Computing Smooth Time-Trajectories for Camera and Deformable Shape in Structure from Motion with Occlusion

Paulo F. U. Gotardo, *Member, IEEE,* Aleix M. Martinez, *Senior Member, IEEE*

Abstract—We address the classical computer vision problems of rigid and non-rigid structure from motion (SFM) with occlusion. We assume that the columns of the input observation matrix $\mathbf{W}$ describe smooth 2D point trajectories over time. We then derive a family of efficient methods that estimate the column space of $\mathbf{W}$ using compact parameterizations in the Discrete Cosine Transform (DCT) domain. Our methods tolerate high percentages of missing data and incorporate new models for the smooth time-trajectories of 2D-points, affine and weak-perspective cameras, and 3D deformable shape. We solve a rigid SFM problem by estimating the smooth time-trajectory of a single camera moving around the structure of interest. By considering a weak-perspective camera model from the outset, we directly compute Euclidean 3D shape reconstructions without requiring post-processing steps such as Euclidean upgrade and bundle adjustment. Our results on real SFM datasets with high percentages of missing data were positively compared to those in the literature. In non-rigid SFM, we propose a novel *3D shape trajectory* approach that solves for the deformable structure as the smooth time-trajectory of a single point in a linear shape space. A key result shows that, compared to state-of-the-art algorithms, our non-rigid SFM method can better model complex articulated deformation with higher frequency DCT components while still maintaining the low-rank factorization constraint. Finally, we also offer an approach for non-rigid SFM when $\mathbf{W}$ is presented with missing data.

Index Terms—Structure from motion, matrix factorization, missing data, camera trajectory, shape trajectory.

1 Introduction

Accurately describing a data matrix as a product of two low-rank factors, matrix factorization, is a fundamental task in computer vision and pattern recognition. This paper focuses on the classical computer vision problem of matrix factorization in rigid and non-rigid structure from motion (SFM) [5], [29]. The goal in SFM is to jointly estimate the 3D scene structure and relative camera motion from corresponding 2D points in a sequence of images. Applications of SFM include autonomous navigation, image augmentation, and the construction of rigid and deformable 3D models from images [11], [15]. The modeling of deformable shapes such as the human hand, face, and body is also of particular importance in computer graphics, and human-computer interaction (e.g., [10]).

While techniques for rigid SFM have matured considerably over the past two decades [6]–[8], [12], [16], [17], [19], [21], [28], non-rigid SFM is still a very difficult problem, especially for complex articulated deformations [30]. The difficulty in providing good solutions reflects the underconstrained nature of SFM once the rigidity assumption is removed. Recent research has thus focused on the definition of new constraints (priors) to solve this problem [1]–[3], [23], [24], [26], [30], [32]–[34].

In the standard matrix factorization approach to SFM [5], [29], each column of the input matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ has a sequence of 2D coordinates of the same 3D structure point as observed from different locations. Considering $\mathbf{W}$ of a predefined low-rank $r \leq \min(m, n)$, the SFM solution is obtained from the factorization

$$\mathbf{W} = \mathbf{MS}, \quad \mathbf{M} \in \mathbb{R}^{m\times r}, \mathbf{S} \in \mathbb{R}^{r\times n}. \tag{1}$$

In rigid SFM, $r = 4$ and $\mathbf{S}$ describes the 3D shape observed by the cameras in $\mathbf{M}$. In non-rigid SFM, each observed shape of the deformable structure is represented in a linear shape space defined by K basis shapes in $\mathbf{S}$. Then, $r = 3K + 1$ and $\mathbf{M}$ includes the cameras and also the shape coordinates in terms of basis $\mathbf{S}$. In both cases, factors $\mathbf{M}$ and $\mathbf{S}$ may be obtained from the singular value decomposition (SVD) of $\mathbf{W}$. In practice, however, a large portion of the 2D observations in $\mathbf{W}$ is often missing because of occlusions. Therefore, standard matrix factorization algorithms such as SVD [13] cannot be directly used. To solve SFM, it is also necessary to overcome other challenges such as tracking errors [19] and degeneracies in the assumed camera motion and shape deformation – *i.e.,* when the above constraints on the rank r of $\mathbf{W}$ do not hold [21], [32], [34].

To compute more accurate and efficient solutions to rigid and non-rigid SFM with occlusion, we start by assuming that each column of $\mathbf{W}$ represents the smooth time-trajectory of a 2D point. Such 2D trajectories are usually provided by a feature tracking algorithm that operates on a monocular video sequence. Equivalently, we assume that the 2D observations were obtained by a single camera moving smoothly around the structure

• *The authors are with the Department of Electrical and Computer Engineering, The Ohio State Univerisy, 2015 Neil Avenue, 205 Dreese Laboratories, Columbus, OH 43210. E-mail: {gotardop,aleix}@ece.osu.edu.*

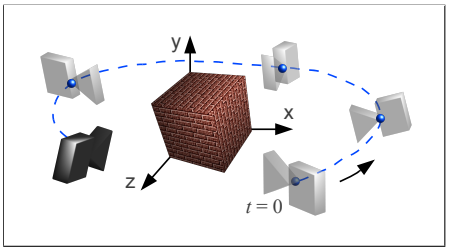


Fig. 1. The smooth trajectory (dashed line) of a single camera moving around the structure of interest (cube) over time (t).

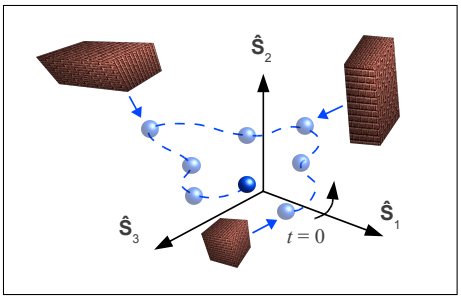


Fig. 2. The smooth time-trajectory (dashed line) of a deforming 3D shape (cube). The object shape is represented by a single point (blue sphere) with coordinates relative to 3D basis shapes $\hat{\mathbf{S}}_1$, $\hat{\mathbf{S}}_2$, and $\hat{\mathbf{S}}_3$ (not shown). As the shape deforms smoothly over time (t), its point representation describes a single, smooth 3D shape trajectory.

of interest (Fig. 1). Our factorization approach tolerates missing data and defines constraints that considerably reduce the number of unknowns that need to be estimated. First, we consider factor $\mathbf{S}$ only implicitly, reformulating the factorization problem in terms of $\mathbf{M}$ alone. Second, because $\mathbf{M}$ is a basis for the smooth 2D trajectories in the columns of $\mathbf{W}$, we solve for a compact representation of $\mathbf{M}$ in a subspace of the Discrete Cosine Transform (DCT) basis vectors. Our algorithms are remarkably efficient in cases of long image sequences and many imaged points (*i.e.*, when $\mathbf{W}$ has high-dimensional column and row spaces).

We solve for rigid SFM by estimating only the smooth time-trajectory of a camera's projection plane, Fig. 1. Also, by considering the weak-perspective camera model from the outset, we directly solve for Euclidean shape and cameras. Therefore, our method does not require post-processing algorithms that upgrade and refine the initial affine solution (*e.g.*, bundle adjustment [15]).

In non-rigid SFM, we assume that the smooth 2D trajectories in $\mathbf{W}$ also reflect the smooth deformation of the observed 3D structure over time. The deformable structure is represented as a single point moving smoothly in a linear shape space, Fig. 2. We then solve for a single, smooth *3D shape trajectory* with time-coordinates in factor $\mathbf{M}$. Thus, the associated basis shapes are defined only implicitly in factor $\mathbf{S}$. Once $\mathbf{M}$ has been estimated, $\mathbf{S}$ is then trivially computed from $\mathbf{M}$ and $\mathbf{W}$ by solving simple linear systems of equations.

Our work is most closely related to the recent non-rigid SFM method by Akhter *et al.* [2]. They propose a factorization approach that does not define a linear shape space, but recovers instead *independent 3D point trajectories* over time. The DCT vectors are used as a basis for individual 3D point trajectories. Their method has provided some of the best results on highly articulated shapes to date. However, the main problem is the method's inability to use additional, higher frequency DCT vectors without increasing the rank of the resulting matrix factors. Thus, its application is restricted to structures with slow and smooth deformation. The method does not address the missing data problem either; $\mathbf{W}$ is assumed complete and factorized using SVD.

In this paper, we also present an interpretation of Akhter *et al.*'s approach in terms of a linear shape space and show that their results correspond to coarse approximations of our solutions. Our non-rigid SFM method provides better results on complex articulated deformations due to its more effective use of higher-frequency DCT components without increasing the factorization rank. Furthermore, we use the DCT basis to model a camera's trajectory and efficiently solve for rigid SFM. Finally, we also contribute with missing data approaches for both rigid and non-rigid SFM.

One main group of related methods that address the missing data problem are known as *batch algorithms* [14], [17], [19], [23], [28]. These methods propose strategies for combining partial rank-r factorizations obtained for complete sub-blocks of $\mathbf{W}$. Examples include the subspace constraint algorithms that reconstruct $\mathbf{W}$ by first building its row null-space [17], [19], column null-space [14], [23], or one of its range spaces [28]. The main problem with these methods is their sub-optimality since errors in the factorization of sub-blocks are propagated to the subsequent optimization stage. Also, the rank-r constraint may not apply to all sub-blocks in case of degenerate motion and deformation [21], [23].

A second group of missing data approaches include iterative methods that use all data at once without searching for complete sub-blocks in $\mathbf{W}$ [6], [8], [16], [24], [30], [31]. Among these, alternation methods [16], [24], [30] iteratively solve for subsets of unknowns while the others remain fixed. For instance, PowerFactorization [16] solves for factors $\mathbf{M}$ and $\mathbf{S}$ in a simple, alternated least-squares manner. It presents very slow convergence when a considerable amount of data is missing. However, the method is useful in initializing faster Newton methods [6] that minimize matrix fitting error in terms of $\mathbf{M}$ and $\mathbf{S}$. The Levenberg-Marquardt-Subspace (LM-S) method in [8] considers $\mathbf{S}$ as an implicit function of $\mathbf{M}$ and $\mathbf{W}$ and solves for $\mathbf{M}$ only. Despite its superior performance, proper initialization of this method remains an open problem. The method's complexity also makes it difficulty to integrate additional constraints into the factorization problem.

In the following, we present our *Column Space Fitting* (CSF) method that computes a rank-r basis $\mathbf{M}$ for the column space of a matrix with missing data. Using a simple Gauss-Newton-based approximation to the Hessian ma-

trix, our method provides equivalent or better solutions for $\mathbf{M}$ when compared to LM-S on matrices with high percentage of missing data. Furthermore, the simplicity of our approach allows us to easily consider additional constraints in the factorization problem, such as a mean column vector and a pre-defined reference basis for $\mathbf{M}$. We then solve for a compact representation of $\mathbf{M}$ in a subspace defined in terms of the DCT basis vectors. As a result, we offer a family of CSF algorithms for rigid and non-rigid SFM problems. Our methods always start from a deterministic initialization corresponding to a "coarse" solution for $\mathbf{M}$.

This paper is organized as follows. Section 2 derives our general CSF approach for matrix factorization with missing data. In Section 3, we solve for rigid SFM by estimating a smooth time-trajectory of an affine or a weak-perspective camera. Section 4 describes our 3D shape trajectory approach to non-rigid SFM. Sections 5 and 6 present experimental results and conclusion.

2 Solving for the Column Space of a Matrix with Missing Data

We formulate the rank-r factorization of incomplete matrix $\mathbf{W}$ as solving exclusively for a column space basis $\mathbf{M}$ of rank-r. To facilitate the introduction of additional constraints into the factorization procedure, especially in SFM problems, we first derive our Column Space Fitting (CSF) algorithm. We then show how to solve for $\mathbf{M}$ in the subspace spanned by a predefined bases. When such a bases is defined in terms of the DCT basis vectors, we propose a simple, deterministic initial form of $\mathbf{M}$.

2.1 General CSF approach

Consider $\mathbf{W} \in \mathbb{R}^{m\times n}$ as in (1) and note the related equation $\mathbf{S} = \mathbf{M}^{\dagger}\mathbf{W}$, where $\dagger$ denotes the Moore-Penrose pseudo-inverse [13].

Assuming $\mathbf{W}$ is presented with missing data, let the complete vector $\mathbf{w}_j \in \mathbb{R}^{m_j}$ $(m_j \leq m)$ denote all the observed entries in the j^{th} column of $\mathbf{W}$. Also, define $\mathbf{\Pi}_j \in \mathbb{R}^{m_j\times m}$ as a row-amputated identity matrix such that $\mathbf{M}_j = \mathbf{\Pi}_j\mathbf{M}$ has the rows in $\mathbf{M}$ that correspond to the rows of entries in $\mathbf{w}_j$. Then $\mathbf{w}_j = \mathbf{M}_j\mathbf{s}_j$, with the complete j^{th} column of $\mathbf{S}$ defined as $\mathbf{s}_j = \mathbf{M}_j^{\dagger}\mathbf{w}_j \in \mathbb{R}^r$.

The goal is to minimize

$$f(\mathbf{M}) = \frac{1}{2}\sum_j \left\| \left(\mathbf{I} - \mathbf{M}_j\mathbf{M}_j^{\dagger}\right)\mathbf{w}_j \right\|_F^2, \tag{2}$$

where $\|\cdot\|_F$ is the Frobenius norm. To better understand the meaning of (2), let the projector on the orthogonal space of $\mathbf{M}_j$ be

$$\mathbf{P}_j^{\perp} = \left(\mathbf{I} - \mathbf{M}_j\mathbf{M}_j^{\dagger}\right) \in \mathbb{R}^{m_j\times m_j}. \tag{3}$$

Rewriting (2) in terms of residual (error) vectors $\mathbf{r}_j \in \mathbb{R}^{m_j}$, we have

$$f(\mathbf{M}) = \frac{1}{2}\sum_j \mathbf{r}_j^T\mathbf{r}_j \ , \quad \mathbf{r}_j = \mathbf{P}_j^{\perp}\mathbf{w}_j \ . \tag{4}$$

Algorithm 1 Column Space Fitting (CSF) method for minimizing the error function $f(\mathbf{M})$.

1: $\mathbf{M} \leftarrow$ initial matrix ($\mathbf{M}_0$).
2: $\delta \leftarrow$ initial damping scalar (δ_0).
3: **repeat**
4: Compute gradient ($\mathbf{g}$) and Hessian ($\mathbf{H}$) from Jacobian terms ($\mathbf{J}_j$).
5: **repeat**
6: $\delta \leftarrow \delta \times 10$.
7: Find $\Delta\mathbf{M}$ from $vec(\Delta\mathbf{M}) \leftarrow (\mathbf{H} + \delta\mathbf{I})^{-1}\mathbf{g}$.
8: **until** $f(\mathbf{M} - \Delta\mathbf{M}) < f(\mathbf{M})$.
9: $\mathbf{M} \leftarrow \mathbf{M} - \Delta\mathbf{M}$.
10: $\delta \leftarrow \delta \times 10^{-2}$.
11: Orthogonalize $\mathbf{M}$ (keep mean vector $\mathbf{t}$ unchanged).
12: **until** convergence.

Therefore, we minimize the overall matrix fitting error as defined by a sum of squared Euclidean distances from each observed column vector to the subspace spanned by the corresponding rows in $\mathbf{M}$.

The error function (2) is minimized using our Column Space Fitting (CSF) method, based on Levenberg-Marquardt optimization [4], which we summarize in Algorithm 1. $\mathbf{M}$ is first set to an initial matrix. Then, in each iteration, we update the current estimate of $\mathbf{M}$ by computing an adjustment matrix $\Delta\mathbf{M}$ in vectorized form, $vec(\Delta\mathbf{M})$, which stacks the columns of $\Delta\mathbf{M}$ in a single vector of unknowns. We solve for $vec(\Delta\mathbf{M})$ using the gradient vector ($\mathbf{g}$) and Hessian matrix ($\mathbf{H}$) of f as given by matrix differential calculus [20]. The damping parameter δ leads to combined Gauss-Newton and steepest-descent iterations when $\mathbf{H}$ becomes singular. For numerical stability, we orthogonalize $\mathbf{M}$ at the end of each iteration.

To compute $\mathbf{g}$ and $\mathbf{H}$, we follow a Gauss-Newton-based derivation in terms of Jacobian matrices $\mathbf{J}_j$ for each vector $\mathbf{w}_j$ (see complete derivation in Appendix A). The first and second differentials of $f(\mathbf{M})$ are

$$df = -\sum_j \left(\mathbf{J}_j^T\mathbf{r}_j\right)^T vec(d\mathbf{M}), \tag{5}$$

$$d^2f = \sum_j vec(d\mathbf{M})^T \left(\mathbf{J}_j^T\mathbf{J}_j\right) vec(d\mathbf{M}), \tag{6}$$

$$\mathbf{J}_j = \mathbf{s}_j^T \otimes \mathbf{P}_j^{\perp}\mathbf{\Pi}_j. \tag{7}$$

In (7), $\otimes$ is the Kronecker product and $\mathbf{s}_j = \mathbf{M}_j^{\dagger}\mathbf{w}_j$ denotes the current (and implicit) estimate of the j^{th} column of factor $\mathbf{S}$. From (5) and (6), we identify

$$\mathbf{g} = -\sum_j \mathbf{J}_j^T\mathbf{r}_j \quad \text{and} \quad \mathbf{H} = \sum_j \mathbf{J}_j^T\mathbf{J}_j \ . \tag{8}$$

For comparison, the forms of $\mathbf{g}$ and $\mathbf{H}$ as computed in [8] are reproduced in our supplementary documentation file. It is important to note that the simplicity of our expressions above makes it easier to consider additional optimization constraints as described next.

