reprojection error of 0.3 pixels was achieved. Adding Gaussian noise with a standard deviation of $\sqrt{2}$ pixel to the image observations increased the average reprojection error to 1.9 pixels for the 80 percent missing data experiment.

All these results were obtained with 1000 sample sweeps which required roughly 1.1 seconds on a Intel Core i7 860 CPU. Note, that this is much faster than other SfM matrix factorization approaches, such as the ones mentioned in Sec. 7.1. Our implementation is parallelized and makes use of all the four cores. However, we did not spend time to optimize the annealing scheme or the iteration count. We leave the tuning of the annealing scheme and convergence check of the Markov chain for future work. The annealing start temperature was set to 100 and the annealing factor was 0.9954. The standard deviation for the measurement variance was set to $\sigma = 0.8$ pixel, whereas the standard deviation for the penalty term was set to $\sigma_\lambda = 0.01$.

These synthetic experiment looked very promising. Hence, we tested the algorithm on the original dinosaur sequence whose observation pat-

tern matrix is strongly band diagonal. Unfortunately, the algorithm no longer behaved that nicely. We could only achieve average reprojection errors of about 2-3 pixels for clean data without any noise added. The following sections provide some possible explanations for these less satisfying results.

7.6.2 Possible Reasons for Breakdown

Two basic problems can be identified in simple MCMC sampling schemes for SfM problems. Firstly, Gibbs and Metropolis-Hastings samplers represent the basic building blocks of the majority of the MCMC sampling methods. Both suffer from some shortcomings. Secondly, the curse of dimensionality makes efficient sampling in high dimensional spaces especially challenging.

Drawbacks of Gibbs and Metropolis-Hastings Samplers

Gibbs samplers are known to converge very slowly if the variables are highly correlated. There are several techniques to decorrelate the variables, e.g. coordinate transformations or the introduction of auxiliary variables. In our SfM application, we have followed the latter approach by introducing auxiliary variables for the projective depths. Moreover, it is well known that blocked Gibbs-sampling is related to block-coordinate optimization methods. By looking at the equations for the mean and the precision in Sec. 7.5.2, a striking similarity to alternating least squares can be seen. Based on the experience with random perturbations in alternating least squares gained in Chap. 3, we have actually hoped that blocked Gibbs sampling in a Markov Chain would perform better. However, as the results with strong band-diagonal sampling patterns have shown, alternately random sampling of cameras and points with auxiliary projective depths variables is insufficient for an efficient MCMC sampling scheme. Fig. 7.1 and Fig. 7.2 visualize common situations we have encountered in these cases and give possible explanations why this is happening. Specifically, we often observed a nearly degenerate, flat reconstruction of the point cloud. Fig. 7.2 explains why this could have been the case. Note that especially the explanation in that figure might also be the reason why alternating least squares does not perform well for bilinear matrix factorizations as investigated in Chap. 3. We have also run some tests with sampling from

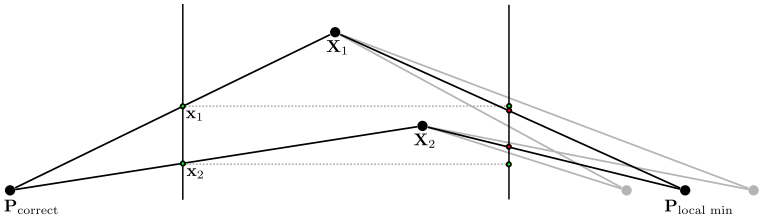
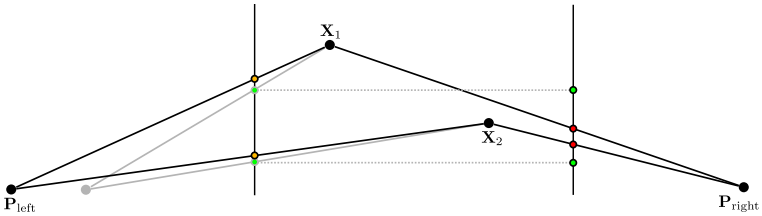


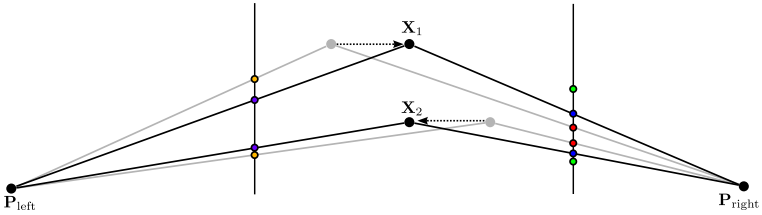
Figure 7.1: Possible explanation for incorrect 3D reconstructions: The camera on the left visualizes the correct position of the camera. Let us assume that the current state of the camera is on the right side. Restricting the possible position of the camera to the horizontal line (e.g. by assuming a symmetrical structure w.r.t. the optical axis), we can see that there is a local minimum for the camera position on the right: moving closer or further away from the structure increases the reprojection error and hence decreases the likelihood of accepting a Metropolis-Hastings proposal. Furthermore, a direct jump to the correct camera position on the left side is unlikely as well due to the low back-ward jumping probability which ensures a reversible Markov chain. Hence, the sampling gets stuck in a local mode.

the inverse depth $\lambda_{f,n}^{-1}$ instead of $\lambda_{f,n}$ and with Log-Normal distributions for $\lambda_{f,n}$, i.e. $\log \lambda_{f,n} = \log(\mathbf{P}_{f,z}\mathbf{X}_n) - \log \epsilon$ with $\log \epsilon \sim \text{Normal}\,(0, \sigma_\lambda)$. This corresponds to $\lambda_{f,n}^{-1} = (\mathbf{P}_{f,z}\mathbf{X}_n)^{-1}\,\epsilon$. However, these changes did not help to improve the results considerably. These observations lead us to the conclusion that instead of alternatively sample for cameras and points one should rather sample the cameras and points jointly, further details about future ideas will be presented in Sec. 7.6.3.

Metropolis-Hastings samplers suffer from the drawback that designing an efficient proposal distribution is highly challenging. The difficulty with choosing an appropriate proposal $q(x^*|x^t)$ is the selection of the scale: if the scale relative to the change of the density around the current state x^t is small, then the Markov chain advances only in very small steps. On the other hand, if the scale is chosen too large, then the acceptance probability may drop almost to zero. Hence, we would like to adaptively set the scale depending on the local neighborhood of the



(a) Correct 3D Structure



(b) Incorrect Flat 3D Structure

Figure 7.2: Possible explanation for flat 3D reconstructions: Fig. 7.2a shows the correct structure with two points and their projections into two cameras. The position of the camera on the left is correct whereas the camera on the right is on the wrong side of the reconstruction (the gray lines denote the correct position of this camera). Holding the cameras fixed, a flat reconstruction helps to improve the L2-reprojection error, as Fig. 7.2b shows. This is because a least-squares objective function tries to even out all the individual reprojection errors and as this figure shows, a flat reconstruction distributes these errors more equally amongst the two cameras.

current state x^t, e.g. by looking at the evolution and acceptance of the last few proposals in previous iterations. However, due to the Markov property, the proposal distribution in a Metropolis-Hastings proposal can only depend on the current state of the chain x^t and *not* on the whole past $x^0, \ldots, x^t$. Adaptively changing the proposal distribution depending on the local neighborhood of the current state results in a time-inhomogeneous Markov-chain whose analysis is much more involved.

Adapting the kernels without paying attention to convergence theorems will invalidate all convergence guarantees and all the nice asymptotic guarantees of Markov chains will be lost.

Sampling in High-Dimensional State Spaces

Exploring large spaces with a random-walk behaviour can take a long time. We would like to quickly reach high-probability regions, for example by combining deterministic non-linear optimization within a Markov chain framework by computing the proposal distribution in a Metropolis-Hastings sampler based on the result of a local optimization. However, this is challenging since backward-jumping probabilities $q(x^t|x^*)$ (which guarantee a reversible Markov chain) in the Metropolis-Hastings ratio can become very small. We have already experienced this issue with the Hessian-based MCMC algorithm where the learning rate needs to be introduced in order counteract the otherwise resulting low acceptance ratios.

Adaptive Direction Samplers [GRG94] allow the selection of adaptive search directions. Adaptive direction sampling is based on a population of states, i.e. at each point in time there are multiple "current" states. Choose randomly a candidate $\mathbf{x}_c^t$ and sample $\mathbf{x}_c^{t+1} = \mathbf{x}_c^t + r(\mathbf{v}^t + u^t\mathbf{x}_c^t)$ where $\mathbf{v}^t \in \mathbb{R}^n$ and $u^t \in \mathbb{R}$ are any functions of the current population excluding the candidate $\mathbf{x}_c^t$. As shown in [GRG94], the resulting distribution needs to be adjusted by the determinant of the Jacobian of the coordinate transformation. This leads to the adjusted distribution $p(\mathbf{x}_c^{t+t})|1 + ru^t|^{n-1}$. The snooker algorithm (by choosing $u^t = -1$ and $\mathbf{v}^t = \mathbf{x}_a^t$ where $\mathbf{x}_a^t$ is a random point from the population excluding the candidate), the Gibbs sampler (by choosing $u^t = 0$ and a random axis aligned coordinate direction $\mathbf{v}^t = \mathbf{e}_i$), and many more samplers are specific instances of this general framework. In general, if $u^t = 0$ the correction factor disappears and simpler sampling schemes result.

However, all the Metropolis-Hastings samplers for Markov chains involving proposals based on local optimization require a correction factor in the Metropolis-Hastings ratio based on the determinant of the Jacobian of a coordinate transformation. This correction factor can become rather small, for example if a large step towards a local minimum is taken. This results in very low acceptance probabilities. Note that the Adaptive Direction Sampler (and its related samplers) keep track of a population of states. This resembles particle filters, a

sequential Monte Carlo procedure well known in the engineering and computer vision community. Particle filters are generally used for online filtering applications where a target density is propagated through time by multiple weighted particles which are updated whenever new measurements become available. Such sequential Monte Carlo methods are fundamentally different in that they do not necessarily build upon Markov chains and they are usually not used for global, randomized optimization. As mentioned in the next section however, recent work tries to apply adaptive sequential Monte Carlo techniques to global optimization problems, surely an interesting approach to keep in mind.

7.6.3 Future Work

Based on the previous discussion, we can draw the following conclusion. Blocked Gibbs sampling for rank-constrained problems with matrix factors as blocks only works well if the measurement pattern is well-behaved, e.g. like in the collaborative filtering application in the Bayesian probabilistic matrix factorization [SM08]. For more challenging measurement operators, such as the strong band diagonal pattern in structure-from-motion applications, the performance of blocked Gibbs sampling decreases considerably, similarly to alternating least squares approaches. Local optimization is crucial in order to efficiently explore large state spaces. Unfortunately, the backward-jumping probabilities in the Metropolis-Hastings ratio which ensure a reversible Markov Chain lead to low acceptance probabilities.

Hence, as future work, we suggest using a different blocking in Gibbs sampling. In the structure-from-motion problem, instead of separating cameras and points into separate blocks, the blocking should rather slice a submatrix out of the full observation matrix. For example, a block could consists of several successive frames and several points. The sampling is then performed over the joint distribution of the cameras and points. We have seen that the Wiberg algorithm has a high probability to converge to the global optimum of a bilinear matrix factorization problem. A combination between such local optimization within a sampling framework seems to offer high potential. However, if we restrict ourselves to Markov chains, then care must be taken not to violate any conditions ensuring asymptotic convergence, e.g. by properly adapting the Metropolis-Hastings ratio with a correction factor. We are unaware of any Markov chain method which can still achieve reasonable

acceptance ratios under these circumstances.

As mentioned previously sequential Monte Carlo methods are more flexible than Markov chain methods since they are inherently based on a population of states. Note that sequential Monte Carlo methods generally also require a proposal distribution and hence offer the advantages of a hypothesize-and-verify framework as well. However, in contrast to Markov chain transition kernels, the conditions on the evolution of the samples in the population of states are less restrictive. We refer to [CGM07; Jas+08] for population-based, sequential, and adaptive Monte Carlo methods.

8 The Subspace Intersection Problem

Rank-constraints have a particularly rich algebraic structure. We have already seen in Chap. 4 how multiple low-dimensional subspaces can interact with each other and can give rise to a low-rank tensor. The underlying algebraic structure however is much deeper than the structure used in that chapter. Obviously, rank-constraints are actually multilinear polynomial constraints in the matrix or tensor entries. This gives rise to *determinantal ideals*, a specific topic mostly considered in algebraic geometry.

In contrast to the previous chapter, this chapter completely ignores any probabilistic interpretation and solely focuses on the algebra beneath low-dimensional subspaces models. Specifically, this chapter introduces a novel and very general problem termed the *subspace intersection problem*. The goal is to infer an unknown subspace from a set of given subspaces which are known to each intersect this unknown subspace. The intersections however are unknown, as well. As an example, an intuitive instance of such a subspace intersection problem is given by 4 lines in 3D space where an unknown additional line has to be determined such that the known 4 lines intersect with it. A general algebraic formulation based on the Laplace expansion for determinants is presented enabling the computation of the unknown subspace in closed-form, given sufficiently many constraining known subspaces. Moreover, an efficient numerical scheme based on a partial reduced row-echelon form is introduced.

The theory and algorithms for subspace intersection problems are showcased on a structure-from-sound problem. The theory enables the computation of an unknown synchronization of sound sources in closed-form. Furthermore, improving upon previous work, new cases such as missing observations can be handled in closed-form.

While developing our theory for subspace intersections, we realized that an almost identical problem has already been considered by Hermann Schubert in the nineteenth century [Sch79]. His findings are nowadays known as as *Schubert calculus* and laid the foundation for various branches in algebraic geometry. A detailed treatment of Schu-

bert calculus in relation to our subspace intersection problem is given in Sec. 8.6.

8.1 Introduction

Let us assume there are multiple given subspaces which are known to intersect with another unknown subspace. The goal of the *subspace intersection problem* is to compute this unknown subspace from the known ones. An intuitive example is given by four lines in 3D which all intersect another unknown line in 4 different unknown points, a problem considered also by Teller and Hohmeyer [TH99].

