Multilinear Factorizations for Multi-Camera Rigid Structure from Motion Problems

Int J Comput Vis (2013) 103:240–266
DOI 10.1007/s11263-012-0581-0

Multilinear Factorizations for Multi-Camera Rigid Structure from Motion Problems

Roland Angst · Marc Pollefeys

Received: 28 January 2011 / Accepted: 28 September 2012 / Published online: 13 November 2012

Abstract 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. The present article 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 article 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.

Keywords Computer vision · 3D reconstruction · Structure from motion · Multilinear factorizations · Tensor algebra

1 Introduction

1.1 Related Work and Motivation

Factorization-based solutions to the structure from motion (SfM) problem have been heavily investigated and extended ever since Tomasi's and Kanade's (1992) seminal work about rigid factorizations. 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 (Tron and Vidal 2007), to perspective cameras (Sturm and Triggs 1996), non-rigid objects (Bregler et al. 2000; Torresani et al. 2001; Brand 2001, 2005; Wang et al. 2008), and articulated objects (Yan and Pollefeys 2008; Tresadern and Reid 2005). More flexible methods which can deal with missing data entries in the data matrix have been proposed in order to

R. Angst (✉) · M. Pollefeys
ETH Zürich, Universitätstrasse 6, 8092 Zürich, Switzerland
e-mail: rangst@inf.ethz.ch

M. Pollefeys
e-mail: marc.pollefeys@inf.ethz.ch

[1] Throughout this paper, 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 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.

overcome shortcomings of singular value based decompositions which can not cope with such situations (Bregler et al. 2000; Hartley and Schaffalitzky 2004; Wang et al. 2008; Guerreiro and Aguiar 2002). 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 Sect. 10).

We therefore propose in Sect. 4 to model the data as a tensor rather than 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 (Torresani et al. 2001; Bue and de Agapito 2006) already extended the factorization approach to a two-camera setup and Svoboda et al. (2005) 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 cameras attached to a camera rig whose field of view do not intersect at all).

Sequences from two camera views have also been investigated (Zelnik-Manor and Irani 2006) in order to temporally synchronize the cameras 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 (Daniilidis 1999) or mirror-based (Kumar et al. 2008). 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 (2006) approach 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. 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 cannot 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 (Li and Chellappa 2005; Vidal and Oliensis 2002). 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.

1.2 Paper Overview

The present article 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 article 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 (Angst and Pollefeys 2009) with our analysis of planar rigid motion (Angst and Pollefeys 2010). 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 Sect. 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 (Sect. 7) and the planar rigid motions (Sect. 8).

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.

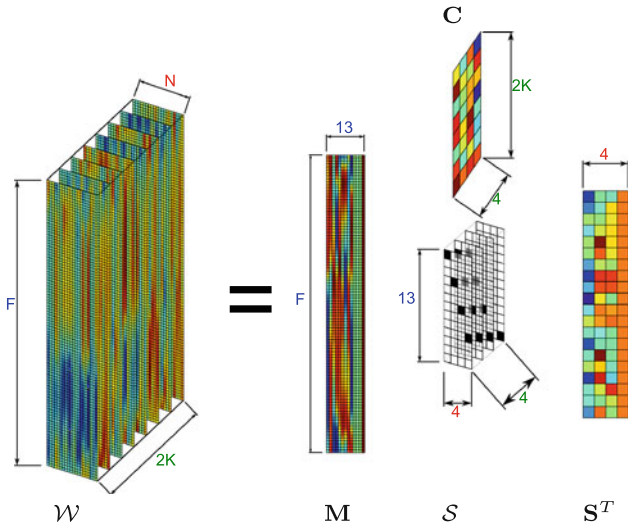


Fig. 1 The data tensor $\mathcal{W}$ containing the feature point tracks can be viewed as a linear combination of third-order tensors each of which equals the outer product of three vectors (so called simple tensors). The coefficients for this linear combination are stored in the core tensor $\mathcal{S}$, 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

The applicability of our algorithm is shown on synthetic as well as on real data sequences in Sects. 11 and 12. 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 paper concludes in Sect. 13 by presenting some ideas for potential future research.

1.3 Why Tensors?

As we will derive in Sect. 4, 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 formulation and interpretation of the data, as visualized in Fig. 1 (The notation used in this figure will described in detail in Sect. 4).

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 (Sect. 4). This avoids a cumbersome formulation on how to reorder the coordinates of feature point tracks into a matrix. Secondly, ambiguities (Sect. 5) and degenerate cases (Sect. 9) 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 Sect. 10) developed especially for multilinear problems can easily be applied if a more general framework and notation is at hand.

2 Notation

The following notation will be used throughout the paper. Matrices are denoted with bold upper case letters $\mathbf{A}$ whereas vectors are bold lower case $\mathbf{a}$. We use calligraphic letters $\mathcal{A}$ for tensors. Matrices built up from multiple submatrices are enclosed whenever possible by square brackets $[\cdots]$, vectors built from multiple subvectors by round brackets $\binom{\cdot}{\cdot}$. The identity matrix of dimension $D \times D$ is denoted as $\mathbf{I}_D$. $\mathbf{A}^*$ denotes the Moore–Penrose pseudo-inverse of matrix $\mathbf{A}$. 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 $\mathbb{P}_{\mathbf{A}}^{\perp} = \mathbf{I} - \mathbb{P}_{\mathbf{A}}$. A matrix whose columns span the orthogonal complement of the columns of matrix $\mathbf{A}$ is denoted as $[\mathbf{A}]_{\perp}$. Concatenation of multiple matrices indexed with a sub- or superscript 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 $\searrow_k$ stacks multiple matrices $\mathbf{A}_k$ into a block-diagonal matrix $[\ \searrow_k \ A_k]$. 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 $\mathbf{a}$.

K is the total number of static cameras, $k \in \{1, \ldots, K\}$ denotes one specific camera, F is the total number of observed frames and $f \in \{1, \ldots, F\}$ 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 $\hat{\mathbf{A}}$.

3 Multilinear Algebra

Concepts from tensor calculus will be introduced in this section. More specifically the mode-i product and the Tucker tensor decomposition (Tucker 1966) are defined and several relationships between tensors, the Kronecker product $\otimes$, and the vec $(\cdot)$-operator are stated. We refer to (Lathauwer et al. 2000; Magnus and Neudecker 1999; Kolda and Bader 2009) for an introductory text on multilinear algebra, tensor operations and decomposition.

3.1 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 (first-order tensors) and matrices (second-order tensors). A tensor of order n can be thought of as a n-dimensional array of numbers. Varying the ith 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 nth-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 ith-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) (Carroll and Chang 1970; Harshman 1970) and the Tucker decomposition (Tucker 1966). A more recent decomposition (Lathauwer et al. 2000), 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 nth-order tensor $\mathcal{A}$ expresses the tensor as n mode-i products between a smaller core tensor $\mathcal{S} \in \mathbb{R}^{r_1 \times \ldots r_n}$ and n matrices $\mathbf{M}_i \in \mathbb{R}^{I_i \times r_i}$

$$\mathcal{A} = \mathcal{S} \times_1 \mathbf{M}_1 \times_2 \mathbf{M}_2 \times_3 \cdots \times_n \mathbf{M}_n, \tag{1}$$

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 (Lathauwer et al. 2000). 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 Sect. 5.

3.2 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)th block equals the matrix $\mathbf{B}$ scaled by the (i, j)th element of $\mathbf{A}$. This implies for example that the first column of $\mathbf{A} \otimes \mathbf{B}$ equals the vectorized outer product $\operatorname{vec}\left(\mathbf{B}_{:,1}\mathbf{A}_{:,1}^T\right)$ of the first column of $\mathbf{A}$ and $\mathbf{B}$. Here, the vectorization operator $\operatorname{vec}(\mathbf{A})$ has been used which is usually 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 a permutation matrix $\mathbf{T}_{m,n} \in \mathbb{R}^{mn \times mn}$ such that $\operatorname{vec}\left(\mathbf{A}^T\right) = \mathbf{T}_{m,n}\operatorname{vec}(\mathbf{A})$.

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

$$\operatorname{vec}(\mathbf{C}) = \operatorname{vec}\left(\mathbf{A}\mathbf{X}\mathbf{B}^T\right) = [\mathbf{B} \otimes \mathbf{A}]\operatorname{vec}(\mathbf{X}). \tag{2}$$

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

$$[\mathbf{A} \otimes \mathbf{B}][\mathbf{C} \otimes \mathbf{D}] = [\mathbf{AC}] \otimes [\mathbf{BD}]. \quad (3)$$

Closer inspection of Eq. (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

$$\mathcal{A} = \sum_{1 \le i_1 \le r_i, \ldots, 1 \le i_n \le r_n} \mathcal{S}_{i_1,\ldots,i_n} \mathbf{M}_{1,(:,i_1)} \otimes \cdots \otimes \mathbf{M}_{n,(:,i_n)}. \quad (4)$$

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 $\operatorname{vec}(\mathcal{A}) = [\mathbf{M}_1 \otimes \cdots \otimes \mathbf{M}_n] \operatorname{vec}(\mathcal{S})$ (assuming a consistent vectorization of the tensor entries). The previous equations clearly show the relation to Eq. (2): The core tensor generalizes the matrix $\mathbf{X}$ in Eq. (2) by capturing interactions between more than just two subspaces (induced by the column and row span of matrix $\mathbf{C}$ in Eq. (2)).

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 $\mathbf{A} \odot \mathbf{B} = \left[\Rightarrow_i \mathbf{A}_{:,i} \otimes \mathbf{B}_{:,i}\right]$. The Khatri-Rao product also enjoys a product property for matrices of appropriate dimensions $[\mathbf{C} \otimes \mathbf{D}]\,[\mathbf{A} \odot \mathbf{B}] = [\mathbf{CA} \odot \mathbf{DB}]$.

3.3 Solving Multilinear Matrix Equations

The paper is 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 (Magnus and Neudecker 1999) 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 the paper in standard form (these identities will become especially handy in Appendix A and Appendix B).