As an alternative to the derivations given above, we

can solve for rank-$(r-1)$ factors $\mathbf{M}$ and $\mathbf{S}$, and an additional mean column $\mathbf{t} \in \mathbb{R}^m$ such that $\mathbf{W} = \mathbf{MS} + \mathbf{t}\mathbf{1}^T$, where $\mathbf{1} \in \mathbb{R}^n$ is a vector of all ones. This is a very useful model in applications such as rigid and non-rigid SFM. Equivalently, we solve for a rank-r factorization

$$\mathbf{W} = \widetilde{\mathbf{M}}\widetilde{\mathbf{S}} = [\mathbf{M}\ \mathbf{t}] \begin{bmatrix} \mathbf{S} \\ \mathbf{1}^T \end{bmatrix}, \tag{9}$$

with extended factors $\widetilde{\mathbf{M}}$ and $\widetilde{\mathbf{S}}$. The mean column interpretation for the last column of $\widetilde{\mathbf{M}}$ constrains the last row of $\widetilde{\mathbf{S}}$ to be $\mathbf{1}^T$. Therefore, the model in (9) has fewer degrees of freedom than that in (1) and the final matrix fitting error, $f(\widetilde{\mathbf{M}})$, is expected to be higher.

The goal now is to minimize

$$f(\mathbf{M}, \mathbf{t}) = \frac{1}{2} \sum_j \mathbf{r}_j^T \mathbf{r}_j\ , \quad \mathbf{r}_j = \mathbf{P}_j^{\perp} (\mathbf{w}_j - \mathbf{t}_j), \tag{10}$$

with $\mathbf{t}_j = \mathbf{\Pi}_j \mathbf{t}$ and $\mathbf{P}_j^{\perp}$ computed from $\mathbf{M}_j$ only. The new Jacobian terms are then derived for $vec(d\widetilde{\mathbf{M}})$ as

$$\mathbf{J}_j = \left[\mathbf{s}_j^T\ 1\right] \otimes \mathbf{P}_j^{\perp} \mathbf{\Pi}_j, \tag{11}$$

with $\mathbf{s}_j = \mathbf{M}_j^{\dagger}(\mathbf{w}_j - \mathbf{t}_j)$ (see derivation in Appendix A).

In this case, at the end of each iteration, only the left-most $r-1$ columns of $\widetilde{\mathbf{M}}$ are orthogonalized and $\mathbf{t}$ is kept unchanged.

2.2 Solution with a predefined basis

Let the columns of a matrix $\mathbf{B} \in \mathbb{R}^{m \times d}$ contain a predefined set of basis vectors (*e.g.*, a truncated DCT basis). We now consider a factor $\mathbf{M}$ of the form

$$\mathbf{M} = \mathbf{BX}, \quad \mathbf{X} \in \mathbb{R}^{d \times r}.$$

Each column of $\mathbf{X}$ has $d \leq m$ coordinates for the corresponding column of $\mathbf{M}$ as represented in the d-dimensional space spanned by $\mathbf{B}$.

Previously, we had implicitly considered a canonical basis $\mathbf{B} = \mathbf{I}_m$, the $m \times m$ identity matrix, and solved for $\mathbf{X} = \mathbf{M}$. Now, in each iteration we solve for the update step $vec(d\mathbf{X})$ with gradient and Hessian as in (8). The Jacobian terms are given by

$$\mathbf{J}_j = \mathbf{s}_j^T \otimes \mathbf{P}_j^{\perp} \mathbf{\Pi}_j \mathbf{B}. \tag{12}$$

It is important to describe our motivation for using a predefined basis: Problems in different application domains involve matrices ($\mathbf{W}$) whose columns have observations of random variables that change only gradually over time. Observations in each column can be considered as samples of a smooth signal over time with a narrow bandwidth spectrum in the DCT domain. This means that most of the energy of a signal (column) is captured by a small number of low-frequency DCT components, leading to a compact representation $\mathbf{X}$ (*i.e.*, small d).

In the next sections, we consider matrices $\mathbf{B}$ constructed from DCT basis vectors, with the right-most columns corresponding to higher-frequency components. Since signal smoothness leads to high-frequency coefficients close to zero, we propose an initialization of $\mathbf{X}$, denoted $\mathbf{X}_0$, of the form

$$\mathbf{X}_0 = \begin{bmatrix} \mathbf{Q} \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{Q} \in \mathbb{R}^{r \times r}.$$

The goal becomes initializing a small block $\mathbf{Q}$, which must be full-rank (*i.e.*, the rank of $\mathbf{M} = \mathbf{BX}_0$ is $rank(\mathbf{Q})$). For any full-rank $\mathbf{Q}$, an equally good initial solution is given by the simple and deterministic initialization

$$\mathbf{X}_0 = \begin{bmatrix} \mathbf{Q} \\ \mathbf{0} \end{bmatrix} \mathbf{Q}^{-1} = \begin{bmatrix} \mathbf{I}_r \\ \mathbf{0} \end{bmatrix}. \tag{13}$$

As a proof, we note that the factorization in (1) is defined only up to a rank-r ambiguity matrix $\mathbf{Q}$ such that $\mathbf{W} = \mathbf{MS} = (\mathbf{MQ}^{-1})(\mathbf{QS})$.

As a result, we can even fix the values for the top-most block of $\mathbf{X}$ as $\mathbf{I}_r$ and solve only for the remaining $(d-r) \times r$ submatrix. Note that $(d-r)r$ is the number of parameters defining an r-dimensional linear space embedded in another d-dimensional linear space.

When the last column of $\mathbf{X}$ is interpreted as the coefficients of the mean column vector $\mathbf{t}$, the ambiguity matrix $\mathbf{Q}$ is of rank $(r-1)$ only, *i.e.*, the last row of $\mathbf{S}$ is constrained to be $\mathbf{1}^T$. This fact suggests to initialize the last column of $\mathbf{X}$ with zeros ($\mathbf{t} = \mathbf{0}$).

3 Rigid SFM: Estimating the Smooth Time-Trajectory of a Camera

This section focuses on rigid SFM with occlusion. We assume the point tracks in $\mathbf{W}$ were obtained from a monocular video sequence provided by a single, smoothly moving camera. First, we present a method using a compact DCT basis to represent the smooth trajectory of a general affine camera. Subsequently, we further improve on this method by using a weak-perspective camera model from the outset to directly estimate Euclidean cameras and shape. Once the final solution for $\mathbf{M}$ is defined, each column $\mathbf{s}_j$ of $\mathbf{S}$ is obtained independently as in the computation of the Jacobian terms above.

3.1 Affine camera trajectory

In rigid SFM with T images, $\mathbf{W} \in \mathbb{R}^{2T \times n}$ is

$$\mathbf{W} = \begin{bmatrix} x_{11} & \dots & x_{1n} \\ y_{11} & \dots & y_{1n} \\ \vdots & & \vdots \\ x_{T1} & \dots & x_{Tn} \\ y_{T1} & \dots & y_{Tn} \end{bmatrix} = \mathbf{M} \begin{bmatrix} \mathbf{s}_1 & \dots & \mathbf{s}_n \\ 1 & & 1 \end{bmatrix}, \tag{14}$$

with $[x_{tj}, y_{tj}]^T$ the 2D projection of 3D point $\mathbf{s}_j$ in the t^{th} image ($t = 1, 2, \dots, T$; $j = 1, 2, \dots, n$). Here, the motion factor $\mathbf{M} \in \mathbb{R}^{2T \times 4}$ describes a stack of T affine camera

matrices $\widehat{\mathbf{M}}_t \in \mathbb{R}^{2\times 4}$,

$$\mathbf{M} = \begin{bmatrix} \widehat{\mathbf{M}}_1 \\ \vdots \\ \widehat{\mathbf{M}}_T \end{bmatrix}, \quad \widehat{\mathbf{M}}_t = \begin{bmatrix} \widehat{\mathbf{A}}_t & \widehat{\mathbf{t}}_t \end{bmatrix}, \tag{15}$$

where $\widehat{\mathbf{A}}_t \in \mathbb{R}^{2\times 3}$ is a general affine projection and $\widehat{\mathbf{t}}_t \in \mathbb{R}^2$ is a 2D translation. The rows of $\widehat{\mathbf{A}}_t$ are 3D vectors defining the x and y axes (not necessarily orthonormal) of the camera's projection plane in the t^{th} observation.

Assuming the 8 camera parameters vary smoothly over time, we define the camera as a sample of a smooth matrix function of time, $\widehat{\mathbf{M}}_t = \widehat{\mathbf{M}}(t)$. We then parameterize $\widehat{\mathbf{M}}(t)$ using 8 independent cosine series whose f^{th} frequency coefficients are given in $\widehat{\mathbf{X}}_f \in \mathbb{R}^{2\times 4}$,

$$\widehat{\mathbf{M}}(t) = \left[\widehat{\mathbf{A}}(t)\ \widehat{\mathbf{t}}(t)\right] = \sum_{f=1}^{d} \omega_{tf} \widehat{\mathbf{X}}_f. \tag{16}$$

Each constant ω_{tf} above is the f^{th} frequency cosine term at time t,

$$\omega_{tf} = \frac{\sigma_f}{\sqrt{T}} \cos\left(\frac{\pi(2t-1)(f-1)}{2T}\right),$$

with $\sigma_1 = 1$ and, for $f \geq 2$, $\sigma_f = \sqrt{2}$.

The goal is then to solve for $d \leq T$ matrices $\widehat{\mathbf{X}}_f$. Let $\mathbf{\Omega}_d \in \mathbb{R}^{T\times d}$ be a truncated, orthonormal DCT matrix whose $(t,f)^{th}$ entry is ω_{tf}, as above. Substituting (16) into (15) yields

$$\mathbf{M} = (\mathbf{\Omega}_d \otimes \mathbf{I}_2) \begin{bmatrix} \widehat{\mathbf{X}}_1 \\ \vdots \\ \widehat{\mathbf{X}}_d \end{bmatrix} = \mathbf{B}_{af}\mathbf{X}.$$

Thus, with basis $\mathbf{B}_{af} = (\mathbf{\Omega}_d \otimes \mathbf{I}_2) \in \mathbb{R}^{2T\times 2d}$, we solve for $\mathbf{M} \in \mathbb{R}^{2T\times 4}$ as a function of $\mathbf{X} \in \mathbb{R}^{2d\times 4}$. Note that the smoothness assumption implies $d \ll T$. We solve for $\mathbf{X}$ as in Section 2.2, with extended terms $\widetilde{\mathbf{s}}_j^T = [\mathbf{s}_j^T\ 1]$ in (12).

For simplicity, and unless stated otherwise, we assume in this paper that the number of DCT components d has been fixed *a priori* based on empirical observations (*e.g.*, expected motion or noise level). An alternative *coarse-to-fine* strategy can start with a small d and increment its value until a convergence criterion is verified. For instance, d can be incremented until the energy of the highest frequency coefficients falls below a small percentage $p_d \in (0,1)$ of the total energy,

$$\left\|\mathbf{x}_d^T\right\|_F < p_d \left\|\mathbf{X}\right\|_F,$$

where $\mathbf{x}_d^T$ is the last row of $\mathbf{X}$.

3.2 Euclidean camera trajectory

The solution $\mathbf{X}$ above leads to an affine shape $\mathbf{S}$. Recovering Euclidean shape and cameras requires a subsequent upgrading step and, ideally, a final bundle adjustment [15]. Because an affine camera has more degrees of freedom than the final Euclidean camera, the initial solution can overfit measurement noise and also lead to incorrect results in cases of degenerate, planar camera motion [21]. These are problems faced by most SFM methods based on an initial affine solution. To avoid them and also render post-processing unnecessary, we further improve on our method above.

We now solve directly for the smooth trajectory of an Euclidean camera, also obtaining the optimal Euclidean shape directly. We consider the trajectory of a weak-perspective (*i.e.*, scaled-orthographic) camera,

$$\widehat{\mathbf{M}}_t = \begin{bmatrix} \widehat{\mathbf{R}}_t & \widehat{\mathbf{t}}_t \end{bmatrix},$$

where the vector $\widehat{\mathbf{t}}_t$ is defined as before. The rows of $\widehat{\mathbf{R}}_t \in \mathbb{R}^{2\times 3}$ define the Euclidean 2D imaging plane of the camera at time t. These rows are constrained to be two equal-length, orthogonal 3D vectors and are obtained from a scaled 3D rotation (see also (17)),

$$\widehat{\mathbf{R}}_t = \lambda_t \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{R}_Z(\alpha_t)\mathbf{R}_Y(\beta_t)\mathbf{R}_Z(\gamma_t).$$

The scalar λ_t is the weak-perspective scale and the three Euler angles α_t, β_t, and γ_t determine a 3D rotation as a sequence of simpler rotations around the Z and Y world coordinate axes.

We model a smooth camera trajectory by considering a cosine series for each of the 6 camera parameters given by the weak-perspective model above,

$$\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_T \end{bmatrix} = \mathbf{\Omega}_d\mathbf{x}_1, \quad \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_T \end{bmatrix} = \mathbf{\Omega}_d\mathbf{x}_2, \quad \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_T \end{bmatrix} = \mathbf{\Omega}_d\mathbf{x}_3,$$

$$\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_T \end{bmatrix} = \mathbf{\Omega}_d\mathbf{x}_4, \quad \mathbf{t} = \mathbf{B}_{af}\mathbf{x}_5,$$

where vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4 \in \mathbb{R}^d$ and $\mathbf{x}_5 \in \mathbb{R}^{2d}$ have the unknown DCT coefficients. For simplicity of presentation, the notation above assumes that d is the same for $\mathbf{x}_1, \ldots, \mathbf{x}_5$. Note, however, that our approach is not limited to such case – an alternative derivation is given in the supplementary file for the case in which $\mathbf{x}_1, \ldots, \mathbf{x}_5$ have different numbers of DCT components.

A deterministic initial solution is defined with $\mathbf{t} = \mathbf{0}$ ($\mathbf{x}_5 = \mathbf{0}$), as before. Also, note that the rigid SFM solution we seek here is defined only up to an ar-

$$\widehat{\mathbf{R}}_t = \lambda_t \begin{bmatrix} \cos\alpha_t\cos\beta_t\cos\gamma_t - \sin\alpha_t\sin\gamma_t & -\cos\alpha_t\cos\beta_t\sin\gamma_t - \sin\alpha_t\cos\gamma_t & \cos\alpha_t\sin\beta_t \\ \sin\alpha_t\cos\beta_t\cos\gamma_t + \cos\alpha_t\sin\gamma_t & -\sin\alpha_t\cos\beta_t\sin\gamma_t + \cos\alpha_t\cos\gamma_t & \sin\alpha_t\sin\beta_t \end{bmatrix} \tag{17}$$

bitrary scale and a 3×3 rotation. We thus initialize the sequence of scaling factors λ_t to a constant signal by setting $\mathbf{x}_4 = [1, 0, \ldots, 0]^T$. We avoid initializing the angles β_t ($\mathbf{x}_2$) with zeros because the resulting $\widehat{\mathbf{R}}_t$ would then represent only 2D rotations within a plane. Setting $\mathbf{x}_2 = [0, 1, 0, \ldots, 0]^T$ provides a coarse initial solution with a smooth sequence of *non-coincident* camera planes that allows $\mathbf{x}_1$ and $\mathbf{x}_3$ to be initialized with zeros.