More generally, let us assume there are m given subspaces $\mathcal{A}_i$ of dimension d_i for $i \in [m]$ embedded in ambient n-dimensional Euclidean space $\mathbb{R}^n$. Subspace i is spanned by the columns of the matrix $\mathbf{A}_i \in \mathbb{R}^{n \times d_i}$, i.e. the columns of $\mathbf{A}_i$ are a basis for the i-th subspace. Let the unknown subspace $\mathcal{B}$ have known dimensionality D and be spanned by the columns of $\mathbf{B} \in \mathbb{R}^{n \times D}$. Throughout this chapter, we assume that the intersection between the i-th subspace $\mathcal{A}_i$ and the unknown subspace $\mathcal{B}$ is one dimensional, i.e. $\dim(\mathcal{A}_i \cap \mathcal{B}) = 1$. With this notation in place, the intersection constraint for each subspace i can be written as

$$\begin{bmatrix}\mathbf{A}_i & \mathbf{B}\end{bmatrix}_{n \times d_i + D} \begin{pmatrix}\mathbf{x}_i \\ \mathbf{y}_i\end{pmatrix} = \mathbf{0}_{n \times 1}, \tag{8.1}$$

with the unknown coordinate vectors $\mathbf{x}_i \in \mathbb{R}^{d_i \times 1}$ and $\mathbf{y}_i \in \mathbb{R}^{D \times 1}$. The subspace intersection problem tries to infer the unknown subspace $\mathcal{B}$ from m given subspaces spanned by the columns of $\mathbf{A}_i$.

Note that the assumption of a one-dimensional intersection between the subspace $\mathcal{A}_i$ and $\mathcal{B}$ does not impose any restriction in general. Indeed, higher dimensional intersections can be reduced to multiple one dimensional intersections. For example, let $\dim(\mathcal{A} \cap \mathcal{B}) = r > 1$ with the columns of $\mathbf{A} \in \mathbb{R}^{n \times d}$ and $\mathbf{B} \in \mathbb{R}^{n \times D}$ spanning the subspaces $\mathcal{A}$ and $\mathcal{B}$, respectively. For $i \in [r]$, randomly choose a matrix $\mathbf{H}_i \in \mathbb{R}^{d \times d - r + 1}$ e.g. with normally distributed entries and consider $\mathbf{A}_i = \mathbf{A}\mathbf{H}_i \in \mathbb{R}^{n \times d - r + 1}$. Due to the random $\mathbf{H}_i$, the intersection between $\operatorname{span}(\mathbf{A}_i = \mathbf{A}\mathbf{H}_i)$ and $\mathcal{B}$ is in general one-dimensional. Therefore, a r-dimensional intersection can be reduced to r one-dimensional intersections. The algebra beneath higher-dimensional intersections will however be more involved due

to the product structure $\mathbf{A}_i = \mathbf{A}\mathbf{H}_i$ and due to the induced linear dependence between the subspaces $\mathcal{A}_i$.

This chapter analyses this subspace intersection problem from an algebraic point of view leading to two novel algorithms which provide the solution for the unknown subspace $\mathcal{B}$ in closed-form by solving a linear system, given sufficiently many constraining subspaces $\mathcal{A}_i$ or their representations as matrices $\mathbf{A}_i$. Moreover, missing observation or entries in $\mathbf{A}_i$, a situation often arising in practice, can be handled in closed-form as well. The developed theory and algorithms are exemplified on a structure-from-sound problem where the goal is to compute unknown synchronizations of sound sources which is the initial step for computing microphone and sound locations.

8.2 Notation and Preliminaries

8.2.1 Notation

The determinant of a matrix $\mathbf{A}$ is denoted as $|\mathbf{A}|$. A minor of a matrix $\mathbf{A}$ is the determinant of the matrix which results by slicing a submatrix out of the matrix $\mathbf{A}$. Hence, for given row-index set I and column index set J[1], the minor $|\mathbf{A}|_{I\times J}$ equals the determinant of the submatrix $\mathbf{A}_{I,J}$ which results by selecting rows $i \in I$ and columns $j \in J$. The number of inversions between two sets I and J is needed in Sec. 8.2.4. This number is defined as $\operatorname{inv}(I|J) = |\{(i,j) \in I \times J | i > j\}|$. A disjoint partition $A \cup B = C$ of a set C with $A \cap B = \varnothing$ is written as $A \sqcup B = C$. The standard notation $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is used for the binomial coefficients.

8.2.2 Operations of Subspaces

Multiple subspaces $\mathcal{A}_i \subset \mathbb{R}^n$ for $i \in [m]$ can be combined in various ways yielding new constructs. The intersection between subspaces $\bigcap_{i=1}^m \mathcal{A}_i$ is defined as the set $\{\mathbf{x} \in \mathbb{R}^n | \forall i \in [m] : \mathbf{x} \in \mathcal{A}_i\}$. Note that this set is again a subspace. This is in contrast to the union of subspaces $\bigcup_{i=1}^m \mathcal{A}_i = \{\mathbf{x} \in \mathbb{R}^n | \exists i \in [m] : \mathbf{x} \in \mathcal{A}_i\}$ which is not necessarily again a subspaces. Think of the union of two lines going through the origin in $\mathbb{R}^3$. This union consists of all the points on the two lines. However, points in the span of these two lines are i.g. not in their union. If one

[1] Strictly speaking, I and J are not sets but rather index vectors since the order of their elements determines the sign of the determinant.

intends to construct a larger subspace consisting of the joint span of the two basis of two subspaces then the sum of subspaces has to be considered. The sum of subspaces is denoted as $\mathcal{A}_1 + \mathcal{A}_2 + \ldots + \mathcal{A}_m$ and is defined as $\mathcal{A}_1 + \ldots \mathcal{A}_m = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \sum_{i=1}^m \mathbf{a}_i \text{ where } \mathbf{a}_i \in \mathcal{A}_i\}$.

Let $\mathbf{a}_i \in \mathcal{A}_i$, then subspaces $\mathcal{A}_i$ for $i \in [m]$ are called independent if $\sum_{i=1}^m \mathbf{a}_i = \mathbf{0}$ implies $\forall i \in [m] : \mathbf{a}_i = \mathbf{0}$. The notion of independent subspaces generalizes the notion of independent vectors which can be interpreted as independent 1D subspaces. The sum of independent subspaces $\mathcal{A}_i$ with $i \in [m]$ is called the direct sum and is denoted as $\mathcal{A}_1 \oplus \mathcal{A}_2 \oplus \ldots \oplus \mathcal{A}_m$. An important fact about direct sums is that any vector $\mathbf{x} \in \bigoplus_{i=1}^m \mathcal{A}_i$ can be uniquely decomposed into $\mathbf{x} = \sum_{i=1}^m \mathbf{a}_i$ with $\mathbf{a}_i \in \mathcal{A}_i$. Furthermore, for direct sums it holds that $\dim\left(\bigoplus_{i=1}^m \mathcal{A}_i\right) = \sum_{i=1}^m d_i$ where $d_i = \dim(\mathcal{A}_i)$. For sums of subspaces, this equality no longer holds but we rather have $\dim\left(+_{i=1}^m \mathcal{A}_i\right) \leq \sum_{i=1}^m d_i$ due to the possible linear dependence between the subspaces $\mathcal{A}_i$.

8.2.3 Plucker Coordinates

When solving problems with subspaces as primary entities of interest, a natural question is how to represent a subspace. Algebraically, a subspace of dimension D embedded in $\mathbb{R}^n$ is a point on the Grassmannian manifold $G_{D,n}$. A well-known method to specify a point on this Grassmannian manifold is to find a basis $\{\mathbf{a}_1, \ldots, \mathbf{a}_D\}$ for the subspace and write the basis vectors as columns in a matrix $\mathbf{A} = [\Rightarrow_i \mathbf{a}_i]$: the subspace is then given by the column span of this matrix. Unfortunately, this representation is far from unique which can be a major drawback depending on the algorithms which are applied. Another representation for subspaces is the projection matrix $\mathbb{P}_\mathbf{A} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}$ onto this subspace. This representation is unique but uses many more variables than the number of inherent degrees of freedom since the projection matrix has n^2 entries whereas the inherent degrees of freedom of a D-dimensional subspace in $\mathbb{R}^n$ equals $D(n - D)$. A unique representation of a subspace with minimal number of parameters is given by the reduced row-echelon form of a matrix. Let the subspace again be spanned by the linearly independent columns of matrix $\mathbf{A} \in \mathbb{R}^{n \times D}$. Any regular transformation $\mathbf{H} \in \mathbb{R}^{D \times D}$ of the columns does not change the column subspace and hence we can choose the canonical representation

which maps the first $D \times D$ block of $\mathbf{A}$ to the identity

$$\begin{bmatrix} \mathbf{I}_D \\ \mathbf{B} \end{bmatrix} = \mathbf{AH}, \tag{8.2}$$

where $\mathbf{B} \in \mathbb{R}^{n-D \times D}$. We will encounter the usefulness of this reduced row-echelon form again in Sec. 8.5.

A slightly more algebraic representation for subspaces is given by the Plucker coordinates (also known as the Grassmannian coordinates). Plucker coordinates are widely used for the representation of lines in 3D. The vector of Plucker coordinates for lines in 3D has six elements and is given by $\left(\mathbf{x}^T - \mathbf{y}^T, (\mathbf{x} \times \mathbf{y})^T\right)^T$ where $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{y} \in \mathbb{R}^3$ are two points on the line. Plucker coordinates are not minimal as can easily be seen in the line example: due to the property of the cross-product, the Plucker coordinates $\mathbf{p} \in \mathbb{R}^6$ for a line in 3D always fulfill $\langle \mathbf{p}_{1:3}, \mathbf{p}_{4:6} \rangle = 0$ where $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T\mathbf{b}$ denotes the standard inner product. The concept of Plucker coordinates can be generalized to subspaces of arbitrary dimensions. Specifically, the Plucker coordinates of a subspace on the Grassmannian $G_{D,n}$ are defined as the $\binom{n}{D}$ maximal minors of a matrix $\mathbf{A} \in \mathbb{R}^{n \times D}$ whose columns span the subspace. Since the choice of such matrix is not unique, the Plucker coordinates are projective entities, i.e. they are only defined up to a scalar multiplication: Choosing a different basis for the subspace $\mathbf{AH}$ with $\mathbf{H} \in \mathbb{R}^{D \times D}$ leads to a scaling of the Plucker coordinates by $|\mathbf{H}|$. In order to map the line example to this more general notion of Plucker coordinates, the two points on the line need to be represented in homogeneous coordinates (and are thus non-zero vectors in $\mathbb{R}^4$) which are then written as columns of a 4-by-2 matrix whose $\binom{4}{2} = 6$ maximal minors are the Plucker coordinates.

The Plucker coordinates are not independent since as maximal minors they need to fulfill a set of quadratic polynomials. These quadratic constraints are known as the Plucker relations. The quadratic Plucker relations are given by the sum of pairwise products of certain subdeterminants. Note that here, we represent a subspace as the row-span of a $m \times n$ matrix which is the more commonly used representation in algebraic geometry. Specifically, let $\mathbf{i} \in [n]^{m-1}$ and $\mathbf{j} \in [n]^{m+1}$ be two column index vectors. The quadratic Plucker relations can be derived with the Laplace expansion (see the next section) and read like

$$\sum_{k=1}^{m+1} (-1)^k \, |\mathbf{A}|_{:\times[\mathbf{i},\mathbf{j}_k]} \, |\mathbf{A}|_{:\times[\mathbf{j}_1,\ldots,\mathbf{j}_{k-1},\mathbf{j}_{k+1},\ldots,\mathbf{j}_{m+1}]} = 0. \tag{8.3}$$

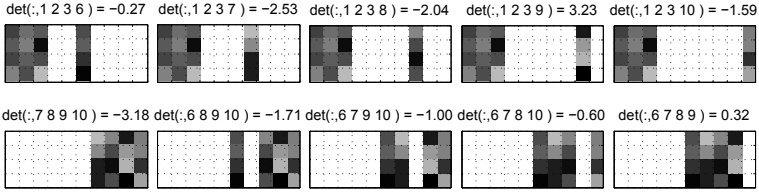


Figure 8.1: One quadratic Plucker relation for a 4×10 matrix: The column subsets are chosen as $\mathbf{i} = (1, 2, 3)$ and $\mathbf{j} = (6, 7, 8, 9, 10)$. The first row visualizes $|\mathbf{A}|_{:\times[\mathbf{i},\mathbf{j}_k]}$ whereas the second shows $|\mathbf{A}|_{:\times[\mathbf{j}_1,\ldots,\mathbf{j}_{k-1},\mathbf{j}_{k+1},\ldots,\mathbf{j}_{m+1}]}$. As this numerical example shows, even though none of the resulting 4×4 subdeterminants equals zero, the sum of their pairwise products with alternating sign equals zero and hence fulfills one of the quadratic Plucker relations.

One of the Plucker relations for a four dimensional subspace in ten dimensional space is visualized in Fig. 8.1.

The Plucker relations can be generalized to summations over more than just one column exchange. However, the resulting polynomials are linear combinations of the ones with just one single column exchange and hence, we stick with this definition. Interestingly, the Plucker relations completely define the Grassmannian in the sense that there is a bijection between the Grassmannian and $\binom{n}{D}$-dimensional vectors which fulfill all the Plucker relations. Hence, any set of $\binom{n}{D}$ coordinates which fulfill all the Plucker relations define a valid subspace: each of the $\binom{n}{D}$ coordinates equals the minor of some matrix $\mathbf{A} \in \mathbb{R}^{n \times D}$ whose columns span this subspace. On the other hand, a vector which does not comply with all the Plucker relations does not specify a valid subspace. We refer to [KL72; Mag07] for more details.

