Static Multi-Camera Factorization Using Rigid Motion

Roland Angst and Marc Pollefeys
Computer Vision and Geometry Lab, Department of Computer Science
ETH Zürich, Switzerland
{rangst, marc.pollefeys}@inf.ethz.ch

Abstract

Camera networks have gained increased importance in recent years. Previous approaches mostly used point correspondences between different camera views to calibrate such systems. However, it is often difficult or even impossible to establish such correspondences. In this paper, we therefore present an approach to calibrate a static camera network where no correspondences between different camera views are required. Each camera tracks its own set of feature points on a commonly observed moving rigid object and these $2D$ feature trajectories are then fed into our algorithm. By assuming the cameras can be well approximated with an affine camera model, we show that the projection of any feature point trajectory onto any affine camera axis is restricted to a 13-dimensional subspace. This observation enables the computation of the camera calibration matrices, the coordinates of the tracked feature points, and the rigid motion of the object with a non-iterative trilinear factorization approach. This solution can then be used as an initial guess for iterative optimization schemes which make use of the strong algebraic structure contained in the data. Our new approach can handle extreme configurations, e.g. *a camera in a camera network tracking only one single feature point. The applicability of our algorithm is evaluated with synthetic and real world data.*

1. Introduction

Factorization-based solutions to the structure from motion problem have been heavily investigated and extended ever since Tomasis' and Kanade's seminal work about rigid factorizations [15]. Such factorization based approaches enjoy interesting properties: *e.g.* given an almost affine camera these techniques provide an optimal, closed-form solution using only non-iterative techniques from linear algebra. The factorization approach, which is based on the singular value decomposition of the data matrix, has been further extended to multi-body motion segmentation [7], to perspective cameras [13], non-rigid objects [4, 16, 2, 3, 19], and articulated objects [21]. More flexible methods which can deal with missing data entries in the data matrix have been proposed in order to overcome shortcomings of singular value based decompositions which can not cope with such situations [4, 10, 19, 9]. 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. We therefore propose 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.

The goal of this paper is to extend rigid factorizations 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. Several methods [16, 5] already extended the factorization approach to a two-camera setup and Svoboda *et al.* [14] 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. Sequences from two camera views have also been investigated [22] 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 [8] or mirror-based [11]. These methods make strong assumptions about the captured data of each camera since in a first step, a reconstruction for each camera is usually computed separately. Wolf's and Zomet's approach [20] 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 can not provide a robust estimate of the whole motion and structure on its own then this method is doomed to failure. In contrast, our method readily combines partial observations from any number of cameras into one consistent solution.

The present paper 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. 2D trajectories of feature points on a commonly observed moving rigid object serve as an input to our algorithm. Sec. 3 derives a rank constraint on such affinely projected rigid motion trajectories. This analysis allows us to derive a new rank-13 constraint on the data matrix in Sec. 4. This rank constraint together with the knowledge of the algebraic structure contained in the data enables the computation of suitable basis vectors for the corresponding 13-dimensional subspace with linear, non-iterative methods (Sec. 4.2). Such a basis readily reveals the unknown camera matrices, coordinates of the points, and rigid motion. Sec. 5 shortly discusses minimal configurations in order to find such a linear solution. Non-linear iterative optimization methods, specifically tailored towards the underlying algebraic structure of the data, can then refine such an initial guess as provided by the linear method (Sec. 6). The applicability of our algorithm is shown on synthetic as well as on real world data in Sec. 7. 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.

2. Notation

The following notation will be used throughout the paper: K is the total number of static cameras, $k \in \{1, \cdots, K\}$ denotes one specific camera, F is the total number of observed frames and $f \in \{1, \cdots, F\}$ labels one specific frame. The number of tracked feature points in camera k is given by N_k. The identity matrix of dimension $D \times D$ is denoted as $\mathbf{I}_D$. 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 $\mathbf{A}$. A matrix which spans the same subspace as matrix $\mathbf{A}$ but which does not comply with the correct underlying algebraic structure is denoted with a hat $\hat{\mathbf{A}}$. We sometimes make use of Matlab® matrix indexing notation, 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.

Concepts from tensor calculus, specifically the mode-i product, the Tucker tensor decomposition [17], the Kronecker product $\otimes$, and the $\operatorname{vec}(\cdot)$-operator, are used as well. For example, the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ between matrix $\mathbf{A} \in \mathbb{R}^{2\times 2}$ and matrix $\mathbf{B}$ equals the block-structured matrix

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \otimes \mathbf{B} = \begin{bmatrix} A_{1,1}\mathbf{B} & A_{1,2}\mathbf{B} \\ A_{2,1}\mathbf{B} & A_{2,2}\mathbf{B} \end{bmatrix}. \quad (1)$$

We define the $\operatorname{vec}(\cdot)$-operator as a column-vector valued operator which stacks all the columns of a matrix below each other. The Kronecker product and the $\operatorname{vec}(\cdot)$-operator enjoy the following property

$$\operatorname{vec}(\mathbf{C}) = \operatorname{vec}\left(\mathbf{A}\mathbf{X}\mathbf{B}^T\right) = (\mathbf{B} \otimes \mathbf{A}) \operatorname{vec}(\mathbf{X}), \quad (2)$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{X}$ are matrices. This property is useful to extract an unknown matrix $\mathbf{X}$ in a matrix-equation. Readers unfamiliar with these concepts are referred to [12, 1] for a short introduction.