Here, we iteratively solve for $vec(d\mathbf{X})$ defined as

$$vec(d\mathbf{X}) \equiv \left[d\mathbf{x}_1^T, d\mathbf{x}_2^T, d\mathbf{x}_3^T, d\mathbf{x}_4^T, d\mathbf{x}_5^T\right]^T \in \mathbb{R}^{6d}.$$

To this end, modified expressions for $vec(d\mathbf{M})$ and the Jacobian terms are needed. First, let $\boldsymbol{\omega}_t^T$ be the t^{th} row of $\boldsymbol{\Omega}_d$. The differential of each camera parameter at time t is $d\alpha_t = \boldsymbol{\omega}_t^T d\mathbf{x}_1$, $d\beta_t = \boldsymbol{\omega}_t^T d\mathbf{x}_2$, $d\gamma_t = \boldsymbol{\omega}_t^T d\mathbf{x}_3$, $d\lambda_t = \boldsymbol{\omega}_t^T d\mathbf{x}_4$, and $d\mathbf{t}_t = (\boldsymbol{\omega}_t^T \otimes \mathbf{I}_2) d\mathbf{x}_5$. The differential of each rotation matrix (17) is

$$d\widehat{\mathbf{R}}_t = \frac{\partial \widehat{\mathbf{R}}_t}{\partial \alpha_t} d\alpha_t + \frac{\partial \widehat{\mathbf{R}}_t}{\partial \beta_t} d\beta_t + \frac{\partial \widehat{\mathbf{R}}_t}{\partial \gamma_t} d\gamma_t + \frac{\partial \widehat{\mathbf{R}}_t}{\partial \lambda_t} d\lambda_t.$$

Considering the three columns $\widehat{\mathbf{r}}_{t,1}$, $\widehat{\mathbf{r}}_{t,2}$, and $\widehat{\mathbf{r}}_{t,3} \in \mathbb{R}^2$ of a partial derivative matrix $\frac{\partial \widehat{\mathbf{R}}_t}{\partial \alpha_t}$, for all t, we stack all α-terms associated with $d\mathbf{x}_1$ into

$$\mathbf{B}_\alpha = \begin{bmatrix} \widehat{\mathbf{r}}_{1,1} \otimes \boldsymbol{\omega}_1^T \\ \vdots \\ \widehat{\mathbf{r}}_{T,1} \otimes \boldsymbol{\omega}_T^T \\ \widehat{\mathbf{r}}_{1,2} \otimes \boldsymbol{\omega}_1^T \\ \vdots \\ \widehat{\mathbf{r}}_{T,2} \otimes \boldsymbol{\omega}_T^T \\ \widehat{\mathbf{r}}_{1,3} \otimes \boldsymbol{\omega}_1^T \\ \vdots \\ \widehat{\mathbf{r}}_{T,3} \otimes \boldsymbol{\omega}_T^T \end{bmatrix} \in \mathbb{R}^{6T \times d}.$$

Analogously, we define $\mathbf{B}_\beta$, $\mathbf{B}_\gamma$, and $\mathbf{B}_\lambda$. Hence,

$$\begin{aligned} vec(d\mathbf{M}) &= \underbrace{\begin{bmatrix} \mathbf{B}_\alpha & \mathbf{B}_\beta & \mathbf{B}_\gamma & \mathbf{B}_\lambda & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{af} \end{bmatrix}}_{\widetilde{\mathbf{B}}_{wp}} \begin{bmatrix} d\mathbf{x}_1 \\ d\mathbf{x}_2 \\ d\mathbf{x}_3 \\ d\mathbf{x}_4 \\ d\mathbf{x}_5 \end{bmatrix} \\ &= \widetilde{\mathbf{B}}_{wp}\ vec(d\mathbf{X}), \end{aligned}$$

with $\widetilde{\mathbf{B}}_{wp} \in \mathbb{R}^{8T \times 6d}$ and $\mathbf{0}$ is a matrix of zeros.

Finally, from (5)-(7), the Jacobian terms of this camera model can be computed as

$$\mathbf{J}_j = \left(\begin{bmatrix} \mathbf{s}_j^T & 1 \end{bmatrix} \otimes \mathbf{P}_j^\perp \boldsymbol{\Pi}_j\right) \widetilde{\mathbf{B}}_{wp}.$$

Unlike in the previous sections, here the extended basis $\widetilde{\mathbf{B}}_{wp}$ must be recomputed in each iteration to update the partial derivatives in $\mathbf{B}_\alpha$, $\mathbf{B}_\beta$, $\mathbf{B}_\gamma$, and $\mathbf{B}_\lambda$. Thus, $\widetilde{\mathbf{B}}_{wp}$ can be considered only as a local basis at the current location of the smooth parameter manifold.

The last step in each iteration shown in Algorithm 1, orthogonalization of $\mathbf{M}$, is no longer necessary.

4 Non-Rigid SFM: Estimating a Smooth-Trajectory in Shape Space

In this section, we solve for non-rigid SFM by estimating the smooth time-trajectory of a 3D shape as represented by a point moving in a linear shape space. Without loss of generality, we assume $\mathbf{W}$ is complete and derive our *3D shape trajectory* approach. Subsequently, we offer an algorithm for cases with missing data.

4.1 Solving for a shape trajectory

Let $\mathbf{W} \in \mathbb{R}^{2T \times n}$ as in (14) and consider the rank-r factorization method, with $r = 3K + 1$, proposed by Bregler *et al.* [5] for non-rigid scenes,

$$\mathbf{W} = \underbrace{\mathbf{D}\,(\mathbf{C} \otimes \mathbf{I}_3)}_{\mathbf{M}}\,\mathbf{S} + \mathbf{t}\mathbf{1}^T,$$

with $\mathbf{D} \in \mathbb{R}^{2T \times 3T}$, $\mathbf{C} \in \mathbb{R}^{T \times K}$, and $\mathbf{S} \in \mathbb{R}^{3K \times n}$. The predefined constant K is the number of 3D basis shapes $\widehat{\mathbf{S}}_k \in \mathbb{R}^{3 \times n}$ defining a linear shape space in $\mathbf{S} = [\widehat{\mathbf{S}}_1^T\ \widehat{\mathbf{S}}_2^T\ \ldots\ \widehat{\mathbf{S}}_K^T]^T$.

The column space factor $\mathbf{M}$ is composed of a block-diagonal rotation matrix $\mathbf{D}$,

$$\mathbf{D} = \begin{bmatrix} \widehat{\mathbf{R}}_1 & & & \\ & \widehat{\mathbf{R}}_2 & & \\ & & \ddots & \\ & & & \widehat{\mathbf{R}}_T \end{bmatrix},$$

and a shape coordinate matrix $\mathbf{C}$. Each row $\mathbf{c}_t^T \in \mathbb{R}^K$ of $\mathbf{C}$ has the coordinates of the 3D shape in the t^{th} image with respect to the shape basis in $\mathbf{S}$. Here, we also consider $\mathbf{c}_t^T = c(t)$ as a single point in shape space that defines *a single smooth 3D shape trajectory over time*.

We assume that the smooth 2D trajectories in $\mathbf{W}$ reflect not only smooth camera motion, but also the smooth deformation of the observed 3D structure over time. This means that each shape coordinate $c_{tk} \in \mathbb{R}$ ($k = 1, 2, \ldots, K$) is assumed to vary smoothly with t. Then, we represent $\mathbf{C}$ using K compact cosine series,

$$\mathbf{C} = \begin{bmatrix} c_{1,1} & \ldots & c_{1,K} \\ \vdots & \ddots & \vdots \\ c_{T,1} & \ldots & c_{T,K} \end{bmatrix} = \boldsymbol{\Omega}_d \begin{bmatrix} \mathbf{x}_1, & \ldots, & \mathbf{x}_K \end{bmatrix},$$

with $\mathbf{x}_k \in \mathbb{R}^d$ ($k = 1, \ldots, K$),

$$\mathbf{C} = \boldsymbol{\Omega}_d \mathbf{X}, \quad \mathbf{X} \in \mathbb{R}^{d \times K}.$$

Assuming $\mathbf{W}$ is complete, $\mathbf{t}$ is estimated simply as the mean column of $\mathbf{W}$. Then, to recover Euclidean shapes and cameras, we consider $\mathbf{S}$ implicitly and solve for

$$\mathbf{M} = \mathbf{D}\,(\boldsymbol{\Omega}_d \mathbf{X} \otimes \mathbf{I}_3)\,, \tag{18}$$

with $\mathbf{D}$ subject to camera orthonormality constraints, *i.e.*, $\widehat{\mathbf{R}}_t \widehat{\mathbf{R}}_t^T = \mathbf{I}_2, \forall t$. For now, let's assume $\mathbf{D}$ has been computed by an initialization algorithm. We will define this algorithm below. Thus, we only need to solve for the rank-K shape trajectory $\mathbf{X}$ in the DCT domain.

For a fixed $\mathbf{D}$, the factor $\mathbf{C} = \mathbf{\Omega}_d\mathbf{X}$ is defined only up to a full-rank ambiguity $\mathbf{Q} \in \mathbb{R}^{K\times K}$; equivalently, $\mathbf{M}$ in (18) is defined only up to an ambiguity $\mathbf{Q} \otimes \mathbf{I}_3$. Therefore, as in Section 2.2, we can initialize $\mathbf{X}$ with a coarse solution $\mathbf{X}_0 = [\ \mathbf{I}_K\ \mathbf{0}\ ]^T$, leading to

$$\mathbf{M}_0 = \mathbf{D}\left(\mathbf{\Omega}_d\mathbf{X_0} \otimes \mathbf{I}_3\right) = \mathbf{D}\left(\mathbf{\Omega}_K \otimes \mathbf{I}_3\right). \tag{19}$$

Note that $K < d$ and the initial rank-$3K$ solution in (19) can only use K low-frequency vectors in the DCT basis.

The *3D point trajectory* approach (PTA) in [2] defines $\mathbf{M} = \mathbf{D\Theta}$, where $\mathbf{\Theta}$ has the same columns as $(\mathbf{\Omega}_K \otimes \mathbf{I}_3)$ in (19), but in a different order. Therefore, our coarse initial solution $\mathbf{M}_0$ is equivalent to the final solution $\mathbf{M}$ of PTA (with factor $\mathbf{S}$ presenting a different order of rows). Note that the PTA method cannot consider additional DCT basis vectors without increasing K, leading to a higher rank of $\mathbf{M}$. Our *3D shape trajectory* approach, on the other hand, can consider any number $d = K, \ldots, T$ of DCT basis vectors because the linear combination represented by $\mathbf{X}$ constrains $\mathbf{M}$ to be of rank-$3K$. That is, our method can better model structure deformation presenting higher-frequency components in the DCT domain, yielding better 3D shape reconstructions.

Empirically, we have found that the coarse solutions computed by PTA contain accurate estimates of the rotation matrices in $\mathbf{D}$. We iteratively run PTA with increasing values of $K \in \{1, 2, \ldots, \lfloor \frac{n}{3} \rfloor\}$, obtaining a solution denoted as $\mathbf{D}_K$. Iterations stop automatically when there is no additional improvement in the average camera orthonormality,

$$\varepsilon(\mathbf{D}_K) = \frac{1}{T}\sum_{t=1}^{T}\left\|\mathbf{I}_2 - \widehat{\mathbf{R}}_t\widehat{\mathbf{R}}_t^T\right\|_F^2.$$

Given $\mathbf{D} = \mathbf{D}_K$, we then solve for $\mathbf{M}$ as a function of $\mathbf{X}$ only. The first differential of $\mathbf{M}$ in (18) is given by

$$d\mathbf{M} = \underbrace{\mathbf{D}\left(\mathbf{\Omega}_d \otimes \mathbf{I}_3\right)}_{\mathbf{B}_{nr}}\left(d\mathbf{X} \otimes \mathbf{I}_3\right) = \mathbf{B}_{nr}\left(d\mathbf{X} \otimes \mathbf{I}_3\right). \tag{20}$$

Next, consider $vec(d\mathbf{X} \otimes \mathbf{I}_3) = \mathbf{V}vec(d\mathbf{X})$, with $\mathbf{V} \in \mathbb{R}^{9dK\times dK}$ a binary mapping matrix. From (12) and (20), the Jacobian terms for the update step $vec(d\mathbf{X})$ are

$$\mathbf{J}_j = \left(\mathbf{s}_j^T \otimes \mathbf{P}_j^{\perp}\mathbf{\Pi}_j\mathbf{B}_{nr}\right)\mathbf{V}.$$

with $\mathbf{s}_j = \mathbf{M}_j^{\dagger}(\mathbf{w}_j - \mathbf{t}_j)$. For $\mathbf{X}$ with dimensions $d \times K$, the constant and sparse matrix $\mathbf{V}$ is defined as [20],

$$\mathbf{V} = \mathbf{I}_K \otimes \left[\left(\mathbf{K}_{3d} \otimes \mathbf{I}_3\right)\left(\mathbf{I}_d \otimes vec(\mathbf{I}_3)\right)\right],$$

where the permutation matrix $\mathbf{K}_{3d} \in \mathbb{R}^{3d\times 3d}$ satisfies $vec(\mathbf{A}^T) = \mathbf{K}_{3d}vec(\mathbf{A})$, for any $\mathbf{A} \in \mathbb{R}^{3\times d}$.

4.2 Non-Rigid SFM with missing data

The method above can estimate $\mathbf{X}$ using only the observed data in $\mathbf{W}$. We now consider the initial estimation of $\mathbf{t}$ and $\mathbf{D}$ (the camera motion) in cases of missing data. To compute $\mathbf{t}$ and $\mathbf{D}$ from a coarse solution as above, we first recover a complete, rank-r $\mathbf{W}$ via the factorization

$$\mathbf{W} = \underbrace{\mathbf{B}_{af}\mathbf{X}}_{\mathbf{M}}\begin{bmatrix}\mathbf{S}\\ \mathbf{1}^T\end{bmatrix}, \quad \mathbf{X} \in \mathbb{R}^{2d\times r},$$

with a predefined $r \in \{4, 5, \ldots, 3K+1\}$ and the DCT basis $\mathbf{B}_{af}$ of Section 3. Instead of considering the trajectory of an affine camera, here $\mathbf{B}_{af}$ is a basis for *individual, smooth 2D point trajectories* in the column space of $\mathbf{W}$.

Note that a common solution for $\mathbf{M}$ corresponds to the eigenvector matrix $\mathbf{U}$ of $\mathbf{W}\mathbf{W}^T$ (if $\mathbf{W}$ is complete). Here, our solution $\mathbf{M} = \mathbf{B}_{af}\mathbf{X}$ can be seen as using linear combinations of the DCT basis vectors to approximate the eigenvectors in $\mathbf{U}$. Indeed, the DCT basis vectors have been found to be good approximations of the Karhunem-Loeve (eigenfunction) transform of first-order Markov processes [18]. Also, the Markov assumption is used in algorithms that track the 2D points in $\mathbf{W}$ [11].

5 Experimental Results

In this section we provide extensive experimental validation for the proposed algorithms. General matrix factorization performance is first assessed on different, synthetic and real datasets. Subsequently, we apply our methods on rigid and non-rigid SFM data.

5.1 Fitting low-rank matrices

Our initial experiments analyze the general performance of our CSF method in its simplest form (*i.e.*, with a canonical basis $\mathbf{I}$ and without a mean column vector $\mathbf{t}$) in the low-rank factorization of synthetic and real data matrices with missing data. The method is evaluated against the LM-S algorithm [8] and its Gauss-Newton variant, LM-S$_{GN}$ (presented in the supplementary file). We also provide results using PowerFactorization (PF) [16] as a baseline algorithm.

We start by generating a random 20×30 matrix $\mathbf{W}$ of rank 3 with values uniformly distributed in the interval $[0, 1]$. Then, we add Gaussian noise with standard deviation σ_n and randomly occlude $\rho\%$ of the matrix entries. The algorithms above are used to compute a rank-3 factorization $\mathbf{W} = \mathbf{MS}$ starting from the same initial factor $\mathbf{M} = \mathbf{M}_0$, which is generated randomly and then refined with 20 iterations of the PF method. We always set the initial damping scalar to $\delta_0 = 10^{-4}$. Each algorithm runs until the change in the cost value (2) is less than 10^{-10} or until the number of iterations reaches 1,000. Note that, although we know the ground truth data, we do not know where the optimal solution for the matrix with missing data is [8]. Therefore, we first run all algorithms and, considering the solution obtained by each algorithm, we then define the trial's target (*i.e.*, "optimal") cost as the smallest cost observed.

We perform 500 trials, with σ_n and ρ fixed, and report the frequency (%) with which a method failed to match the trial's "optimal cost". We consider two cost values as equivalent if their absolute difference is below 10^{-7}.

TABLE 1
Algorithm performances on random matrices with missing data: frequency of sub-optimal solutions (%) and average number of iterations (in parenthesis).

$\rho(\%)$, σ_n	PF	LM-S	LM-S$_{GN}$	CSF
25, .1	23 (313)	**1 (16)**	4 (82)	3 (57)
25, .2	30 (391)	**3 (17)**	6 (93)	**3** (65)
25, .4	28 (381)	**3 (16)**	8 (108)	5 (74)
50, .1	71 (813)	12 (51)	14 (96)	**11 (79)**
50, .2	76 (831)	15 (58)	14 (105)	**12 (89)**
50, .4	77 (869)	18 (61)	19 (120)	**16 (103)**
75, .1	100 (1000)	69 (356)	62 (380)	**42 (549)**
75, .2	100 (1000)	72 (376)	54 (419)	**46 (586)**
75, .4	100 (1000)	77 (421)	65 (473)	**44 (698)**

TABLE 2
Algorithm performances on real datasets with missing data: frequency of sub-optimal convergence (%) and average number of iterations (in parenthesis).

Dataset	PF	LM-S	LM-S$_{GN}$	CSF	CSF-B$_{af}$
Dinosaur	100 (500)	26 (225)	3 (92)	**1 (101)**	**0 (12)**
Giraffe	100 (500)	0 (192)	0 (86)	**0 (61)**	**0 (49)**
Face	100 (500)	65 (36)	62 (64)	**62 (43)**	–

Results for different values of σ_n and ρ are shown in Table 1, which also includes the average number of iterations performed by each method. As expected, the simple PF method converges very slowly and is competitive only when ρ is very small. For $\rho \in \{25\%, 50\%\}$, CSF's performance is either comparable or better than that of LM-S and LM-S$_{GN}$. CSF also converges faster than LM-S$_{GN}$ for smaller fractions of missing data. With $\rho = 75\%$, CSF clearly outperforms the other methods. These results are highly revelant to SFM, where the number of missing entries is usually large ($> 75\%$).

A second, similar experiment considers the three real datasets used in [6] and [8]: (i) the *dinosaur* on a turn table, with rank-4 $\mathbf{W} \in \mathbb{R}^{72\times319}$ and 76.9% missing data; (ii) the occluded motion of a *giraffe*, with rank-6 $\mathbf{W} \in \mathbb{R}^{240\times166}$ and 30.2% missing data; and (iii) the *face* illumination data under a moving light source, with rank-4 $\mathbf{W} \in \mathbb{R}^{20\times2944}$ and 41.7% missing data. For each dataset, the smallest known root-mean-square error (RMSE) of the observed entries, as previously published [6], gives the target value that indicates optimal convergence.