8.2.4 Laplace Expansion for Determinants

The Laplace expansion for determinants is needed for the derivations in Sec. 8.3. The Laplace expansion is basically a generalization of the well-known co-factor expansion method for computing a determinant of a matrix. Let $\mathbf{X} \in \mathbb{R}^{m \times m}$ and fix a partition of the columns $J' \sqcup J'' = [m]$. Then the determinant of $\mathbf{X}$ can be computed as a sum over all possible

disjoint row partitions $I' \sqcup I'' = [m]$ of appropriate size $|I'| = |J'|$ and $|I''| = |J''|$

$$|\mathbf{X}| = \sum_{I' \sqcup I'' = [m]} (-1)^{\mathrm{inv}(I'|I'') + \mathrm{inv}(J'|J'')} |\mathbf{X}|_{I' \times J'} |\mathbf{X}|_{I'' \times J''} . \tag{8.4}$$

In Sec. 8.3 the Laplace expansion needs to be applied to a minor $|\mathbf{X}|_{I \times J}$ of a matrix. Fix again a disjoint column partition $J' \sqcup J'' = J$, then the Laplace expansion for minors reads

$$|\mathbf{X}|_{I \times J} = \sum_{I' \sqcup I'' = I} (-1)^{\mathrm{inv}(I'|I'') + \mathrm{inv}(J'|J'')} |\mathbf{X}|_{I' \times J'} |\mathbf{X}|_{I'' \times J''} . \tag{8.5}$$

8.3 An Algebraic View on the Subspace Intersection Problem

Coming back to the subspace intersection problem, the null-vector on the right-hand side of Eq. (8.1) implies that the matrix $[\mathbf{A}_i, \mathbf{B}]_{n \times d_i + D}$ must be singular. This in turn implies that all the $d_i + D$-by-$d_i + D$ minors $|[\mathbf{A}_i, \mathbf{B}]|_{I \times [d_i + D]}$, where a subset of rows $i \in I$ with $|I| = d_i + D$ has been selected, must vanish. The Laplace expansion can be used to represent such a minor. Specifically, a column partition $J' = [d_i]$ and $J'' = \{d_i + 1, \ldots, d_i + D\}$ is chosen such that the terms due to entries of $\mathbf{A}_i$ are maximally separated from $\mathbf{B}$ in the following way

$$|[\mathbf{A}_i, \mathbf{B}]|_{I \times [d_i + D]} = \sum_{I' \sqcup I'' = I} (-1)^{\mathrm{inv}(I'|I'') + \mathrm{inv}(J'|J'')} |\mathbf{A}_i|_{I' \times [d_i]} |\mathbf{B}|_{I'' \times [D]} . \tag{8.6}$$

Note that by this choice, all the columns of $\mathbf{A}_i$ and all the columns of $\mathbf{B}$ will be selected.

The crucial point in this representation is that the minors $|\mathbf{A}_i|_{I' \times [d_i]}$ are all known and hence the condition that all the $d_i + D$-by-$d_i + D$ minors of $[\mathbf{A}_i, \mathbf{B}]_{n \times d_i + D}$ must vanish results in linear equations in the minors $|\mathbf{B}|_{I'' \times [D]}$ of $\mathbf{B}$ with coefficients determined by minors $|\mathbf{A}_i|_{I' \times [d_i]}$ of $\mathbf{A}_i$

$$\begin{aligned} |[\mathbf{A}_i, \mathbf{B}]|_{I \times [d_i + D]} = \sum_{I' \sqcup I'' = I} (-1)^{\mathrm{inv}(I'|I'') + \mathrm{inv}(J'|J'')} |\mathbf{A}_i|_{I' \times [d_i]} |\mathbf{B}|_{I'' \times [D]} = 0 \\ \Rightarrow \quad \mathbf{M}_i \mathbf{b} = \mathbf{0}. \end{aligned} \tag{8.7}$$

Here matrix $\mathbf{M}_i$ is a $\binom{n}{d_i+D}$-by-$\binom{n}{D}$ matrix where each row represents one constraint due to choosing a certain row-subset I with $|I| = d_i + D$. Such a row holds the coefficients $(-1)^{\text{inv}(I'|I'')+\text{inv}(J'|J'')} |\mathbf{A}_i|_{I'\times[d_i]}$ at the appropriate index position. The vector $\mathbf{b}$ stores the $\binom{n}{D}$ minors $|\mathbf{B}|_{I''\times[D]}$ of $\mathbf{B}$. As we have seen in Sec. 8.2.3, the vector $\mathbf{b}$ actually holds the Plucker coordinates of the unknown subspace spanned by the matrix $\mathbf{B}$.

The homogeneous linear equations in Eq. (8.7) due to subspaces $\mathbf{A}_i$ for $i \in [m]$ can be stacked in one linear system with system matrix $\mathbf{M} = [\Downarrow_{i=1}^{m} \mathbf{M}_i]$. Note that not all the linear equations in Eq. (8.7) are linearly independent. Not even the linear equations between two subspaces i and j with $i \neq j$ are completely linearly independent. Nevertheless, given sufficiently many known subspaces $\mathbf{A}_i$, this system matrix has exactly a one-dimensional nullspace which equals the Plucker coordinates of the subspace $\mathbf{B}$. Such a nullspace can robustly be computed with the singular value decomposition of matrix $\mathbf{M}$ (or an iterative variant of it since we are only interested in the nullspace [Gv96]). This results in a linear method to compute the originally unknown subspace $\mathbf{B}$ from multiple known subspaces $\mathbf{A}_i$. Hence, the subspace intersection problem as defined in Sec. 8.1 can be solved linearly in this way.

If there are not sufficiently many subspaces $\mathbf{A}_i$ at hand, the quadratic Plucker relations can be used to infer the unknown subspace $\mathbf{B}$. First, the nullspace $\mathbf{N}$ of the system matrix $\mathbf{M}$ is computed, i.e. $\mathbf{MN} = \mathbf{0}$. Let d_N denote the dimensionality of this nullspace. Then the solution can be parameterised as $\mathbf{Nz} = \mathbf{b}$. Plugging this parameterization into the Plucker relations results in a multivariate quadratic polynomial system in the unknowns $\mathbf{z} \in \mathbb{R}^{d_N \times 1}$ which can be solved by any method of choice, e.g. such as a Grobner basis approach [CLO97]. Note that this approach is feasible as long as d_N is rather small. Otherwise, the resulting polynomial system will become computationally intractable.

8.4 A First Example: Intersecting Lines in 3D

As a concrete example in order to clarify the derivations, let us consider the problem of m lines in 3D intersecting another unknown line in 3D. As previously described, a line in 3D is actually an element of $G_{2,4}$ and as such can be represented as the column span of a 4-by-2 matrix. Hence, we are looking for a line represented as a $D = 2$ dimensional

subspace spanned by a $\mathbf{B} \in \mathbb{R}^{4\times 2}$. As such, there will be $\binom{4}{2} = 6$ Plucker coordinates. Each line $i \in [m]$ defines a $d_i = 2$ dimensional subspace spanned by the columns of $\mathbf{A}_i \in \mathbb{R}^{4\times 2}$ and leads to exactly one minor constraint

$$\left|[\mathbf{A}_i, \mathbf{B}]_{4\times 4}\right|_{[4]\times[4]} = 0. \tag{8.8}$$

Given $m \geq 5$ lines, the system matrix $\mathbf{M} \in \mathbb{R}^{m\times 6}$ will have exactly a one-dimensional nullspace which equals the Plucker coordinates of $\mathbf{B}$.

For $m = 4$ lines, there is a two-dimensional nullspace $\mathbf{N} \in \mathbb{R}^{6\times 2}$ of $\mathbf{M}$. Plugging the parametrization $\mathbf{b} = \mathbf{N}\mathbf{z}$ with $\mathbf{z} \in \mathbb{R}^2$ into the single Plucker relation for $G_{2,4}$ results in a quadratic constraint which can readily be solved (note that $\mathbf{z}_1$ can be fixed to 1 since the Plucker coordinates are a projective entity). For this line intersection problem, this approach is equivalent to the minimal method proposed by Teller and Hohmeyer [TH99] where an algorithm has been derived specifically tailored to this line intersection problem in 3D. However, our derivation is more general and reveals more clearly the connection to Plucker coordinates of arbitrary subspaces and can thus be applied more broadly. For example, our method also extends to cases where not all the d_i are equal.

8.5 Alternative Algorithm

The total number of linear equations due to all the $d_i + D$-by-$d_i + D$ minor constraints equals $\sum_{i=1}^{m} \binom{n}{d_i+D}$ and the number of unknown Plucker coordinates is $\binom{n}{D}$. These numbers can become large rather quickly with increasing n, D, and d_i. This section therefore presents a numerically motivated approach to solve subspace intersection problems. The advantage is its efficiency, on the downside is the loss of the relationship to Plucker coordinates.

Starting again from Eq. (8.1), all these constraints can be collected in a single matrix equation

$$\left[\Rightarrow_{i=1}^{m} \mathbf{A}_i\right]_{n\times\sum_i d_i} \left[\searrow_{i=1}^{m} \mathbf{x}_i\right]_{\sum_i d_i \times m} = \mathbf{B}_{n\times D} \left[\Rightarrow_{i=1}^{m} \mathbf{y}_i\right]_{D\times m}. \tag{8.9}$$

This equation is basically a low-rank constrained linear problem in the unknowns $\mathbf{x}_i \in \mathbb{R}^{d_i}$ and $\mathbf{C} = \mathbf{B}\left[\Rightarrow_i \mathbf{y}_i\right] \in \mathbb{R}^{n\times m}$

$$\left[\Rightarrow_{i=1}^{m} \mathbf{A}_i\right]_{n\times\sum_i d_i} \left[\searrow_{i=1}^{m} \mathbf{x}_i\right]_{\sum_i d_i \times m} = \mathbf{C} \quad \text{s.t} \quad \operatorname{rank}(\mathbf{C}) = D. \tag{8.10}$$

Recently, such rank constrained linear problems have gained much interest. Specifically as seen in Chap. 6, convex relaxations of the rank function with the trace norm have been proposed resulting in a semidefinite programming problem [FHB01; AL11]. Convex relaxations of this kind can lead to impressive results but can as well fail catastrophically. Based on our experience, the success strongly depends on the problem structure at hand and on the input data. Especially for strict algebraically constrained problems, i.e. for problems where the solution must have *exactly* rank D and not just approximatively, convex relaxations often fail. For example, in the sound synchronization problem considered in Sec. 8.7 where the solution must have rank 5, a convex relaxation with trace norm failed miserably. Even though convex relaxations of rank-constrained linear problems are an interesting line of future research, in this chapter we focus on techniques which are guaranteed to give the correct solution (barring noise or outliers).

Fortunately, the low-rank problem in Eq. (8.9) has a very unique structure which can be exploited in order to find a closed-form solution. Choose a subset $I \subset [m]$ with cardinality $|I| = D$ and consider a matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ which reduces the matrix $\left[\Rightarrow_{i \in I} \mathbf{A}_i\right]$ to a partial reduced row-echelon form

$$\mathbf{Q}\left[\Rightarrow_{i \in I} \mathbf{A}_i, \Rightarrow_{i \notin I} \mathbf{A}_i\right] = \begin{bmatrix} \mathbf{I}_{\sum_{i \in I} d_i} & \mathbf{F} \\ \mathbf{0}_{n - \sum_{i \in I} d_i \times \sum_{i \in I} d_i} & \mathbf{G} \end{bmatrix}, \tag{8.11}$$

with matrix blocks $\mathbf{F} \in \mathbb{R}^{\sum_{i \in I} d_i \times \sum_{i \notin I} d_i}$ and $\mathbf{G} \in \mathbb{R}^{n - \sum_{i \in I} d_i \times \sum_{i \notin I} d_i}$. In this step, we have assumed that the subspaces $\mathcal{A}_i$ for $i \in I$ are independent. This can be a restrictive assumption, e.g. for the line intersection example the subspaces $\mathcal{A}_i \in G_{2,4}$ are definitely not independent for $m > 2$.

Such a matrix $\mathbf{Q}$ can for example be computed with $\hat{\mathbf{Q}} = \left[\Rightarrow_{i \in I} \mathbf{A}_i\right]^\dagger \in \mathbb{R}^{\sum_{i \in I} d_i \times n}$ and $\mathbf{Q} = \begin{bmatrix} \hat{\mathbf{Q}}^T, \left[\hat{\mathbf{Q}}^T\right]_\perp \end{bmatrix}^T$. Due to the block-diagonal structure of $\left[\searrow_{i=1}^m \mathbf{x}_i\right]$, this change of basis of the rows of $\left[\Rightarrow_{i=1}^m \mathbf{A}_i\right]$ leads to

$$\mathbf{Q}\left[\Rightarrow_{i \in I} \mathbf{A}_i, \Rightarrow_{i \notin I} \mathbf{A}_i\right] \begin{bmatrix} \left[\searrow_{i \in I} \mathbf{x}_i\right] & \mathbf{0} \\ \mathbf{0} & \left[\searrow_{i \notin I} \mathbf{x}_i\right] \end{bmatrix} = \tag{8.12}$$

$$\begin{bmatrix} \begin{bmatrix} \left[\searrow_{i \in I} \mathbf{x}_i\right]_{\sum_{i \in I} d_i \times D} \\ \mathbf{0}_{n - \sum_{i \in I} d_i \times \sum_{i \in I} d_i} \end{bmatrix} & \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \end{bmatrix} \left[\searrow_{i \notin I} \mathbf{x}_i\right] \end{bmatrix} = \mathbf{Q}\mathbf{B}\left[\Rightarrow_{i \in I} \mathbf{y}_i, \Rightarrow_{i \notin I} \mathbf{y}_i\right]. \tag{8.13}$$

Since the left-most block is already of rank D and since the lower part of this block equals the zero matrix, we can deduce that in order for the result being of rank D it must hold $\mathbf{G}\left[\searrow_{i \notin I} \mathbf{x}_i\right] = \mathbf{0}_{n-\sum_{i \in I} d_i \times m-D}$. Furthermore, again due to the block-diagonal structure of the second factor, this equation splits into $m - D$ separate equations

$$\mathbf{G}_i \mathbf{x}_i = \mathbf{0}_{n-\sum_{i \in I} d_i \times 1}, \tag{8.14}$$

for $i \notin I$ and with $\mathbf{G} = \left[\Rightarrow_{i \notin I} \mathbf{G}_i\right]$ where each block $\mathbf{G}_i \in \mathbb{R}^{n-\sum_{j \in I} d_j \times d_i}$. This enables the computation of $\mathbf{x}_i$ for $i \notin I$ in closed-form. By choosing different subsets I, all the $\mathbf{x}_i$ can be computed in this way. In order to maximize the number of equations in Eq. (8.14), the subsets I should be chosen such that $\sum_{i \in I} d_i$ is minimized. Sec. 8.7 shows an example of how this approach works in practice.