3. Rank-Constraint on Motion Trajectories

3.1. Derivation of Rank-13 Constraint

The affine projection $\mathcal{W}_{k,f,n}$ of the n^{th} feature point with homogeneous coordinates $\mathbf{s}_n \in \mathbb{R}^{4\times 1}$ undergoing a rigid motion $\begin{bmatrix}\mathbf{R}_f & \mathbf{t}_f\end{bmatrix}$ onto an affine camera axis $\mathbf{c}_k^T \in \mathbb{R}^{1\times 4}$ reads like

$$\begin{aligned} \mathcal{W}_{k,f,n} &= \mathbf{c}_k^T \begin{bmatrix} \mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{s}_n & (3) \\ &= \operatorname{vec}\left(\begin{bmatrix} \mathbf{R}_f & \mathbf{t}_f \\ \mathbf{0} & 1 \end{bmatrix}\right)^T \begin{bmatrix}\mathbf{s}_n \otimes \mathbf{c}_k\end{bmatrix} & (4) \\ &= \begin{bmatrix}\operatorname{vec}(\mathbf{R}_f)^T & \mathbf{t}_f^T & 1\end{bmatrix} \mathcal{S}_{(f)} \begin{bmatrix}\mathbf{s}_n \otimes \mathbf{c}_k\end{bmatrix}, & (5) \end{aligned}$$

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 $\mathcal{S} \in \mathbb{R}^{4\times 13\times 4}$ flattened along the temporal mode in order to get rid of the zero columns from the vectorized rigid motion. This flattened tensor thus looks like

$$\mathcal{S}_{(f)} = \begin{bmatrix} \mathbf{I}_3 \otimes \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_{3\times 1}\end{bmatrix} & \mathbf{0}_{9\times 4} \\ \mathbf{0}_{4\times 12} & \mathbf{I}_4 \end{bmatrix} \in \mathbb{R}^{13\times 16}. \quad (6)$$

The camera axes of all the K cameras can be stacked vertically into a matrix $\mathbf{C} = \begin{bmatrix}\Downarrow_k \mathbf{c}_k^T\end{bmatrix} \in \mathbb{R}^{2K\times 4}$. In a similar way, the tracked feature points can be stacked into a matrix $\mathbf{S} = \begin{bmatrix}\Rightarrow_n \mathbf{s}_n\end{bmatrix} \in \mathbb{R}^{4\times \sum_k N_k}$. By introducing the motion matrix $\mathbf{M} = \begin{bmatrix}\Downarrow_f \begin{bmatrix}\operatorname{vec}(\mathbf{R}_f)^T & \mathbf{t}_f^T & 1\end{bmatrix}\end{bmatrix} \in \mathbb{R}^{F\times 13}$ we finally get the equation for the coordinates of the trajectory of the n^{th} feature point projected onto the k^{th} camera axis $\mathcal{W}_{k,:,n} = \mathbf{M}\mathcal{S}_{(f)}(\mathbf{S}_{:,n}^T \otimes \mathbf{C}_{k,:})^T$. Fixing a column ordering scheme $\Rightarrow_{n,k}$ consistent with the Kronecker product, we derive the equation for a 3rd-order data tensor $\mathcal{W} \in \mathbb{R}^{2K\times F\times \sum_k N_k}$ flattened along the temporal mode f

$$\mathcal{W}_{(f)} = \begin{bmatrix}\Rightarrow_{n,k} \mathcal{W}_{k,:,n}\end{bmatrix} = \mathbf{M}\mathcal{S}_{(f)}(\mathbf{S}^T \otimes \mathbf{C})^T. \quad (7)$$