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

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

$$\begin{aligned} \operatorname{vec}(\mathbf{X} \otimes \mathbf{Y}) &= \left[\mathbf{I}_n \otimes \mathbf{T}_{q,m} \otimes \mathbf{I}_p\right] (\operatorname{vec}(\mathbf{X}) \otimes \operatorname{vec}(\mathbf{Y})) \\ &= \left[\mathbf{I}_n \otimes \mathbf{T}_{q,m} \otimes \mathbf{I}_p\right] \cdot \\ &\quad [\mathbf{I}_{mn} \otimes \operatorname{vec}(\mathbf{Y})] \operatorname{vec}(\mathbf{X}) \quad (5) \\ &= \left[\mathbf{I}_n \otimes \mathbf{T}_{q,m} \otimes \mathbf{I}_p\right] \cdot \\ &\quad \left[\operatorname{vec}(\mathbf{X}) \otimes \mathbf{I}_{pq}\right] \operatorname{vec}(\mathbf{Y}) \quad (6) \end{aligned}$$

(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 $\mathbf{x} = \operatorname{vec}(\mathbf{X})$ and $\mathbf{y} = \operatorname{vec}(\mathbf{Y})$ is

$$\begin{aligned} \mathbf{J}_{\mathbf{x},\mathbf{y}} = &\left[\mathbf{I}_n \otimes \mathbf{T}_{q,m} \otimes \mathbf{I}_p\right] \cdot \\ &\left[\mathbf{I}_{mn} \otimes \operatorname{vec}(\mathbf{Y}), \operatorname{vec}(\mathbf{X}) \otimes \mathbf{I}_{pq}\right]. \end{aligned} \quad (7)$$

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

$$\mathbf{J}_{\mathbf{x},\mathbf{y}} \begin{pmatrix} \operatorname{vec}(\mathbf{X}) \\ \operatorname{vec}(\mathbf{Y}) \end{pmatrix} = \operatorname{vec}(\mathbf{C}). \quad (8)$$

4 Applying Tensor Algebra to SfM Problems

This section applies the techniques introduced in Sect. 3 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 (1992). 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. 1 in mind which shows a graphical illustration of the tensor decomposition of the structure-from-motion data tensor.

4.1 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 nth feature point with homogeneous coordinates $\mathbf{s}_n \in \mathbb{R}^{4 \times 1}$ undergoing a rigid motion $\left[\mathbf{R}_f \ \mathbf{t}_f\right]$ at frame f, onto the kth affine camera axis $\mathbf{c}_k^T \in \mathbb{R}^{1 \times 4}$ reads like

$$\mathcal{W}_{[k,f,n]} = \mathbf{c}_k^T \begin{bmatrix} \mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{s}_n = \operatorname{vec}\left(\begin{bmatrix} \mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0} & 1 \end{bmatrix} \right)^T \left[\mathbf{s}_n \otimes \mathbf{c}_k \right]$$
$$= \left[\operatorname{vec}\left(\mathbf{R}_f\right)^T \ \mathbf{t}_f^T \ 1 \right] \mathcal{S}_{(f)} \left[\mathbf{s}_n \otimes \mathbf{c}_k \right], \tag{9}$$

where the Kronecker product property of Eq. (2) and $\mathcal{W}_{[k,f,n]} = \mathcal{W}_{[k,f,n]}^T \in \mathbb{R}$ was used in the second step. In the last line, we introduced the core tensor $\mathcal{S} \in \mathbb{R}^{4\times13\times4}$ 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

$$\mathcal{S}_{(f)} = \begin{bmatrix} \mathbf{I}_3 \otimes \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times1} \end{bmatrix} & \mathbf{0}_{9\times4} \\ \mathbf{0}_{4\times12} & \mathbf{I}_4 \end{bmatrix} \in \mathbb{R}^{13\times16}. \tag{10}$$

We recall that by Eq. (4), 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 $\mathbf{C} = \left[\Downarrow_k \mathbf{c}_k^T \right] \in \mathbb{R}^{2K\times4}$ (each camera has a x- and y-axis). In a similar way, the tracked feature points can be stacked into a structure matrix $\mathbf{S} = \left[\Rightarrow_n \mathbf{s}_n \right] \in \mathbb{R}^{4\times\sum_k N_k}$. By introducing the motion matrix

$$\mathbf{M} = \left[\Downarrow_f \left[\operatorname{vec}\left(\mathbf{R}_f\right)^T \ \mathbf{t}_f^T \ 1 \right] \right] \in \mathbb{R}^{F\times13}$$

we finally get the equation for the coordinates of the trajectory of the nth feature point projected onto the kth 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 third-order data tensor $\mathcal{W} \in \mathbb{R}^{2K\times F\times\sum_k N_k}$ flattened along the temporal mode f

$$\mathcal{W}_{(f)} = \left[\Rightarrow_{n,k} \mathcal{W}_{[k,:,n]} \right] = \mathbf{M}\mathcal{S}_{(f)}[\mathbf{S}^T \otimes \mathbf{C}]^T. \tag{11}$$

This leads to the following

Observation 1 *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*

$$\mathbf{M} = \left[\Downarrow_f \left[vec\left(\mathbf{R}_f\right)^T \ \mathbf{t}_f^T \ 1 \right] \right] \in \mathbb{R}^{F\times13}, \tag{12}$$

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

Equation (11) 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 third order tensor $\mathcal{S} \in \mathbb{R}^{4\times13\times4}$ 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:

$$\mathcal{W} = \mathcal{S} \times_f \mathbf{M} \times_k \mathbf{C} \times_n \mathbf{S}^T \tag{13}$$

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.

4.2 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

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

Let the columns of $\mathbf{V} = [\mathbf{a}]_\perp \in \mathbb{R}^{3\times2}$ 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

$$\begin{pmatrix} \operatorname{vec}\left(\mathbf{R}_{\mathbf{a},\alpha}\right) \\ \operatorname{vec}(\mathbf{V}\mathbf{t}) \\ 1 \end{pmatrix} = \begin{bmatrix} \operatorname{vec}(\mathbf{I}_3) & \operatorname{vec}\left(\mathbf{a}\mathbf{a}^T\right) & \operatorname{vec}\left([\mathbf{a}]_\times\right) & \mathbf{0}_{9\times2} \\ \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{V} \\ 1 & 1 & 0 & \mathbf{0}_{1\times2} \end{bmatrix} \begin{pmatrix} \cos\alpha \\ 1-\cos\alpha \\ \sin\alpha \\ \mathbf{t} \end{pmatrix}, \tag{14}$$

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 $\operatorname{vec}\left(\mathbf{a}\mathbf{a}^T\right)$ 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 three 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 Eq. (14) into the motion matrix of Eq. (12) we get

$$\mathbf{M} = [\Downarrow_f (\cos\alpha_f, (1-\cos\alpha_f), \sin\alpha_f, \mathbf{t}_f^T)] \cdot \begin{bmatrix} \text{vec}(\mathbf{I}_3)^T & \mathbf{0}_{1\times 3} & 1 \\ \text{vec}(\mathbf{a}\mathbf{a}^T)^T & \mathbf{0}_{1\times 3} & 1 \\ \text{vec}([\mathbf{a}]_\times)^T & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{2\times 9} & \mathbf{V}^T & \mathbf{0}_{2\times 1} \end{bmatrix}, \tag{15}$$

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. (15) with Eq. (11), the temporally constant matrix built from the axis of rotation and plane of rotation can be absorbed into a new core tensor

$$\mathcal{C}_{(f)} = \begin{bmatrix} \text{vec}(\mathbf{I}_3)^T & \mathbf{0}_{1\times 3} & 1 \\ \text{vec}(\mathbf{a}\mathbf{a}^T)^T & \mathbf{0}_{1\times 3} & 1 \\ \text{vec}([\mathbf{a}]_\times)^T & \mathbf{0}_{1\times 3} & 0 \\ \mathbf{0}_{2\times 9} & \mathbf{V}^T & \mathbf{0}_{2\times 1} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{I}_3 \otimes [\mathbf{I}_3 \ \mathbf{0}_{3\times 1}] & \mathbf{0}_{9\times 4} \\ \mathbf{0}_{4\times 12} & \mathbf{I}_4 \end{bmatrix}}_{=\mathcal{S}_{(f)}} \in \mathbb{R}^{5\times 16} \tag{16}$$

and the number of columns in the new planar motion matrix

$$\mathbf{M} = [\Downarrow_f (\cos\alpha_f, (1-\cos\alpha_f), \sin\alpha_f, \mathbf{t}_f^T)] \tag{17}$$

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 $\mathbf{M}\in\mathbb{R}^{F\times 5}$ (see Eq. (17)) and core tensor $\mathcal{C}\in\mathbb{R}^{4\times 5\times 4}$ as given in Eq. (16). We summarize these findings in

Observation 2 *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 $\mathbf{V} = [\mathbf{a}]_\perp \in \mathbb{R}^{3\times 2}$) 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*

$$\mathbf{M} = [\Downarrow_f (\cos\alpha_f, (1-\cos\alpha_f), \sin\alpha_f, \mathbf{t}_f^T)] \in \mathbb{R}^{F\times 5}, \tag{18}$$

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

5 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 structure-from-motion problems. Thanks to the tensor formulation, the derivation of such ambiguities is rather straightforward, at least for the general rigid motion case. Note that these gauge freedoms will lead to rank deficient linear systems (e.g.see Sect. 7) 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 Sect. 8 is strongly based on exploiting the ambiguities due to the inherent gauge freedom in order to derive a closed-form solution.

5.1 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 Sect. 3.1). This would obviously result in changing the entries of the known core tensor (Eq. (10)). 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

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

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

$$\mathcal{W}_{[:,f,:]} = \mathbf{C}\mathbf{Q}_C^{-1}\mathbf{Q}_C\begin{bmatrix} \mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}\mathbf{Q}_S\mathbf{Q}_S^{-1}\mathbf{S}. \tag{19}$$

In tensor notation, this equation looks like

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

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

$$\mathbf{Q}_M = \mathcal{S}_{(f)}[\mathbf{Q}_S^{-T}\otimes\mathbf{Q}_C^{-1}]^T\mathcal{S}_{(f)}^T. \tag{20}$$

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. (9) is a product involving three factors and hence two ambiguities arise between these factors as shown in Eq. (19).

5.2 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

$$\mathcal{W}_{[:,f,:]} = \mathbf{C}\mathbf{Q}_C^{-1}\mathbf{Q}_C \begin{bmatrix} \mathbf{R}_{\mathbf{a},\alpha_f} & \mathbf{V}\mathbf{t}_f \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \mathbf{Q}_S\mathbf{Q}_S^{-1}\mathbf{S}. \tag{21}$$

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 $\mathbf{Z}_{\mathbf{a},\lambda} = \begin{bmatrix}\mathbf{V} & \mathbf{a}\end{bmatrix} \text{diag}\,(\mathbf{I}_2, \lambda) \begin{bmatrix}\mathbf{V} & \mathbf{a}\end{bmatrix}^T$ 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 $\mathcal{C}_{\mathbf{a},\mathbf{V}} \times_k \mathbf{Q}_C \times_n \mathbf{Q}_S^T = \mathcal{C}_{\mathbf{R}\mathbf{a},\mathbf{R}\mathbf{V}}$ 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.

6 Rank-4 versus Rank-8 Factorization

We first state the classical factorization approach for rigid motions for one single camera (Tomasi and Kanade 1992) 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

$$\mathbf{W} = \left[\Downarrow_f \mathbf{C}_f\right][\Rightarrow_n \mathbf{s}_n] \in \mathbb{R}^{2F\times N}, \tag{22}$$

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. (22), 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. (12). 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

$$\mathbf{W} = \mathbf{U}\Lambda\mathbf{V}^T = [\mathbf{U}\Lambda^{\frac{1}{2}}][\Lambda^{\frac{1}{2}}\mathbf{V}^T] = \hat{\mathbf{P}}\hat{\mathbf{S}} \tag{23}$$

results in an affine reconstruction of the moving camera matrices $\hat{\mathbf{P}} = \mathbf{U}\Lambda^{\frac{1}{2}}$ and the structure matrix $\hat{\mathbf{S}} = \Lambda^{\frac{1}{2}}\mathbf{V}^T$. 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. (Brand 2001) 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 Akhter et al.'s (2011) 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. (11) 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 (Xiao et al. 2004) for some insights on how to compute the corrective transformation in closed-form in the case of the non-rigid blend shape formulation.