We performed 100 trials corresponding to 100 different initial factors $\mathbf{M}$ generated as above. Each algorithm was allowed to run for at most 500 iterations. Table 2 shows the frequency of sub-optimal convergence (%) and average number of iterations for each algorithm. We first note that on the face dataset, all algorithms except PF often converged to a sub-optimal solution whose cost is only $.45\%$ (about 10^{-4}) higher than that of the optimal solution. If this solution is considered equivalent to the optimal one, the new sub-optimal convergence frequencies are 9% (LM-S), 8% (LM-S$_{GN}$), and 1% (CSF).

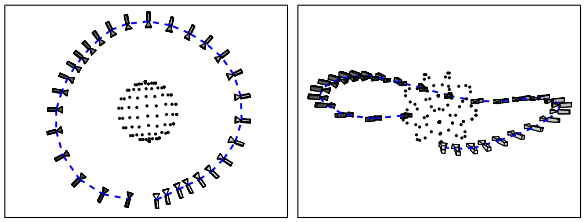


Fig. 3. Smooth camera trajectory in the synthetic sphere dataset.

Nevertheless, in the results above, CSF either provides better solutions or computes equivalent solutions with fewer iterations as compared to the other methods.

Because the columns of the dinosaur and giraffe matrices have measurements (alternated x- and y-coordinates) that vary smoothly over subsequent pairs of rows, we also computed factorizations with the CSF-B$_{af}$ variant of our method. We used the full $\mathbf{B}_{af}$ basis and started with the deterministic initialization in (13) as a coarse initial solution. CSF-B$_{af}$ presented optimal convergence on both datasets, without the initial use of PF.

5.2 Computing rigid SFM

Our first experiment on rigid SFM considers a synthetic *sphere* dataset with known 3D points ($n = 100$), located on its surface, and also known camera matrices ($T = 90$) describing a smooth trajectory over time, Fig. 3. The real occlusion pattern for the simple spherical shape can be easily determined and the resulting observation matrix $\mathbf{W} \in \mathbb{R}^{180\times100}$ is missing 50% of its entries.

Results of our SFM methods with affine (CSF-B$_{af}$) and weak-perspective (CSF-B$_{wp}$) cameras are compared against those of PF, LM-S, LM-S$_{GN}$, and the Wiberg algorithm [31]. Euclidean upgrade is applied to the results of affine methods. All results are aligned (rotated and scaled) with the ground-truth shape and motion before comparison. On the noiseless $\mathbf{W}$, the methods above provide camera and shape reconstructions with nearly zero error. Our CSF methods are run with only 30% of the DCT basis vectors ($d = .3T$).

In our experiment, we analyze how the performance of these methods degrades with different levels of Gaussian noise added to $\mathbf{W}$. We now set $\sigma_n = \hat{\sigma}_n\sigma(\mathbf{W})$, where $\hat{\sigma}_n \in [.05, .25]$ and $\sigma(\mathbf{W})$ indicates the scale of the entries of $\mathbf{W}$. For any matrix $\mathbf{A}$ with m rows, let

$$\sigma(\mathbf{A}) = \frac{1}{m}\sum_{r=1}^{m}\sigma_r \ , \tag{21}$$

where σ_r is the standard deviation of the available entries in the r^{th} row of $\mathbf{A}$. For each value of $\hat{\sigma}_n$, we perform 100 trials and report the average error of the recovered 3D shape (e_S) and camera rotations (e_R). The error e_S is the average Euclidean distance between original and recovered 3D points, normalized by the radius of the original 3D sphere. Let the original and

TABLE 3
Results of rigid SFM methods on the sphere dataset.

Method	$\hat{\sigma}_n = .05$		$\hat{\sigma}_n = .125$		$\hat{\sigma}_n = .25$	
	e_S	e_R	e_S	e_R	e_S	e_R
PF	.0163	.0250	.0818	.1247	.2092	.2839
LM-S	.0169	.0255	.0845	.1284	.1780	.2690
LM-S$_{GN}$	.0170	.0255	.0845	.1284	.1777	.2686
Wiberg	.0163	.0250	.0806	.1242	.1695	.2569
CSF-B$_{af}$	.0163	.0151	.0806	**.0731**	.1651	**.1513**
CSF-B$_{wp}$	.0156	.0130	.0779	**.0606**	.1586	**.1280**

estimated rotations be $\widehat{\mathbf{R}}_t^*$ and $\widehat{\mathbf{R}}_t$, then e_R is

$$e_R = \frac{1}{T}\sum_{t=1}^{T}\left\|\widehat{\mathbf{R}}_t^* - \widehat{\mathbf{R}}_t\right\|_F. \quad (22)$$

CSF-B$_{af}$ and CSF-B$_{wp}$ were run with deterministic initializations. The other methods require random initializations and often provided very poor solutions in our experiments. For this reason, in each trial they were run with five different random initializations; the result with the smallest RMSE for the reconstructed $\mathbf{W}$ was chosen.

The average errors in Table 3 show that the shape estimates of all methods seem to be similarly affected by noise up to $\hat{\sigma}_n = .125$. Also, a significant difference is seen for the camera estimates with $\hat{\sigma}_n > .05$. Affine methods overfit noise in the data due to the extra degrees of freedom in their camera model. CSF-B$_{af}$ and CSF-B$_{wp}$ attenuate this problem by enforcing smoothness on the camera trajectory. CSF-B$_{wp}$ outperforms all methods by further constraining the camera axes to be orthogonal and of equal length. This result is of special importance to SFM applications that rely more on the motion factor.

The average runtimes in seconds (per initialization, on a single-core 2.6 GHz processor) were: .2 (PF), 18.2 (LM-S), 14.9 (LM-S$_{GN}$), 128.5 (Wiberg), 1.1 (CSF-B$_{af}$), and 8.0 (CSF-B$_{wp}$). The simple PF is fast but often provides poor estimates, even with a maximum of 10,000 iterations (versus 500 for the other methods). CSF-B$_{af}$ and CSF-B$_{wp}$ are also very fast. Typically, CSF-B$_{wp}$ performs more iterations than CSF-B$_{af}$ – for better convergence on long sequences (large T), we run CSF-B$_{wp}$ in a coarse-to-fine manner, increasing $d \in \{.1T, .2T, .3T\}$. We also note that the Wiberg method does not scale well for application on large matrices due to the size of the system of equations it has to solve in each iteration.

Our experiments also considered the complete dinosaur dataset with 4,983 tracks (of which 2,300 are defined on only two images).[1] The dinosaur sequence is arguably the most popular dataset used to evaluate rigid SFM algorithms, with results published on different subsets of its 2D point tracks. For comparison against previously published results, we consider a subset of 2,683 tracks (points tracked in at least 3 images) and the subset of 319 tracks described above.

1. Available at *http://www.robots.ox.ac.uk/∼vgg/*.

Results of the methods above are also compared to those of the following algorithms: Damped-Newton (DN) [6], the Deviation Parameter (DP) for subspace constraints [19], Minimal Missing Elements (MME) [7], Camera Basis (subspace) Constraints (CBC) [28], and the Euclidean PowerFactorization (EPF) method in [21]. Finally, we also compare the result of CSF-B$_{wp}$ to that obtained with projective SFM followed by Euclidean bundle adjustment (ProjSFM-BA) [22]. Because the ground-truth 3D shape and cameras are not available, we compare RMSE values for the reconstructed $\mathbf{W}$ and also the mean/maximum 2D reprojection errors (in pixels) of the best solutions provided by each method. For this numerical comparison on the relatively short dinosaur sequence ($T = 36$), our CSF methods were both initialized (deterministically) with full DCT bases.

In Table 4, each incomplete row contains only the results reported in the original publication due to the algorithm not being available in our experiments. Because these algorithms use factorization models with different numbers of degrees of freedom, we define three main groups of methods. In this case, models with more degrees of freedom are expected to be associated with lower RMSE and 2D error values. However, we note that CSF-B$_{wp}$ yielded lower 2D reprojection errors compared to the affine SFM methods. In the dinosaur dataset, the performance of CSF-B$_{wp}$ is very close to that of the projective SFM method with bundle adjustment, despite the non-negligible perspective distortion in the data. Fig. 4(b)-(c) show the 3D shape recovered by CSF-B$_{wp}$.

Information on the ground-truth motion is actually available because we know the dinosaur is on a turntable. Thus, the reconstructed $\mathbf{W}$ must describe 2D point trajectories that are all concentric ellipses. Indeed, the result of CSF-B$_{wp}$ indicates correct motion recovery, Fig. 4(d). The best affine methods in our comparison (CSF-B$_{af}$, LM-S, LM-S$_{GN}$, and Wiberg) overfit noise and outliers and recover an incorrect motion pattern, Fig. 4(e). This problem is inherent in all affine SFM methods (incorrect 2D trajectories are also shown in [6], [8]). Also note this fact is independent of the Euclidean upgrade step, which does not affect reprojection.

The robust CBC method uses random sampling to tolerate outliers when computing subspace constraints and triangulating 3D points. Thus, CBC yields higher RMSEs as compared to the other methods. Table 4 shows that 2D reprojection error decreases as more point tracks are available for outlier/inlier identification – we computed camera constrains from triplets of frames presenting at least 15 tracks in common. In [28], CBC was found to provide the best solutions among the main subspace constraint methods. Here, error values are higher than those in [28] because we did not remove outliers before the comparison against non-robust methods. We note that our CSF methods do not explicitly address the problem of outliers and may fail if the input data present gross measurement errors. Future work will investigate the use of robust error terms [9], [12] in our cost functions.

TABLE 4
Results of rigid SFM methods on different subsets of the dinosaur dataset.

Method	(model)	319 point tracks (76.9% missing)		2,683 point tracks (87.8% missing)		4,983 point tracks (90.8% missing)	
		RMSE	mean/max 2D error	RMSE	mean/max 2D error	RMSE	mean/max 2D error
CSF-B$_{af}$	(1)	1.0847	.6286 / 38.8381	1.3370	.7129 / 39.9999	1.1346	.5133 / 39.9999
LM-S, LM-S$_{GN}$	(1)	1.0847	.6287 / 38.8376	1.3370	.7131 / 39.9999	1.1346	.5135 / 39.9997
DN	(1)	1.0847	–	–	–	–	–
DP	(1)	1.5482	1.0131 / 39.4277	2.0201	1.9103 / 40.2124	1.7142	1.6233 / 40.1713
MME	(1)	–	–	–	2.4017 / 72.4467	–	1.8438 / 72.4467
CSF-B$_{af}$	(2)	1.2702	.8810 / 41.4078	1.4294	.8610 / 42.5002	1.2176	.6808 / 42.4961
Wiberg	(2)	1.2702	.8810 / 41.4081	1.4294	.8609 / 42.5002	1.2176	.6808 / 42.4959
DN	(1*)	1.2702	–	–	–	–	–
CBC	(2)	1.8120	.9730 / 76.4071	2.9126	.8884 / 92.7365	2.4538	.7044 / 92.7399
PF	(2)	1.4139	1.0091 / 41.1094	2.0604	1.2238 / 64.1713	2.4531	1.5846 / 61.3726
CSF-B$_{wp}$	(3)	1.3031	.8757 / 43.8770	1.4833	.8122 / 44.6721	1.2641	.6472 / 44.7587
EPF	(3)	1.3705	– / 21	–	–	–	–
ProjSFM-BA	(3*)	–	–	–	–	–	.64 / 41.5

†Models: (1) general rank-4 factors (no mean column); (2) affine SFM; (3) Euclidean (weak-perspective) SFM; (3*) projective SFM with Euclidean bundle adjustment; (1*) penalty terms imposed on model (1) to favor orthogonality of camera axes. Error values are given in pixels.

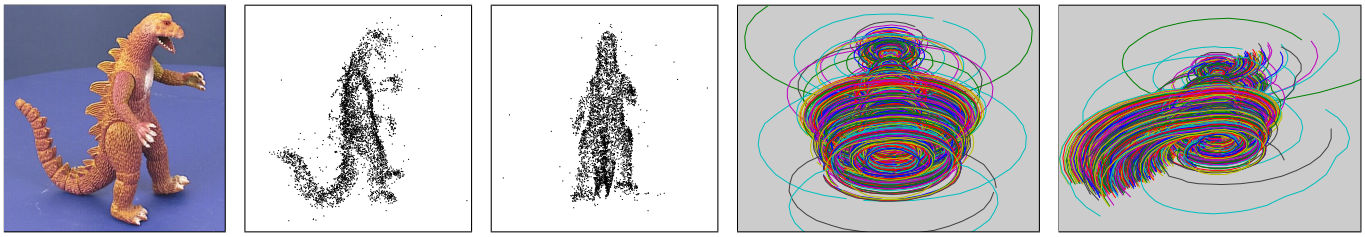


Fig. 4. Results on the 4,983 point set (90.8% missing) of the dinosaur dataset: (a) one of the 36 images of the sequence; (b)-(c) side and frontal views of the Euclidean 3D shape recovered by CSF-B$_{wp}$; and 2D point tracks reconstructed by CSF-B$_{wp}$ (d) and by the best affine methods (e).

TABLE 5
Results of rigid SFM methods on the teddy bear dataset.

Method	(model)	RMSE	mean/max 2D error
CSF-B$_{af}$	(1)	.4925	.4527 / 7.4998
LM-S, LM-S$_{GN}$	(1)	.4925	.4527 / 7.4998
DP	(1)	.5911	.4977 / 13.8324
CSF-B$_{af}$	(2)	.6174	.5689 / 10.7385
Wiberg	(2)	.6174	.5689 / 10.7385
CBC	(2)	.7487	.6478 / 14.1339
PF	(2)	1.2125	.8649 / 27.5116
CSF-B$_{wp}$	(3)	.6310	.5792 / 10.6524

†Models are as in Table 4. Error values are given in pixels.

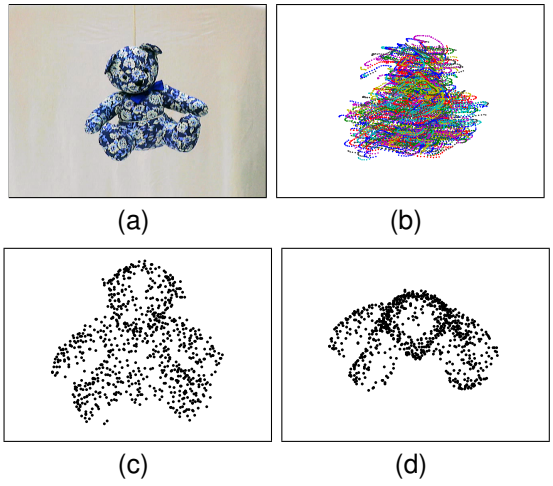
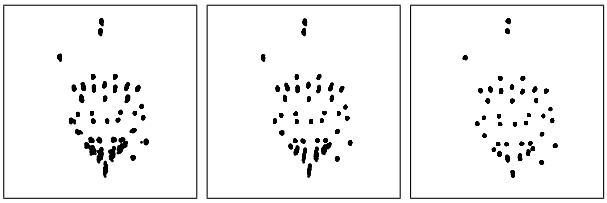


Fig. 5. Result of CSF-B$_{wp}$ on 806 point tracks (88.6% missing) of the teddy bear sequence: (a) image in the sequence; (b) input 2D tracks; (c)-(d) frontal and top views of the recovered Euclidean 3D shape.

In another experiment, we applied the Kanade-Lucas-Tomasi (KLT) feature tracker [27] on a 200-image *teddy bear* sequence [28]. We selected 806 point tracks appearing in at least 10 images. Due to occlusion, the resulting matrix $\mathbf{W} \in \mathbb{R}^{400\times 806}$ is missing 88.6% of its entries. For numerical comparison, both CSF methods considered a full DCT basis. CSF-B$_{wp}$ adopted a coarse-to-fine strategy with $d \in \{.1T, .2T, .3T, \ldots, T\}$ increased each time convergence to a coarser solution was detected.

Table 5 gives the 2D errors obtained with the algorithms described above on our teddy bear dataset. CSF-B$_{af}$ provides equivalent or better solutions as compared to the other methods. Also, CSF-B$_{wp}$ directly reconstructs

the Euclidean 3D shape with only a slight increase in 2D error – its camera model captures less of the noise in the data. Fig. 5 shows the 3D shape reconstructed with CSF-B$_{wp}$. On this long sequence ($T = 200$) with smooth motion, the difference in the results of CSF-B$_{wp}$ with a full DCT basis ($d = T$) and with $d = .3T$ is very small, $e_R = .0175$ and $e_S = .0229$ (here e_S is normalized by $\sigma(\mathbf{S})$ as in (21)). With $d = .3T$, the runtime of CSF-B$_{wp}$ drops from 27.3 to 3.6 minutes (4 to .7 minute for CSF-B$_{af}$). Wiberg took on average 4.2 hours.



(a) Original (b) CSF-B$_{nr}$ (c) PTA

Fig. 6. Overlay of 316 unrotated 3D shapes in the face2 sequence. Deformation is seen predominantly on the lower-lips and chin. CSF-B$_{nr}$ can capture this high-frequency deformation better than PTA.

5.3 Computing non-rigid SFM

First, we evaluate our non-rigid SFM algorithm on complete datasets with known 3D shapes for each frame, also simulating missing data and noise. Then, we present results on real datasets with and without occlusion. The number of frames (T) and the number of point tracks (n) are indicated as (T/n) after a dataset's name.