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*

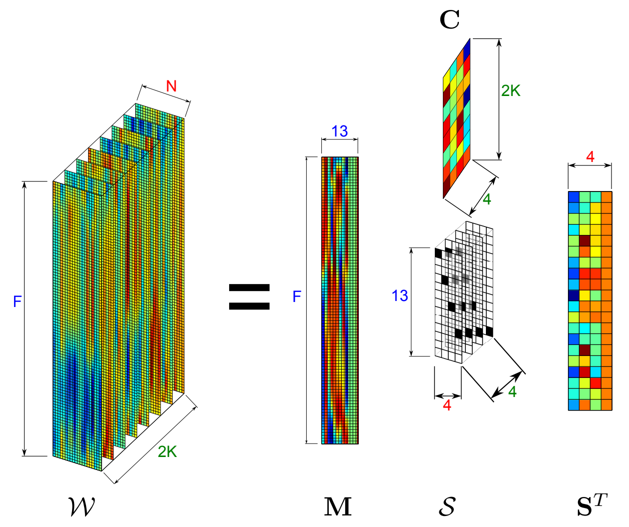


Figure 1. The tensor $\mathcal{W}$ can be viewed as a linear combination of 3^{rd}-order tensors each of which equals the outer product of three vectors (so called simple tensors). The coefficients for this linear combination are stored in the core tensor $\mathcal{S}$, which in our case simply consists of only 13 non-zero entries (visualized as black squares). Stacking all the vectors of these simple tensors according to their mode next to each other reveals the motion matrix $\mathbf{M}$, the camera matrix $\mathbf{C}$, and the coordinates of the points $\mathbf{S}$. The difficulty lies in decomposing the given data tensor into these 13 simple tensors in the presence of missing data entries.

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*

$$\mathbf{M} = \left[\Downarrow_f \left[\text{vec}\left(\mathbf{R}_f\right)^T \quad \mathbf{t}_f^T \quad 1\right]\right] \in \mathbb{R}^{F\times 13}, \tag{8}$$

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

Eq. (7) 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 3^{rd} order tensor $\mathcal{S} \in \mathbb{R}^{4\times 13\times 4}$ and by applying the three mode-i products between the core tensor and the mode-f, mode-k, and mode-n subspaces $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{S}^T$, respectively:

$$\mathcal{W} = \mathcal{S} \times_f \mathbf{M} \times_k \mathbf{C} \times_n \mathbf{S}^T \tag{9}$$

Note that f, k, and n are used for readability reasons for labeling of 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. Fig. 1 depicts this decomposition visually.

3.2. Ambiguities

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

$$\begin{aligned} \mathcal{W} = (\mathcal{S} \times_f \mathbf{Q}_f \times_k \mathbf{Q}_k \times_n \mathbf{Q}_n^T) & \qquad (10)\\ \times_f \mathbf{M}\mathbf{Q}_f^{-1} \times_k \mathbf{C}\mathbf{Q}_k^{-1} \times_n (\mathbf{Q}_n^{-1}\mathbf{S})^T. & \qquad (11) \end{aligned}$$

This would obviously result in changing the entries of the known core tensor (Eq. (6)). 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 the last row of $\mathbf{Q}_k$ and $\mathbf{Q}_n$ be equal to $\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$, i.e. $\mathbf{Q}_k$ and $\mathbf{Q}_n$ are affine transformations. Then we can solve for $\mathbf{Q}_f$ in the equation $\mathcal{S} = \mathcal{S} \times_f \mathbf{Q}_f \times_k \mathbf{Q}_k \times_n \mathbf{Q}_n^T$ which leads to

$$\mathbf{Q}_f = \mathcal{S}_{(f)}(\mathbf{Q}_n^{-T} \otimes \mathbf{Q}_k^{-1})^T \mathcal{S}_{(f)}^T. \tag{12}$$

The inverse of this transformation is then applied to the motion matrix $\mathbf{M} \leftarrow \mathbf{M}\mathbf{Q}_f^{-1}$ 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 $\mathbf{M}\mathbf{Q}_f^{-1}$ still fulfills the rotational constraints. This clearly shows that the factorization is only unique up to two affine resp. two Euclidean transformations $\mathbf{Q}_n$ and $\mathbf{Q}_k$ which should not come as a surprise since Eq. (3) is a product involving three factors and hence two ambiguities arise between these factors. Brand [3] mentions a similar result in the case of deformable objects where the ambiguity arises from an arbitrary basis transformation of the subspace of the blend shape weights and an Euclidean transformation of the camera frame.

4. Rank-13 Factorization

With the previously derived formulation, our problem can be restated in the following way. Given the knowledge of certain elements of the 3^{rd}-order tensor $\mathcal{W}$, compute the underlying mode-f, mode-n, and mode-k subspaces ($\mathbf{M}$, $\mathbf{S}^T$, and $\mathbf{C}$, respectively) which generate the data tensor according to Eq. (9). 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 [17] 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.