7 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 third-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. (13). 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 (Tucker 1966) 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, such as bioinformatics, signal processing, and collaborative filtering. We think this is an interesting route of research and needs further investigation. We only glimpse at the tip of the iceberg of iterative techniques in Sect. 10 bearing in mind that there are many different approaches to attack factorization problems with general patterns of missing entries (see e.g. Aguiar et al. (2008) 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). This paper however focuses on how a certain pattern of missing entries can be combined 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 Algorithm 1 from high level whereas the upcoming sections provide a detailed description of each individual step.

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 $\mathbf{S} \in \mathbb{R}^{4 \times \sum_k N_k}$
Rank-13 factorization according to Eq. (24) ; // Sec. 7.1
Stratified upgrade ; // Sec. 7.2
  Affine upgrade ; // Sec. 7.2.1
  Affine cameras ; // Sec. 7.2.2
  Enforcing Kronecker-structure ; // Sec. 7.2.4
  Similarity upgrade ; // Sec. 7.2.5

Algorithm 1: Closed-form algorithm for general rigid motions.

7.1 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. 2, only

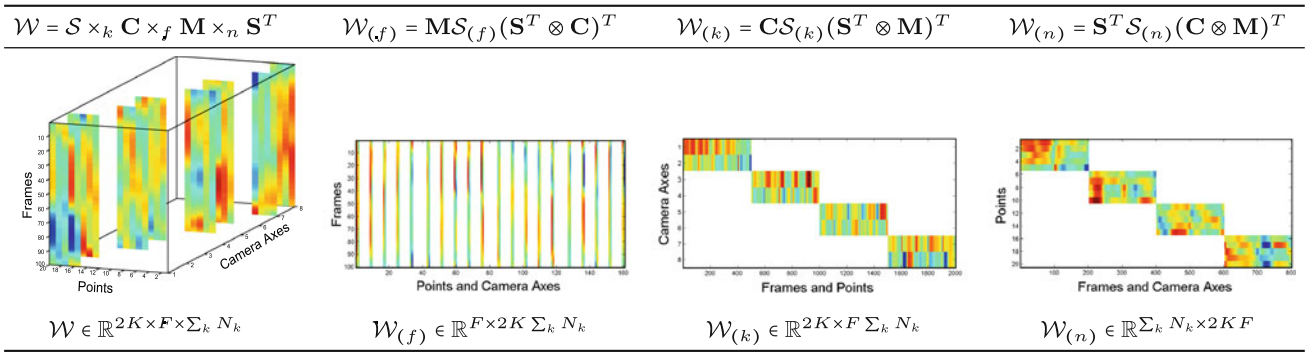


Fig. 2 If there are no feature point correspondences between different camera views then the data tensor $\mathcal{W}$ has many missing data entries (missing data entries are visualized transparently). Along the third order data tensor itself, its three flattened versions are shown as well. Note that 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 consistent subspace for the camera matrices or the coordinates of the points

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 $\mathbf{W} = [\Rightarrow_k \mathbf{W}^k]$. A rank-13 matrix factorization (e.g. with SVD $\mathbf{W} = \mathbf{U}\Sigma\mathbf{V}^T$) reveals the two factors $\hat{\mathbf{M}} = \mathbf{U} \in \mathbb{R}^{F\times 13}$ and $\hat{\mathbf{A}} = \Sigma\mathbf{V}^T \in \mathbb{R}^{13\times 2\sum_k N_k}$ which fulfill

$$\mathbf{W} = \mathbf{M}\mathcal{S}_{(f)}\left[\Downarrow_k \mathbf{S}^{k^T} \otimes \mathbf{C}^k\right]^T = [\hat{\mathbf{M}}\mathbf{Q}][\mathbf{Q}^{-1}\hat{\mathbf{A}}]. \tag{24}$$

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 $\mathbf{Q}^{-1}\hat{\mathbf{A}}$.

7.2 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 Chap. 10.4 in Hartley and Zisserman (2004) 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.

7.2.1 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 $\mathbf{Q}_{aff} = \left[\left[\mathbf{q}_{aff}\right]_\perp \mathbf{q}_{aff}\right]$, where $\left[\mathbf{q}_{aff}\right]_\perp$ denotes an orthogonal basis for the nullspace of $\mathbf{q}_{aff}$. We cannot gain any more knowledge by analyzing $\hat{\mathbf{M}}$, yet. We therefore turn our attention toward $\hat{\mathbf{A}}$.

7.2.2 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

$$\tilde{\mathbf{A}} = \mathbf{Q}_{kron}\mathbf{Q}^{-1}_{aff}\hat{\mathbf{A}} = \mathcal{S}_{(f)}\left[\Downarrow_k \tilde{\mathbf{S}}^{k^T} \otimes \tilde{\mathbf{C}}^k\right]^T. \tag{25}$$

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. (20) implies that the last column of $\mathbf{Q}_{kron}$ equals $(\mathbf{0}_{1\times 12}, 1)^T$ and thus only the first 12 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. (25) 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 Appendix A for details).

7.2.3 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 $\mathbf{K}^k = \text{diag}\,(s_k, s_k, 1)$ and we can use self-calibration methods using the dual quadric, see Hartley and Zisserman (2004), Sect. 19.3. Since we computed affine camera matrices, the dual quadric is of the form $\Omega_\infty^* = \begin{bmatrix} \Omega_{3\times 3} & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 4} \end{bmatrix}$, which means we only require at least three rather than four cameras observing the scene to compute symmetric Ω_∞^*. 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.

7.2.4 Enforcing the Kronecker-Structure

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

$$\tilde{\mathbf{A}} = \mathcal{S}_{(f)}[\Downarrow_k \tilde{\mathbf{S}}^{k\,T} \otimes \tilde{\mathbf{C}}^k]^T = \mathbf{Q}_{kron}\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}} \tag{26}$$

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 (Appendix B provides details on how to set up and solve this system).

7.2.5 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 Sect. 7.2.1, 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. (20). We are therefore looking for affine transformations $\mathbf{Q}_C$ and $\mathbf{Q}_S$ such that

$$\begin{aligned}\mathbf{M} &= \left[\Downarrow_f \left((\text{vec}\,(\mathbf{R}_f))^T, \mathbf{t}_f^T, 1\right)\right] \\ &= \underbrace{\left[\Downarrow_f \left((\text{vec}\left(\tilde{\mathbf{R}}_f\right))^T, \tilde{\mathbf{t}}_f^T, 1\right)\right]}_{=\tilde{\mathbf{M}}=\hat{\mathbf{M}}\mathbf{Q}_{aff}\mathbf{Q}_{kron}^{-1}} \underbrace{\mathcal{S}_{(f)}[\mathbf{Q}_S^T \otimes \mathbf{Q}_C]^T \mathcal{S}_{(f)}^T}_{=\mathbf{Q}_{metric}}\end{aligned} \tag{27}$$

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), the above equation Eq. (27) is equivalent to the set of equations

$$\mathbf{R}_f = \mathbf{T}_C\tilde{\mathbf{R}}_f\mathbf{T}_S^{-1} \tag{28}$$

$$\mathbf{t}_f = \mathbf{T}_C\tilde{\mathbf{R}}_f\mathbf{t}_S + \mathbf{T}_C\tilde{\mathbf{t}}_f + \mathbf{t}_C \tag{29}$$

for $f \in \{1 \ldots F\}$. Eq. (28) 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

$$\left[\mathbf{T}_C\tilde{\mathbf{R}}_f\right]^T\left[\mathbf{T}_C\tilde{\mathbf{R}}_f\right] = \left[\mathbf{R}_f\mathbf{T}_S\right]^T\left[\mathbf{R}_f\mathbf{T}_S\right] \tag{30}$$

$$= \tilde{\mathbf{R}}_f^T\mathbf{T}_C^T\mathbf{T}_C\tilde{\mathbf{R}}_f = \mathbf{T}_S^T\mathbf{T}_S. \tag{31}$$

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 one presented in (Brand 2001, 2005) 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. (29) 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.

Note that 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 Sect. 11 are computed without the camera self-calibration approach of Sect. 7.2.3.

8 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, Algorithm 2 provides an overview of the multiple sequential steps whereas the following sections give detailed explanations.

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 $\mathbf{S} \in \mathbb{R}^{4\times\sum_k N_k}$
Rank-5 factorization according to Eq. (32) ; // Sec. 8.1
Trigonometric upgrade ; // Sec. 8.2
Metric upgrade in plane of motion ; // Sec. 8.3
  Projection onto plane of motion ; // Sec. 8.3.1
  Per-camera metric reconstruction inside plane of motion ; // Sec. 8.3.2
  Registration in common coordinate system in plane of motion ; // Sec. 8.3.3
  Metric upgrade of component outside of plane of motion ; // Sec. 8.3.4

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

8.1 Rank-5 Factorization

In a similar spirit to Sect. 7, 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 $\text{span}(\mathbf{M}) = \text{span}\left(\hat{\mathbf{M}}\right)$ of the motion matrix

$$\mathbf{W} = \hat{\mathbf{M}}\hat{\mathbf{A}} = \underbrace{\left[\Downarrow_f \cos\alpha_f \;\; 1-\cos\alpha_f \;\; \sin\alpha_f \;\; t_{f,1} \;\; t_{f,2}\right]}_{=\hat{\mathbf{M}}\mathbf{Q}} \cdot \underbrace{\mathcal{C}_{(f)}\left[\Rightarrow_k \mathbf{S}^k \otimes \mathbf{C}^{k\,T}\right]}_{=\mathbf{Q}^{-1}\hat{\mathbf{A}}}, \tag{32}$$

where we have introduced the corrective transformation $\mathbf{Q} \in \mathbb{R}^{5\times 5}$ 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. (17) should sum to the constant one vector. Hence, only a rank 4 factorization of the data matrix $\mathbf{W}$ 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 $\mathbf{Q} = \mathbf{Q}_{trig}\mathbf{Q}^{-1}_{orient}\mathbf{Q}^{-1}_{transl}$ 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 (Fig. 3).