We start with the motion capture sequences: *drink* (1102/41), *pick-up* (357/41), *yoga* (307/41), *stretch* (370/41), and *dance* (264/75) used in [2]; *face1* (74/37) of [24]; *face2* (316/40) and *walking* (260/55) of [30]. We also use the synthetic bending *shark* (240/91) of [30]. Note that a different *shark* dataset appears in [2]. We use only the original one in [30].

To allow for comparison against the results reported in [2], we used the same procedure and error metrics therein. For each dataset, the complete 2D point trajectories in $\mathbf{W}$ are obtained by applying an orthographic projection on the sequence of 3D shapes. Because the solution of non-rigid SFM methods is defined up to an arbitrary 3×3 rotation, we compute a single rotation that best aligns all reconstructed and original 3D shapes. Let e_{tj} be the reconstruction error (*i.e.*, Euclidean distance) for the j^{th} 3D point of frame t. We then compute a normalized mean 3D error over all points and frames,

$$e_{3D} = \frac{1}{\sigma_e T n} \sum_{t=1}^{T} \sum_{j=1}^{n} e_{tj}, \quad \sigma_e = \frac{1}{T} \sum_{t=1}^{T} \sigma(\mathbf{S}_t),$$

with $\sigma(\cdot)$ as in (21) and $\mathbf{S}_t \in \mathbb{R}^{3\times n}$ the original 3D shape in frame t. The first four motion capture sequences have artificial rotations applied on them. We thus compare original and estimated rotations using e_R as in (22).

Table 6 compares the performance of our 3D shape trajectory approach, CSF-B$_{nr}$, against four state-of-the-art, non-rigid SFM methods: (i) the shape basis constraints (XCK) method[2] [32]; (ii) the algorithm modeling 3D shape using probabilistic principal component analysis (EM-PPCA) [30]; (iii) the Metric Projections (MP) method[3] [24]; and (iv) the DCT-based 3D point trajectory approach (PTA) [2].

Following the methodology in [2], we ran the algorithms with different values of $K \in \{2, 3, \ldots, 13\}$, reporting the best result. Table 6 also shows the value of K for the best solutions obtained with PTA and CSF-B$_{nr}$ for comparison. We also report the initial error of CSF-B$_{nr}$ because it shows the error that PTA would provide with the same K and rotations in $\mathbf{D}$. In all runs, CSF-B$_{nr}$ had the number of DCT basis set to $d = .1T$, except for the two face datasets on which we set $d = \frac{T}{3}$ due to the presence of higher frequency deformations.

Our results, as shown in Table 6, are consistently similar or better than the best results provided by the other methods on each dataset. As compared to PTA, CSF-B$_{nr}$ computes better solutions and at a lower rank $r = 3K + 1$ by more efficiently using higher DCT frequency components. Furthermore, our simple strategy of iterating over K while computing factor $\mathbf{D}$ also provides better Euclidean camera estimates than plain PTA.

EM-PPCA, MP, and CSF-B$_{nr}$ provide comparable results on the face2 dataset, which has mostly rigid motion with high-frequency deformation seen on the lower-lips and chin. In this case, CSF-B$_{nr}$ requires at least 75% of all DCT components ($d = .75T$) to provide a mean 3D error (.0328) that is smaller than that of EM-PPCA. PTA is not capable of modeling this high-frequency deformation and recovers a mostly rigid mouth, Fig. 6. On the much shorter face1 sequence, CSF-B$_{nr}$ obtains an error improvement of only .0012 with a larger d. Setting $d > .1T$ resulted in no significant improvement to the CSF-B$_{nr}$ solutions on the other datasets.

CSF-B$_{nr}$ is the only method that can accurately reconstruct the deformation of the bending shark. Furthermore, with a full DCT basis, the resulting e_{3D} is negligible (.00004) and perfect reconstruction is achieved. Fig. 7 shows the three best shark reconstructions of Table 6. XCK fails on this dataset due to the bending shark presenting a 2D (degenerate) deformation mode [30].

The motion capture sequences containing highly articulated bodies also show the superiority of CSF-B$_{nr}$

2. We did not implement the XCK algorithm. Its results on the first five sequences are reproduced from [2]. Results on the shark, face, and walking datasets are given in [30], with a different error metric, and found to be significantly inferior to those of EM-PPCA.

3. Two MP methods are presented in [24]. Here we experiment with the method using a generic model of deformable shapes. The supplementary file shows comparative results against MP with a specialized articulated model that requires the columns of $\mathbf{W}$ to be initially grouped as corresponding to separate object parts.

TABLE 6
Performance of non-rigid SFM methods on synthetic and motion capture data. For the related PTA and CSF-B$_{nr}$ methods, factorization rank is also indicated by the value of K in parenthesis.

Dataset	XCK		EM-PPCA		MP		PTA		CSF-B$_{nr}$		
	e_R	e_{3D}	e_R	e_{3D}	e_R	e_{3D}	e_R	e_{3D} (K)	e_R	e_{3D} (K)	initial e_{3D} (K)
Drink	.3359	3.5186	.2906	.3393	.2859	.4604	**.0058**	**.0250** (13)	**.0055**	**.0223** (6)	.0854 (6)
Pick-up	.4687	3.3721	.4277	.5822	.2506	.4332	**.1549**	**.2369** (12)	**.1546**	**.2301** (6)	.2685 (6)
Yoga	1.2014	7.4935	.8089	.8097	.8711	.8039	**.1059**	.1625 (11)	**.1021**	**.1467** (7)	.1528 (7)
Stretch	.9489	4.2415	.7594	1.1111	.8174	.8549	.0549	.1088 (12)	**.0489**	**.0710** (8)	.0966 (8)
Dance	–	2.9962	–	.9839	–	**.2639**	–	.2958 (5)	–	**.2705** (2)	.3259 (2)
Face1	–	–	–	**.0434**	–	.0734	–	.1247 (3)	–	.0637 (5)	.1487 (3)
Face2	–	–	–	**.0329**	–	**.0357**	–	.0444 (5)	–	**.0363** (3)	.0451 (3)
Shark	–	–	–	.0501	–	.1571	–	.1796 (9)	–	**.0081** (3)	.3195 (3)
Walking	–	–	–	.4917	–	.5607	–	.3954 (2)	–	**.1863** (2)	.6823 (2)

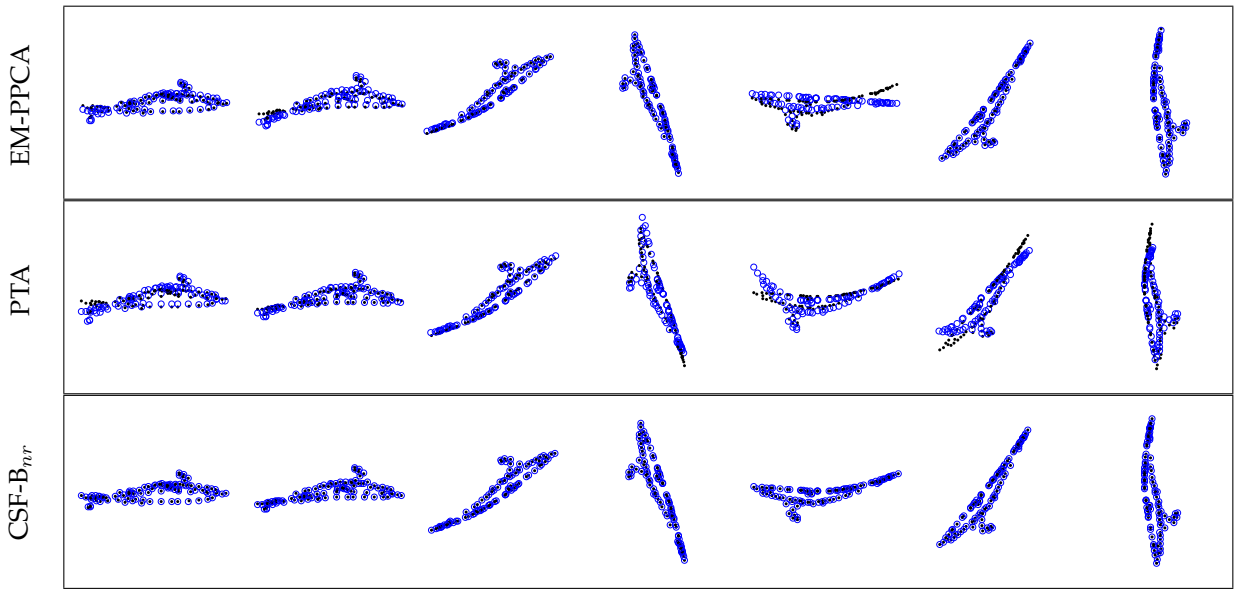


Fig. 7. Results on the bending shark sequence. Reconstructed 3D shapes (blue circles) are shown against the original 3D data (dark dots). Frames 20, 50, 80, 115, 148, 175, and 200 are displayed above.

and PTA compared to XCK, EM-PPCA, and MP. On the smoothly deforming shapes in the drink, pick-up, and yoga sequences (used in [2]), the improvement offered by CSF-B$_{nr}$ over PTA is marginal. On the other hand, the results on the more difficult walking sequence of [30] (Fig. 8) highlight the advantages of using our 3D shape trajectory approach (no artificial rotation was added to this sequence). MP provides comparable results with CSF-B$_{nr}$ on the dance sequence. The supplementary file includes additional examples.

To simulate missing data in the shark and walking datasets, we randomly discard $\rho\%$ of the 2D entries in $\mathbf{W}$. Before applying CSF-B$_{nr}$ (with K as in Table 6), we compute $\mathbf{D}$ and $\mathbf{t}$ by first using CSF-B$_{af}$ to reconstruct the complete 2D point trajectories in $\mathbf{W}$. CSF-B$_{af}$ was run with $d = .25T$ and rank $r = 7$. Let $\mathbf{W}_0$ be the complete matrix, we normalize the 2D reconstruction error for the incomplete $\mathbf{W}$ by $\sigma(\mathbf{W}_0)$ as in (21).

On the smooth shark deformation, results are visually similar to those in Fig. 7 with ρ up to 95% (see images in the supplementary file). Of 10 runs with $\rho = 95\%$, the average (maximum) 2D reconstruction error for $\mathbf{W}$ was .0015 (.0029). The average (maximum) 3D error after running CSF-B$_{nr}$ was .0163 (.0488). On the walking sequence, results with $\rho = 75\%$ are still visually similar to those in Fig. 8. After 10 runs, the average (maximum) 2D and 3D errors were .0508 (.0548) and .2063 (.3910), respectively. These results on incomplete data are still better than those of XCK, EM-PPCA, MP, and PTA on the complete shark and walking datasets.

On the complete face2 dataset, the performances of EM-PPCA, MP, PTA, and CSF-B$_{nr}$ are more similar than on the other sequences. We thus analyzed their average e_{3D} with different levels of random occlusion and noise (simulated independently) on $\mathbf{W}$ of face2. EM-PPCA and MP handle cases of missing data by adopting an alternation approach as in PowerFactorization. PTA does not handle occlusions and was tested with added noise

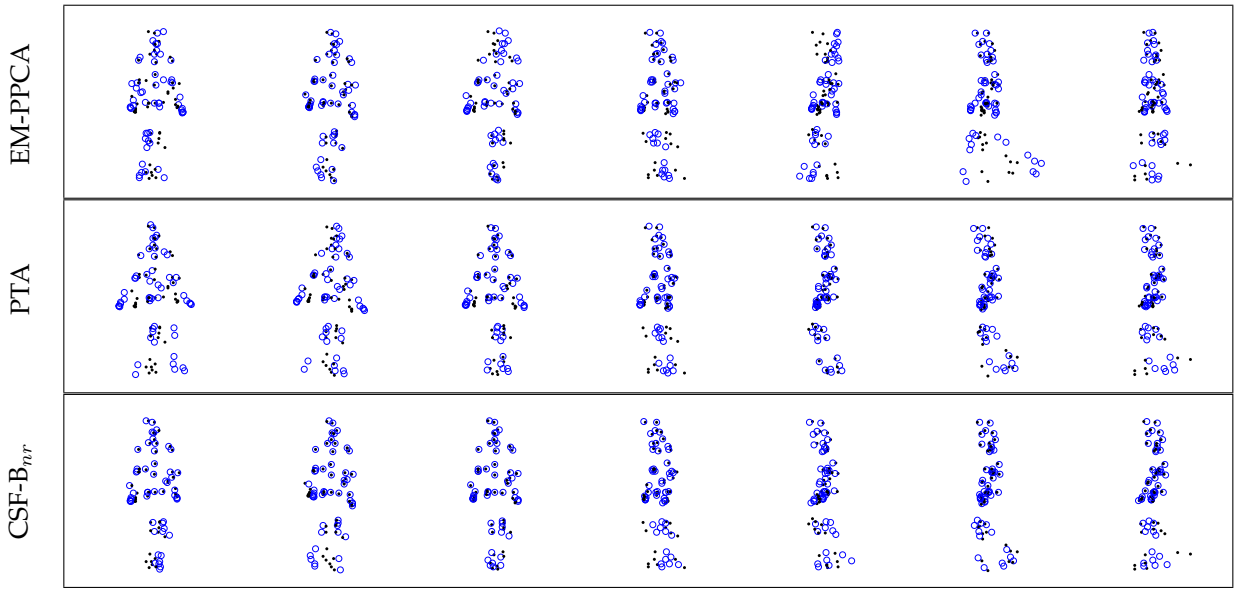


Fig. 8. Results on the walking sequence. Reconstructed 3D shapes (blue circles) are shown against the original 3D data (dark dots). Frames 34, 74, 122, 160, 198, 223, and 255 are displayed above.

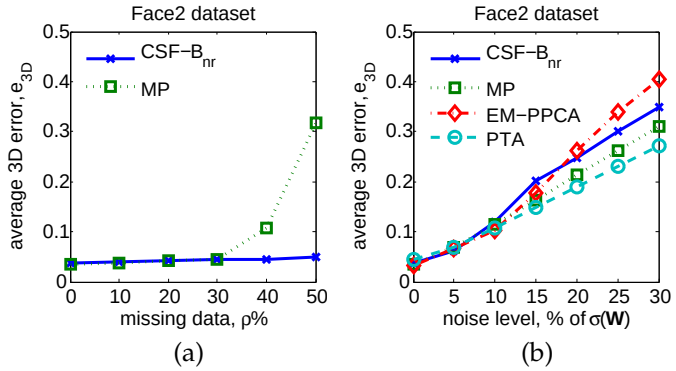


Fig. 9. Reconstruction errors on the face2 sequence with missing data and noise simulated independently (averages over 100 trials).

only. All methods were run with their best parameter K for the complete data ($K = 5$ for EM-PPCA, MP, PTA; $K = 3$ for CSF-B$_{nr}$). After random occlusion, we ensured that $\mathbf{W}$ had at least $3K_{max}+1$ entries ($K_{max} = 5$) in each row and column. On the high-frequency deformation and motion of face2 (combined in the columns of $\mathbf{W}$), the initial CSF-B$_{af}$ step of CSF-B$_{nr}$ reconstructed the 2D point trajectories in $\mathbf{W}$ with $d = .5T$ and $r = 5$. Results were averaged over 100 trials and are shown in Fig. 9.

In our trials on face2 with missing data, EM-PPCA was unstable and presented both small and very large errors for random occlusions ρ as low as 10%. Thus, Fig. 9(a) shows the results of CSF-B$_{nr}$ in comparison to those of MP, which was shown in [24] to outperform EM-PPCA in a similar experiment on face2. Note that the average e_{3D} of MP begins to increase with ρ above 30%, while that of CSF-B$_{nr}$ presents almost no variation over all the tested levels of random occlusion.

For levels of added Gaussian noise below $.15\sigma(\mathbf{W})$ (15%), the increase in e_{3D} is similar for all four methods, Fig. 9(b). At 15% and above, small differences in performance are observed but the reconstructions still degrade only gradually with the level of noise. Because PTA largely oversmoothes the deformation of face2, its performance suffers a smaller penalty as compared to that of the other methods. At high noise levels, correct 3D shape reconstruction seems to require additional information on the nature of the 3D shapes and noise [12].

The average runtimes of the algorithms on the face2 trials at 5% noise level were, in minutes: 2.52 (EM-PPCA), .34 (MP), .05 (PTA), and .40 (CSF-B$_{nr}$, of which .15 was spent to compute $\mathbf{D}$). These times were similar to those observed for the algorithms on the original data. PTA is fast due to its use of SVD. Runtimes in minutes on face2 trials with 30% missing data were: 2.85 (EM-PPCA), 3.66 (MP), and 2.29 (CSF-B$_{nr}$). This runtime of CSF-B$_{nr}$ includes time spent running CSF-B$_{af}$ (.82) and computing $\mathbf{D}$ (.19). At above 20% random occlusion, CSF-B$_{nr}$ was faster than MP and EM-PPCA.

We also applied CSF-B$_{nr}$ to the (complete) real dataset *cubes* (200/14) of [2]. With $K = 2$ and $d = .1T$, the solution of CSF-B$_{nr}$ has mean (maximum) 2D reprojection error of .4958 (2.0672) pixel. The solution of PTA has an error of 1.6589 (4.8602) also with $K = 2$. Images and additional results are given in the supplementary file.