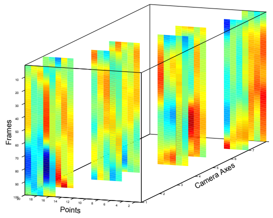
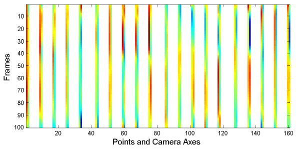
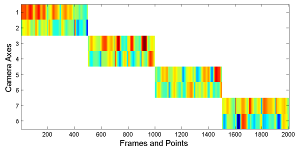
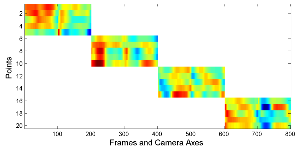
<table>
<tr><th>$\mathcal{W} = \mathcal{S} \times_k \mathbf{C} \times_f \mathbf{M} \times_n \mathbf{S}^T$</th><th>$\mathcal{W}_{(f)} = \mathbf{M}\mathcal{S}_{(f)}(\mathbf{S}^T \otimes \mathbf{C})^T$</th><th>$\mathcal{W}_{(k)} = \mathbf{C}\mathcal{S}_{(k)}(\mathbf{S}^T \otimes \mathbf{M})^T$</th><th>$\mathcal{W}_{(n)} = \mathbf{S}^T\mathcal{S}_{(n)}(\mathbf{C} \otimes \mathbf{M})^T$</th></tr>
<tr><td>
</td><td>
</td><td>
</td><td>
</td></tr>
<tr><td>$\mathcal{W} \in \mathbb{R}^{2K \times F \times \sum_k N_k}$</td><td>$\mathcal{W}_{(f)} \in \mathbb{R}^{F \times 2K \sum_k N_k}$</td><td>$\mathcal{W}_{(k)} \in \mathbb{R}^{2K \times F \sum_k N_k}$</td><td>$\mathcal{W}_{(n)} \in \mathbb{R}^{\sum_k N_k \times 2KF}$</td></tr>
</table>

Table 1. 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.

4.1. 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 Tab. 1, only the flattened tensor $\mathcal{W}_{(f)}$ along the temporal mode f contains some columns whose entries are all known, amongst many completely unknown columns. These known columns however still span the complete 13-dimensional mode-f subspace. Analogously to the well-known rank-4 factorization approach, this rank-13 constraint can be used to robustly decompose the known $2\sum_k N_k$ columns of the flattened data tensor $\mathcal{W}_{(f)}$ with a singular value decomposition into a product of two rank-13 matrices $\hat{\mathbf{M}} = \mathbf{U} \in \mathbb{R}^{F \times 13}$ and $\hat{\mathbf{A}} = \mathbf{\Lambda}\mathbf{V}^T \in \mathbb{R}^{13 \times 2\sum_k N_k}$

$$\mathbf{M}\mathcal{S}_{(f)} \left[\Downarrow_k \mathbf{S}_k^T \otimes \mathbf{C}_k\right]^T = (\hat{\mathbf{M}}\mathbf{Q})(\mathbf{Q}^{-1}\hat{\mathbf{A}}). \tag{13}$$

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.

4.2. Stratified Corrective Transformation

For that reason we propose to use a stratified approach to compute the unknown transformation matrix $\mathbf{Q}$

$$\mathbf{Q} = \mathbf{Q}_{\mathit{aff}}\mathbf{Q}_{\mathit{kron}}^{-1}\mathbf{Q}_{\mathit{metric}}. \tag{14}$$

$\mathbf{Q}_{\mathit{aff}}$ isolates the camera translations from the remaining columns 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 $\mathbf{Q}_{\mathit{kron}}^{-1}$, whereas $\mathbf{Q}_{\mathit{metric}}$ finally performs a similarity upgrade. The following subsections present each step in detail.

4.2.1 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 $\mathbf{q}_{\mathit{aff}}^T$ in $\mathbf{1}_{F \times 1} = \hat{\mathbf{M}}\mathbf{q}_{\mathit{aff}}$. A guaranteed non-singular affine upgrade is then given by $\mathbf{Q}_{\mathit{aff}} = \left[\mathcal{N}\left(\mathbf{q}_{\mathit{aff}}\right) \quad \mathbf{q}_{\mathit{aff}}\right]$, where $\mathcal{N}\left(\mathbf{q}_{\mathit{aff}}\right)$ denotes an orthogonal basis for the nullspace of $\mathbf{q}_{\mathit{aff}}$. We can not gain any more knowledge by analyzing $\hat{\mathbf{M}}$, yet. We therefore turn our attention toward $\hat{\mathbf{A}}$.