8.2 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 $\hat{\mathbf{M}}_{[f,:]}\mathbf{q}_i\mathbf{q}_i^T\hat{\mathbf{M}}^T_{[f,:]} = \mathbf{M}_{[f,i]}{}^2$ we have

$$\begin{aligned} 1 &= \hat{\mathbf{M}}_{[f,:]}\left[(\mathbf{q}_1+\mathbf{q}_2)(\mathbf{q}_1+\mathbf{q}_2)^T\right]\hat{\mathbf{M}}^T_{[f,:]} \\ &= \left(\cos\alpha_f + (1-\cos\alpha_f)\right)^2 \\ 1 &= \hat{\mathbf{M}}_{[f,:]}\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right]\hat{\mathbf{M}}^T_{[f,:]} = \cos^2\alpha_f + \sin^2\alpha_f. \end{aligned} \tag{33}$$

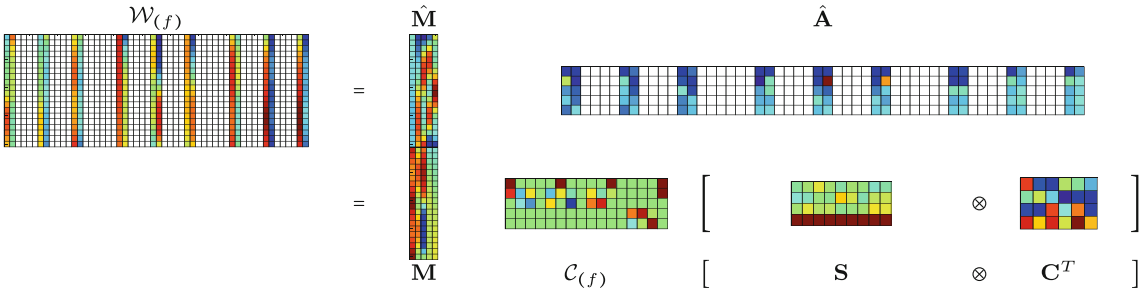


Fig. 3 Visual representation of the rank-5 factorization. Missing data entries due to missing correspondences between different cameras are depicted transparently

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 $b \in \mathbb{R}$:

$$\begin{aligned} 1 &= \hat{\mathbf{M}}_{[f,:]}\left[(\mathbf{q}_1+\mathbf{q}_2)(\mathbf{q}_1+\mathbf{q}_2)^T\right]\hat{\mathbf{M}}_{[f,:]}^T \\ &= \hat{\mathbf{M}}_{[f,:]}\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right]\hat{\mathbf{M}}_{[f,:]}^T \\ &= \hat{\mathbf{M}}_{[f,:]}\left[b\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right]\right. \\ &\quad \left. +(1-b)\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_2\mathbf{q}_2^T\right]\right]\hat{\mathbf{M}}_{[f,:]}^T \end{aligned} \tag{34}$$

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 $b \in \mathbb{R}$). Appendix C 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 $\mathbf{Q}_{trig} = \left[\mathbf{q}_1\ \mathbf{q}_2\ \mathbf{q}_3\ \left[\mathbf{q}_1,\mathbf{q}_2,\mathbf{q}_3\right]_\perp\right]$ 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-diagonal structure with an identity matrix of dimension 3 in the upper left corner. The inverse of such an upper block-diagonal matrix has exactly the same non-zero pattern, i.e.

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

8.3 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

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{Q}_{3\times 2} \\ \mathbf{0}_{2\times 3} & \mathbf{Q}_{2\times 2} \end{bmatrix}\tilde{\mathbf{A}} = \mathcal{C}_{(f)}\left[\Rightarrow_k \mathbf{S}^k \otimes \mathbf{C}^{kT}\right]. \tag{35}$$

This is a particularly tricky instance of a bilinear system of equations in $\mathbf{Q}_{3\times 2}$, $\mathbf{Q}_{2\times 2}$, $\mathbf{S}^k$, and $\mathbf{C}^k$. Based on our experiences, even algebraic computer software does not succeed in finding a closed-form solution. Nevertheless, we succeeded in deriving manually a solution using geometric intuition and reasoning.

8.3.1 Projection onto Plane of Motion

Equation (35) together with the known matrix $\mathcal{C}_{(f)}$ in Eq. (16) tells that $\tilde{\mathbf{A}}_{[4:5,:]} = \left[\Rightarrow_k \mathbf{1}_{1\times N_k} \otimes \left[\mathbf{C}^k_{[:,1:3]}\mathbf{V}\mathbf{Q}_{2\times 2}^{-T}\right]^T\right]$, 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. Appendix D shows that this can be done by subtracting the second from the first row and keeping the third row of Eq. (35)

$$\begin{aligned} &\begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\tilde{\mathbf{A}}_{[1:3,:]} + \underbrace{\begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{=\mathbf{T}_{2\times 2}}\mathbf{Q}_{3\times 2}\,\tilde{\mathbf{A}}_{[4:5,:]} \\ &= \begin{bmatrix} \mathrm{vec}\left(\mathbb{P}_\mathbf{V}\right)^T \\ \mathrm{vec}\left([\mathbf{a}]_\times\right)^T \end{bmatrix}\underbrace{\left[\Rightarrow_k \left[\mathbb{P}_\mathbf{V}\mathbf{S}^k_{[1:3,:]}\right] \otimes \left[\mathbb{P}_\mathbf{V}\mathbf{C}^k_{[:,1:3]}{}^T\right]\right]} \\ &= \begin{bmatrix} \mathrm{vec}\left(\mathbb{P}_\mathbf{V}\right)^T \\ \mathrm{vec}\left([\mathbf{a}]_\times\right)^T \end{bmatrix}. \\ &\quad \left[\Rightarrow_k \left[\mathbb{P}_\mathbf{V}\mathbf{S}^k_{[1:3,:]}\right] \otimes \left[\mathbf{V}\mathbf{Q}_{2\times 2}\right]\left[\mathbf{Q}_{2\times 2}^{-1}\mathbf{V}^T\mathbf{C}^k_{[:,1:3]}{}^T\right]\right]. \end{aligned} \tag{36}$$

In the last step we have used $\mathbb{P}_\mathbf{V} = \mathbf{V}\mathbf{Q}_{2\times2}\mathbf{Q}_{2\times2}^{-1}\mathbf{V}^T$ and the parenthesis in the last term should stress out that for all the cameras the term $\mathbf{Q}_{2\times2}^{-1}\mathbf{V}^T{\mathbf{C}_{[:,1:3]}^k}^T$ 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 $\mathbf{T}_{2\times2}$ and $\mathbf{Q}_{2\times2}$.

8.3.2 Per-Camera Reconstruction in the Plane of Rotation

Equation (36) 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 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 $\mathbf{T}_{2\times2}$ and $\mathbf{Q}_{2\times2}$ 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. $\frac{1}{N}\mathbf{S}\mathbf{1}_{N\times1} = \mathbf{0}$. We will make the same choice $\frac{1}{N_k}\mathbf{S}^k\mathbf{1}_{N_k\times1} = \mathbf{0}$ 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. (36) and with the Kronecker product property of Eq. (3) we get

$$\begin{aligned}&\left[\begin{bmatrix}1 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}\tilde{\mathbf{A}}_{[1:3,:]}^k + \mathbf{T}_{2\times2}\tilde{\mathbf{A}}_{[4:5,:]}^k\right]\left[\frac{1}{N_k}\mathbf{1}_{N_k\times1}\otimes\mathbf{I}_2\right]\\ &\quad= \begin{bmatrix}\mathrm{vec}\,(\mathbb{P}_\mathbf{V})^T\\ \mathrm{vec}\,([\mathbf{a}]_\times)^T\end{bmatrix}\cdot\\ &\quad\left(\mathbb{P}_\mathbf{V}\mathbf{S}_{[1:3,:]}^k\frac{1}{N_k}\mathbf{1}_{N_k\times1}\right)\otimes\left(\mathbb{P}_\mathbf{V}{\mathbf{C}_{[:,1:3]}^k}^T\right) = \mathbf{0}_{2\times2}.\end{aligned}\tag{37}$$

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 $\mathbf{T}_{2\times2}$.

[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 is only one planar similarity transformation left from the two mentioned in Sect. 5.

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 $\begin{pmatrix}1 & 0\end{pmatrix}\mathbf{C}_{[:,1:3]}^k\mathbf{V}$ of the kth camera matrix equals the known row $\begin{pmatrix}1 & 0\end{pmatrix}\mathbf{C}_{[:,1:3]}^k\mathbf{V}\mathbf{Q}_{2\times2}^{-T}$. Such a choice poses two constraints on $\mathbf{Q}_{2\times2}$

$$\begin{aligned}\begin{pmatrix}1 & 0\end{pmatrix}\mathbf{C}_{[:,1:3]}^k\mathbf{V} &= \begin{pmatrix}1 & 0\end{pmatrix}\left[\mathbf{C}_{[:,1:3]}^k\mathbf{V}\mathbf{Q}_{2\times2}^{-T}\right]\\ &= \begin{pmatrix}1 & 0\end{pmatrix}\left[\mathbf{C}_{[:,1:3]}^k\mathbf{V}\mathbf{Q}_{2\times2}^{-T}\right]\mathbf{Q}_{2\times2}^T.\end{aligned}\tag{38}$$

Knowing $\mathbf{T}_{2\times2}$ as well as the first row of $\mathbf{C}_{[:,1:3]}^k\mathbf{V}$ 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 $\mathbb{P}_\mathbf{V}\mathbf{S}_{[1:3,:]}^k$. After solving this system, only the entries of $\mathbf{Q}_{2\times2}$ are not yet known. The two linear constraints of Eq. (38) 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. (36) 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.

8.3.3 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 $\mathbf{C}_{[:,1:3]}^k\mathbb{P}_\mathbf{V}$ 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 $\mathbf{C}_{[:,1:3]}^k\mathbf{V}\mathbf{Q}_{2\times2}^{-T}$ 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

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

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 $\mathbf{E}\Lambda\mathbf{E}^T = \mathbf{Q}^T_{2\times 2}\mathbf{Q}_{2\times 2}$ provides a mean to recover $\mathbf{Q}_{2\times 2} = \mathbf{E}^T\Lambda^{\frac{1}{2}}$ 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.

8.3.4 Orthogonality and Equality of Norm Constraints

As has been previously mentioned in Sect. 5.2, 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

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

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 second-order polynomial system of equations with three independent equations in the three unknowns $\mathbf{w}^k$ and λ_k. Straight-forward algebraic manipulation will reveal the closed-form solution to this system (see Appendix E for details). Once $\mathbf{w}^k$ is recovered, the camera matrix is given by solving the linear system $\mathbf{C}^k_{[:,1:3]}[\mathbb{P}_\mathbf{V}, \mathbf{a}] = \left[\mathbf{C}^k_{[:,1:3]}\mathbb{P}_\mathbf{V}, \mathbf{w}^k\right]$. 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

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

in the three unknowns of $\mathbf{Q}^T_{2\times 2}\mathbf{Q}_{2\times 2}$, 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 $\mathbf{Q}_{2\times 2}$ 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.

8.3.5 Final Step

Once the first three columns of the camera matrices are known in an Euclidean reference frame, the first three rows in Eq. (35) become linear in the unknowns $\mathbf{Q}_{3\times 2}$, $\mathbf{S}$, 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 can be compensated by the planar motion) and K degrees of freedom for the per-camera translation ambiguities along the axis of rotation.

9 Minimal Configurations

Our two algorithms require the union of all the feature trajectories spanning the complete 13- respectively 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.