Finally, an application of our non-rigid SFM method is in the interpretation of the facial expression component of sign languages from video [10]. In this case, head rotation and hand gesticulation often cause the occlusion of facial features, leading to incomplete 2D point tracks. We now consider a 115-image (4 seconds long) face close-up sequence of an American Sign Language (ASL) sentence. Facial landmarks were manually annotated in each image when visible. The resulting matrix $\mathbf{W} \in \mathbb{R}^{230\times77}$ is missing 17.4% of its data and has small magnitude noise due to annotation errors caused by partial occlusion of facial features and motion blur in the video images.

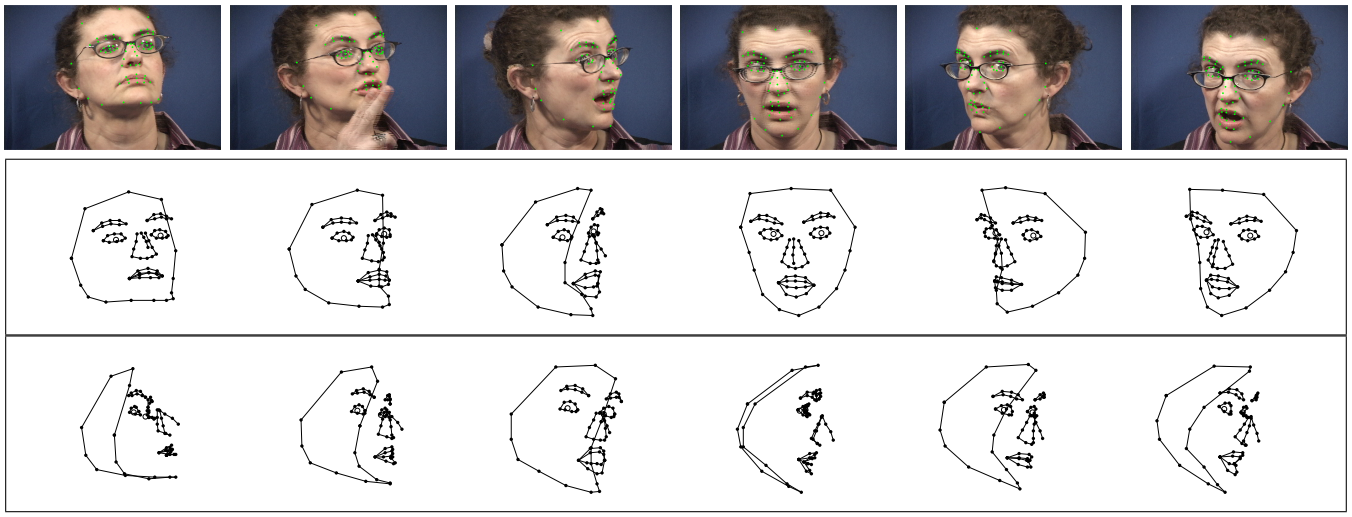


Fig. 10. Results of CSF-B$_{nr}$ on the ASL sequence (77 points, 17.4% missing data): (*top*) six out of 115 images with annotated facial landmarks in green; (*middle,bottom*) two orthogonal views of the recovered 3D shapes.

Fig. 10 shows six example images and their respective 3D face shapes recovered using CSF-B$_{nr}$ ($K = 2$ and $d = .5T$, with the initial CSF-B$_{af}$ step run as for face2 above). Figures with the results of EM-PPCA and MP are given in the supplementary file. For all tested values of $K \in \{2, 3, \ldots, 13\}$, the result obtained with EM-PPCA presented visibly large 3D reconstruction errors for a number of facial points that are occluded frequently in each sequence (*e.g.*, the lateral contours of the nose and the face). The results of CSF-B$_{nr}$ and MP are visually similar and indicate correct recovery of pose and non-rigid 3D shape despite the occurrence of occlusion.

6 Conclusion

In this paper, we have addressed the classical computer vision problems of rigid and non-rigid SFM with occlusion. We started by assuming that the columns of the input data matrix $\mathbf{W}$ describe smooth 2D point trajectories over time. This assumption is equivalent to considering that 2D observations in $\mathbf{W}$ are obtained by a single camera moving smoothly around a rigid structure. In non-rigid SFM, our assumption also requires the structure to deform only smoothly over time.

Our main contributions are two-fold: (i) we provide new models for the smooth time-trajectories of camera and deformable shape with a compact parameterization in the DCT domain; and (ii) we derive a family of efficient Column Space Fitting (CSF) methods to estimate such trajectories while tolerating cases in which $\mathbf{W}$ is presented with missing data.

In rigid SFM, we consider a weak-perspective camera model from the outset and directly reconstruct Euclidean 3D shape without requiring post-processing steps. Our results on synthetic and real SFM datasets with noise and high percentages of missing data were positively compared to the state of the art.

In non-rigid SFM, we propose a novel 3D shape trajectory approach that solves for the deformable structure as the trajectory of a single point in an implicitly defined linear shape space. A comparison against state-of-the-art algorithms show that our method can better model complex articulated deformation with higher frequency DCT components while still maintaining the low-rank factorization constraint. We also demonstrate that our non-rigid SFM algorithm can tolerate high percentages on missing data in the input matrix $\mathbf{W}$ with only a small penalty in 3D reconstruction accuracy.

Future work will investigate the integration of our weak-perspective camera model into our non-rigid SFM approach. We will also consider the automatic selection of the number of elements in the DCT and shape bases using regularization terms that balance the tradeoff between higher model complexity (r, K, and d) and smaller fitting error [23], [25], [30]. Our CSF algorithms may also benefit from a robust error term in order to better tolerate the presence of outliers in $\mathbf{W}$. Finally, non-linear models for the column space of $\mathbf{W}$ shall be considered.

Appendix A
Derivation of the Jacobian Terms in CSF

Consider $vec(\mathbf{u}^T\mathbf{v}) = vec(\mathbf{u})^T vec(\mathbf{v})$, and $vec(\mathbf{BXA}) = (\mathbf{A}^T \otimes \mathbf{B})vec(\mathbf{X})$, for any column vectors $\mathbf{u}$ and $\mathbf{v}$, and any matrices $\mathbf{B}$, $\mathbf{X}$, and $\mathbf{A}$. The operator $\otimes$ is the Kronecker product. Also, let the matrix $\mathbf{K}_{mr} \in \mathbb{R}^{mr \times mr}$ be a permutation satisfying $vec(\mathbf{M}^T) = \mathbf{K}_{mr} vec(\mathbf{M})$.

By following a Gauss-Newton approach, the first and second differentials of f in (4) are

$$df = \frac{1}{2}\sum_j \left(d\mathbf{r}_j^T \mathbf{r}_j + \mathbf{r}_j^T d\mathbf{r}_j\right) = \sum_j \mathbf{r}_j^T d\mathbf{r}_j,$$
$$d^2 f \approx \sum_j d\mathbf{r}_j^T d\mathbf{r}_j,$$

with second order terms $d^2\mathbf{r}_j$ neglected in $d^2 f$ and

$$d\mathbf{r}_j = -d\left(\mathbf{M}_j \mathbf{M}_j^{\dagger}\right)\mathbf{w}_j. \tag{23}$$

From [20], we have

$$d\left(\mathbf{M}_j\mathbf{M}_j^\dagger\right) = \mathbf{P}_j^\perp d\mathbf{M}_j\mathbf{M}_j^\dagger + \left(\mathbf{P}_j^\perp d\mathbf{M}_j\mathbf{M}_j^\dagger\right)^T. \quad (24)$$

with $\mathbf{P}_j^\perp$ as in (3). Considering $\mathbf{M}_j = \mathbf{\Pi}_j\mathbf{M}$ and expressing (23) as $d\mathbf{r}_j = -\mathbf{J}_j vec(d\mathbf{M})$, the Jacobian terms are identified as

$$\mathbf{J}_j = \mathbf{s}_j^T \otimes \mathbf{P}_j^\perp\mathbf{\Pi}_j + \left(\mathbf{\Pi}_j^T\mathbf{r}_j \otimes \mathbf{M}_j^\dagger\right)^T \mathbf{K}_{mr}. \quad (25)$$

We now depart from the Gauss-Newton solution (25). First, notice that the right-most term in (24) vanishes when multiplied by $\mathbf{r}_j^T = (\mathbf{P}_j^\perp\mathbf{w}_j)^T$ on the left,

$$\mathbf{r}_j^T\left(\mathbf{P}_j^\perp d\mathbf{M}_j\mathbf{M}_j^\dagger\right)^T = \left(\mathbf{P}_j^\perp d\mathbf{M}_j(\mathbf{M}_j^\dagger\mathbf{P}_j^\perp)\mathbf{w}_j\right)^T = \mathbf{0},$$

because the property $\mathbf{M}_j^\dagger = \mathbf{M}_j^\dagger\mathbf{M}_j\mathbf{M}_j^\dagger$ implies

$$\mathbf{M}_j^\dagger\mathbf{P}_j^\perp = \mathbf{M}_j^\dagger\left(\mathbf{I} - \mathbf{M}_j\mathbf{M}_j^\dagger\right) = \mathbf{0}.$$

Thus, a Jacobian term (7) in CSF neglects the right-most term in (24) and (25). While there is no difference in the gradient vector, it can be shown that the Hessian matrix in (8) neglects terms of the form $\mathbf{\Pi}_j^T\mathbf{r}_j\mathbf{r}_j^T\mathbf{\Pi}_j \otimes \mathbf{M}_j^\dagger\mathbf{M}_j^{\dagger T}$.

Our Hessian approximation, combined with the damping parameter δ in Algorithm 1, is efficient in providing adequate solution updates, $vec(d\mathbf{M})$, despite the simpler form. CSF's performance is positively compared to that of LM-S in Section 5. We also consider a Gauss-Newton derivation of LM-S (see the supplementary file).

For the rank-r factorization with a mean column $\mathbf{t}$, we define $\mathbf{s}_j = \mathbf{M}_j^\dagger(\mathbf{w}_j - \mathbf{t}_j) \in \mathbb{R}^{r-1}$. Then, the differential of the residual terms in (10) is

$$\begin{aligned} d\mathbf{r}_j &= d\mathbf{P}_j^\perp(\mathbf{w}_j - \mathbf{t}_j) - \mathbf{P}_j^\perp d\mathbf{t}_j \\ &\approx -\mathbf{P}_j^\perp\mathbf{\Pi}_j d\mathbf{M}\mathbf{s}_j - \mathbf{P}_j^\perp\mathbf{\Pi}_j d\mathbf{t} \\ &\approx -\mathbf{P}_j^\perp\mathbf{\Pi}_j\left[d\mathbf{M}\ d\mathbf{t}\right]\begin{bmatrix}\mathbf{s}_j\\1\end{bmatrix} \\ &\approx -\left(\widetilde{\mathbf{s}}_j \otimes \mathbf{P}_j^\perp\mathbf{\Pi}_j\right) vec(d\widetilde{\mathbf{M}}), \end{aligned}$$

revealing the form of the Jacobian in (11). It is similar to (7) but has the extended term $\widetilde{\mathbf{s}}_j$ instead of $\mathbf{s}_j$.

Acknowledgments

We thank the reviewers for their constructive comments. This research was supported in part by the National Science Foundation, grant 0713055, and the National Institutes of Health, grant R01 EY 020834.

REFERENCES

[1] I. Akhter, Y. Sheikh, and S. Khan, "In defense of orthonormality constraints for nonrigid structure from motion," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009, pp. 1534–1541.

[2] I. Akhter, Y. A. Sheikh, S. Khan, and T. Kanade, "Nonrigid structure from motion in trajectory space," in *Neural Information Processing Systems*, December 2008.

[3] A. Bartoli, V. Gay-Bellile, U. Castellani, J. Peyras, S. Olsen, and P. Sayd, "Coarse-to-fine low-rank structure-from-motion," in *IEEE Conference on Computer Vision and Pattern Recognition*, no. 1, 2008, pp. 1–8.

[4] D. Bertsekas, *Nonlinear Programming*. Athena Scientific Press, 1999.

[5] C. Bregler, A. Hertzmann, and H. Biermann, "Recovering non-rigid 3d shape from image streams," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 690–696.

[6] A. M. Buchanan and A. W. Fitzgibbon, "Damped newton algorithms for matrix factorization with missing data," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 316–322.

[7] P. Chen and D. Suter, "Recovering the missing components in a large noisy low-rank matrix: Application to sfm," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1051–1063, 2004.

[8] P. Chen, "Optimization algorithms on subspaces: Revisiting missing data problem in low-rank matrix," *International Journal on Computer Vision*, vol. 80, no. 1, pp. 125–142, 2008.

[9] F. de la Torre and M. Black, "Robust principal component analysis for computer vision," in *Proc. Int. Conf. Computer Vision*, vol. 1, 2001, pp. 362–369.

[10] L. Ding and A. Martinez, "Modelling and recognition of the linguistic components in american sign language," *Image and Vision Computing*, vol. 27, no. 12, pp. 1826–1844, 2009.

[11] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[12] J. Fortuna and A. M. Martinez, "Rigid structure from motion from a blind source separation perspective," *International Journal of Computer Vision*, 2010, (in press).

[13] G. H. Golub and C. F. Van Loan, *Matrix Computations*, ser. Johns Hopkins Studies in Mathematical Sciences. The Johns Hopkins University Press, 1996.

[14] N. Guilbert, A. Bartoli, and A. Heyden, "Affine approximation for direct batch recovery of euclidian structure and motion from sparse data," *International Journal of Computer Vision*, vol. 69, no. 3, pp. 317–333, 2006.

[15] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[16] R. Hartley and F. Schaffalizky, "Powerfactorization: 3d reconstruction with missing or uncertain data," in *Proc. Australia-Japan Advanced Workshop on Computer Vision*, 2003.

[17] D. Jacobs, "Linear fitting with missing data for structure-from-motion," *Computer Vision and Image Understanding*, vol. 82, no. 1, pp. 57–81, 2001.

[18] A. Jain, "A sinusoidal family of unitary transforms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 1, no. 4, pp. 356–365, 1979.

[19] H. Jia and A. M. Martinez, "Low-rank matrix fitting based on subspace perturbation analysis with applications to structure from motion," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 841–854, 2009.

[20] J. Magnus and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics*, 2nd ed. Wiley, 1999.

[21] M. Marques and J. Costeira, "Estimating 3d shape from degenerate sequences with missing data," *Computer Vision and Image Understanding*, vol. 113, no. 2, pp. 261–272, 2009.

[22] D. Martinec and T. Pajdla, "Structure from many perspective images with occlusions," in *Proc. European Conf. Computer Vision*, 2002, pp. 355–369.

[23] S. I. Olsen and A. Bartoli, "Implicit non-rigid structure-from-motion with priors," *J. Math. Imaging Vis.*, vol. 31, no. 2-3, pp. 233–244, 2008.

[24] M. Paladini, A. Del Bue, M. Stošić, M. Dodig, J. Xavier, and L. Agapito, "Factorization for non-rigid and articulated structure using metric projections," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009, pp. 2898–2905.

[25] M. Pollefeys, F. Verbiest, and L. Van Gool, "Surviving dominant planes in uncalibrated structure and motion recovery," in *In Proc. European Conference on Computer Vision*, vol. 2, 2002, pp. 837–851.

[26] V. Rabaud and S. Belongie, "Rethinking nonrigid structure from motion," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, 2008, pp. 1–8.

[27] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, no. 1, 1994, pp. 593–600.

[28] J.-P. Tardif, A. Bartoli, M. Trudeau, N. Guilbert, and S. Roy, "Algorithms for batch matrix factorization with application to structure-from-motion," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, no. 1, 2007, pp. 1–8.

[29] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, November 1992.
[30] L. Torresani, A. Hertzmann, and C. Bregler, "Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 878–892, 2008.
[31] T. Wiberg, "Computation of principal components when data is missing," in *Proc. Second Symp. Computational Statistics*, 1976, pp. 229–236.
[32] J. Xiao, J. Chai, and T. Kanade, "A closed form solution to non-rigid shape and motion recovery," *International Journal on Computer Vision*, vol. 67, no. 2, pp. 233–246, 2006.
[33] J. Xiao and T. Kanade, "Non-rigid shape and motion recovery: Degenerate deformations," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. 668–675.
[34] J. Yan and M. Pollefeys, "A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 865–877, 2008.