4.2.2 Computing Affine Cameras

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

$$\tilde{\mathbf{A}} = \mathbf{Q}_{\mathit{kron}}\mathbf{Q}_{\mathit{aff}}^{-1}\hat{\mathbf{A}} = \mathcal{S}_{(f)}(\Downarrow_k \tilde{\mathbf{S}}_k^T \otimes \tilde{\mathbf{C}}_k)^T. \tag{15}$$

This is a bilinear problem in the unknowns $\mathbf{Q}_{\mathit{kron}}$, $\tilde{\mathbf{S}}$, and $\tilde{\mathbf{C}}$. Since the product between $\mathbf{Q}_{\mathit{kron}}^{-1}$ and $\mathbf{Q}_{\mathit{metric}}$ should not modify the last column of $\mathbf{Q}_{\mathit{aff}}$ anymore, the last column of $\mathbf{Q}_{\mathit{metric}}^{-1}\mathbf{Q}_{\mathit{kron}}$ has to be equal to $(\mathbf{0}_{1 \times 12}, 1)^T$. This together with the fact that the structure of $\mathbf{Q}_{\mathit{metric}}$ must follow the structure in Eq. (12) implies that the last column of $\mathbf{Q}_{\mathit{kron}}$ equals $(\mathbf{0}_{1 \times 12}, 1)^T$ and thus only the first twelve columns of $\mathbf{Q}_{\mathit{kron}}$ 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. (15) actually describe an over-determined linear problem in the $4 \cdot 12 + 2K \cdot 4$ unknowns of $\mathbf{Q}_{\mathit{kron},10:13,1:12}$ and $\tilde{\mathbf{C}}$. The resulting system can be solved linearly in the least-squared sense (see [1] for details).

4.2.3 Enforcing the Kronecker-Structure

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

$$\tilde{\mathbf{A}} = \mathcal{S}_{(f)}(\Downarrow_k \tilde{\mathbf{S}}_k^T \otimes \tilde{\mathbf{C}}_k)^T = \mathbf{Q}_{kron}\mathbf{Q}_{\mathit{aff}}^{-1}\hat{\mathbf{A}} \tag{16}$$

in the unknowns $\tilde{\mathbf{S}}_k$ and $\mathbf{Q}_{kron}$. This is again an overdetermined 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 (the technical report [1] again provides details on how to set up and solve this system).

4.2.4 Metric Upgrade

There is not enough information contained in $\mathcal{S}_{(f)}(\tilde{\mathbf{S}}^T \otimes \tilde{\mathbf{C}})^T = \mathbf{Q}_{kron}\mathbf{Q}_{\mathit{aff}}^{-1}\hat{\mathbf{A}}$ to perform the metric upgrade and we thus have to turn our attention again to the rigid motion matrix $\hat{\mathbf{M}}$. However, in contrast to Sec. 4.2.1, 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. (12). We are therefore looking for affine transformations $\mathbf{Q}_n$ and $\mathbf{Q}_k$ such that

$$\begin{aligned} \mathbf{M} &= \Downarrow_f \left[(\mathrm{vec}\,(\mathbf{R}_f))^T \quad \mathbf{t}_f^T \quad 1 \right] \qquad (17) \\ &= \underbrace{\Downarrow_f \left[(\mathrm{vec}\left(\tilde{\mathbf{R}}_f\right))^T \quad \tilde{\mathbf{t}}_f^T \quad 1 \right]}_{\tilde{\mathbf{M}} = \hat{\mathbf{M}}\mathbf{Q}_{\mathit{aff}}\mathbf{Q}_{kron}^{-1}} \underbrace{\mathcal{S}_{(f)}(\mathbf{Q}_n^T \otimes \mathbf{Q}_k)^T \mathcal{S}_{(f)}^T}_{\mathbf{Q}_{metric}} \end{aligned}$$

conforms to an Euclidean rigid motion matrix. Let

$$\mathbf{Q}_n = \begin{bmatrix} \mathbf{T}_n^{-1} & \mathbf{t}_n \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{Q}_k = \begin{bmatrix} \mathbf{T}_k & \mathbf{t}_k \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}. \tag{18}$$

Using the Kronecker product property of Eq. (2), the above equation Eq. (17) is equivalent to the set of equations

$$\mathbf{R}_f = \mathbf{T}_k\tilde{\mathbf{R}}_f\mathbf{T}_n^{-1} \tag{19}$$

$$\mathbf{t}_f = \mathbf{T}_k\tilde{\mathbf{R}}_f\mathbf{t}_n + \mathbf{T}_k\tilde{\mathbf{t}}_f + \mathbf{t}_k \tag{20}$$

for $f \in \{1 \ldots F\}$. Eq. (19) is equivalent to $\mathbf{T}_k\tilde{\mathbf{R}}_f = \mathbf{R}_f\mathbf{T}_n$. Since $\mathbf{R}_f$ is a rotation matrix we have

$$\begin{aligned} \left(\mathbf{T}_k\tilde{\mathbf{R}}_f\right)^T \left(\mathbf{T}_k\tilde{\mathbf{R}}_f\right) &= (\mathbf{R}_f\mathbf{T}_n)^T (\mathbf{R}_f\mathbf{T}_n) \qquad (21) \\ = \quad \tilde{\mathbf{R}}_f^T\mathbf{T}_k^T\mathbf{T}_k\tilde{\mathbf{R}}_f &= \mathbf{T}_n^T\mathbf{T}_n. \qquad (22) \end{aligned}$$