The derivation of Eq. (13) assumed a rank-4 structure matrix $\mathbf{S}$. This assumption is violated if the observed object is planar. Our algorithm currently cannot 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 Sect. 11.2).

As mentioned in Sect. 8, 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. A detailed look at the algorithm for the rank-13 factorization however 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.

9.1 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 $i = 1,\ldots,n$. 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 $\mathbf{C} = [\mathbf{C}_1,\ldots,\mathbf{C}_{i-1}]$ 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

$$\mathbf{A}_k = [\Rightarrow_{i<k} \mathbf{A}_i]\left[\Downarrow_{i<k} \mathbf{X}_i\right] = \sum_{i<k} \mathbf{A}_i \mathbf{X}_i \tag{39}$$

$$\mathbf{B}_k = [\Rightarrow_{i<k} \mathbf{B}_i]\left[\Downarrow_{i<k} \mathbf{Y}_i\right] = \sum_{i<k} \mathbf{B}_i \mathbf{Y}_i. \tag{40}$$

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

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

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 Eqs. (39) and (40) 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. (39) 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

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

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.

Table 1 Minimal cases: 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 Sect. 7.2 succeeds in computing a valid factorization or not (summarized in the last row)

# points per camera	$(3, N_2 \geq 4)$	$(4, 4)$	$(1, 3, 3)$	$(2, 3, 3)$	$(2, 2, N_3 \geq 4)$	$(2, 2, 2, 2)$	$(2, 2, 2, 3)$	$(2, 2, 2, 2, 2)$
rank $(\mathbf{A}) \stackrel{?}{=} 13$	rank $(\mathbf{A}) \leq 12$	$13 = 13$	$13 = 13$	$13 = 13$	$13 = 13$	$13 = 13$	$13 = 13$	$13 = 13$
Sect. 7.2 applicable	×	✓	×	✓	×	×	✓	✓

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 Sect. 7.2.2). 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 $N_3 = 1$)

9.2 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

$$
\begin{aligned}
\mathbf{S}_1 \otimes \mathbf{C}_1 &\rightarrow 4 \cdot 2\text{basis vectors}\\
\mathbf{s}_2 \otimes \mathbf{c}_2 &= \mathbf{S}_1\mathbf{x} \otimes \mathbf{c}_2 \rightarrow 1 \text{ additional basis vector}\\
\mathbf{s}_3 \otimes \mathbf{c}_3 &= \mathbf{s}_3 \otimes [\mathbf{C}_1, \mathbf{c}_2]\begin{pmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{pmatrix}\\
&= \mathbf{S}_1\mathbf{x} \otimes \mathbf{C}_1\mathbf{y}_1 + \mathbf{s}_2 \otimes \mathbf{c}_2\mathbf{y}_2\\
&= [\mathbf{S}_1 \otimes \mathbf{C}_1][\mathbf{x} \otimes \mathbf{y}_1] + [\mathbf{s}_2 \otimes \mathbf{c}_2][\mathbf{I}_4 \otimes \mathbf{y}_2]\\
&\rightarrow \text{no new basis vectors.}
\end{aligned}
$$

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 Table 1. Note that for some cases, even though the rank of their mode-f subspace is 13, the computation of the affine cameras (Sect. 7.2.2) still fails because the points do not provide enough linear independent constraints for solving the linear system of Eq. (25) due to reasons akin to the one shown above. Interestingly, experiments showed that direct nonlinear iterative minimization such as the ones presented in Sect. 10 sometimes succeeded in solving

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

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

10 Iterative Optimization

Even though the algorithmic part of this paper 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 Sect. 7.2 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. (13) 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, we provide two algorithms which proved to work very well in our application. (Chen 2008) recently analyzed several iterative algorithms for bilinear matrix factorization problems with missing entries, amongst others the alternating least squares (ALS) method and the Wiberg algorithm. 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 Sect. 3.3, it should be possible to apply these ideas to other multilinear matrix and tensor equations.

10.1 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 be rewritten as a sum of squares over all the elements of the matrix

$$\begin{aligned}\Phi &= \frac{1}{2}\left\|\mathbf{M}\mathcal{S}_{(f)}\left(\mathbf{S}^T\otimes\mathbf{C}\right)^T-\mathcal{W}_{(f)}\right\|_F^2 \\ &= \frac{1}{2}\sum_{f,k,n}\left(\mathbf{M}_{[f,:]}\mathcal{S}_{(f)}\left(\mathbf{S}^T_{[:,n]}\otimes\mathbf{C}_{[k,:]}\right)^T-\mathcal{W}_{[f,k,n]}\right)^2.\end{aligned} \tag{41}$$

We introduce $\mathbf{w}_{(f)}=\operatorname{vec}\left(\mathcal{W}_{(f)}\right)$, $\mathbf{w}_{(k)}=\operatorname{vec}\left(\mathcal{W}_{(k)}\right)$, and $\mathbf{w}_{(n)}=\operatorname{vec}\left(\mathcal{W}_{(n)}\right)$ in addition to $\mathbf{m}=\operatorname{vec}(\mathbf{M})$, $\mathbf{c}=\operatorname{vec}(\mathbf{C})$, and $\mathbf{s}=\operatorname{vec}(\mathbf{S})$. The sum of squares problem in Eq. (41) is then equivalent to $\Phi=\frac{1}{2}\|\mathbf{r}\|_2^2$ with the residuum

$$\begin{aligned}\mathbf{r} &= \left[\left(\mathbf{S}^T\otimes\mathbf{C}\right)\mathcal{S}^T_{(f)}\otimes\mathbf{I}_F\right]\mathbf{m}-\mathbf{w}_{(f)} \\ &= \left[\mathbf{\Pi}\right]_{k\to f}\left(\left[\left(\mathbf{S}^T\otimes\mathbf{M}\right)\mathcal{S}^T_{(k)}\otimes\mathbf{I}_{2K}\right]\mathbf{c}-\mathbf{w}_{(k)}\right) \\ &= \left[\mathbf{\Pi}\right]_{n\to f}\left(\left[\left(\mathbf{M}^T\otimes\mathbf{C}\right)\mathcal{S}^T_{(n)}\otimes\mathbf{I}_N\right]\mathbf{s}-\mathbf{w}_{(n)}\right),\end{aligned} \tag{42}$$

where Eq. (2) has been used to expose the unknown vectors $\mathbf{m}$, $\mathbf{c}$, and $\mathbf{s}$ and the row-permutation matrices $\Pi_{k\to f}$ and $\Pi_{n\to f}$ reorder the rows of the residuum to match those of Eq. (42). For later reference, we also note the partial derivatives of the residuum

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

With this notation in place, the ALS algorithm 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 Algorithm 3). 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.

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**

$$\begin{aligned}\mathbf{m} &= \arg\min_{\mathbf{m}}\left\|\left[\left(\mathbf{S}^T\otimes\mathbf{C}\right)\mathcal{S}^T_{(f)}\otimes\mathbf{I}_F\right]\mathbf{m}-\mathbf{w}_{(f)}\right\|_2^2 \\ \mathbf{c} &= \arg\min_{\mathbf{c}}\left\|\left[\left(\mathbf{S}^T\otimes\mathbf{M}\right)\mathcal{S}^T_{(k)}\otimes\mathbf{I}_{2K}\right]\mathbf{c}-\mathbf{w}_{(k)}\right\|_2^2 \\ \mathbf{s} &= \arg\min_{\mathbf{s}}\left\|\left[\left(\mathbf{M}^T\otimes\mathbf{C}\right)\mathcal{S}^T_{(n)}\otimes\mathbf{I}_N\right]\mathbf{s}-\mathbf{w}_{(n)}\right\|_2^2\end{aligned}$$

end

Algorithm 3: ALS applied to our trilinear factorization problem

10.2 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.

The Wiberg algorithm is a variation of the Gauss–Newton algorithm with an interleaved variable projection step. Thus, we again minimize the norm $\Phi=\frac{1}{2}\|\mathbf{r}\|_2^2$ of the residuum

$$\mathbf{r} = \left[\left(\mathbf{S}^T\otimes\mathbf{C}\right)\mathcal{S}^T_{(f)}\otimes\mathbf{I}_F\right]\mathbf{m}-\mathbf{w}_{(f)}, \tag{43}$$

but this time Eq. (43) 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 second-order Taylor expansion of the objective function at the current ith iteration is minimized

$$\begin{pmatrix}\mathbf{dc}\\ \mathbf{ds}\end{pmatrix} = \arg\min_{\mathbf{dc},\mathbf{ds}} \left(\Phi_{\mathbf{c}_i,\mathbf{s}_i} + \mathbf{r}^T d_{\mathbf{c}_i,\mathbf{s}_i}\mathbf{r} \begin{pmatrix}\mathbf{dc}\\ \mathbf{ds}\end{pmatrix} + \frac{1}{2}\begin{pmatrix}\mathbf{dc}\\ \mathbf{ds}\end{pmatrix}^T \mathbf{H}_{\mathbf{c}_i,\mathbf{s}_i} \begin{pmatrix}\mathbf{dc}\\ \mathbf{ds}\end{pmatrix} \right) \tag{44}$$

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 solved 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

$$\mathbf{0} = \partial_{\mathbf{m}}\Phi = (\partial_{\mathbf{m}}\mathbf{r})^T \mathbf{r} = \left[\left(\mathbf{S}^T \otimes \mathbf{C}\right) \mathcal{S}_{(f)}^T \otimes \mathbf{I}_F \right]^T \mathbf{r}. \tag{45}$$

As the left-hand side of Eq. (45) 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

$$\begin{aligned}\mathbf{0} &= d_{\mathbf{c},\mathbf{s}} \left((\partial_{\mathbf{m}}\mathbf{r})^T \mathbf{r} \right) \\ &= d_{\mathbf{c},\mathbf{s}} \left((\partial_{\mathbf{m}}\mathbf{r})^T \right) \mathbf{r} + (\partial_{\mathbf{m}}\mathbf{r})^T d_{\mathbf{c},\mathbf{s}}\mathbf{r}.\end{aligned} \tag{46}$$

With a similar justification as in the Gauss–Newton algorithm, the multiplication between the residuum 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

$$d_{\mathbf{c},\mathbf{s}}\mathbf{r} = \partial_{\mathbf{m}}\mathbf{r}\partial_{\mathbf{c},\mathbf{s}}\mathbf{m} + \partial_{\mathbf{c},\mathbf{s}}\mathbf{r} \tag{47}$$

allows to rewrite Eq. (46) as

$$\begin{aligned}\mathbf{0} &= (\partial_{\mathbf{m}}\mathbf{r})^T d_{\mathbf{c},\mathbf{s}}\mathbf{r} \\ &= (\partial_{\mathbf{m}}\mathbf{r})^T \left(\partial_{\mathbf{m}}\mathbf{r}\partial_{\mathbf{c},\mathbf{s}}\mathbf{m} + \partial_{\mathbf{c},\mathbf{s}}\mathbf{r} \right)\end{aligned} \tag{48}$$

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

$$\partial_{\mathbf{c},\mathbf{s}}\mathbf{m} = -\left((\partial_{\mathbf{m}}\mathbf{r})^T \partial_{\mathbf{m}}\mathbf{r} \right)^{-1} (\partial_{\mathbf{m}}\mathbf{r})^T \partial_{\mathbf{c},\mathbf{s}}\mathbf{r} \tag{49}$$

Finally, an approximation of the sought after total derivative is given by (Eqs. (47), (49))

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

In the last equation, the projection matrix $\mathbb{P}_{\partial_{\mathbf{r}}\mathbf{r}}^{\perp}$ onto the orthogonal complement of the columns (i.e. the column nullspace) of $\partial_{\mathbf{m}}\mathbf{r}$ has been introduced. Note that the residuum is also expressible in terms of this projection matrix $\mathbf{r} = -\mathbb{P}_{\partial_{\mathbf{r}}\mathbf{r}}^{\perp}\mathbf{w}$. Now, we have everything in place to state the Wiberg algorithm applied to our trilinear factorization problem (Algorithm 4).