8.6 Determinantal Ideals

As seen in Sec. 8.2.3, Plucker coordinates are inherently linked to maximal minors of matrices. Similarly, a rank-D constraint like in Eq. (8.10) actually encodes multiple constraints: all the $D + 1 \times D + 1$ minors must vanish. Minors are multilinear polynomials in the matrix entries. A collection of polynomials give rise to an ideal whose zero-set is called a variety [CLO97]. The determinantal ideals generated by rank constraints have a particularly rich algebraic structure and several of their properties will be discussed in the following paragraphs.

8.6.1 Schubert Calculus

Hermann Schubert was amongst the first who studied specific determinantal ideals in detail at the end of the nineteenth century [Sch79]. He was mostly interested in enumerative problems: finding the number of geometric primitives like points and lines which satisfy certain geometric conditions. Obviously, the line intersection problem of Sec. 8.4 is a representative instance for these enumerative problems. In the seminal paper [KL72], Kleiman and Laksov reconsidered Schubert's theory, rephrased it in more modern terms, and established the connection to so-called flags. A flag is a nested sequence of increasing subspaces. More specifically, let $\mathcal{V}_i$ denote a subspace of dimension d_i of a n-dimensional

vector space $\mathcal{V}$. A nested sequence of increasing subspaces

$$\mathcal{V}_1 \subset \mathcal{V}_2 \subset \ldots \subset \mathcal{V}_m \subset \mathcal{V} \tag{8.15}$$

with $d_1 < d_2 < \ldots < d_m$ is called a *flag*. If $m = n$ (and hence $d_i = i$), the sequence is known as a *complete flag*. Kleiman and Laksov introduced the *Schubert conditions* induced by a flag

$$\dim(\mathcal{V}_i \cap \mathcal{B}) \geq i \text{ for all } i \in [m] \tag{8.16}$$

for some m-dimensional subspace $\mathcal{B}$. The set of all subspaces $\mathcal{B}$ which fulfill these Schubert conditions are characterized in detail by Corollary 5 in [KL72]: This set equals the intersection in $\mathbb{R}^{\binom{n}{m}}$ between the Grassmannian $G_{m,n}$ and a specific set of linear equations. This means that a point on the Grassmannian is represented by its $\binom{n}{m}$ Plucker coordinates and the linear equations act on those Plucker coordinates. Hence, this set is actually a subvariety of $G_{m,n}$ which is known as the *Schubert variety*. The proof of this corollary also states how the number of linearly independent equations in this set of linear equations can be computed. Interestingly, this number only depends on $\dim(\mathcal{V}_i)$ and not on the specific orientations of the subspaces $\mathcal{V}_i$. For concreteness, let us consider again the line example. We have already seen that a line in 3D space can be represented as a 2D-subspace $\mathcal{V}_1 \in G_{2,4}$ embedded in $\mathcal{V} = \mathbb{R}^4$. The flag $\mathcal{V}_1 \subset \mathcal{V}_2 = \mathcal{R}^4$ with $d_1 = 2$, $d_2 = 4$, and $m = 2$ leads to a single linear equation according to this corollary and hence, four lines give four linear equations. Assuming these four equations are linearly independent, then these four linear equations together with the single quadratic Plucker relation for $\mathcal{B} \in G_{2,4}$ give two different solutions for $\mathcal{B}$ in general.

8.6.2 Subspace Intersection Problem in Terms of Flags

The theory of varieties implied by flags has two shortcomings when applied to the subspace intersection problem defined according to Eq. (8.1).

The previously mentioned line intersection example assumed the linear equations due to the four flags induced by four lines to be linearly independent. Even though this is the case for lines in $G_{2,4}$ in general position, this is no longer true for subspaces embedded in higher dimensional space $n > 4$. For example, Corollary 5 in [KL72] predicts 28 linearly independent equations for the flag $\mathcal{V}_1 \subset \mathcal{V}_2 = \mathbb{R}^{10}$ with $\mathcal{V}_1 \in G_{2,10}$

(and again $m = 2$) which is also equal to the rank $\text{rank}(\mathbf{M}_1) = 28$ of the matrix $\mathbf{M}_1$ capturing the linear constraints due to $\mathcal{A}_1 = \mathcal{V}_1$ derived in Sec. 8.3. However, the number of linearly independent constraints for two such flags $\mathcal{V}_1 \subset \mathcal{V}_2 = \mathbb{R}^{10}$ and $\mathcal{V}_1' \subset \mathcal{V}_2 = \mathbb{R}^{10}$ with $\mathcal{V}_1$ and $\mathcal{V}_1'$ in general position does not equal $28 + 28 = 56$ but rather 41 as can be seen by considering $\text{rank}(\mathbf{M}) = 41$ with $\mathbf{M} = [\mathbf{M}_1^T, \mathbf{M}_2^T]^T$ as defined in Sec. 8.3.

The Schubert conditions in Eq. (8.16) are obviously related to the subspace intersection problem in Eq. (8.1). To see that, let us introduce the flag $\mathcal{V}_1 \subset \mathcal{V}_2 \subset \ldots \subset V_m$ with the sum of subspaces $\mathcal{V}_i = \mathcal{A}_1 + \ldots + \mathcal{A}_i$ where $\mathcal{A}_i \in G_{d_i,n}$. Note that this step makes an important implicit assumption, namely that each $\mathcal{A}_i$ contributes at least one additional dimension to this sum, i.e. $\dim(\mathcal{A}_1 + \ldots + \mathcal{A}_{i-1}) < \dim(\mathcal{A}_1 + \ldots + \mathcal{A}_i)$. For the sake of simplicity, we also assume that the subspaces are independent, i.e. $\dim\left(\mathcal{V}_i = \bigoplus_{j=1}^{i} \mathcal{A}_j\right) = \sum_{j=1}^{i} d_j$. In order to apply the theory of Schubert conditions introduced in the previous section, we must assume in addition that the dimensionality of the unknown subspace $\mathcal{B} \in G_{D,n}$ equals the number of subspaces in the flag, i.e. $m = D$. These are strong assumptions and often do not hold, e.g. for the line intersection problem these assumptions are not satisfied. Given these assumptions hold, the 1D intersections between $\mathcal{A}_i$ and $\mathcal{B}$ imposed by the subspace intersection problem implies that also the Schubert conditions in Eq. (8.16) are fulfilled (assuming each $\mathcal{A}_i$ intersects $\mathcal{B}$ in a different point). Hence, Corollary 5 in [KL72] can again be used to compute the number of linearly independent equations for the flag $\mathcal{V}_1 \subset \mathcal{V}_2 \subset \ldots \subset V_m$. This number is however an underestimate of the true number of linearly independent equations contained in matrix $\mathbf{M}$ of Sec. 8.3. Let us again consider the previous example of two general subspaces $\mathcal{V}_1, \mathcal{V}_1' \in G_{2,10}$. According to Corollary 5 in [KL72], the flag $\mathcal{V}_1 \subset (\mathcal{V}_1 \oplus \mathcal{V}_1')$ yields 40 linearly independent equations, however considering $\text{rank}(\mathbf{M}) = 41$ shows that the number of linearly independent constraints due to the derivation in Sec. 8.3 is slightly higher (the difference between these two numbers can become larger for other choices of d_i, m, and n).

We have the following explanation for this discrepancy. The Schubert conditions in Eq. (8.16) can be rephrased using the notation introduced in Sec. 8.1 (here for $m = D = 2$)

$$[\mathbf{A}_1, \mathbf{A}_2] \begin{bmatrix} \mathbf{x}_1 & \mathbf{z} \\ \mathbf{0}_{d_2 \times 1} & \mathbf{x}_2 \end{bmatrix} = \mathbf{B}\,[\mathbf{y}_1, \mathbf{y}_2]\,. \tag{8.17}$$

The linear equations derived in Sec. 8.3 implicitly set $\mathbf{z}$ to the zero vector since they build upon Eq. (8.1) where each subspace $\mathcal{A}_i$ needs to intersect the unknown subspace $\mathcal{B}$ separately. By assumption, the matrix $[\mathbf{A}_1, \mathbf{A}_2]$ has full column-rank and hence, solving Eq. (8.17) instead of Eq. (8.1) yields the same solution with $\mathbf{z} = \mathbf{0}_{d_1 \times 1}$. In contrast, the Schubert variety is based entirely on Eq. (8.16) and the analysis in [KL72] allows for non-zero vectors $\mathbf{z}$. Imposing $\mathbf{z} = \mathbf{0}$ is obviously an additional constraint which explains the larger number of linearly independent rows in $\mathbf{M}$.

The cases where the $\mathcal{A}_i$ are dependent $\dim\left(\mathcal{V}_i = +_{j=1}^{i} \mathcal{A}_j\right) < \sum_{j=1}^{i} d_i$, or $m \neq D$, or where the subspaces $\mathcal{A}_i$ can not be embedded in a flag (e.g. because the dimension of the ambient space is too small) require a more careful analysis. For instance, the case of $\mathcal{A}_1, \mathcal{A}_2 \in G_{3,10}$ with $\dim(\mathcal{A}_1 \cap \mathcal{A}_2) = 1$ leads to $\dim(\mathcal{V}_1) = 3$ and $\dim(\mathcal{V}_2) = 5$ which provides 36 linearly independent constraints according to Corollary 5 in [KL72], however $\operatorname{rank}(\mathbf{M}) = 32$. Hence, in case of intersecting subspaces $\mathcal{A}_i$, the Schubert conditions in Eq. (8.16) can be stronger than the subspace intersection in Eq. (8.1)[2]. All these cases basically lead to more general determinantal ideals, known as *quiver ideals* which are generated by minors of products of matrices. Eq. (8.10) clearly reveals that the underlying ideal of the subspace intersection problem is actually a quiver ideal where all minors of size $D+1 \times D+1$ of the product between the matrices $[\Rightarrow_{i=1}^{m} \mathbf{A}_i]$ and $[\searrow_{i=1}^{m} \mathbf{x}_i]$ must vanish. The block-diagonal pattern of the second matrix factor introduces additional challenges in the analysis. We leave a detailed analysis of theses cases for future work and refer to Chap. 15 and 17 of [MS05] as a starting point.

8.7 Instances of Subspace Intersection Problems

One simple example involving intersecting lines in 3D has already been shown in Sec. 8.4. Further instances of our general theory are multiple-view tensors in multiple-view geometry in computer vision [HZ04]. For example, the fundamental matrix computation with the 7- or 8-point algorithm can be seen as instances of our general theory. As shown by Hartley and Schaffalitzky in [HS09], the derivation of the fundamental

[2] Based on several numerical examples we have run, the cases where the Schubert conditions provide more linearly independent constraints than the subspace intersections are rare.

matrix starts with equations of the form

$$\begin{bmatrix} \mathbf{P} & \mathbf{z}_i & \mathbf{0}_{3\times 1} \\ \mathbf{P}' & \mathbf{0}_{3\times 1} & \mathbf{z}_i' \end{bmatrix}_{6\times 6} \begin{pmatrix} \mathbf{Z}_i \\ \lambda_i \\ \lambda_i' \end{pmatrix} = \mathbf{0}_{6\times 1}, \tag{8.18}$$

where the two projection matrices $\mathbf{P}$ and $\mathbf{P}'$, the point $\mathbf{Z}_i \in \mathbb{R}^4$, and the projective depths $\lambda \in \mathbb{R}$ and $\lambda' \in \mathbb{R}$ are all unknown. The correspondence between two points $\mathbf{z}_i \in \mathbb{R}^3$ and $\mathbf{z}_i' \in \mathbb{R}^3$ in two images is known. Eq. (8.18) is an instance of a subspace intersection problem which can easily be seen by a change of notation with $\mathbf{B} = \begin{bmatrix} \mathbf{P} \\ \mathbf{P}' \end{bmatrix}$, $\mathbf{A}_i = \begin{bmatrix} \mathbf{z}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{z}_i' \end{bmatrix}$, $\mathbf{y}_i = \mathbf{Z}_i$, and $\mathbf{x}_i = (\lambda, \lambda')^T$. Hence, having sufficiently many subspaces $\mathbf{A}_i$ or equivalently correspondences at hand, the subspace spanned by the cameras can be computed. If only 7 correspondences are available, then a Plucker relation which translates to a rank-2 contraint on the fundamental matrix can be used to find the correct fundamental matrix in a 2D nullspace given by applying the algorithm of Sec. 8.3. Note that [HS09] uses similar techniques like e.g. the Laplace expansion and Plucker coordinates but is specifically tailored toward multiple-view geometry settings. As such, it is less general but offers a more thorough analysis of multiple-view relations, i.e. the Plucker relations arising due to constraints between multiple views. The interested reader is referred to that paper [HS09] for more details.

In the upcoming subsections, we would rather like to showcase the application of the theory derived so far to a new problem for which the novel interpretation as a subspace intersection problem opens up new possibilities. The structure-from-sound (SfS) problem is chosen for that because not all the dimensions d_i of the subspaces $\mathcal{A}_i$ are the same and because it allows to incorporate additional knowledge about the unknown subspace, like for example if a vector is known to lie in the unknown subspace.

8.7.1 Preliminaries for Unsynchronized Structure-from-Sound Problems

The SfS problem is an instance of an Euclidean distance matrix problem. Given pairwise measurements of distances between points, the Euclidean distance matrix problem tries to find a reconstruction of these points

in an Euclidean space such that the constraints due to the distance measurements are met. We refer to Chapter 30 of the recent book [AL11] and references therein for further details about Euclidean distance problems.