This set of equations is linear in symmetric $\mathbf{T}_k^T\mathbf{T}_k$ and $\mathbf{T}_n^T\mathbf{T}_n$ and can be solved by similar techniques as the one presented in [2, 3] for the rigid case. Each frame provides 6 constraints on the 12 unknowns (every dot product between columns of $\tilde{\mathbf{R}}_f$ provides one constraint) and a solution for $\mathbf{T}_k^T\mathbf{T}_k$ and $\mathbf{T}_n^T\mathbf{T}_n$ can be found given sufficient frames are available. A final eigenvalue decomposition of these symmetric matrices finally yields the matrices $\mathbf{T}_n$ and $\mathbf{T}_k$. These matrices are then used to render Eq. (20) linear in the unknowns, i.e., the translations $\mathbf{t}_n$, $\mathbf{t}_k$, 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 $\mathbf{t}_k$ and $\mathbf{t}_n$. 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.

5. Minimal Configurations

Our algorithm requires the union of all the feature trajectories spanning the 13-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 and we can assume the rigid motion being general enough such that the whole 13 dimensional motion subspace gets explored. On the other hand, the constraints on the minimal number of camera axes and feature points are more interesting. Tab. 2 shows a summary which minimal cases meet these constraints. Note that for some cases, even though the rank of their mode-f subspace is 13, the computation of the affine cameras (Sec. 4.2.2) still fails because the points do not provide enough linear independent constraints for solving the linear system of Eq. (15).

Let us shortly discuss why a single affine camera is insufficient to capture the whole 13-dimensional motion space. If the camera is considered as static and the rigid object as moving, the rigid factorization approach [15] provides two axes of the rigid rotation and rigid translations along two axes at every frame. The rotation matrix can obviously be completed by taking the cross product of the known axes. Furthermore, the last column of the motion matrix $\mathbf{M}$ is known to be equal to the constant one-vector. One single camera can therefore reconstruct $9 + 2 + 1 = 12$ dimensions of the underlying motion space. The missing dimension corresponds to the rigid translation along the z-axis of the camera. This is is not surprising since affine cameras suffer from a depth-ambiguity. The complete motion space is therefore only determined with at least two camera views with non-parallel image planes (three linearly independent camera axes would actually be sufficient).

The derivation of Eq. (9) assumed a rank-4 structure matrix $\mathbf{S}$. This assumption is violated if the observed object is planar. Our algorithm currently can not handle such situations and thus planar objects represent degenerated cases. Note however that each camera is allowed to track feature points which lie in a plane, as long as they are not contained in a common plane and the combined structure matrix $\left[\Rightarrow_k \mathbf{S}_k\right]$ is thus non-planar (see also the evaluation of the real data sequence in Sec. 7.2).

# points per camera	(3, 4)	(4, 4)	(1, 3, 3)	(2, 3, 3)	(2, 2, 4)	(2, 2, 2, 2)	(2, 2, 2, 3)	(2, 2, 2, 2, 2)
$\text{rank}\,(\mathbf{A}) \overset{?}{=} 13$	$12 \neq 13$	$13 = 13$	$13 = 13$	$13 = 13$	$13 = 13$	$13 = 13$	$13 = 13$	$13 = 13$
Eq. (15) solvable	✗	✓	✗	✓	✗	✗	✓	✓
Sec. 4.2 applicable	✗	✓	✗	✓	✗	✗	✓	✓

Table 2. Minimal cases: This table lists the number of points per camera (for example, $(N_1,\, N_2)$ means the first camera observes N_1 points whereas the second tracks N_2 points) and whether the linear algorithm of Sec. 4.2 succeeds in computing a valid factorization or not (summarized in the last row). The first condition states that the observed points should span the complete 13-dimensional mode-f subspace. The second condition ensures that valid affine camera matrices are computable (see Sec. 4.2.2). 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 $N_3 = 1$).

6. Iterative Optimization

The solution given by the linear algorithm described in Sec. 4.2 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 next 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 solution still provides an accurate initial guess for an iterative non-linear optimization scheme. [6] 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 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 can be adapted to the matrix factorization of $\mathcal{W}_{(f)}$ where the gradient is taken with respect to the unknown mode-k subspace $\mathbf{C}$ and mode-n subspace $\mathbf{S}$. 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 if initialized with the linear solution. The linear solution as provided by the algorithm described in Sec. 4.2 thus seems to suit perfectly as an initial guess for an iterative refinement.

7. Evaluation

The steps described in Sec. 4.2.1, Sec. 4.2.2, and Sec. 4.2.3 were applied sequentially to synthetically generated data and to a real data sequence in order to get an initial estimate for an iterative non-linear refinement. The ALS scheme was then iterated up to 10 times, with the previously computed solution as an initial guess. This already provided a very good reconstruction which could be even further improved by performing a couple of Wiberg iterations with the ALS solution as initialization. Finally, the metric upgrade was performed as described in Sec. 4.2.4. Because the metric upgrade step is based on a least squares formulation, only soft constraints are enforced 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 $\mathbf{R}_f = \mathbf{R}\mathbf{P}$ at every frame, replaced $\mathbf{R}_f$ by $\mathbf{R}$ and ignored the non-rotational part $\mathbf{P}$ [1].