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

Algorithm 4: Wiberg algorithm applied to our trilinear factorization problem

11 Results: Rank-13 Factorization

The steps described in Sects. 7.2.1, 7.2.2, and 7.2.4 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 Sect. 7.2.5. Because the metric upgrade step is based on a least squares formulation, orthogonality constraints are not enforced strictly on the 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].

[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 $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$.

11.1 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 $\frac{1080 \text{ pixels}}{4.035 \text{ mm}}$, 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°). The translation vector of the rigid motions and the feature points were drawn from a normal distribution with a standard deviation of 5 cm. $K = 4$ cameras with focal length $f = 90$ mm were placed randomly 7.5 m 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. 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. 4b shows that our algorithm with non-strict orthogonality constraints even 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

$$\frac{1}{\sqrt{F \sum_k N_k}} \left\| \mathbf{W} - \mathbf{M} \mathcal{S}_{(f)} \left[\Rightarrow_k \mathbf{S}^k \otimes \mathbf{C}^{k^T} \right] \right\|_F \tag{50}$$

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 $m = \frac{61 \text{ mm}}{5 \text{ m}}$ which can be interpreted as choosing a focal length of 61 mm with an average distance between camera and rigid object of 5 m. 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. 4c).

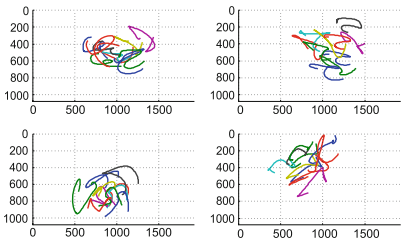
In a third synthetic experiment, the stability of the method w.r.t. the number of points observed per camera is investigated. The data has been generated in the same way as for the first experiment, but this time with $K = 6$ affine cameras. The method has been applied several times to an increasing number of points visible per camera by adding additional feature trajectories (each camera still tracked a different set of points). Figure 5 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 3–4 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. 6, our methods does not seem to depend strongly on the number of cameras.

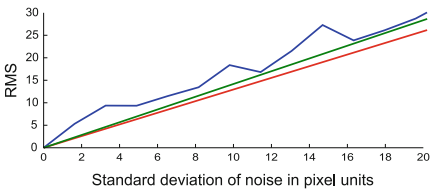
Combining the results from Fig. 6 with Fig. 5, we conclude that the method is considerably robust as soon as there is sufficient redundancy in the input data. For example, the accuracy decreases slightly in the $K = 6$ with $N_k = 2$ case as seen in Fig. 5, however in the case $K = 10$ with $N_k = 2$ shown in Fig. 6 the accuracy stays more or less the same.

11.2 Real Data Sequence

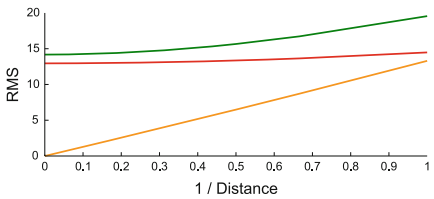
We evaluated our algorithm on a real sequence of a rotating rigid box. The cameras recorded in a resolution of 1920×1080 pixels. In order to ease the tracking we used a template based tracking algorithm (Wagner and Schmalstieg 2007) which provides 5 points per template (the 4 vertices and the middle point). The cameras were not aware of the fact



(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

Fig. 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

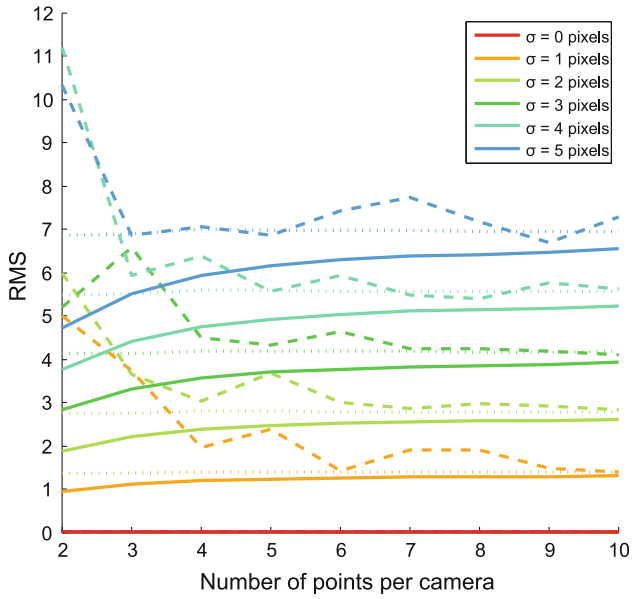


Fig. 5 Synthetic data experiment showing the influence of an increasing number of points observed per camera while holding the number of cameras fixed: The x-axis shows the number of points tracked in each of the $K = 6$ cameras whereas the y-axis shows the resulting RMS. The *continuous lines* show 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

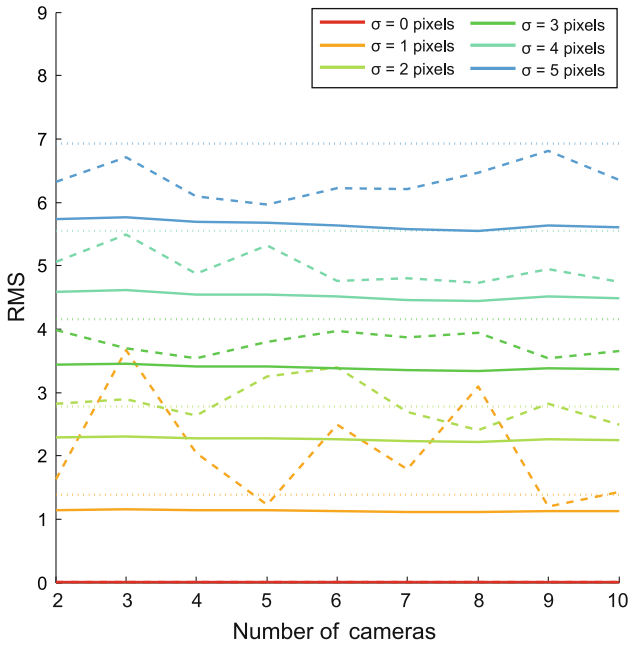


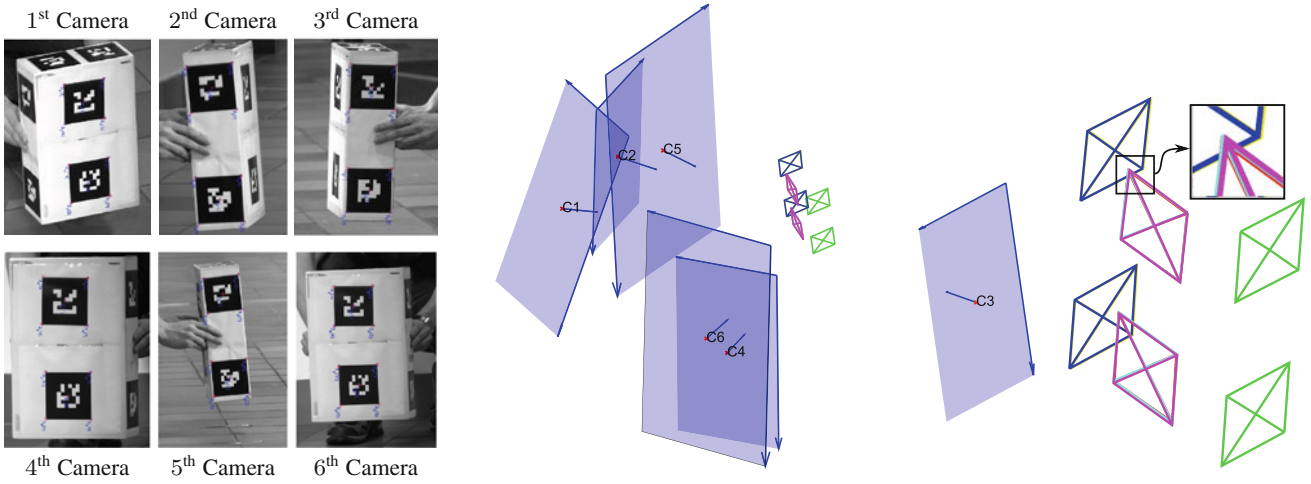
Fig. 6 Synthetic data experiment showing the influence of an increasing number of cameras while holding the total number of points fixed: The x-axis shows the number of cameras whereas the y-axis shows the resulting RMS. The total number of points was fixed to $N = \sum_k N_k = 20$ and the number of 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 in Fig. 5

that they might have tracked the very same feature points (i.e., no correspondences were established between different camera views). Each camera tracked two 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. Figure 7 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 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 Figs. 7 and 8 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. Figure 8 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.

