# CP decomposition-based algorithms for completion problem of motion capture data

Citation for the original published paper (version of record):
Mohaoui, S., Dmytryshyn, A. (2024). CP decomposition-based algorithms for completion problem of motion capture data. Pattern Analysis and Applications, 27(4).
http://dx.doi.org/10.1007/s10044-024-01342-4

Pattern Analysis and Applications (2024) 27:133
https://doi.org/10.1007/s10044-024-01342-4

ORIGINAL ARTICLE

# CP decomposition-based algorithms for completion problem of motion capture data

Souad Mohaoui[1] · Andrii Dmytryshyn[1,2]

Received: 3 May 2024 / Accepted: 13 September 2024

**Abstract**

Motion capture (MoCap) technology is an essential tool for recording and analyzing movements of objects or humans. However, MoCap systems frequently encounter the challenge of missing data, stemming from mismatched markers, occlusion, or equipment limitations. Recovery of these missing data is imperative to maintain the reliability and integrity of MoCap recordings. This paper introduces a novel application of the tensor framework for MoCap data completion. We propose three completion algorithms based on the canonical polyadic (CP) decomposition of tensors. The first algorithm utilizes CP decomposition to capture the low-rank structure of the tensor. However, relying only on low-rank assumptions may be insufficient to deal with complex motion data. Thus, we propose two modified CP decompositions that incorporate additional information, SmoothCP and SparseCP decompositions. SmoothCP integrates piecewise smoothness prior, while SparseCP incorporates sparsity prior, each aiming to improve the accuracy and robustness of MoCap data recovery. To compare and evaluate the merit of the proposed algorithms over other tensor completion methods in terms of several evaluation metrics, we conduct numerical experiments with different MoCap sequences from the CMU motion capture dataset.

**Keywords** MoCap data · Missing markers · Gap-filling · Candecomp/parafac (CP) decomposition · Tensor recovery · Smooth CP · Sparse CP

## 1 Introduction

Motion Capture (MoCap) systems are popular technologies used for recording and analyzing the three-dimensional movements of individuals or objects by employing specialized cameras, sensors, or wearable devices. The advanced techniques for capturing and analyzing motion data have made motion capture an indispensable tool in various practical applications within medicine [1, 2], sports sciences [3, 4], robotics [5–8], etc. Most human MoCap systems use the optical motion capture technique, where specialized cameras track markers attached to specific joints of the human body. These cameras capture the position and orientation of markers in three-dimensional space, and software is then used to reconstruct the skeleton and posture of the human subject.

Despite the use of professional systems and advanced software, recorded trajectories of the markers may still suffer from noise or incompleteness due to occlusions, mismatched markers, and lighting conditions. For instance, if a marker is hidden from all cameras at a specific recording time, its trajectory remains incomplete and cannot be accurately recorded. Missing markers during the motion capture process leads to gaps in the data, making it challenging to have a complete and continuous representation of the motion being captured. Notably, the missing markers problem also known as the gap-filling problem in MoCap data belongs to the broader class of data completion problems. A classical approach to addressing this problem is Low-Rank Matrix Completion (LRMC), which seeks to recover missing entries in a matrix from partial observations [9]. These approaches rely on the low-rank structure to capture the underlying low-order patterns and relations in the data. The problem of missing markers in MoCap data can be seen as a partial case of LRMC problem, allowing, e.g., the singular value

✉ Souad Mohaoui
souad.mohaoui@oru.se

Andrii Dmytryshyn
andrii@chalmers.se; andrii.dmytryshyn@oru.se

[1] Mathematics Department, School of Science and Technology, Örebro University, Örebro, Sweden

[2] Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, 41296 Gothenburg, Sweden

Published online: 04 October 2024

thresholding algorithm [10] to be used for MoCap data completion [11, 12].

Improving the recovery of missing markers can be achieved by exploring specific properties of human motion. For instance, in [13, 14], bone length constraints are introduced to prevent unrealistic skeleton properties. Moreover, human motion exhibits temporal smoothness characterized by continuous transitions in position, velocity, and acceleration. This observation has led to the development of robust matrix completion models that incorporate both low-rank properties and temporal smoothness [14–16]. Furthermore, a hybrid MoCap completion approach [17] has been proposed, combining skeleton constraints with smoothness to maintain consistent bone lengths and explore spatial smoothness in skeleton sequences. Human motion often exhibits strong nonlocal self-similarity, where similar actions are replicated across multiple instances. Motivated by this observation, [18] introduces a nonlocal low-rank framework. In [19], the authors propose hierarchical block-based incomplete MoCap data recovery using nonnegative matrix factorization. This method decomposes the human skeleton into five blocks, treating MoCap sequences as sub-motion chain clips, and applies nonnegative matrix factorization to recover missing markers for each sub-motion chain individually. Low-rank matrix completion may fail to deliver satisfactory performance when the data is subject to complex nonlinear changes, such as significant pose variations or varying illumination conditions. To cope with the nonlinear data effectively, [20] proposes a nonlinear matrix completion method by approximating low-rank kernel using a multiple kernel learning process.