Supplementary Documentation

Paulo F. U. Gotardo, Aleix M. Martinez

1 The LM-S Method

Two methods are proposed in [8] and it is concluded that the LM-S variant is the recommended one. It minimizes the cost function (2) by computing updates $vec(d\mathbf{M})$ with a gradient vector $\mathbf{g}$ and a Hessian matrix $\mathbf{H}$ as given next. Consider our notation in the main manuscript and, for $\mathbf{M}_j \in \mathbb{R}^{m_j \times r}$, let matrix $\mathbf{K}_{m_j r} \in \mathbb{R}^{m_j r \times m_j r}$ be a permutation satisfying $vec(\mathbf{M}_j^T) = \mathbf{K}_{m_j r} vec(\mathbf{M}_j)$. Then

$$\mathbf{g} = -\frac{1}{2}\sum_j \left(\mathbf{I}_r \otimes \mathbf{\Pi}_j^T\right)\mathbf{g}_j, \qquad \mathbf{g}_j = \left(\mathbf{s}_j \otimes \mathbf{r}_j\right) + \mathbf{K}_{m_j r}^T\left(\mathbf{r}_j \otimes \mathbf{s}_j\right),$$

and

$$\mathbf{H} = -\frac{1}{2}\sum_j \left(\mathbf{I}_r \otimes \mathbf{\Pi}_j^T\right)\left(\mathbf{H}_j + \mathbf{H}_j^T\right)\left(\mathbf{I}_r \otimes \mathbf{\Pi}_j\right),$$

$$\mathbf{H}_j = \left(\mathbf{I}_r \otimes \mathbf{r}_j\right)\left[\mathbf{r}_j^T \otimes (\mathbf{M}_j^T\mathbf{M}_j)^{-1}\right]\mathbf{K}_{m_j r} - 2\left(\mathbf{I}_r \otimes \mathbf{r}_j\right)\left(\mathbf{s}_j^T \otimes \mathbf{M}_j^{\dagger}\right) - \mathbf{K}_{m_j r}^T\left(\mathbf{I}_{m_j} \otimes \mathbf{s}_j\right)\left(\mathbf{s}_j^T \otimes \mathbf{P}_j^{\perp}\right).$$

The algorithm's source code available on the author's website (*http://sist.sysu.edu.cn/∼chenpei/*) also includes an unpublished Gauss-Newton implementation. It computes $\mathbf{g}$ and $\mathbf{H}$ as in (8) but with Jacobian terms

$$\mathbf{J}_j = \left((\mathbf{s}_j^T \otimes \mathbf{P}_j^{\perp}) + (\mathbf{r}_j \otimes \mathbf{M}_j^{\dagger})^T\mathbf{K}_{m_j r}\right)\left(\mathbf{I}_r \otimes \mathbf{\Pi}_j\right).$$

Our experimental results show that the derivation in CSF, (7) and (8), is not only simpler but also performs better than the two derivations above when applied on low-rank matrices with a high percentage of missing data.

2 Modeling the Time-Trajectory of an Euclidean Camera

In this section, we model the smooth time-trajectory of a weak-perspective camera using cosine series of different lengths d for the camera parameters $\alpha_t, \beta_t, \gamma_t, \lambda_t$, and $\widehat{\mathbf{t}}_\mathbf{t}$. The resulting model and algorithm allow for the enforcement of different degrees of smoothness on the camera parameters and may be used to reduce the total number of unknowns that need to be estimated. These capabilities are specially useful in applications for which *a priori* information on the camera motion is available.

Consider integer numbers $d_1, d_2, \ldots, d_5 \in \{1, 2, \ldots, T\}$ and define

$$\begin{bmatrix}\alpha_1\\ \vdots \\ \alpha_T\end{bmatrix} = \mathbf{\Omega}_{d_1}\mathbf{x}_1, \quad \begin{bmatrix}\beta_1\\ \vdots \\ \beta_T\end{bmatrix} = \mathbf{\Omega}_{d_2}\mathbf{x}_2, \quad \begin{bmatrix}\gamma_1\\ \vdots \\ \gamma_T\end{bmatrix} = \mathbf{\Omega}_{d_3}\mathbf{x}_3, \quad \begin{bmatrix}\lambda_1\\ \vdots \\ \lambda_T\end{bmatrix} = \mathbf{\Omega}_{d_4}\mathbf{x}_4, \quad \begin{bmatrix}\widehat{\mathbf{t}}_\mathbf{1}\\ \vdots \\ \widehat{\mathbf{t}}_\mathbf{T}\end{bmatrix} = \underbrace{\left(\mathbf{\Omega}_{d_5} \otimes \mathbf{I}_2\right)}_{\mathbf{B}_{af}}\mathbf{x}_5,$$

with $\mathbf{x}_1 \in \mathbb{R}^{d_1}, \mathbf{x}_2 \in \mathbb{R}^{d_2}, \mathbf{x}_3 \in \mathbb{R}^{d_3}, \mathbf{x}_4 \in \mathbb{R}^{d_4}$, and $\mathbf{x}_5, \in \mathbb{R}^{2d_5}$. We now iteratively solve for

$$vec(d\mathbf{X}) \equiv \left[d\mathbf{x}_1^T, d\mathbf{x}_2^T, d\mathbf{x}_3^T, d\mathbf{x}_4^T, d\mathbf{x}_5^T\right]^T \in \mathbb{R}^D, \quad D = d_1 + d_2 + d_3 + d_4 + 2d_5.$$

As before, we compute the differential of each rotation matrix, $d\widehat{\mathbf{R}}_t$, and stack the terms associated with $d\mathbf{x}_1, \ldots, d\mathbf{x}_4$ into, respectively, $\mathbf{B}_\alpha \in \mathbb{R}^{6T \times d_1}, \mathbf{B}_\beta \in \mathbb{R}^{6T \times d_2}, \mathbf{B}_\gamma \in \mathbb{R}^{6T \times d_3}$, and $\mathbf{B}_\lambda \in \mathbb{R}^{6T \times d_4}$. These matrices and $\mathbf{B}_{af} \in \mathbb{R}^{2T \times 2d_5}$ are blocks of the local basis matrix

$$\widetilde{\mathbf{B}}_{wp} = \begin{bmatrix}\mathbf{B}_\alpha & \mathbf{B}_\beta & \mathbf{B}_\gamma & \mathbf{B}_\lambda & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{af}\end{bmatrix} \in \mathbb{R}^{8T \times D}$$

Finally, we use our CSF algorithm to iteratively solve for the update step

$$vec(d\mathbf{M}) = \widetilde{\mathbf{B}}_{wp} vec(d\mathbf{X}).$$

The Jacobian terms of this camera model remain the same and are given in the main manuscript.

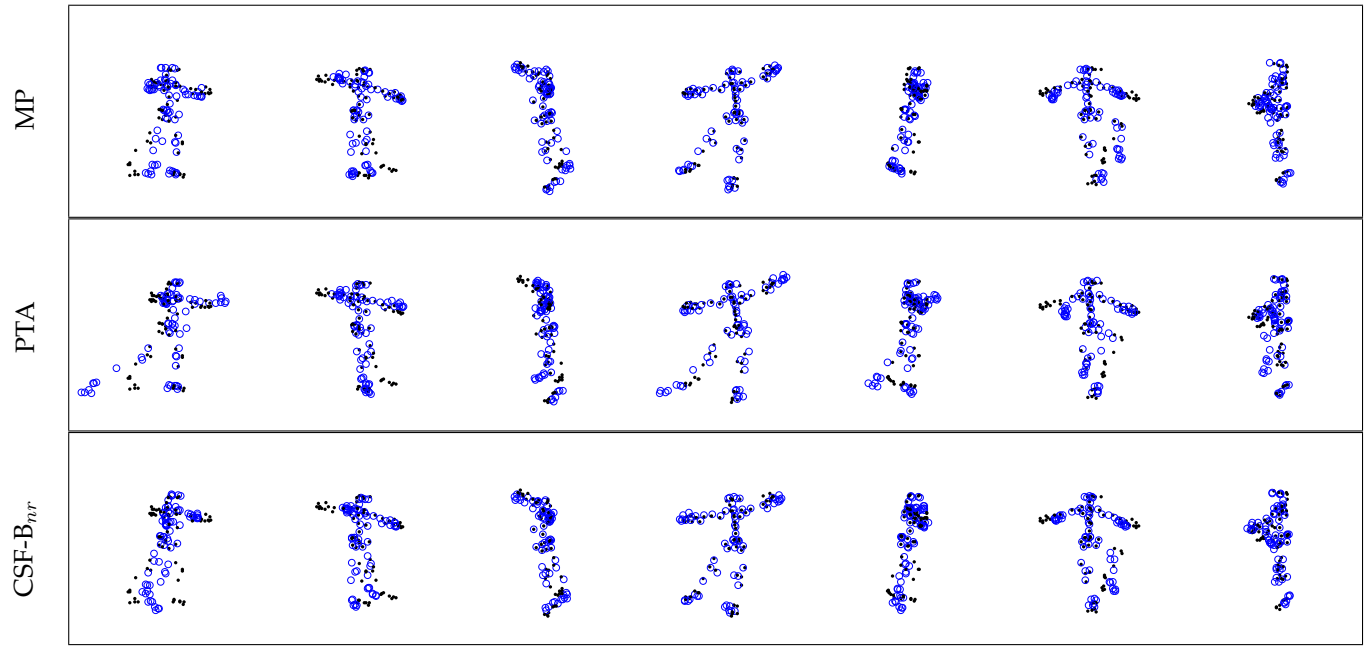


Fig. S1. Results on the *dance* sequence. Reconstructed 3D shapes (blue circles) are shown against the original 3D data (dark dots). Frames 7, 44, 81, 118, 155, 192, and 229 are displayed above.

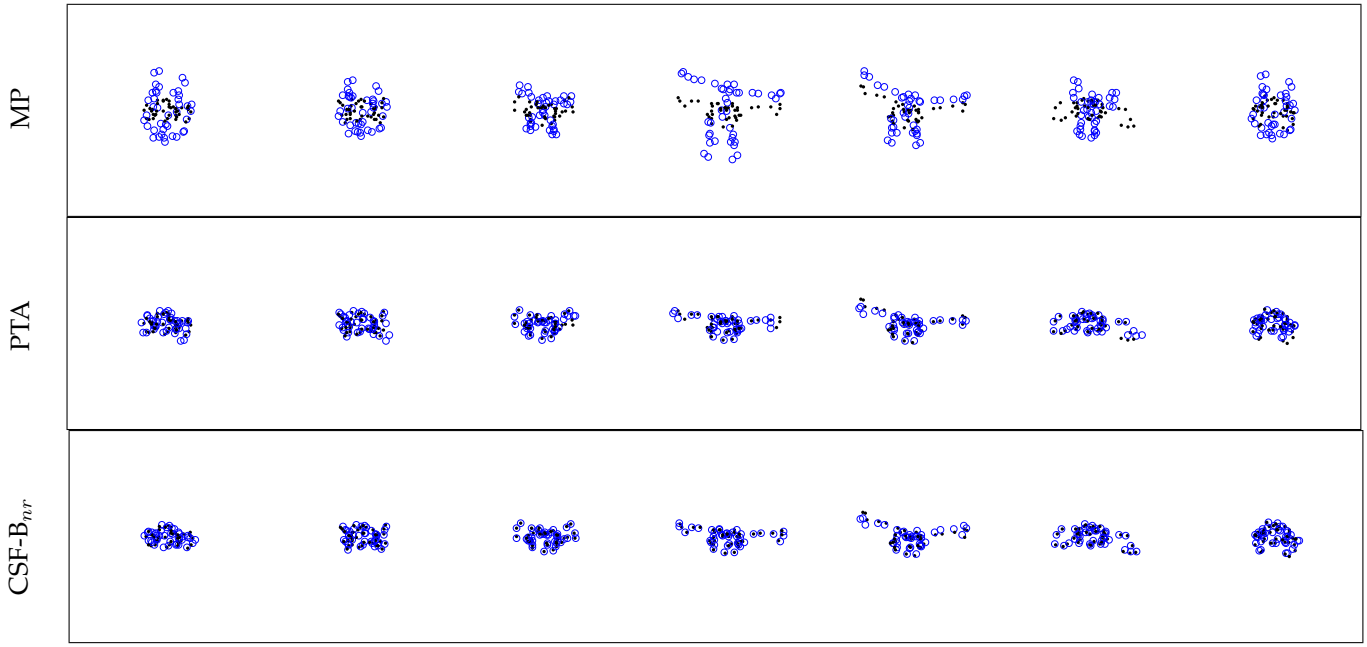


Fig. S2. Results on the *stretch* sequence as seen from above the person stretching their arms. Reconstructed 3D shapes (blue circles) are shown against the original 3D data (dark dots). Frames 7, 59, 111, 163, 215, 267, and 319 are displayed above.

3 Non-rigid SFM Results on Motion Capture Sequences

This section offers additional non-rigid SFM results on the motion capture sequences *dance* and *stretch*. We present and discuss only the results of the three best algorithms on each of these two datasets, as in Table 6.

The results of MP, PTA, and CSF-B$_{nr}$ on the motion capture sequence *dance* are shown in Fig. S1. Because the deformable structure in this sequence presents natural rotations, the addition of artificial rotations was not necessary. On this sequence, CSF-B$_{nr}$ provided a mean 3D reconstruction error smaller than that of PTA and equivalent to that of MP (see Table 6 in the main manuscript). However, the results of all three methods show inaccuracies that may suggest the need of non-linear models in some applications of non-rigid SFM. Also, the fact that MP provided

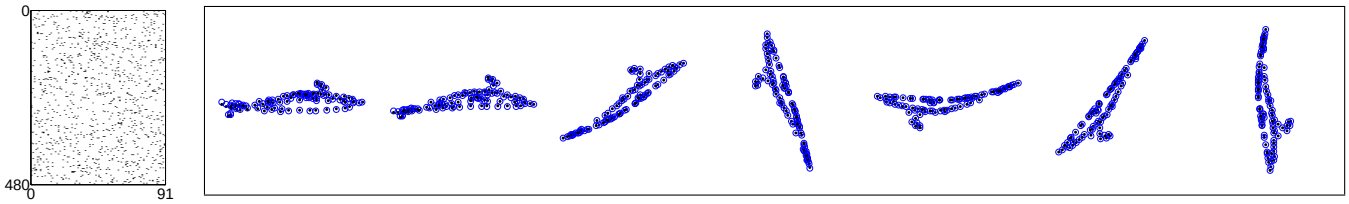


Fig. S3. Results on the *bending shark* sequence with 95% missing data. The left-most plot shows the available entries (in dark) of the input matrix $\mathbf{W}$. The other plots show the reconstructed 3D shapes (blue circles) against the original 3D data (dark dots). Frames 20, 50, 80, 115, 148, 175, and 200 are displayed above.

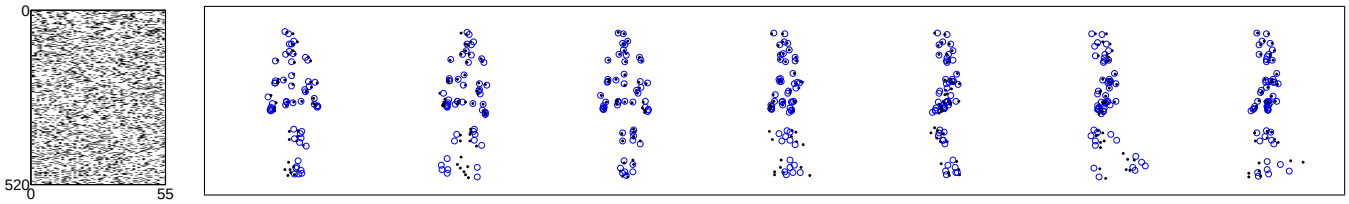


Fig. S4. Results on the *walking* sequence with 75% missing data. The left-most plot shows the visible entries (in dark) of the input matrix $\mathbf{W}$. The other plots show the reconstructed 3D shapes (blue circles) against the original 3D data (dark dots). Frames 34, 74, 122, 160, 198, 223, and 255 are displayed above.

good results on the dance dataset, but not on the other datasets with articulated bodies, needs to be investigated in future experiments. Like the dance dataset, the walking sequence also has natural rotations but MP does not perform as well on this sequence.

Results of the three methods above on the *stretch* sequence are shown in Fig. S2, which has unrotated body shapes – *i.e.* without the artificial rotations initially applied on the sequence to simulate camera motion. Note the results for frames 163 and 267 (the fourth and sixth shapes displayed in each row) in which there is a faster change in hand position as the person stretches their arms out and back in. In these frames the results obtained with PTA are over-smoothed. In comparison, our CSF-B$_{nr}$ method provides more accurate 3D shape reconstructions because it can better fit the high-frequency components of shape deformation. The result of MP on this sequence presents a high 3D reconstruction error.

4 Non-rigid SFM Results With Missing Data

Random occlusion masks were applied on the input matrices $\mathbf{W}$ of the motion capture sequences *bending shark* and *walking*. Fig. S3 shows an example of an occlusion pattern in $\mathbf{W}$ and the corresponding 3D shape reconstructions on the shark dataset with 95% missing data. In this example, the application of CSF-B$_{af}$ to recover complete 2D point trajectories from the available data in $\mathbf{W}$ provided a normalized mean/maximum 2D reprojection error of $0.0015/0.0029$. The normalized mean 3D reconstruction error obtained with CSF-B$_{nr}$ was then 0.0163.

A similar example for the *walking* dataset with 75% missing data is shown in Fig. S4. The normalized mean/maximum 2D error provided by CSF-B$_{af}$ was $0.0508/0.0548$. The normalized mean 3D reconstruction error obtained with CSF-B$_{nr}$ was then 0.2063.

Despite the large amount of missing data, the results in Figs. S3 and S4 are equivalent to those obtained with a complete matrix $\mathbf{W}$ (see Table 6 and Figs. 7 and 8 in the main manuscript).

5 Non-rigid SFM Results on Real Sequences

Results on the real datasets *cubes* (200/14) and *deformable dinosaur* (231/49), also appearing in [2], are given in Fig. S5 and Fig. S6. Because there is no 3D ground truth data available for these datasets, we only show overlays of the results given by PTA and CSF-B$_{nr}$. These results are also compared in terms of their mean 2D reprojection errors.

On the *cubes* dataset, CSF-B$_{nr}$ provides a smaller mean 2D reprojection error than that of PTA (see main manuscript). CSF-B$_{nr}$ can better model the changes in speed of the cube being pulled by a string. These changes in speed introduce high-frequency deformation components in the 3D shape formed by taking together the 3D points of both cubes.