For SfS based Euclidean distance problems, the distance matrix is actually not observed directly but rather the time-of-arrival of signals emitted by speakers (or other sources) is measured at microphones from which a distance can be computed using the velocity of the signal. A homogeneous medium is assumed which leads to a constant velocity. In such SfS problems, the positions of both microphones and sound sources are computed at the same time. A distinguishing feature of SfS problems from standard Euclidean distance problems is the need for temporal synchronization. Only if all the microphones and all the speakers are registered w.r.t. a globally consistent time frame, then the time-of-arrivals (ToA) can be converted into time-of-flight (ToF) measurements which then in turn lead to distance measurements. Sources and microphones can both be synchronized, both unsynchronized, or either the sources or only the microhpones can be synchronized, for example achieved off-line in highly controlled settings. The application in this section addresses the last case of synchronized microphones but unsynchronized sound sources.

For notational convenience let us denote the case of m unsynchronized sources with n synchronized microphones by (m, n). We strive for an non-iterative closed-form solution. Surprisingly, this does not prevent the handling of missing distance measurements entries as we will see shortly. Pollefeys and Nister [PN08] showed previously that the synchronization time offsets of 5 unsynchronized sources can be computed in closed-form if distance measurements to at least 10 synchronized microphones are available. The theory developed in the current chapter allows to handle the same $(5, 10)$ case, even though derived in a completely different, but much more general way. Moreover, the theory for subspace intersection problems allows the handling of missing time-of-arrival measurements in closed-form, a situation which could not be handled by previous algorithms. Lastly, [PN08] could not handle situations with $m \geq 5$ and $n \geq 10$ in a unified framework: one had to fall-back to the $(5, 10)$ case which had to be solved repeatedly before fusing the individual results.

Let us first introduce and formalize the unsynchronized SfS problem. Let $\mathbf{s}_i \in \mathbb{R}^3$ for $i \in [m]$ denote the locations of the sound sources and

$\mathbf{m}_j \in \mathbb{R}^3$ for $j \in [n]$ denote the locations of the microphones. Microphone j measures the time of arrival of the sound from source i at time $\mathbf{T}_{i,j}$. The temporal offsets in order to register the sources to a globally consistent time are encoded in $\mathbf{u} \in \mathbb{R}^m$. Hence, the time of flight of the signal is given by $\mathbf{T}_{i,j} - \mathbf{u}_i$. Then for a homogeneous medium with a constant speed of sound v (at T degrees Celsius, $v \approx 331 + 0.6T\frac{m}{s}$) the basic equation for the distance between source i and microphone j is $\|\mathbf{s}_i - \mathbf{m}_j\| = v(\mathbf{T}_{i,j} - \mathbf{u}_i)$. This equation is non-linear in the sensor positions due to the square root. Squaring this equation leads to the entries of the *Euclidean distance matrix*

$$\mathbf{D}_{i,j} = \|\mathbf{s}_i - \mathbf{m}_j\|^2 = v^2(\mathbf{T}_{i,j} - \mathbf{u}_i)^2. \tag{8.19}$$

The Euclidean distance can be bilinearized [PN08; AL11] by introducing $\mathbf{S}_i^T = \left(\mathbf{s}_i^T\mathbf{s}_i, -2\mathbf{s}_i^T, 1\right) \in \mathbb{R}^{1\times 5}$ and $\mathbf{M}_j^T = \left(1, \mathbf{m}_j^T, \mathbf{m}_j^T\mathbf{m}_j\right) \in \mathbb{R}^{1\times 5}$ which leads to $\mathbf{D}_{i,j} = \mathbf{S}_i^T\mathbf{M}_j$. Gathering all the data from the different sound sources and microphones, we get the rank-5 representation of the data matrix

$$\mathbf{D} = \mathbf{S}\mathbf{M} = v^2\left[\Downarrow_i \Rightarrow_j (\mathbf{T}_{i,j} - \mathbf{u}_i)^2\right] \in \mathbb{R}^{m\times n}, \tag{8.20}$$

where $\mathbf{S} = \left[\Downarrow_i \mathbf{S}_i^T\right] \in \mathbb{R}^{m\times 5}$ and $\mathbf{M} = [\Rightarrow_j \mathbf{M}_j] \in \mathbb{R}^{5\times n}$. Expanding the time data on the right-hand side gives

$$\frac{1}{v^2}\mathbf{D} = \frac{1}{v^2}\mathbf{S}\mathbf{M} = \left[\Downarrow_i \Rightarrow_j \mathbf{T}_{i,j}^2 - 2\mathbf{T}_{i,j}(\mathbf{u}_i) + \mathbf{u}_i^2\right] \tag{8.21}$$

$$= \mathbf{T}^{\odot 2} - 2\,\mathrm{diag}\,(\mathbf{u})\,\mathbf{T} + \mathbf{u}^{\odot 2}\mathbf{1}_{n\times 1}^T. \tag{8.22}$$

Here, $\mathbf{A}^{\odot 2} = \mathbf{A} \odot \mathbf{A}$ denotes the entry-wise squaring of a matrix or equivalently the Hadamard product with itself. The term $\mathbf{u}^{\odot 2}\mathbf{1}_{n\times 1}^T$ can be absorbed in the first column of $\mathbf{S}$ by replacing $\mathbf{s}_i^T\mathbf{s}_i$ with $\mathbf{s}_i^T\mathbf{s}_i - \mathbf{u}_i^2$. This leads to

$$\frac{1}{v^2}\mathbf{D} = \frac{1}{v^2}\left[\Downarrow_i \mathbf{s}_i^T\mathbf{s}_i - \mathbf{u}_i^2, \mathbf{s}_i^T, 1\right]_{m\times 5}\left[\Rightarrow_j (1, \mathbf{m}_j^T, \mathbf{m}_j^T\mathbf{m}_j)^T\right]_{5\times n} \tag{8.23}$$

$$= \mathbf{T}^{\odot 2} - 2\,\mathrm{diag}\,(\mathbf{u})\,\mathbf{T}. \tag{8.24}$$

Note that the matrix $\mathbf{D}$ appearing on the left-hand side is subject to a low-rank constraint and the right-hand side is linear in the unknowns $\mathbf{u}$. Inspired by recent results for convex relaxations of rank constraints with the trace-norm (especially for Euclidean distance problems, see

[AL11]), we also tried such convex semi-definite programming relaxations. Interestingly, even though such relaxations seem to be quite successfull in handling missing entries in completely synchronized cases, the relaxations turned out to be too weak in unsynchronized cases.

Non-linear iterative optimizations such as Gauss-Newton have also been applied in order to solve Eq. (8.22) in the least-squares sense. For example, [MKP03] presented an iterative algorithm for the $2D$ case (i.e. where the microphones and sources are constrained to lie in the same plane) where in addition direction-of-arrival measurements were available. Unfortunately, a good initial guess is crucial for local optimization methods applied to problem Eq. (8.22). The importance of proper initialization has also been verified by Thrun [Thr05] who proposed a low-rank matrix factorization algorithm for a $2D$ far-field approximation with unsynchronized sources. A far field approximation assumes that the sources are infinitely far away from the microphones.

Based on our experience with Gauss-Newton and semidefinite programming, the completely synchronized SfS problem, i.e. where the synchronizations $\mathbf{u}$ are known, is much better-behaved than the unsynchronized case. Hence, knowing the synchronization seems to be highly beneficial. In the next section, we present how these synchronizations can be computed in closed-form based on the subspace intersection theory developed earlier on.

8.7.2 Unsynchronized SfS as a Subspace Intersection Problem

Interestingly, the formulation of the preceding section is an instance of a subspace intersection problem. In order to see that, let us define (by using Matlab-motivated row-slicing notation)

$$\mathbf{A}_i = \left[-2\mathbf{T}_{i,:}^T, \mathbf{T}_{i,:}^{2\,T}\right] \in \mathbb{R}^{n\times 2}. \tag{8.25}$$

Then Eq. (8.24) is equivalent to

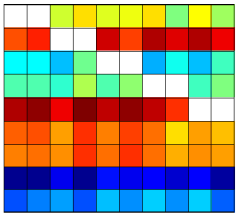
$$\forall i \in [m]: \quad \left[\mathbf{A}_i, \mathbf{M}^T\right]\begin{pmatrix}\mathbf{x}_i\\ \mathbf{y}_i\end{pmatrix} = \mathbf{0}_{n\times 1}, \tag{8.26}$$

with $\mathbf{x}_i = (\mathbf{u}_i, 1)^T \in \mathbb{R}^2$ and $\mathbf{y}_i = -\frac{1}{v^2}(\mathbf{s}_i^T\mathbf{s}_i - \mathbf{u}_i^2, \mathbf{s}_i^T, 1)^T \in \mathbb{R}^5$. Hence, we have m subspaces of dimension $d_i = 2$ intersecting another unknown subspace $\mathcal{B} = \operatorname{span}(\mathbf{B}) = \operatorname{span}\left(\mathbf{M}^T\right)$ of dimension $D = 5$ which is embedded in Euclidean n-dimensional space. Closer inspection of the

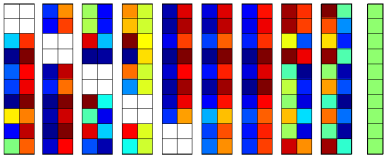
definition of $\mathbf{M}^T = \left[\Downarrow_{j=1}^{n} 1, \mathbf{m}_j^T, \mathbf{m}_j^T \mathbf{m}_j\right]$ even reveals that the constant one vector $\mathbf{1}_{n\times 1}$ must be contained in the unknown subspace $\mathcal{B}$. Such a constraint can be added by introducing a further matrix $\mathbf{A}_{m+1} = \mathbf{1}_{n\times 1}$.

At this point, it is worth to come back to a detail pointed to at the end of Sec. 8.5, namely the choice of the index set I for the algorithm based on the partial reduced row-echelon form. For cases with $n = 10$ and if the index set is chosen such that $m+1 \notin I$ (i.e. the constant one vector is not part of the matrices whose columns will be mapped to the block-matrix composed of the identity and zero matrix), then the algorithm fails since the system matrices $\mathbf{G}_i \in \mathbb{R}^{n-\sum_{j\in I} d_j \times d_i}$ for solving for $\mathbf{x}_i$ as in Eq. (8.14) will be the empty-matrices as $n - \sum_{i\in I} = 0$ for $n = 10$, $|I| = 5$, and $d_i = 2$ for $i \in [m]$. However, if $m+1 \in I$, then $n - \sum_{i\in I} = 9$ and $\mathbf{G}_i \in \mathbb{R}^{1\times 2}$, which have a unique one-dimensional nullspace which encodes $\mathbf{x}_i = (\mathbf{u}_i, 1)^T$ up to a scalar multiple. The unique time offset $\mathbf{u}_i$ can be found by dividing the two entries of $\mathbf{G}_i$, i.e. $\mathbf{u}_i = -\frac{\mathbf{G}_{i,1,2}}{\mathbf{G}_{i,1,1}}$.

Applying the algorithms of Sec. 8.3 or Sec. 8.5 shows that the minimal case of $m = 5$ unsynchronized speakers with $n = 10$ synchronized microphones can be handled linearly in closed-form. This is the same case which could be handled by the previous work [PN08]. Our algorithm can be applied more broadly, though. For example, cases $(7, 8)$ and $(6, 9)$ handled with the algorithm presented in Sec. 8.3 also yield a one-dimensional solution space and hence a unique solution for the time synchronizations $\mathbf{u}$. Previous approaches such as [PN08] could not handle these cases. Even more interestingly, the algorithm in Sec. 8.3 opens up the possibility to handle missing entries with a closed-form algorithm. If a time-of-arrival observation $\mathbf{T}_{i,j}$ is missing, this means that the j-th row of $\mathbf{A}_i$ is unknown. However, since the minor constraints only select $d_i + D = 2 + 5$ out of n rows, minors based on row subsets I, which slice a submatrix out of $\mathbf{A}_i$ whose entries are all known, can still be used. In this way, partial observations in $\mathbf{A}_i$ can still provide valuable constraints for the subspace $\mathbf{B}$. Fig. 8.2 shows an example for a $(9, 10)$ case where entries have been deleted such that no single submatrix is contained in matrix $\mathbf{T}$ whose entries are all known and which could be used to solve for the time synchronizations (i.e. no completely known submatrix of size $(5, 10)$, $(6, 9)$, or $(7, 8)$ exists). Nevertheless, the algorithm of Sec. 8.3 disregarding minor constraints consisting of missing entries succeeded in computing the correct time synchronization in closed-form. Obviously, the handling of missing observations in this



(a) Time-of-Arrival data $\mathbf{T}$ for $(m, n) = (9, 10)$



(b) Resulting subspaces $\mathbf{A}_i$

Figure 8.2: This figure visualizes the input data for the case of $m = 9$ unsynchronized sources and $n = 10$ synchronzied microphones in the presence of missing entries. These unknown entries are visualized transparently. Fig. 8.2a shows the time-of-arrival matrix $\mathbf{T} \in \mathbb{R}^{9\times10}$ whereas Fig. 8.2b shows the resulting subspace matrices $\mathbf{A}_i \in \mathbb{R}^{10\times2}$ for $i \in [m]$ and $\mathbf{A}_{m+1} = \mathbf{1}_{n\times1}$. Note that the pattern of missing entries is designed such that there is no single submatrix contained in $\mathbf{T}$ of minimal size (i.e. either $(5, 10)$, $(6, 9)$, or $(7, 8)$) without any unknown entry. Hence, solving a problem of smaller size where all the entries are known is not possible. Nevertheless, the algorithm of Sec. 8.3 disregarding minor constraints consisting of missing entries succeeded in computing the correct time synchronization in closed-form.

way is not restricted to the SfS problem. Any problem which can be cast in the form of a subspace intersection problem enjoys this property. As far as we know, this is a novel insight.