12 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



(a) Example Frames **(b)** 3D view of reconstruction **(c)** Closeup view

Fig. 7 Resulting reconstruction of the real data sequence. **a** The reprojection of feature points (*red circles*) into three camera views along with the ground truth (*blue crosses*) for one specific frame (the frames are cropped in order to fit in the figure). **b** A 3D view of the reconstructed camera poses together with the points of the box at one specific frame. **c** A closeup view of the reconstructed points at this frame (Color figure online)

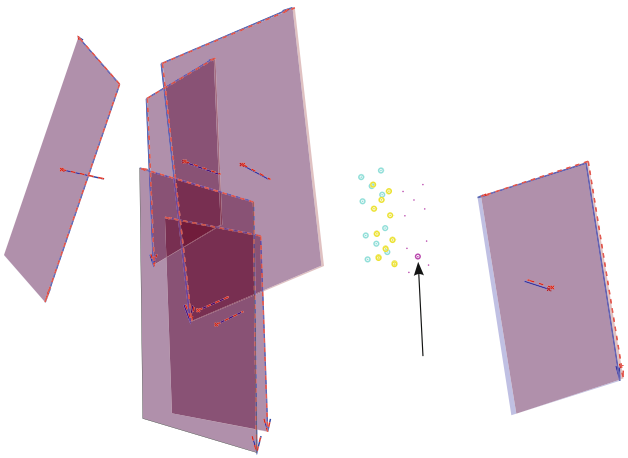


Fig. 8 Comparison between two reconstructions of the real data sequence: All the 10 feature points per camera view are used for the first reconstruction (feature points are drawn as *dots*). In contrast, for the second reconstruction (feature points drawn as *circles*), the rightmost camera 3 only tracked 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 (Color figure online)

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. Figure 9 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 (Wagner and Schmalstieg 2007) has been used to generate the feature trajectories. Each camera tracked between 10 and 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 model and the solution is given in a non-iterative closed-form. The reprojection error of the closed-form solution is

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

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

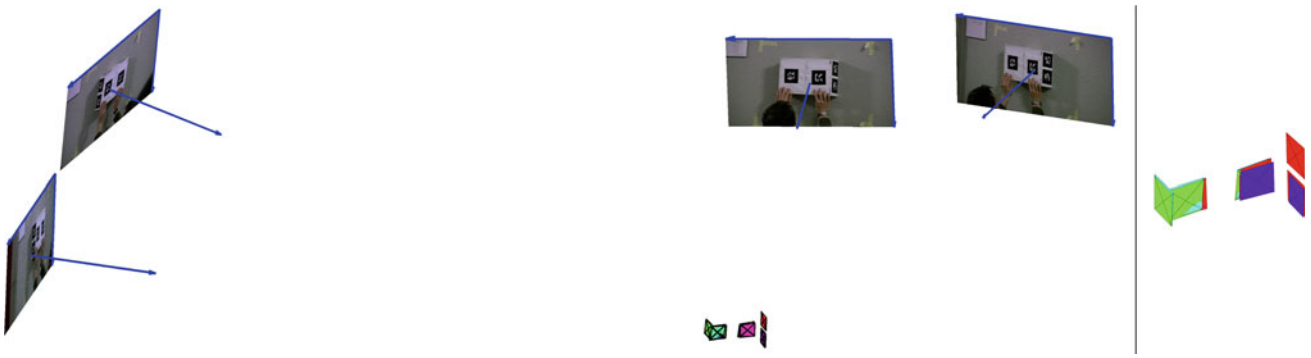


Fig. 9 Reconstruction of a planarly moving box: The *right* image shows a close-up view of the reconstructed structure (tags tracked by one specific camera share the *same color*). Two cameras have been positioned slightly below respectively above the remaining other cameras which is nicely captured by the reconstruction (Color figure online)

13 Conclusions and Future Work

This article brought together the ideas previously presented in Angst and Pollefeys (2009) and Angst and Pollefeys (2010). Specifically, this article presented a unified analysis of rigidly moving objects, for general rigid motions as well as for the special case of planar rigid motions (Sect. 4). 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 SfM reconstruction problem where no feature point correspondences between the different camera views exist (Sects. 7, 8). 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 respectively 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 (Sects. 11 and 12).

We hope the analysis and techniques presented in this article will be stimulating and boost potential future work. We see several opportunities which build upon the present work. 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. 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 so-called matrix completion problem. The power of iterative factorization methods which can deal with general patterns of missing data is not yet completely understood. The methods presented in Sect. 10 are only a first step towards this goal. 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.

Acknowledgments Roland Angst is a recipient of the Google Europe Fellowship in Computer Vision, and this research is supported in part by this Google Fellowship. We gratefully acknowledge the support of the 4DVideo ERC Starting Grant Nr. 210806 and a Packard Foundation Fellowship.

Appendix A: Linear System for Affine Reconstruction of Camera matrices

The affine camera matrices must fulfill

$$\mathcal{S}_{(f),[10:13,:]}\left[\Downarrow_k \tilde{\mathbf{S}}^{k\,T} \otimes \tilde{\mathbf{C}}^k\right]^T = \mathbf{Q}_{kron,[10:13,:]}\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}} \tag{51}$$

$$= \left[\Rightarrow_k \mathbf{1}_{1\times N_k} \otimes \tilde{\mathbf{C}}^{k\,T}\right] = \mathbf{Q}_{kron,[10:13,:]}\mathbf{Q}_{aff}^{-1}\left[\Rightarrow_k \hat{\mathbf{A}}^k\right], \tag{52}$$

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. (6) and Eq. (3) gives

$$\mathbf{G}^k \mathrm{vec}\left(\tilde{\mathbf{C}}^{k\,T}\right) = \left[(\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}}^k)^T \otimes \mathbf{I}_4\right] \mathrm{vec}\left(\mathbf{Q}_{kron,[10:13,:]}\right) \tag{53}$$

$$= \mathbf{H}^k \mathrm{vec}\left(\mathbf{Q}_{kron,[10:13,1:12]}\right) + \mathbf{b}^k \tag{54}$$

where the following matrices were introduced for abbreviation

$$\mathbf{G}^k = \left[\mathbf{I}_{N_k} \otimes \mathbf{T}_{2,1} \otimes \mathbf{I}_4\right]\left[\mathbf{1}_{N_k\times 1} \otimes \mathbf{I}_{2\cdot 4}\right] \tag{55}$$

$$\mathbf{H}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[1:12,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_4\Big] \tag{56}$$

$$\mathbf{b}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[13,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_4\right] \mathrm{vec}\left(\mathbf{Q}_{kron,[10:13,13]}\right). \tag{57}$$

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

$$\left[\searrow_k \mathbf{G}^k \; - \Downarrow_k \mathbf{H}^k\right]\begin{pmatrix} \Downarrow_k \mathrm{vec}\left(\tilde{\mathbf{C}}^{k\,T}\right) \\ \mathrm{vec}\left(\mathbf{Q}_{kron,[10:13,1:12]}\right)\end{pmatrix} = \left(\Downarrow_k \mathbf{b}^k\right). \tag{58}$$

However, closer inspection of Eq. (52) 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

$$\mathbf{G}^k = \left[\mathbf{I}_{N_k} \otimes \mathbf{T}_{2,1} \otimes \mathbf{I}_1\right]\left[\mathbf{1}_{N_k\times 1} \otimes \mathbf{I}_2\right] \tag{59}$$

$$\mathbf{H}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[1:12,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_1\Big] \tag{60}$$

$$\mathbf{b}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[13,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_1\right] \mathrm{vec}\left(\mathbf{Q}_{kron,(9+i,13)}\right). \tag{61}$$

The resulting over-constrained linear system reads like

$$\left[\searrow_k \mathbf{G}^k \; - \Downarrow_k \mathbf{H}^k\right]\begin{pmatrix} \Downarrow_k \mathrm{vec}\left(\tilde{\mathbf{C}}^{k\;T}_{[:,i]}\right) \\ \mathrm{vec}\left(\mathbf{Q}_{kron,[9+i,1:12]}\right)\end{pmatrix} = \left(\Downarrow_k \mathbf{b}^k\right), \tag{62}$$

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. 62 however has a three-dimensional nullspace, and therefore provides four linear independent solutions for the four rows.

Appendix B: Linear System for Affine Reconstruction of Points

This derivation closely follows the one from Sect. 13. Let $\mathbf{X}^k$ denote the non-homogeneous part of the points $\mathbf{S}^{k\,T} = \left[\mathbf{X}^{k\,T}\; \mathbf{1}_{N_k\times 1}\right]$ and $\mathbf{P}^k$ stands for the non-translational columns of the camera matrix $\mathbf{C}^k = \left[\mathbf{P}^k\; \mathbf{t}^k\right]$. Using this notation, a valid affine reconstruction must fulfill

$$\mathcal{S}_{(f),[1:9,:]}[\Downarrow_k \tilde{\mathbf{S}}^{k\,T} \otimes \tilde{\mathbf{C}}^k]^T = \mathbf{Q}_{kron,[1:9,:]}\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}} \tag{63}$$

$$= \left[\Rightarrow_k \tilde{\mathbf{X}}^k \otimes \tilde{\mathbf{P}}^{k\,T}\right] = \mathbf{Q}_{kron,[1:9,:]}\mathbf{Q}_{aff}^{-1}\left[\Rightarrow_k \hat{\mathbf{A}}^k\right] \tag{64}$$

Vectorization of the submatrix equation due to camera k using Eqs. (5) and (3) leads to

$$\mathbf{G}^k \mathrm{vec}\left(\tilde{\mathbf{X}}^k\right) = \left[[\mathbf{Q}_{aff}^{-1}\hat{\mathbf{A}}^k]^T \otimes \mathbf{I}_9\right] \mathrm{vec}\left(\mathbf{Q}_{kron,[1:9,:]}\right) \tag{65}$$

$$= \mathbf{H}^k \mathrm{vec}\left(\mathbf{Q}_{kron,[1:9,1:12]}\right) + \mathbf{b}^k \tag{66}$$

where the following matrices were introduced for abbreviation

$$\mathbf{G}^k = [\mathbf{I}_{N_k} \otimes \mathbf{T}_{2,3} \otimes \mathbf{I}_3][\mathbf{I}_{3N_k} \otimes \mathrm{vec}\left(\tilde{\mathbf{P}}^{k\,T}\right)] \tag{67}$$

$$\mathbf{H}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[1:12,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_9\Big] \tag{68}$$

$$\mathbf{b}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[13,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_9\right] \mathrm{vec}\left(\mathbf{Q}_{kron,[1:9,13]}\right). \tag{69}$$

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

$$\left[\searrow_k \mathbf{G}^k \; - \Downarrow_k \mathbf{H}^k\right]\begin{pmatrix} \Downarrow_k \mathrm{vec}\left(\tilde{\mathbf{X}}^k\right) \\ \mathrm{vec}\left(\mathbf{Q}_{kron,[1:9,1:12]}\right)\end{pmatrix} = \left(\Downarrow_k \mathbf{b}^k\right). \tag{70}$$

A similar observation as in Sect. 13 holds true for Eq. 70. More specifically, successive row-triples in Eq. (64) 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

$$\mathbf{G}^k = [\mathbf{I}_{N_k} \otimes \mathbf{T}_{2,1} \otimes \mathbf{I}_3][\mathbf{I}_{N_k} \otimes \mathrm{vec}\left(\tilde{\mathbf{P}}_k^T\right)] \tag{71}$$

$$\mathbf{H}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[1:12,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_3\Big] \tag{72}$$

$$\mathbf{b}^k = \left[\left[\mathbf{Q}_{aff}^{-1}\right]_{[13,:]}\hat{\mathbf{A}}^k\right]^T \otimes \mathbf{I}_3\right] \mathrm{vec}\left(\mathbf{Q}_{kron,[3i-2:3i,13]}\right). \tag{73}$$

The resulting over-constrained linear system reads like

$$\left[\searrow_k \mathbf{G}^k \; - \Downarrow_k \mathbf{H}^k\right]\begin{pmatrix} \Downarrow_k \mathrm{vec}\left(\tilde{\mathbf{X}}_{k,[i,:]}\right) \\ \mathrm{vec}\left(\mathbf{Q}_{kron,[3i-2:3i,1:12]}\right)\end{pmatrix} = \left(\Downarrow_k \mathbf{b}^k\right), \tag{74}$$

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).