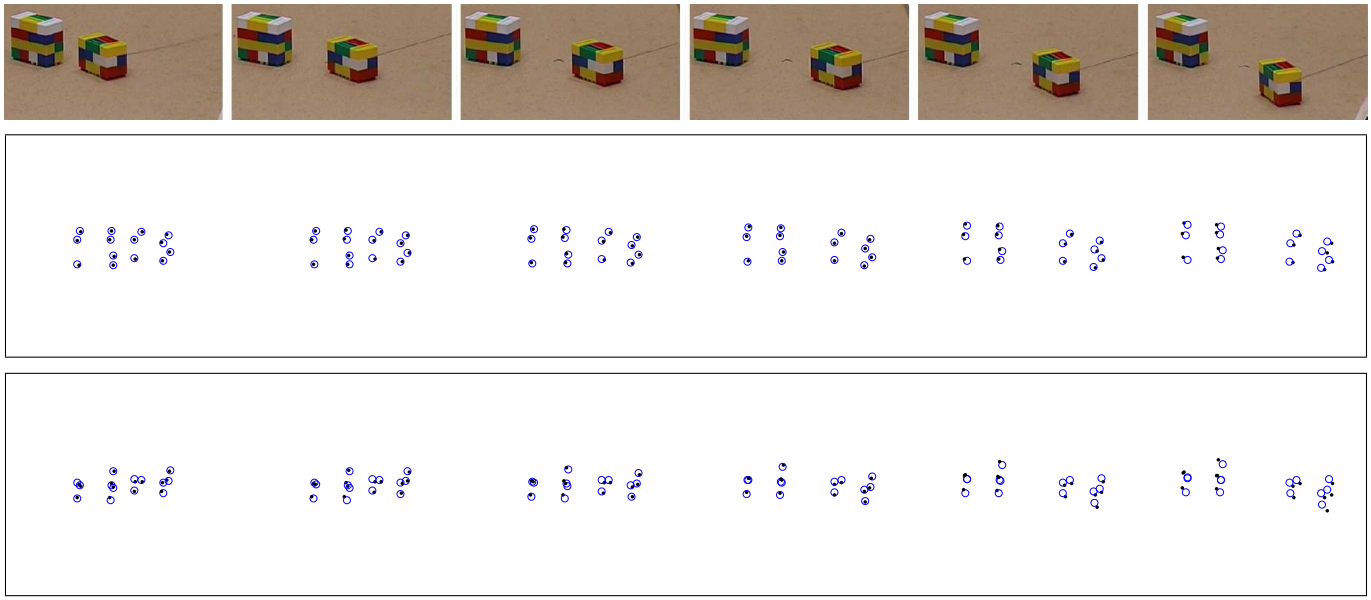


Fig. S5. Results on the real data of the *cubes* sequence. Top row shows example images. The middle and bottom rows give two different views of the 3D shapes reconstructed with PTA (dark dots) and CSF-B$_{nr}$ (blue circles). Results for frames 6, 39, 72, 105, 138, and 171 are displayed above.

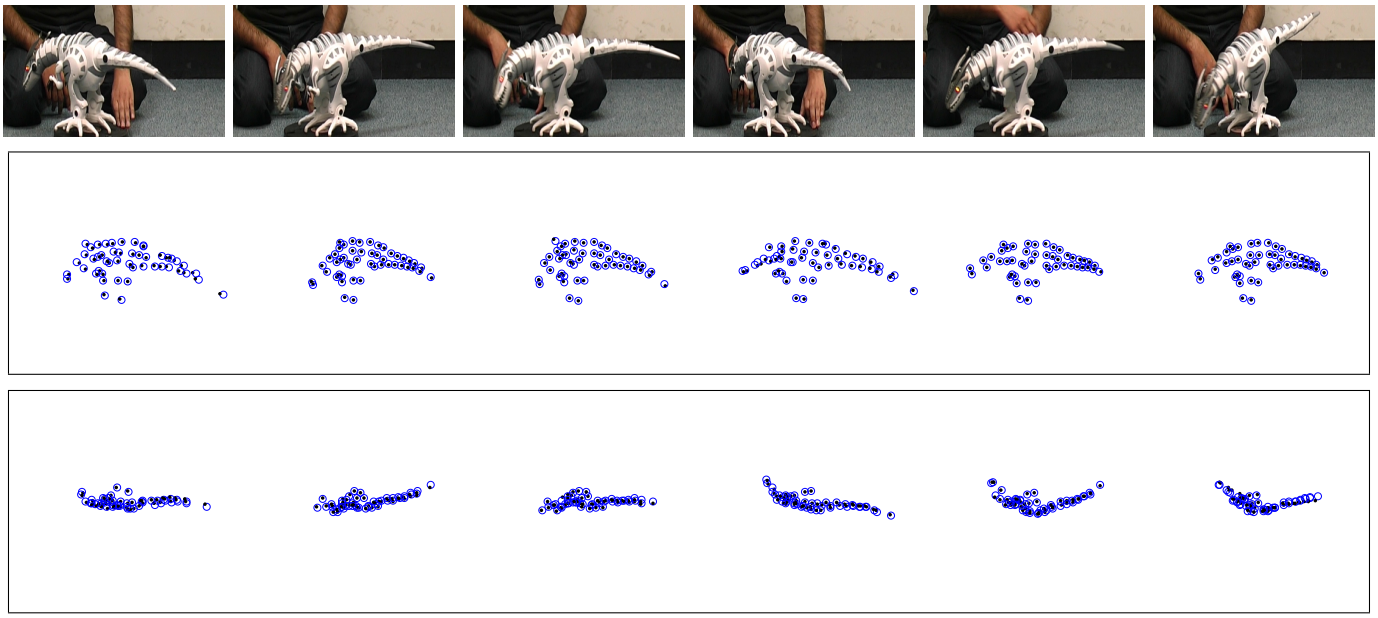


Fig. S6. Results on the real data of the *deformable dinosaur* sequence. Top row shows example images. The middle and bottom rows give two different views of the 3D shapes reconstructed with PTA (dark dots) and CSF-B$_{nr}$ (blue circles). Results for frames 6, 44, 82, 120, 158, and 196 are displayed above.

On the more complex deformation of the dinosaur model in Fig. S6, the best 3D shape reconstructions given by PTA and CSF-B$_{nr}$ were obtained with the same factorization rank and number of DCT components (*i.e.*, $K = d = 12$). The recovered 3D shapes are, thus, visually similar. Because we use a modified procedure for estimating rotations, the mean/maximum 2D reprojection errors (in pixel units) given by CSF-B$_{nr}$ and PTA are only slightly different, respectively, 4.9780/58.3390 and 4.9223/55.9048.

6 Results of EM-PPCA and MP on the ASL Sequence

The algorithms EM-PPCA and MP can handle cases of missing data by adopting a strategy similar to that of PowerFactorization, solving for camera motion and deformable shape in an alternated manner. This is a simple

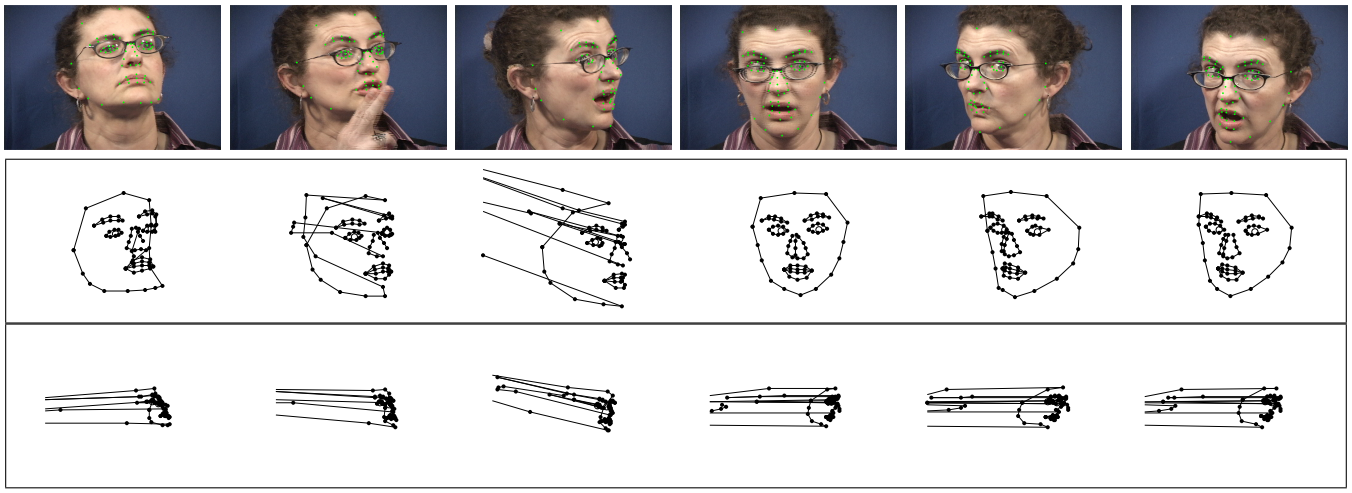


Fig. S7. Results of EM-PCA on the ASL sequence (77 points, 17.4% missing data): (*top*) six out of 115 images with annotated facial landmarks in green; (*middle,bottom*) two orthogonal views of the recovered 3D shapes.

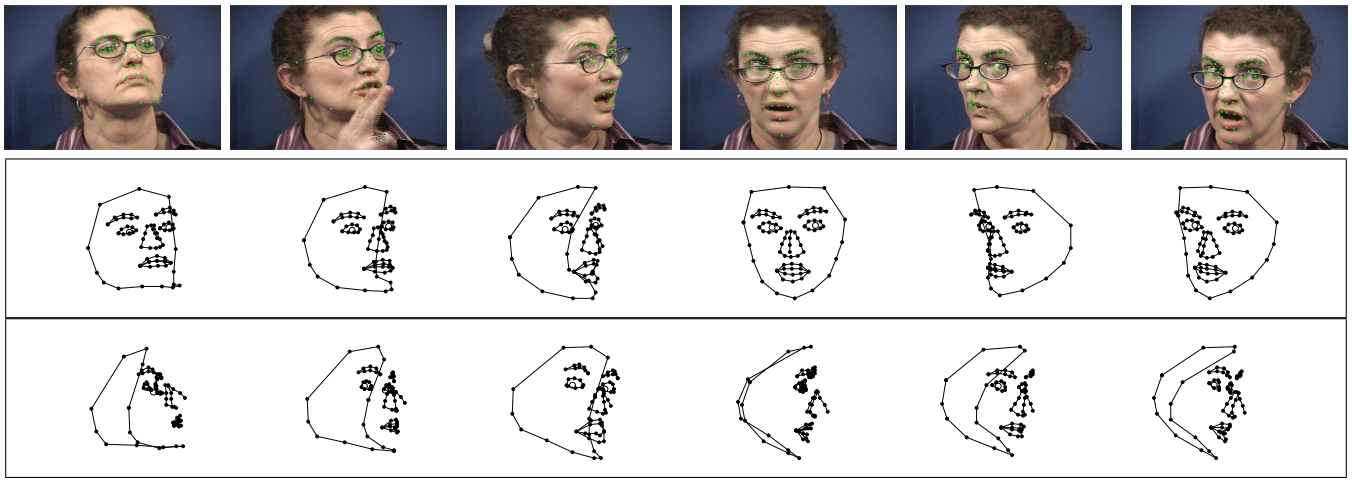


Fig. S8. Results of MP on the ASL sequence (77 points, 17.4% missing data): (*top*) six out of 115 images with annotated facial landmarks in green; (*middle,bottom*) two orthogonal views of the recovered 3D shapes.

strategy that often provides good results on datasets with small amounts of missing data. However, for all tested values of $K \in \{2, 3, \ldots, 13\}$, the result obtained with EM-PPCA on our ASL dataset presented visibly large 3D reconstruction errors for a number of facial points that are occluded frequently in each sequence (*e.g.*, the lateral contours of the nose and the face). Figure S7 shows the result of EM-PPCA with $K = 2$. The result of MP, also with $K = 2$, is shown in Fig. S8 and is visually similar to that of CSF-B$_{nr}$ (see main manuscript). These results present mean/maximum 2D reprojection errors of 3.2910/17.2972 (EM-PPCA) and 2.7666/19.4416 (MP), and 2.9322/18.2654 (CSF-B$_{nr}$) pixels. The smooth 3D shape trajectory computed by CSF-B$_{nr}$ provides a 2D error that is slightly higher than that of MP.

7 Comparison against MP with an Articulated Shape Model

In this section, we compare the performances of PTA and CSF-B$_{nr}$ against that of MP with a specialized, articulated shape model (MPA). Shape articulation, seen as the intersection of motion subspaces, can lead to degeneracies in the rank-$3K$ factorization computed by non-rigid SFM algorithms.

The MPA variant of the MP method requires a pre-processing algorithm to group the 2D trajectories of $\mathbf{W}$ into separate parts of an articulated structure. Here we use a dataset for which such trajectory groups are already available. The synthetic *hinge* sequence is provided with the source code of MP and MPA.[1] This dataset includes a 63-frame sequence of a smoothly deforming 3D shape composed of two 3D boxes joined by a hinge. Each 3D box is

1. Available at *http://www.dcs.qmul.ac.uk/∼lourdes/code.html*

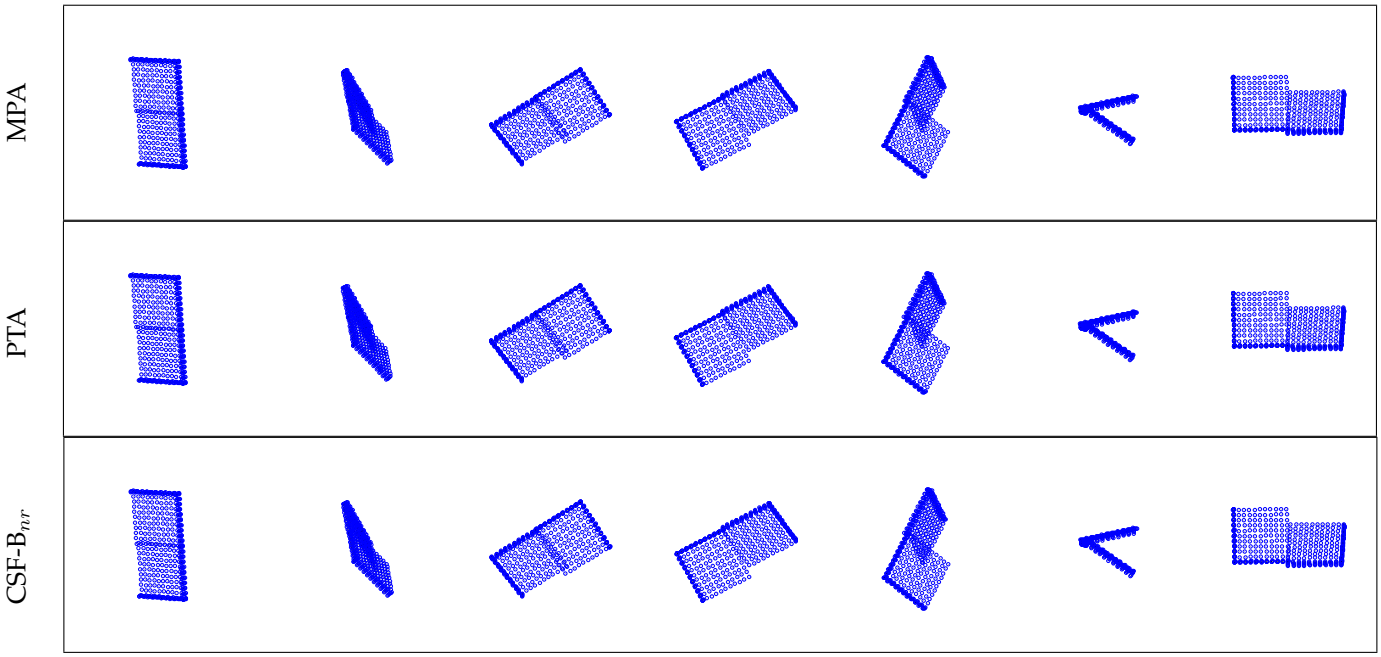


Fig. S9. Results on the *hinge* sequence. Reconstructed 3D shapes (blue circles) are shown against the original 3D data (dark dots). Frames 7, 16, 25, 34, 43, 52, and 61 are displayed above.

composed of 231 points. The supplied observation matrix $\mathbf{W} \in \mathbb{R}^{126\times462}$ was generated by orthographic projection. The dataset also includes two index matrices that describe two groups of points, one for each box. These index matrices are used by MPA but are ignored by PTA and CSF-B$_{nr}$.

From the 2D trajectories in $\mathbf{W}$, the 3D shapes reconstructed with MPA, PTA, and CSF-B$_{nr}$ are shown in Fig. S9. The normalized mean 3D error of these results are 0.0002 (MPA), 0.0148 (PTA), and 0.0071 (CSF-B$_{nr}$). All three methods are capable of accurately reconstructing the original 3D shapes in this sequence. Despite the slightly higher 3D errors, PTA and CSF-B$_{nr}$ are capable of recovering the articulated 3D shapes in this dataset without requiring an error-prone, initial grouping of 2D point trajectories. CSF-B$_{nr}$ demonstrates that this task can be accomplished with the standard linear shape model.