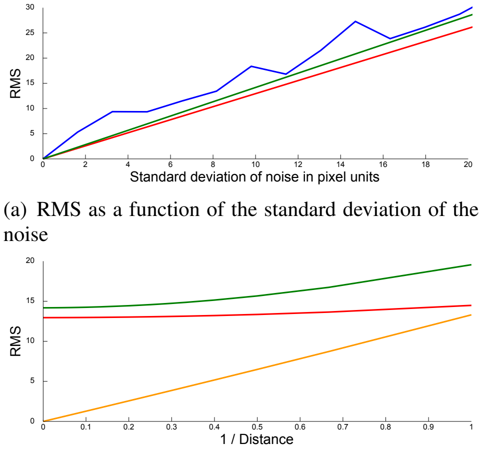
7.1. Synthetic Data

For the synthetic data experiments, the cameras were modeled as similar to the ones used for the real data experiments as possible (pixel density of $\frac{1080 pixels}{4.035 mm}$, image resolution of 1920×1080). We chose a magnification factor of $m = \frac{61mm}{5m}$ which corresponds to a focal length of $61mm$ with an average distance between camera and rigid object of $5m$. $K = 4$ cameras were placed randomly $7.5m$ apart from the scene each of which tracked $N_k = 10$ feature points over 100 frames. The rigid motion was generated by specifying 5 keyframes and interpolating in between. We randomly picked 5 axes of rotation and rotation angles (which were limited to a maximum of 45 degrees). The translation vector of the rigid motions and the feature points were drawn from a normal distribution with a standard deviation of $5cm$.

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. 2(a) shows that our algorithm with soft metric constraints even slightly overfits the ground truth. This is mainly due to the fact, that only soft constraints on the rigidity of the motion were imposed. Enforcing truly rigid motions using polar decompositions increased the root mean squared error of the reprojected moving points (RMS) slightly.

Secondly, the influence of the distance between cameras and rigid object was investigated. In order to keep the magnification factor constant, the focal length of the cameras

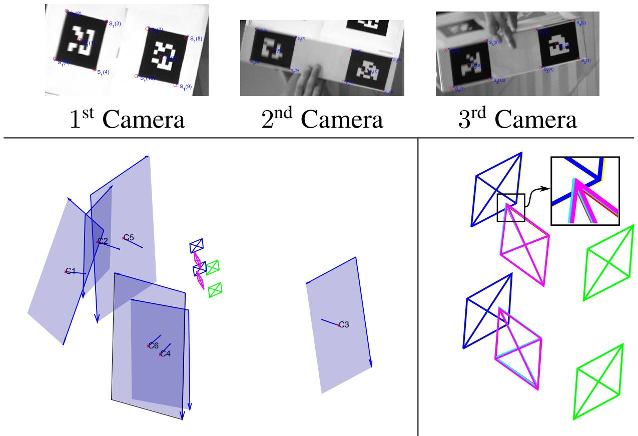
[1] The polar decomposition $\mathbf{A} = \mathbf{R}\mathbf{P}$ provides the optimal approximation $\mathbf{R} \approx \mathbf{A}$ of a matrix $\mathbf{A}$ with an orthogonal matrix $\mathbf{R}$ w.r.t. the Frobenius-norm $\mathbf{R} = \arg\min_{\mathbf{Q}} \|\mathbf{A} - \mathbf{Q}\|_F$ subject to $\mathbf{Q}^T\mathbf{Q}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



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

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

Figure 2. 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 (with soft constraints on rigid motion) whereas the blue line shows the resulting error if 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.



(b) 3D view of reconstruction

(c) Closeup view

Figure 3. Resulting reconstruction of the real data sequence. The top three images show 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). Fig. 3(b) shows a 3D view of the reconstructed camera poses together with the points of the box at one specific frame. Fig. 3(c) shows a closeup view of the reconstructed points at this frame.

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. 2(b)).

7.2. 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 [18] which provides 5 points per template (the 4 vertices and the middle point). The cameras were not aware of the fact that they might have tracked the very same feature points (i.e., no correspondences were established between different camera views). Each camera tracked 2 templates which were located on the same side of the box and hence, the structure of the points tracked by one single camera was actually planar. As the results show, our algorithm can handle this configuration without problems. Fig. 3 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 Fig. 4 and Fig. 3 are based on the reconstruction which enforces true rigid motions.