Appendix C: Extracting Rank-Degenerate Solutions

The linear system in Eq. (34) is concisely formulated as

$$\left[\hat{\mathbf{M}}^T \odot \hat{\mathbf{M}}^T\right]^T \mathbf{K}_5 \mathrm{vecs}\,(\mathbf{Q}) = \mathbf{1}, \tag{75}$$

where $\odot$ denotes the Khatri-Rao product with column-wise block partitioning (i.e. column-wise Kronecker product), vecs () vectorizes the upper triangular part of a matrix, and $\mathbf{K}_5$ is the duplicity matrix s.t. $\mathrm{vec}\,(\mathbf{Q}) = \mathbf{K}_5 \mathrm{vecs}\,(\mathbf{Q})$ (we refer to reference (Magnus and Neudecker 1999) for more details about these operators). Eq. (75) can be solved in the least squares sense. The solution will in general have rank 3.

Let $\mathbf{Q}_0 \in \mathbb{R}^{5\times 5}$ denote a particular solution and $\mathbf{N} \in \mathbb{R}^{5\times 5}$ denote the nullspace of the linear system in Eq. (75). 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 third-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 $\hat{\mathbf{M}}(\mathbf{q}_1+\mathbf{q}_2) = \mathbf{1}_{F\times 1}$ for the vector $\mathbf{q}_1+\mathbf{q}_2$. Then we have

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

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

$$\begin{aligned}&\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\mathbf{Q}_0\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\\ &\quad= \left(1 - b_{\mathbf{Q}_0}\right)\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\left(\mathbf{q}_1+\mathbf{q}_2\right)\left(\mathbf{q}_1+\mathbf{q}_2\right)^T\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\end{aligned} \tag{76}$$

and

$$\begin{aligned}&\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\mathbf{N}\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\\ &\quad= \left(1 - b_{\mathbf{N}}\right)\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T}\left(\mathbf{q}_1+\mathbf{q}_2\right)\left(\mathbf{q}_1+\mathbf{q}_2\right)^T\mathbb{P}^{\perp}_{\mathbf{q}_1\mathbf{q}_1^T+\mathbf{q}_3\mathbf{q}_3^T},\end{aligned} \tag{77}$$

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

$$\begin{aligned}\mathbf{Q}_0 - \frac{1-b_{\mathbf{Q}_0}}{1-b_{\mathbf{N}}}\mathbf{N} &= \left(b_{\mathbf{Q}_0} - \frac{1-b_{\mathbf{Q}_0}}{1-b_{\mathbf{N}}}b_{\mathbf{N}}\right)\left[\mathbf{q}_1\mathbf{q}_1^T + \mathbf{q}_3\mathbf{q}_3^T\right]\\ &\quad+ \underbrace{\left(1 - b_{\mathbf{Q}_0} - \frac{1-b_{\mathbf{Q}_0}}{1-b_{\mathbf{N}}}\left(1-b_{\mathbf{N}}\right)\right)}_{=0}\left(\mathbf{q}_1+\mathbf{q}_2\right)\left(\mathbf{q}_1+\mathbf{q}_2\right).\end{aligned} \tag{78}$$

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. (33)) 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 $-1-\cos\alpha_f \le 0 \le 1-\cos\alpha_f$), $\mathbf{q}_2$ is replaced with $-\mathbf{q}_2 - 2\mathbf{q}_1$ (because $-(-1-\cos\alpha_f) - 2\cos\alpha_f = 1-\cos\alpha_f$).

Appendix D: 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

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

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

$$\begin{bmatrix}1 & -1 & 0 & \mathbf{0}_{1\times 2}\\ 0 & 0 & 1 & \mathbf{0}_{1\times 2}\\ \mathbf{0}_{2\times 1} & \mathbf{0}_{2\times 1} & \mathbf{0}_{2\times 1} & \mathbf{I}_2\end{bmatrix}\mathcal{C}_{(f)}\mathbf{S}\otimes\mathbf{C}^T$$

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

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.

Appendix E: 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

$$
\mathbf{G}_{[1,1]}^k - \mathbf{G}_{[2,2]}^k + \mathbf{w}_{k,[1]}^2 - \mathbf{w}_{k,[2]}^2 = 0 \tag{79}
$$

$$
\mathbf{G}_{[1,2]}^k + \mathbf{w}_{k,[1]}\mathbf{w}_{k,[2]} = 0. \tag{80}
$$

The second equation Eq. (80) can be solved for $\mathbf{w}_{k,[1]} = -\frac{\mathbf{G}_{[1,2]}^k}{\mathbf{w}_{k,[2]}}$ (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. (79) leads to a polynomial in the monomials $\mathbf{w}_{k,[2]}^2$ and $\mathbf{w}_{k,[2]}^4$. This polynomial can be solved for $\mathbf{w}_{k,[2]}^2$ 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.

References

Aguiar, P. M. Q., Xavier, J. M. F., & Stosic, M. (2008). Spectrally optimal factorization of incomplete matrices. In *IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE Computer Society.

Akhter, I., Sheikh, Y., Khan, S., & Kanade, T. (2011). Trajectory space: A dual representation for nonrigid structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 33*(7), 1442–1456. doi:10.1109/TPAMI.2010.201.

Angst, R., & Pollefeys, M. (2009). Static multi-camera factorization using rigid motion. In *Proceedings of IEEE international conference on computer vision 2009 (ICCV '09), Washington, DC* (pp. 1203–1210). IEEE Computer Society.

Angst, R.,& Pollefeys, M. (2010). 5D motion subspaces for planar motions. In *Proceedings of the 11th European conference on computer vision conference on computer vision 2010 (ECCV'10): Part III* (pp. 144–157). Berlin: Springer.

Brand, M. (2001). Morphable 3D models from video. In *IEEE conference on computer vision and pattern recognition (CVPR)* (Vol. 2, pp. 456–463). IEEE Computer Society.

Brand, M. (2005). A direct method for 3D factorization of nonrigid motion observed in 2D. In *IEEE conference on computer vision and pattern recognition (CVPR)* (Vol. 2, pp. 122–128). IEEE Computer Society.

Bregler, C., Hertzmann, A., & Biermann, H. (2000). Recovering non-rigid 3D shape from image streams. In *IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2690–2696). IEEE Computer Society.

Bue, A. D., & de Agapito, L. (2006). Non-rigid stereo factorization. *International Journal of Computer Vision, 66*(2), 193–207.

Carroll, J., & Chang, J. J. (1970). Analysis of individual differences in multidimensional scaling via an *n*-way generalization of Eckart–Young decomposition. *Psychometrika*, *35*(3), 283–319.

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

Daniilidis, K. (1999). Hand-eye calibration using dual quaternions. *International Journal of Robotics Research, 18*(3), 286–298.

Guerreiro, R. F. C., & Aguiar, P. M. Q. (2002). 3D structure from video streams with partially overlapping images. In *International conference on image processing (ICIP)* (Vol. 3, pp. 897–900).

Harshman, R. (1970). *Foundations of the parafac procedure: Models and conditions for an explanatory multi-modal factor analysis*. Working papers in phonetics, Vol. 16.

Hartley, R., & Schaffalitzky, F. (2004). Power factorization: 3D reconstruction with missing or uncertain data. In *Japan-Australia workshop on computer vision*.

Hartley, R. I., & Zisserman, A. (2004). *Multiple view geometry in computer vision* (2nd ed.). Cambridge: Cambridge University Press. ISBN: 0521540518.

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review, 51*(3), 455–500. doi:10.1137/07070111X.

Kumar, R. K., Ilie, A., Frahm, J. M., & Pollefeys, M. (2008). Simple calibration of non-overlapping cameras with a mirror. In *IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE Computer Society.

Lathauwer, L. D., Moor, B., & Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM Journal of Matrix Analysis and Applications,21*(4), 1253–1278. doi:10.1137/S0895479896305696.

Li, J., & Chellappa, R. (2005). A factorization method for structure from planar motion. In *7th IEEE workshop on applications of computer vision/IEEE workshop on motion and video computing (WACV/MOTION)* (pp. 154–159). IEEE Computer Society.

Magnus, J. R., & Neudecker, H. (1999). *Matrix differential calculus with applications in statistics and econometrics* (2nd ed.). New York: Wiley.

Sturm, P. F., & Triggs, B. (1996). A factorization based algorithm for multi-image projective structure and motion. In Buxton, B. F., & Cipolla, R. (Eds.). *European conference on computer vision (ECCV)* (Vol. 2, pp. 709–720). Lecture Notes in Computer Science, Vol. 1065. Berlin: Springer

Svoboda, T., Martinec, D., & Pajdla, T. (2005). A convenient multicamera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments, 14*(4), 407–422.

Tomasi, C., & Kanade, T. (1992). Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision, 9*(2), 137–154.

Torresani, L., Yang, D. B., Alexander, E. J., & Bregler, C. (2001). Tracking and modeling non-rigid objects with rank constraints. In *IEEE conference on computer vision and pattern recognition (CVPR)* (Vol. 1, pp. 493–500). IEEE Computer Society.

Tresadern, P. A., & Reid, I. D. (2005). Articulated structure from motion by factorization. In: *IEEE conference on computer vision and pattern recognition (CVPR)* (Vol. 2, pp. 1110–1115). IEEE Computer Society.

Tron, R., & Vidal, R. (2007). A benchmark for the comparison of 3-D motion segmentation algorithms. In: *IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE Computer Society.

Tucker, L. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika, 31*(3), 279–311.

Vidal, R., & Oliensis, J. (2002). Structure from planar motions with small baselines. In Heyden, A., Sparr, G., Nielsen, M., & Johansen, P. (eds). *European conference on computer vision (ECCV)* (Vol. 2, pp. 383–398). Lecture Notes in Computer Science, Vol. 2351. Berlin: Springer.

Wang, G., Tsui, H. T., & Wu, Q. M. J. (2008). Rotation constrained power factorization for structure from motion of nonrigid objects. *Pattern Recognition Letters, 29*(1), 72–80.

Wagner, D., & Schmalstieg, D. (2007). Artoolkitplus for pose tracking on mobile devices. In *Proceedings of 12th computer vision winter workshop*.

Wolf, L., & Zomet, A. (2006). Wide baseline matching between unsynchronized video sequences. *International Journal of Computer Vision, 68*(1), 43–52.

Xiao, J., Chai, J., & Kanade, T. (2004). A closed-form solution to non-rigid shape and motion recovery. In Pajdla, T., & Matas, J. (Eds.). *European conference on computer vision (ECCV)* (Vol. 4, pp. 573–587). Lecture Notes in Computer Science, Vol. 3024. Berlin: Springer.

Yan, J., & Pollefeys, M. (2008). A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 30*(5), 865–877.

Zelnik-Manor, L., & Irani, M. (2006). On single-sequence and multi-sequence factorizations. *International Journal of Computer Vision, 67*(3), 313–326.