In several practical fields, data inherently possesses a multidimensional structure and often can be organized in multidimensional arrays. This is typically the case with the analysis of MoCap data in which the movements of individuals or objects are recorded in three dimensions. Thus, a valuable method for studying and analyzing MoCap data is to use multilinear algebra, and, in particular, tensors and their decompositions. In the context of missing data, tensor completion techniques have attracted significant attention due to their flexibility and broad applicability across various fields.

Analog to the LRMC approach [9], tensor completion aims to reconstruct multidimensional data by approximating a partially observed tensor using the rank structure of the data. Unlike in the matrix case, the definition of the tensor's rank is not well-established. Thus, to characterize the low-rankness of tensors, various tensor rank definitions, as well as their convex relaxations, have been studied. Overall, the tensor rank definitions have been proposed based on different decomposition methods such as Canonical Polyadic (CP) decomposition [21], tucker decomposition [22], and tensor train decomposition [23, 24]. Based on these definitions, several tensor completion approaches have been proposed. In [25], the authors propose the tensor trace norm as a relaxation of the tucker rank. This is defined by taking the average sum of the nuclear norm of the unfolding matrices across all dimensions of an observed tensor. In [23], authors propose the tensor multi-rank and tubal rank definitions based on the tensor tensor product. Besides, a tensor train rank has been proposed in [26] for the tensor completion problem. The tensor train rank is established by unfolding the tensor along with permutations of modes instead of one model versus the rest.

An alternative to minimizing tensor rank directly is tensor decomposition methods. Tensor decompositions break down a high-dimensional tensor into a series of lower-dimensional tensors, facilitating the extraction of meaningful patterns and simplifying tensor completion. Various tensor decomposition techniques have been applied to tensor completion problems, including CP decomposition [27, 28], tucker decomposition [29], tensor singular value decomposition [30, 31], and tensor networks [32]. Each of these methods leverages the low-rank property of the tensor to address the completion problem.

While the low-rank prior is crucial for effective tensor completion, it may not be sufficient to deal with complex data. Thus, to address this limitation, several approaches incorporate additional prior knowledge [13, 15, 27, 28, 33–37]. These enhanced methods aim to refine the completion process by integrating contextual or structural insights that complement the low-rank properties, such as manifold information, smoothness, sparsity, and spatial or temporal constraints. For instance, one popular method involves regularizing the factor matrices of the tensor decomposition with the $l_2$ norm [38]. Another approach proposes simultaneous tensor decomposition and completion using manifold information and the nuclear norm to regularize the tucker decomposition. Additionally, a Bayesian CP decomposition with mixture priors and an automatic rank determination procedure has been proposed, along with methods that exploit smoothness priors to constrain the decomposition factors, enhancing applicability and robustness. However, these methods often lack comparisons with baseline tensor decomposition methods in their reported results. This absence of comparative evaluation leaves unanswered questions regarding the extent of improvement achieved by these auxiliary priors in the context of tensor decompositions for data completion. To bridge this gap, it is essential to present a comprehensive framework that thoroughly investigates and evaluates both the baseline tensor decomposition and its modified versions. To address the identified gap and demonstrate the robustness of tensor completion techniques in MoCap data, this paper develops and evaluates three tensor completion algorithms. In the first algorithm, we consider the CP decomposition to reveal the latent patterns that may be present in the tensor MoCap data. Besides, when Mocap data is subject to complex changes, such as significant pose variations, the CP decomposition only relies on the low-rank property, and may fail to deliver satisfactory results. Therefore, given the inherent complexity of MoCap data, we propose two modified variations of CP decomposition: SmoothCP and SparseCP. These modified decompositions incorporate constraints on the factor vectors to enhance the extraction of meaningful patterns by leveraging smoothness and sparsity properties, respectively.

Smoothness is an essential characteristic of MoCap data, which can be effectively utilized to predict missing elements more accurately. In the SmoothCP decomposition, we incorporate 1D total variation constraints on the factor vectors. This approach ensures that the decomposition respects the smooth transitions inherent in human motion. Conversely, sparsity focuses on identifying and utilizing the most informative parts of the data, which is crucial in scenarios where only a few significant elements contribute to the overall structure. In the SparseCP decomposition, we introduce sparsity constraints by applying the $l_1$-norm to the factor vectors. This method guides the decomposition process to produce sparse factor vectors, highlighting the most informative components and reducing redundancy in the decomposition. In the experiment section, we evaluate the effectiveness of these priors in enhancing the performance of tensor completion regarding the CP decomposition-based algorithm. Briefly, the contributions of this paper are as follows:

- We introduce and demonstrate the application of tensor completion techniques for the gap-filling problem in MoCap data.
- We propose three CP-based approaches to address the gap-filling problem taking into consideration the complexity of the data.
- We develop three minimization algorithms to solve the three completion problems and we study their convergence analysis.
- We evaluate the merit of these algorithms in the experimental section, as well as compare their performance with state-of-the-art tensor completion algorithms.