8.8 Conclusion and Future Work

In this chapter we have introduced the novel subspace intersection problem which tries to infer an unknown subspace constrained by other subspaces which are known to intersect this unknown subspace. An algebraic derivation has clearly highlighted the relation to Plucker coordinates which enabled a deeper understanding of the subspace intersection problem. For computational reasons, another algorithm

can be used which is based on a trick using a partial reduced row-echelon form. Line intersection problems in 3D and unsynchronized structure-from-sound problems have been shown to be instances of such subspace intersection problems. As a major novelty for the SfS problem, it has been shown that new cases can be handled in closed-form and that missing observation can still be handled in closed form by the presented algorithm. Missing observations in other instances of subspace intersection problems can be handled in a similar way.

A thread for future research is to specify conditions, under which circumstances there are sufficiently many linearly independent equations available, e.g. in the presence of missing entries. The easiest situation is probably when the subspaces $\mathcal{A}_i$ are independent. The situation becomes increasingly complex when the subspaces are pairwise disjoint, i.e. $\mathcal{A}_i \cap \mathcal{A}_j = \varnothing$ for $i \neq j$, but still the whole set of subspaces $\mathcal{A}_i$ for $i \in [m]$ is not independent (which is the case in the line intersection problem in Sec. 8.4, for example), or if the subspaces are not even pairwise disjoint. We suspect this to be a very delicate problem which requires a thorough understanding of determinantal ideals. As a starting point, we refer to Chap. 14-17 in [MS05]. Lastly, we plan to investigate which further applications can be cast as instances of subspace intersection problems.

9 Summary and Conclusion

9.1 Summary

This thesis has presented a holistic view on subspace models and algorithms. The ideas and concepts have mostly been introduced by building upon geometric computer vision applications, foremost upon the structure-from-motion problem. The chapters have introduced various different subspace models. The major optimization techniques have been presented and applied to low-dimensional subspace models.

Specifically, we have seen in Chap. 4 and Chap. 5 how multilinear interactions between variables can be modeled with tensors and how tensor decompositions such as the Tucker decomposition can reveal the underlying subspaces contained in a data tensor. Specific properties of the affine multi-camera structure-from-motion problem have been exploited in these chapters in order to derive algorithms which provide the solution to a trilinear factorization problem for general rigid, planar rigid, and non-rigid motions in closed-form.

If the problem does not enjoy such properties which can be exploited, standard least-squares non-linear optimization methods can be used to compute a low-rank decompositions. Chap. 3 has presented a comparison between well-known least-squares methods such as Levenberg-Marquardt and the Wiberg algorithm. The Wiberg algorithm has been shown to work exceptionally well for bilinear low-rank factorization problems.

An entirely different approach is taken by convex relaxations which replace the originally non-convex problem with a convex surrogate which guarantees that a globally optimal solution to this relaxed problem can be found. This approach has been followed in Chap. 6 where the generalized trace norm has been introduced as an alternative regularization term compared to the standard trace norm. The generalized trace norm enables a non-uniform penalization of subspace orientations, much similar to a weighted L1-regularization. That chapter has also presented an efficient first order primal-dual method based on proximal operators.

Another competing optimization technique is based on sampling. Sampling methods have attractive properties, e.g. they can be made less sensitive to outliers or to local minima in comparison to local optimization methods. Unfortunately, the design of efficient sampling strategies is highly challenging. Chap. 7 has presented a Markov Chain Monte Carlo method for rigid structure-from-motion based on simulated annealing, Gibbs, and Metropolis-Hastings samplers. The experiments showed mixed success: the performance depends largely on the pattern of known entries. Nonetheless, as we argue in Sec. 9.3, we think that sampling methods have great potential as optimization techniques for low-dimensional subspace models.

Lastly, this thesis has presented a purely algebraic view on subspace models. Specifically, Chap. 8 has motivated to represent a subspace in its Plucker coordinates. This chapter has also introduced the subspace intersection problem where an unknown subspace has to be found such that this unknown subspace intersects multiple known subspaces in an unknown intersection point. Our insights have lead to algebraic constraints expressed as linear equations in the lifted space of Plucker coordinates. Even though multiple view constraints in 3D computer vision are an instance of such a subspace intersection problem, Chap. 8 has focused on a structure-from-sound application where the developed algorithms allowed to compute an unknown temporal synchronization between sound sources in closed-form.

9.2 Conclusion

From a theoretical point of view, the findings of Chap. 4 and Chap. 5 are quite surprising: given a set of trajectories of feature points tracked in different affine cameras, there exist factorization based algorithms which provide the solution to the multi-camera rigid or non-rigid SfM problem in closed-form. Once we had realized how to formulate the problem in tensor notation, the relations between trajectories from different points seen in different cameras became obvious. The motion correspondence due to the commonly observed motion leads to a low-rank constraint which links all the data from the cameras even though the cameras might not track the same points. Unfortunately, the closed-form algorithms suffer from several short-comings in practice. Firstly, the trajectories must be complete, i.e. a feature point must be tracked from the first

to the last frame. Secondly, outliers can not be handled since the algorithms are based on a non-robust singular value decomposition of a flattened tensor. Thirdly, the derivations are based on an affine camera assumption which is only an approximation to the commonly used and more exact projective pinhole camera model. Motivated by these challenges, we started to explore more flexible and robust optimization techniques for subspace models.

As Chap. 3 showed, the Wiberg algorithm performed exceptionally well for low-rank matrix factorization problems. Unfortunately, we currently have no theoretically justified explanation for this. We have also applied the Wiberg algorithm to more general bilinear problem formulations where the problem no longer had the same highly organized structure as is the case for low-rank matrix factorization problems. Interestingly, the performance of the Wiberg algorithm dropped considerably under these circumstances. Hence, it seems that the Wiberg algorithm is the method of choice for low-rank factorization problems but performs comparable to its competitors for more general bilinear problems. Unfortunately, the Wiberg algorithm is inherently based on a least-squares formulation and hence, is not robust w.r.t. outliers.

Convex relaxations of the rank with the trace norm and with a robust Huber cost function are more robust to outliers and are guaranteed to converge to a global optimum, even though of the relaxed problem. The generalized trace norm presented in Chap. 6 indeed achieves its goal to encode prior knowledge about the subspace orientations in a low-rank regularization term. Such convex relaxations work well as soft-low rank priors for problems where the model complexity needs to be controlled. Results in compressive sensing provide a strong justification for the use of the trace-norm relaxations under idealized theoretical conditions. Unfortunately, for problems where the correct solution must have a strict low-rank (e.g. rank 4 for SfM factorizations) the success of convex relaxations largely depends on the problem at hand. Based on our experience, it is hard to judge beforehand whether or not a trace-norm relaxation or generalized trace-norm relaxation will yield satisfying results because the theoretical conditions required for the results derived in the compressive sensing community can often not be verified in practice. We argue that for many problems with a strict low-rank constraint, a local optimization with multiple random restarts often leads to better results than a convex relaxation with the trace or generalized trace norm. One of the reasons is that even though a convex

relaxation of the rank results in a convex problem and hence a global optimum is guaranteed to be found, the algorithms required to find this global optimum can be rather slow. For example, first-order methods for trace-norm relaxations are often based on soft-thresholding of singular values and the computation of a singular value decomposition of a large matrix in each iteration can be time consuming. Hence, rather than optimizing the convex relaxation once, one could as well run multiple instances of a non-linear iterative optimizations in the same amount of time.

Sampling methods (or randomized algorithms even more generally) offer several advantages compared to deterministic optimization techniques. Sampling methods are often less prone to converge to a local optimum and are more robust w.r.t. outliers. It is important to note however, that randomized algorithms do not exclude deterministic optimization techniques. For example, in a RANSAC loop for SfM, the computation of two-view relations inside the loop are based on deterministic techniques such as the 8-point algorithm for the fundamental matrix or the 5-point algorithm for the essential matrix. Hence, we see sampling techniques as complementary methods to deterministic techniques: an outer sampling loop can render deterministic techniques more robust w.r.t. outliers for example. Chap. 7 showed an example how a Markov Chain Monte Carlo sampling method can be combined with a Newton-step direction. Unfortunately, the experiments in that chapter have shown that the straight-forward blocks we have chosen for the Gibbs sampler in Chap. 7 lead to an unsatisfying performance for challenging patterns of known entries. Hence, we suspect that a different blocking can lead to a considerable performance increase. However, it is not clear how to efficiently combine deterministic optimization inside a Gibbs sampler with more general blocks of variables. We leave that as an open question for future work.

Lastly, the algebraic view on subspaces as presented in Chap. 8 establishes connections to other related fields such as algebraic geometry. The algebraic constructs (i.e. ideals and varieties) due to minor constraints on subsets of variables in a product of matrices or tensors have a rich algebraic structure and an understanding of these constructs is invaluable for a holistic view on low-dimensional subspace models. Chap. 8 also showed that such a theoretical understanding can lead to novel algorithms.

In conclusion, the holistic approach taken in this thesis enabled a

deeper understanding of subspace models and algorithms in geometric computer vision. Establishing connections to related fields served as inspiration and allowed us to derive new insights, models, and algorithms. The topics covered in this thesis also lay the foundations for future work as will be explained in the next section.

9.3 Future Work

9.3.1 Practical Considerations

As previously mentioned, robustness to a great number of outliers can be achieved by sampling frameworks. RANSAC has proved its power in many applications in the last decades. Unfortunately, plain RANSAC does not learn from previously generated hypothesis. Adaptive Monte Carlo methods on the other hand can adjust the proposal distributions based on the past. Hence, these methods are more suitable in an exploration-exploitation framework. As usually in these framework, the difficulty lies in finding the right trade-off between exploration and exploitation. We see a lot of potential in the combination of such adaptive Monte Carlo methods with deterministic optimization methods such as the ones presented in this thesis. The outer Monte Carlo sampling loop adds robustness to outliers and also protects against local minima whereas deterministic optimizations can lead to the global optimum for a subset of the variables of the problem with high probability. If the asymptotic guarantees of Monte Carlo methods are of lesser importance, then Genetic Algorithms offer similar advantages as adaptive Monte Carlo methods. Genetic Algorithms are similar to particle filter methods in that they maintain a population of states. However, the rules how this population is updated in each iteration are purely based on heuristics and generally come without any asymptotic guarantee. For that reason, we favour Monte Carlo methods. In the future, we plan to apply adaptive Monte Carlo methods to bilinear matrix completion problems with outliers. Once this is working, we expect the generalization to more complex low-dimensional subspaces models to be straight-forward.

9.3.2 Theoretical Questions

From a purely theoretical point of view, two remaining major questions can be identified. Firstly, the trace-norm for tensors as presented in

Sec. 6.7 still lacks a formal proof under which conditions it succeeds with high probability. Empirical tests provide strong evidence that the trace-norm regularization for tensors indeed behaves very similar to the trace-norm convex relaxation for low-rank matrices. Unfortunately, the proofs from the matrix case can not easily be extended to tensors because in the tensor case, the sum over multiple trace norms lead to a complex coupling between the tensor entries which make a proof more challenging.

Secondly, the algebra beneath subspace intersections is a complex issue and we still need to get a deeper understanding of the involved ideals, varieties, and constraints. This would allow to state clear conditions how many constraining subspaces must be known in order that the subspace intersection problem of Chap. 8 has a unique solution or a discrete number of solutions. Moreover, we can think of other relevant problems which would benefit from a deeper understanding of these topics. As a concrete example we would like to present the case of an uncalibrated camera rig. Consider a camera rig consisting of K projective cameras $\mathbf{P}_k \in \mathbb{R}^{3\times 4}$. The combined camera matrix is denoted with $\mathbf{P} = \left[\Downarrow_{k=1}^{K} \mathbf{P}_k\right] \in \mathbb{R}^{3K\times 4}$. A similar derivation as for the fundamental matrix between two cameras leads to the following equation capturing the motion of the camera rig due to a projective transformation $\mathbf{H} \in \mathbb{R}^{4\times 4}$

$$\begin{bmatrix} \mathbf{P} & \left[\searrow_{k=1}^{K} \mathbf{x}_k\right] & \mathbf{0} \\ \mathbf{PH} & \mathbf{0} & \left[\searrow_{k=1}^{K} \mathbf{x}'_k\right] \end{bmatrix} \begin{pmatrix} \mathbf{X} \\ -\left[\Downarrow_{k=1}^{K} \lambda_k\right] \\ -\left[\Downarrow_{k=1}^{K} \lambda'_k\right] \end{pmatrix} = \mathbf{0}_{2\cdot 3K\times 1}, \tag{9.1}$$

where $\mathbf{x}_k \in \mathbb{R}^3$ and $\mathbf{x}'_k \in \mathbb{R}^3$ are the projections of feature point $\mathbf{X} \in \mathbb{R}^4$ into the cameras before and after the motion $\mathbf{H}$ is applied and $\lambda_k \in \mathbb{R}$ and $\lambda'_k \in \mathbb{R}$ denote the corresponding projective depths. This equation can be turned into constraints on minors of the matrix on the left-hand side. If there are no feature point correspondences between different cameras of the camera rig, e.g. due to non-overlapping views, then the only minor constraints available are the ones involving submatrices of the form

$$\begin{bmatrix} \mathbf{P}_{k_1} \\ \mathbf{P}_{k_2}\mathbf{H} \end{bmatrix} \in \mathbb{R}^{6\times 4}, \tag{9.2}$$

with $k_1 = k_2$. This certainly leads to polynomial equations generated by minors of products of matrices $\mathbf{P}_k$ and $\mathbf{H}$, both of which are unknown.

Obviously, the same reasoning also applies to relations involving more than two projective transformations of the camera rig.

Bibliography

[Aan+02] Henrik Aanæs, Rune Fisker, Kalle Åström, and Jens Michael Carstensen. "Robust Factorization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.9 (Sept. 2002), pp. 1215–1225. ISSN: 0162-8828 (cit. on p. 152).

[Akh+08] Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. "Nonrigid structure from motion in trajectory space". In: *Advances in Neural Information Processing Systems, NIPS 2008, Vancouver, British Columbia, Canada, December 8-11, 2008*. Curran Associates, Inc., 2008, pp. 41–48 (cit. on pp. 113, 115).