In a second experiment, we tried how robust our algorithm can handle a camera which only tracks one single feature point. We therefore excluded all but one feature trajectory in camera 3 and run the same algorithm again. The resulting reconstruction again had a RMS of about 2.6 pixels, respectively 8.5 pixels with enforced rotation matrices. Fig. 4 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 really be used in practice to calibrate cameras which only track one single point which is not in correspondence with any other point tracked by the remaining cameras.

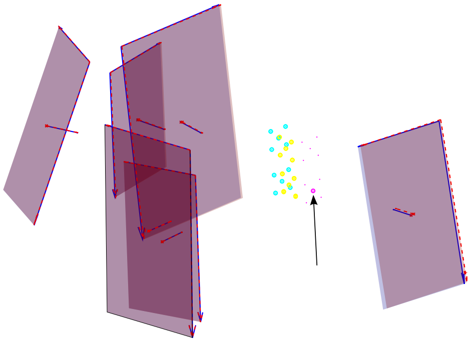


Figure 4. Comparison between two reconstructions of the real data sequence: All the 10 feature points per camera view are used for the first reconstruction (feature points are drawn as dots). In contrast, for the second reconstruction (feature points drawn as circles), the rightmost camera 3 only tracked 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.

8. Conlusion and Future Work

In this paper, we have presented an approach to the structure-from-motion problem for multiple static affine cameras where no point correspondences are available between different camera views. The cameras are only assumed to track feature points on a commonly observed moving rigid object. Tensorial notation provided us with the necessary insight to derive a non-iterative linear solution. We have shown how to further improve such an initial solution with iterative methods. Our proposed method can deal with extreme situations where a camera might only track one single feature point. The algorithm was evaluated on synthetic data and has been shown to work in practice on a real data sequence.

A drawback of our current method is that the linear method assumes 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 Tab. 1). This prevents large rotations of the rigid object due to possible occlusions. As future work, we plan to investigate the potential of iterative factorization methods which can deal with missing data. Another topic for further improvement is the robustness of our algorithm w.r.t. outlier trajectories. Furthermore, we intend to study motion subspaces with higher ranks (non-rigid or multi-body motions) in the multi-camera setting.

References

[1] R. Angst and M. Pollefeys. CVG Lab Technical Report 1. `www.cvg.ethz.ch/people/phdstudents/rangst/Publications`, 2009.

[2] M. Brand. Morphable 3d models from video. In *CVPR (2)*, pages 456–463. IEEE Computer Society, 2001.

[3] M. Brand. A direct method for 3d factorization of nonrigid motion observed in 2d. In *CVPR (2)*, pages 122–128. IEEE Computer Society, 2005.

[4] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *CVPR*, pages 2690–2696. IEEE Computer Society, 2000.

[5] A. D. Bue and L. de Agapito. Non-rigid stereo factorization. *IJCV*, 66(2):193–207, 2006.

[6] P. Chen. Optimization algorithms on subspaces: Revisiting missing data problem in low-rank matrix. *IJCV*, 80(1):125–142, 2008.

[7] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *ICCV*, page 1071. IEEE Computer Society, 1995.

[8] K. Daniilidis. Hand-eye calibration using dual quaternions. *Int. Journal of Robotics Research*, 18:286–298, 1998.

[9] R. F. C. Guerreiro and P. M. Q. Aguiar. 3d structure from video streams with partially overlapping images. In *ICIP (3)*, pages 897–900, 2002.

[10] R. Hartley and F. Schaffalitzky. Powerfactorization : 3d reconstruction with missing or uncertain data. In *Japan-Australia Workshop on Computer Vision*, 2004.

[11] R. Kumar, A. Ilie, J.-M. Frahm, and M. Pollefeys. Simple calibration of non-overlapping cameras with a mirror. In *CVPR*, pages 1–7. IEEE Computer Society, 2008.

[12] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[13] P. F. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *ECCV (2)*, pages 709–720. Springer, 1996.

[14] T. Svoboda, D. Martinec, and T. Pajdla. A convenient multicamera self-calibration for virtual environments. *Presence*, 14(4):407–422, 2005.

[15] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *IJCV*, 9(2):137–154, 1992.

[16] L. Torresani, D. B. Yang, E. J. Alexander, and C. Bregler. Tracking and modeling non-rigid objects with rank constraints. In *CVPR (1)*, pages 493–500, 2001.

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

[18] D. Wagner and D. Schmalstieg. Artoolkitplus for pose tracking on mobile devices. In *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)*, 2007.

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

[20] L. Wolf and A. Zomet. Wide baseline matching between unsynchronized video sequences. *IJCV*, 68(1):43–52, 2006.

[21] J. Yan and M. Pollefeys. A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video. *PAMI*, 30(5):865–877, 2008.

[22] L. Zelnik-Manor and M. Irani. On single-sequence and multi-sequence factorizations. *IJCV*, 67(3):313–326, 2006.