[Akh+11] Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. "Trajectory space: A dual representation for nonrigid structure from motion". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.7 (2011), pp. 1442–1456. ISSN: 0162-8828 (cit. on pp. 68, 113, 115).

[AL06] Krishna B Athreya and Soumendra N Lahiri. *Measure theory and probability theory*. Springer Texts in Statistics. Springer Verlag, 2006. ISBN: 978-0-3873-2903-1 (cit. on p. 170).

[AL11] Miguel F. Anjos and Jean B. Lasserre. *Handbook on semidefinite, conic and polynomial optimization*. International Series in Operations Research & Management Science. Springer Verlag, 2011. ISBN: 978-1-4614-0768-3 (cit. on pp. 198, 204–206).

[AP09] Roland Angst and Marc Pollefeys. "Static multi-camera factorization using rigid motion". In: *IEEE International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1203–1210. ISBN: 978-1-4244-4419-9 (cit. on pp. 54, 56, 101).

[AP10] Roland Angst and Marc Pollefeys. "5D motion subspaces for planar motions". In: *European Conference on Computer Vision, ECCV 2010, Heraklion, Crete, Greece, September 5-11, 2010*. Vol. 6313. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Verlag, 2010, pp. 144–157. ISBN: 978-3-6421-5557-4 (cit. on pp. 54, 56, 101).

[AP12] Roland Angst and Marc Pollefeys. "A unified view on deformable shape factorizations". In: *European Conference on Computer Vision, ECCV 2012, Florence, Italy, October 7-13 2012*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Verlag, 2012 (cit. on p. 111).

[AXS08] Pedro M. Q. Aguiar, João M. F. Xavier, and Marko Stosic. "Spectrally optimal factorization of incomplete matrices". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, Anchorage, Alaska, USA, June 24-26 2008*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 1–8. ISBN: 978-1-4244-2242-5 (cit. on p. 70).

[AZP11] Roland Angst, Christopher Zach, and Marc Pollefeys. "The generalized trace-norm and its application to structure-from-motion problems". In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 2502–2509. ISBN: 978-1-4577-1101-5 (cit. on p. 144).

[BA06] Alessio Del Bue and Lourdes de Agapito. "Non-rigid stereo factorization". In: *International Journal of Computer Vision* 66.2 (2006), pp. 193–207 (cit. on pp. 55, 113, 118).

[Bac08] Francis R. Bach. "Consistency of trace norm minimization". In: *Journal of Machine Learning Research* 9 (2008), pp. 1019–1048. ISSN: 1532-4435 (cit. on pp. 148, 155).

[BF05] A. M. Buchanan and Andrew W. Fitzgibbon. "Damped newton algorithms for matrix factorization with missing data". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2005, San Diego, CA, USA, June 20-26 2005*. Washington, DC, USA: IEEE Computer

Society, 2005, pp. 316–322. ISBN: 0-7695-2372-2 (cit. on p. 23).

[BHB00] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. "Recovering non-rigid 3D shape from image streams". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2000, Hilton Head, SC, USA, June 13-15 2000*. Washington, DC, USA: IEEE Computer Society, 2000, pp. 2690–2696. ISBN: 0-7695-0662-3 (cit. on pp. 55, 112, 115, 118).

[Bis06] Christopher M. Bishop. *Pattern recognition and machine learning*. 1st ed. Information Science and Statistics. Springer Verlag, 2006. ISBN: 978-0-3873-1073-2 (cit. on pp. 168, 177).

[BMP08] Francis Bach, Julien Mairal, and Jean Ponce. "Convex sparse matrix factorizations". In: *Computing Research Repository* abs/0812.1869 (2008) (cit. on pp. 145, 149).

[Bra01] Matthew Brand. "Morphable 3D models from video". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2001, Kauai, HI, USA, December 8-14 2001*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 456–463. ISBN: 0-7695-1272-0 (cit. on pp. 55, 67, 74, 112).

[Bra05] Matthew Brand. "A direct method for 3D factorization of nonrigid motion observed in 2D". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2005, San Diego, CA, USA, June 20-26 2005*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 122–128. ISBN: 0-7695-2372-2 (cit. on pp. 55, 74, 112, 119, 121, 126).

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004. ISBN: 978-0-5218-3378-3 (cit. on p. 143).

[CC70] J. Carroll and Jih-Jie Chang. "Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition". In: *Psychometrika* 35.3 (Sept. 1970), pp. 283–319 (cit. on p. 15).

[CGM07] Olivier Cappe, Simon J. Godsill, and Eric Moulines. "An overview of existing methods and recent advances in sequential Monte Carlo". In: *Proceedings of the IEEE* 95.5 (2007), pp. 899–924. ISSN: 0018-9219 (cit. on p. 188).

[CK95] João Paulo Costeira and Takeo Kanade. "A multi-body factorization method for motion analysis". In: *IEEE International Conference on Computer Vision, ICCV 1995, Cambridge, Massachusetts, USA, June 20-23, 1995*. Washington, DC, USA: IEEE Computer Society, 1995, pp. 1071–1076 (cit. on p. 112).

[CLO97] David A. Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms: An introduction to computational algebraic geometry and commutative algebra*. Undergraduate Texts in Mathematics. Springer Verlag, 1997. ISBN: 978-0-3879-4680-1 (cit. on pp. 9, 196, 199).

[CP11] Antonin Chambolle and Thomas Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging". In: *Journal of Mathematical Imaging and Vision* 40.1 (2011), pp. 120–145 (cit. on pp. 156, 157).

[CT10] Emmanuel J. Candès and Terence Tao. "The power of convex relaxation: near-optimal matrix completion". In: *IEEE Transactions on Information Theory* 56.5 (2010), pp. 2053–2080 (cit. on p. 143).

[Dan99] Konstantinos Daniilidis. "Hand-eye calibration using dual quaternions". In: *International Journal of Robotic Research* 18.3 (1999), pp. 286–298 (cit. on p. 56).

[Das11] Boris Nikolaev Daskalov. "Iterative methods for matrix factorization with missing data". Masters's Thesis. Zurich, Switzerland: ETH Zurich, Aug. 2011 (cit. on pp. 30, 32).

[DLH10] Yuchao Dai, Hongdong Li, and Mingyi He. "Element-Wise Factorization for N-View Projective Reconstruction". In: *European Conference on Computer Vision, ECCV 2010, Heraklion, Crete, Greece, September 5-11, 2010*. Vol. 6313. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Verlag, 2010, pp. 396–409. ISBN:

978-3-6421-5557-4 (cit. on pp. 144, 152, 154, 159, 161, 166).

[DM01] D. van Dyk and X.L. Meng. "The art of data augmentation (with discussion)". In: *Journal of Computational and Graphical Statistics* 10 (2001), pp. 1–111 (cit. on p. 176).

[EH10] Anders Eriksson and Anton van den Hengel. "Efficient computation of robust low-rank matrix approximations in the presence of missing data using the L1 norm". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, June 13-18 2010*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 771–778 (cit. on pp. 152, 165).

[FB81] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395 (cit. on pp. 166, 168).

[FHB01] Maryam Fazel, Haitham Hindi, and Stephen P. Boyd. "A rank minimization heuristic with application to minimum order system approximation". In: *Proceedings of the American Control Conference*. Vol. 6. 2001, pp. 4734–4739 (cit. on p. 198).

[FHI01] David A. Forsyth, John A. Haddon, and Sergey Ioffe. "The joy of sampling". In: *International Journal of Computer Vision* 41 (2001), pp. 109–134. ISSN: 0920-5691 (cit. on p. 169).

[FRA11] João Fayad, Chris Russell, and Lourdes de Agapito. "Automated articulated structure and 3D shape recovery from point correspondences". In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 431–438. ISBN: 978-1-4577-1101-5 (cit. on p. 113).

[GA02] Rui F. C. Guerreiro and Pedro M. Q. Aguiar. "3D structure from video streams with partially overlapping images". In: *IEEE International Conference on Image Processing, ICIP 2011, Rochester, New York, USA, September 22-25, 2011*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 897–900 (cit. on p. 55).

[GC11] Mark Girolami and Ben Calderhead. "Riemann manifold Langevin and Hamiltonian Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (2011), pp. 123–214. ISSN: 1467-9868 (cit. on p. 169).

[GM11] Paulo F. U. Gotardo and Aleix M. Martinez. "Non-rigid structure from motion with complementary rank-3 spaces". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2011, Colorado Springs, CO, USA, 20-25 June 2011*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 3065–3072. ISBN: 978-1-4577-0394-2 (cit. on p. 115).

[GP03] Gene Golub and Victor Pereyra. "Separable nonlinear least squares: the variable projection method and its applications". In: *Inverse Problems* 19.2 (2003), R1 (cit. on p. 28).

[GRG94] Walter R. Gilks, Gareth O. Roberts, and Edward I. George. "Adaptive direction sampling". English. In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 43.1 (1994), pp. 179–189. ISSN: 00390526 (cit. on p. 186).

[Gro11] David Gross. "Recovering Low-Rank Matrices From Few Coefficients in Any Basis". In: *IEEE Transactions on Information Theory* 57.3 (Mar. 2011), pp. 1548–1566. ISSN: 0018-9448 (cit. on pp. 32, 144, 151, 163, 164).

[Gv96] Gene H. Golub and Charles F. van Van Loan. *Matrix computations*. 3rd. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996. ISBN: 978-0-8018-5414-9 (cit. on p. 196).

[GW95] Michel X. Goemans and David P. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM* 42.6 (Nov. 1995), pp. 1115–1145. ISSN: 0004-5411 (cit. on p. 143).

[Han01] Martin Hanke. "On Lanczos based methods for the regularization of discrete ill-posed problems". In: *BIT Numerical Mathematics* 41 (5 2001). 10.1023/A:1021941328858, pp. 1008–1018. ISSN: 0006-3835 (cit. on p. 31).

[Har70] Richard A. Harshman. "Foundations of the PARAFAC procedure: Models and conditions for an explanatory multi-modal factor analysis". In: *UCLA Working Papers in Phonetics* 16.1 (1970), p. 84 (cit. on p. 15).

[HJ94] Roger A. Horn and Charles R. Johnson. *Topics in matrix analysis*. Topics in matrix analysis. Cambridge University Press, 1994. ISBN: 978-0-5214-6713-1 (cit. on pp. 145, 147, 152).

[Hof10] Peter D. Hoff. "Separable covariance arrays via the Tucker product, with applications to multivariate relational data". In: *Arxiv preprint arXiv:1008.2169* (Aug. 2010). arXiv: 1008.2169 (cit. on p. 168).

[Hof11] Peter D. Hoff. "Hierarchical multilinear models for multiway data". In: *Computational Statistics and Data Analysis* 55.1 (Jan. 2011), pp. 530–543. ISSN: 0167-9473 (cit. on pp. 8, 168).

[HS04] Richard Hartley and Frederik Schaffalizky. "PowerFactorization: 3D reconstruction with missing or uncertain data". In: *Australia-Japan Advanced Workshop on Computer Vision*. 2004 (cit. on pp. 25, 26, 55).

[HS09] Richard Hartley and Frederik Schaffalitzky. "Reconstruction from projections using grassmann tensors". In: *International Journal of Computer Vision* 83.3 (2009), pp. 274–293. ISSN: 0920-5691 (cit. on pp. 202, 203).

[HV08] Richard Hartley and René Vidal. "Perspective nonrigid shape and motion recovery". In: *European Conference on Computer Vision, ECCV 2008, Marseille, France, October 12-18, 2008*. Vol. 5302. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Verlag, 2008, pp. 276–289. ISBN: 978-3-540-88681-5 (cit. on p. 119).

[HZ04] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. 2nd. Cambridge University Press, 2004. ISBN: 978-0-5215-4051-3 (cit. on pp. 18, 21, 45, 71, 72, 132, 202).

[Ira99] Michal Irani. "Multi-frame optical flow estimation using subspace constraints". In: *IEEE International Conference on Computer Vision, ICCV 2009, Kerkyra, Corfu, Greece, September 20-25, 1999*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 626–633 (cit. on p. 118).

[Jas+08] Ajay Jasra, Arnaud Doucet, David A. Stephens, and Christopher C. Holmes. "Interacting sequential Monte Carlo samplers for trans-dimensional simulation". In: *Computational Statistics and Data Analysis* 52.4 (Jan. 2008), pp. 1765–1791. ISSN: 0167-9473 (cit. on p. 188).

[KB09] Tamara G. Kolda and Brett W. Bader. "Tensor decompositions and applications". In: *SIAM Review* 51.3 (Aug. 2009), pp. 455–500. ISSN: 0036-1445 (cit. on pp. 14, 25).

[KD10] Michael Kaess and Frank Dellaert. "Probabilistic structure matching for visual SLAM with a multi-camera rig". In: *Computer Vision Image Understanding* 114.2 (2010), pp. 286–296. ISSN: 1077-3142 (cit. on p. 168).

[KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. 1st. Adaptive computation and machine learning. MIT Press, 2009. ISBN: 978-0-2620-1319-2 (cit. on p. 5).

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing". In: *Science, Number 4598, 13 May 1983* 220, 4598 (1983), pp. 671–680 (cit. on p. 175).

[KL72] Steven L. Kleiman and Dan Laksov. "Schubert Calculus". In: *The American Mathematical Monthly* 79.10 (1972), pp. 1061–1082. ISSN: 00029890 (cit. on pp. 194, 199–202).

[Kum+08] Ram Krishan Kumar, Adrian Ilie, Jan-Michael Frahm, and Marc Pollefeys. "Simple calibration of non-overlapping cameras with a mirror". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, Anchorage, Alaska, USA, June 24-26 2008*. Washington, DC, USA: IEEE Computer Society, 2008. ISBN: 978-1-4244-2242-5 (cit. on p. 56).

[LC05] Jian Li and Rama Chellappa. "A factorization method for structure from planar motion". In: *IEEE Workshop on Applications of Computer Vision / IEEE Workshop on Motion and Video Computing, WACV/MOTION, 2005, Breckenridge, CO, USA, January 5-7 2005*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 154–159. ISBN: 0-7695-2271-8 (cit. on p. 56).

[LCM10] Zhouchen Lin, Minming Chen, and Yi Ma. "The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices". In: *Arxiv preprint arXiv:1009.5055* (Sept. 9, 2010). arXiv: 1009.5055 (cit. on p. 154).

[Lla+11] Xavier Lladó, Alessio Del Bue, Arnau Oliver, Joaquim Salvi, and Lourdes de Agapito. "Reconstruction of non-rigid 3D shapes from stereo-motion". In: *Pattern Recognition Letters* 32.7 (2011), pp. 1020–1028 (cit. on pp. 113, 118).

[LLC10] Faming Liang, Chuanhai Liu, and Raymond Carroll. *Advanced Markov chain Monte Carlo methods: Learning from past samples*. Wiley Series in Computational Statistics. John Wiley & Sons, 2010. ISBN: 978-0-4707-4826-8 (cit. on pp. 170, 171, 173).

[LMV00] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. "A multilinear singular value decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (Mar. 2000), pp. 1253–1278. ISSN: 0895-4798 (cit. on pp. 14, 15, 134, 163, 164).

[Mag07] Peter Magyar. *Notes on minors and plucker coordinates*. Tech. rep. Department of Mathematics, Michigan State University, 2007 (cit. on p. 194).

[MHT10] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. "Spectral regularization algorithms for learning large incomplete matrices". In: *Journal of Machine Learning Research* 99 (Aug. 2010), pp. 2287–2322. ISSN: 1532-4435 (cit. on pp. 154, 156).

[MKP03] Randolph L. Moses, Dushyanth Krishnamurthy, and Robert M. Patterson. "A self-localization method for wireless sensor networks". In: *EURASIP Journal on Applied Signal Processing* 2003.4 (2003), pp. 348–358 (cit. on p. 206).

[MN99] Jan R. Magnus and Heinz Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. 2nd. Wiley Series in Probability and Statistics. John Wiley & Sons, 1999. ISBN: 978-0-4719-8633-1 (cit. on pp. 14, 16, 17, 31, 105, 133).

[MS05] Ezra Miller and Bernd Sturmfels. *Combinatorial commutative algebra*. Graduate Texts in Mathematics. Springer Verlag, 2005. ISBN: 978-0-3872-3707-7 (cit. on pp. 9, 202, 209).

[Nea93] Radford M. Neal. *Probabilistic inference using Markov Chain Monte Carlo methods*. Tech. rep. CRG-TR-93-1. Department of Computer Science, University of Toronto, 1993 (cit. on pp. 169, 170).

[Neg+10] Sahand Negahban, Pradeep D. Ravikumar, Martin J. Wainwright, and Bin Yu. "A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers". In: *Computing Research Repository* abs/1010.2731 (2010) (cit. on pp. 144, 149, 151, 163, 164).

[NL11] Sebastian Nowozin and Christoph H. Lampert. *Structured learning and prediction in computer vision*. Foundations and Trends in Computer Graphics and Vision. Now Publishers, 2011. ISBN: 978-1-6019-8456-2 (cit. on p. 6).

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. 2nd. Springer Series in Operations Research. Springer Verlag, 2006. ISBN: 978-0-3879-8793-4 (cit. on pp. 7, 23, 26, 27, 176).

[OB08] Søren I. Olsen and Adrien Bartoli. "Implicit non-rigid structure-from-motion with priors". In: *Journal of Mathematical Imaging and Vision* 31.2-3 (2008), pp. 233–244. ISSN: 0924-9907 (cit. on p. 114).

[OD07] Takayuki Okatani and Koichiro Deguchi. "On the Wiberg algorithm for matrix factorization in the presence of missing components". In: *International Journal of Computer Vision* 72.3 (May 2007), pp. 329–337. ISSN: 0920-5691 (cit. on pp. 28, 30, 31).

[OH07] John Oliensis and Richard Hartley. "Iterative extensions of the sturm/triggs algorithm: Convergence and nonconvergence". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.12 (2007), pp. 2217–2233 (cit. on p. 152).

[OO09] Carl Olsson and Magnus Oskarsson. "A convex approach to low rank matrix approximation with missing data". In: *Scandinavian Conference on Image Analysis, SCIA 2009, Oslo, Norway, June 15-18, 2009*. Vol. 5575. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Verlag, 2009, pp. 301–309. ISBN: 978-3-642-02229-6 (cit. on pp. 144, 152).

[OYD11] Takayuki Okatani, Takahiro Yoshida, and Koichiro Deguchi. "Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms". In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 842–849. ISBN: 978-1-4577-1101-5 (cit. on pp. 23, 28).

[Par+10] Hyun Soo Park, Takaaki Shiratori, Iain Matthews, and Yaser Sheikh. "3D reconstruction of a moving point from a series of 2D projections". In: *European Conference on Computer Vision, ECCV 2010, Heraklion, Crete, Greece,*

September 5-11, 2010. Vol. 6313. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Verlag, 2010, pp. 158–171. ISBN: 978-3-6421-5557-4 (cit. on pp. 119, 122).

[PBA10] Marco Paladini, Adrien Bartoli, and Lourdes de Agapito. "Sequential non-rigid structure-from-motion with the 3D-implicit low-rank shape model". In: *European Conference on Computer Vision, ECCV 2010, Heraklion, Crete, Greece, September 5-11, 2010*. Vol. 6313. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Verlag, 2010, pp. 15–28. ISBN: 978-3-6421-5557-4 (cit. on pp. 113, 114).

[PN08] Marc Pollefeys and David Nister. "Direct computation of sound and microphone locations from time-difference-of-arrival data". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2008, Las Vegas, Nevada, USA, March 30 - April 4, 2008*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 2445–2448. ISBN: 1-4244-1484-9 (cit. on pp. 204, 205, 207).

[PS82] Christopher C. Paige and Michael A. Saunders. "LSQR: An algorithm for sparse linear equations and sparse least squares". In: *ACM Transactions on Mathematical Software* 8.1 (Mar. 1982), pp. 43–71. ISSN: 0098-3500 (cit. on p. 30).

[QM02] Yuan Qi and Thomas P. Minka. *Hessian-based Markov chain Monte-Carlo algorithms*. Tech. rep. 2002 (cit. on pp. 179, 181).

[RC04] Christian Robert and George Casella. *Monte Carlo Statistical Methods*. 2nd. Springer Texts in Statistics. Springer Verlag, 2004. ISBN: 978-0-3872-1239-5 (cit. on pp. 170, 173, 174).

[RFP10] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization". In: *SIAM Review* 52.3 (Aug. 2010), pp. 471–501. ISSN: 0036-1445 (cit. on pp. 32, 143, 145, 149).

[Roc70] Ralph Tyrell Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, 1970. ISBN: 0691015864 (cit. on p. 143).

[Sch79] Hermann Schubert. *Kalkül der abzählenden Geometrie: Neuauflage von 1979*. Springer Verlag, 1879. ISBN: 978-3-5400-9233-9 (cit. on pp. 189, 199).

[SIR95] Heung-Yeung Shum, Katsushi Ikeuchi, and Raj Reddy. "Principal component analysis with missing data and its application to polyhedral object modeling". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.9 (1995), pp. 854–867. ISSN: 0162-8828 (cit. on pp. 28, 31).

[SM08] Ruslan Salakhutdinov and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo". In: *International Conference on Machine Learning, ICML 2008, Helsinki, Finland, June 5-9, 2008*. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 880–887. ISBN: 978-1-60558-205-4 (cit. on pp. 8, 150, 167, 177, 187).

[SMP05] Tomás Svoboda, Daniel Martinec, and Tomás Pajdla. "A convenient multicamera self-calibration for virtual environments". In: *Presence* 14.4 (2005), pp. 407–422 (cit. on p. 55).

[SS10] Ruslan Salakhutdinov and Nathan Srebro. "Collaborative filtering in a non-uniform world: Learning with the weighted trace norm". In: *Conference on Neural Information Processing Systems, NIPS 2010, Vancouver, British Columbia, Canada, December 6-9, 2010*. Curran Associates, Inc., 2010, pp. 2056–2064 (cit. on pp. 149, 154).

[ST96] Peter F. Sturm and Bill Triggs. "A factorization based algorithm for multi-image projective structure and motion". In: *European Conference on Computer Vision, ECCV 1996, Cambridge, UK, April 15-18, 1996*. Vol. 1065. Lecture Notes in Computer Science. Springer Verlag, 1996, pp. 709–720. ISBN: 3-540-61123-1 (cit. on p. 55).

[TD03] Philip H. S. Torr and Colin Davidson. "IMPSAC: Synthesis of importance sampling and random sample consensus". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.3 (2003), pp. 354–364. ISSN: 0162-8828 (cit. on p. 169).

[TH99] Seth Teller and Michael Hohmeyer. "Determining the lines through four lines". In: *Journal of Graphics Tools* 4.3 (Nov. 1999), pp. 11–22. ISSN: 1086-7651 (cit. on pp. 190, 197).

[THB08] Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. "Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.5 (2008), pp. 878–892 (cit. on pp. 113, 138, 140, 141).

[THK10] Ryota Tomioka, Kohei Hayashi, and Hisashi Kashima. "Estimation of low-rank tensors via convex optimization". In: *ArXiv e-prints* (Oct. 2010). arXiv: 1010.0789 (cit. on p. 163).

[Thr05] Sebastian Thrun. "Affine structure from sound". In: *Advances in Neural Information Processing Systems, NIPS 2005, Vancouver, British Columbia, Canada, December 5-8 2005*. Curran Associates, Inc., 2005, pp. 1353–1360 (cit. on p. 206).

[Tie94] Luke Tierney. "Markov chains for exploring posterior distributions". In: *Annals of Statistics* 22.4 (1994), pp. 1701–1728. ISSN: 0090-5364 (cit. on pp. 170, 171).

[TK92] Carlo Tomasi and Takeo Kanade. "Shape and motion from image streams under orthography: a factorization method". In: *International Journal of Computer Vision* 9.2 (1992), pp. 137–154 (cit. on pp. 41, 54, 59, 67, 112).

[Tor+01] Lorenzo Torresani, Danny B. Yang, Eugene J. Alexander, and Christoph Bregler. "Tracking and modeling non-rigid objects with rank constraints". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2001, Kauai, HI, USA, December 8-14 2001*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 493–500. ISBN: 0-7695-1272-0 (cit. on pp. 55, 118).

[TR05] Philip A. Tresadern and Ian D. Reid. "Articulated structure from motion by factorization". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2005, San Diego, CA, USA, June 20-26 2005*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 1110–1115. ISBN: 0-7695-2372-2 (cit. on pp. 55, 112).

[Tuc66] Ledyard Tucker. "Some mathematical notes on three-mode factor analysis". In: *Psychometrika* 31 (3 1966). 10.1007/BF02289464, pp. 279–311. ISSN: 0033-3123 (cit. on pp. 14, 15, 68).

[TV07] Roberto Tron and René Vidal. "A benchmark for the comparison of 3-D motion segmentation algorithms". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2007, Minneapolis, Minnesota, USA, 18-23 June 2007*. IEEE Computer Society, 2007 (cit. on pp. 55, 112).

[VO02] René Vidal and John Oliensis. "Structure from planar motions with small baselines". In: *European Conference on Computer Vision, ECCV 2002, Copenhagen, Denmark, May 28-31 2002*. Vol. 2351. Lecture Notes in Computer Science. Springer Verlag, 2002, pp. 383–398. ISBN: 3-540-43744-4 (cit. on p. 56).

[Wib76] T. Wiberg. "Computation of principal components when data are missing". In: *Second Symposium of Computational Statistics, 1976, Berlin, Germany*. 1976, pp. 229–236 (cit. on p. 28).

[WS07] Daniel Wagner and Dieter Schmalstieg. *ARToolKitPlus for pose tracking on mobile devices*. Tech. rep. Institute for Computer Graphics and Vision, Graz University of Technology, Feb. 2007 (cit. on pp. 96, 99).

[WTW08] Guanghui Wang, Hung-Tat Tsui, and Q. M. Jonathan Wu. "Rotation constrained power factorization for structure from motion of nonrigid objects". In: *Pattern Recognition Letters* 29.1 (2008), pp. 72–80 (cit. on p. 55).

[WZ02] Lior Wolf and Assaf Zomet. "Correspondence-free synchronization and reconstruction in a non-rigid scene". In: *Workshop on Vision and Modelling of Dynamic Scenes*. 2002 (cit. on p. 113).

[WZ06] Lior Wolf and Assaf Zomet. "Wide baseline matching between unsynchronized video sequences". In: *International Journal of Computer Vision* 68.1 (2006), pp. 43–52 (cit. on p. 56).

[XCK04] Jing Xiao, Jinxiang Chai, and Takeo Kanade. "A closed-form solution to non-rigid shape and motion recovery". In: *European Conference on Computer Vision, ECCV 2004, Prague, Czech Republic, May 11-14 2004*. Vol. 3024. Lecture Notes in Computer Science. Springer Verlag, 2004, pp. 573–587. ISBN: 3-540-21981-1 (cit. on pp. 68, 112, 119, 125, 127).

[YP08] Jingyu Yan and Marc Pollefeys. "A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.5 (2008), pp. 865–877 (cit. on pp. 55, 112).

[Zah+11] Aamer Zaheer, Ijaz Akhter, Mohammad Haris Baig, Shabbir Marzban, and Sohaib Khan. "Multiview structure from motion in trajectory space". In: *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 2447–2453. ISBN: 978-1-4577-1101-5 (cit. on pp. 113, 118, 134–136).

[ZH09] Andrei Zaharescu and Radu Horaud. "Robust factorization methods using a gaussian/uniform mixture model". In: *International Journal of Computer Vision* 81.3 (2009), pp. 240–258 (cit. on p. 152).

[ZI06] Lihi Zelnik-Manor and Michal Irani. "On single-sequence and multi-sequence factorizations". In: *International Journal of Computer Vision* 67.3 (2006), pp. 313–326 (cit. on p. 56).