The rest of this paper is organized as follows. After the introduction, in Sect. 2, we introduce the main notations, properties, and preliminaries used throughout the paper. Then, in Sect. 3, we present the proposed tensor completion methods. In Sect. 4, we illustrate and discuss the numerical simulations and the comparison between the proposed methods over different MoCap sequences from the CMU dataset. Finally, we conclude in Sect. 5.

## 2 Tensor notations and preliminaries

In this section, we present notations and definitions used throughout the paper. We also review the CP decomposition and the optimization algorithm used to fit the decomposition.

### 2.1 Basic definitions

The order of a tensor is the number of dimensions. A tensor $\mathcal{X}$ of order $N$ and size $I_1 \times I_2 \times \cdots \times I_N$ is an $N$ dimensional array in $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. Thus, a vector is a first-order tensor and a matrix is a second-order tensor. We use the Euler script to denote tensors, e.g., $\mathcal{X}$, bold upper-case letters to denote matrices, e.g., $\boldsymbol{X}$, bold lowercase letters to denote vectors, e.g., $\boldsymbol{x}$, and light lowercase letters to denote scalars, e.g., $x$. The $j$-th column of the matrix $\boldsymbol{X}$ is denoted as $\boldsymbol{x}_j$. The $\left(i_1, i_2, \ldots, i_n\right)$-th element of a tensor $\mathcal{X}$ is denoted as $x_{i_1 i_2 \ldots i_n}$, the $\left(i_1, i_2\right)$-th element of a matrix $\boldsymbol{X}$ is denoted as $x_{i_1 i_2}$, and the $(i)$-th element of a vector $\boldsymbol{x}$ is denoted as the scalar $x_i$. Following [39], we define the notions below.

**Definition 1** (*The inner product*) The inner product of two same-sized tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the sum of the products of their entries, i.e.,

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1, \ldots, i_N} x_{i_1, \ldots, i_N} \cdot y_{i_1, \ldots, i_N}. \tag{1}$$

**Definition 2** (*The Frobenius norm*) The Frobenius norm of tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the square root of the sum of the squares of all its elements, i.e.,

$$\|\mathcal{X}\|_F = \sqrt{\sum_{i_1} \sum_{i_2} \cdots \sum_{i_N} x^2_{i_1 i_2 \ldots i_N}} = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}. \tag{2}$$

**Definition 3** (*The rank-one tensor*) A tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ of order $N$ is rank one if it can be written as the outer product of $N$ vectors, i.e.,

$$\mathcal{X} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \cdots \circ \boldsymbol{a}^{(N)}, \tag{3}$$

where the symbol "∘" represents the vector's outer product. This means that each element of the tensor is the product of the corresponding vector elements

$$x_{i_1 i_2 \cdots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(N)}_{i_N}, \quad \text{for all } 1 \leq i_n \leq I_n. \tag{4}$$

**Definition 4** (*Super-diagonal tensor*) A tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ of order $N$ is super-diagonal if

$$x_{i_1 i_2 \cdots i_N} = \begin{cases} 1 & \text{if } i_1 = i_2 = \ldots = i_N, \\ 0 & \text{otherwise.} \end{cases}$$

**Definition 5** (*The Khatri–Rao product*) Given two matrices $\boldsymbol{A} = [\boldsymbol{a}_1, ..., \boldsymbol{a}_K] \in \mathbb{R}^{I \times K}$ and $\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_K] \in \mathbb{R}^{J \times K}$, their Khatri–Rao product is the matching column-wise Kronecker product and is denoted by $\boldsymbol{A} \odot \boldsymbol{B}$. The result is a matrix of size $IJ \times K$ defined by

$$\boldsymbol{A} \odot \boldsymbol{B} = \left[\boldsymbol{a}_1 \otimes \boldsymbol{b}_1,\ \boldsymbol{a}_2 \otimes \boldsymbol{b}_2,\ \cdots\ , \boldsymbol{a}_K \otimes \boldsymbol{b}_K\right], \tag{5}$$

where $\otimes$ is the Kronecker product. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are vectors, then the Khatri–Rao and Kronecker products are identical, i.e., $\boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{a} \odot \boldsymbol{b}$.

**Definition 6** (*The n-mode tensor-matrix product*) The $n$-mode tensor-matrix product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{X} \times_n \mathbf{A}$ and is of the size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$. Elementwise,

$$\left(\mathcal{X} \times_n \mathbf{A}\right)_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} a_{j i_n}. \tag{6}$$

**Definition 7** (*The n-mode tensor-vector product*) The $n$-mode tensor-vector product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a vector $\mathbf{v} \in \mathbb{R}^{I_n}$ is denoted by $\mathcal{X} \bar{\times}_n \mathbf{v}$. The result is of order $N-1$, i.e., the size is $I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N$. Elementwise,

$$\left(\mathcal{X} \bar{\times}_n \mathbf{v}\right)_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} v_{i_n}. \tag{7}$$

**Definition 8** (*The mode-n matricization*) The mode-$n$ matricization (unfolding) of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, is the operation of transforming the tensor into a matrix denoted as $\boldsymbol{X}^{(n)} \in \mathbb{R}^{I_n \times S}$, where $S = \prod_{k \neq n}^{N} I_k$. Thus, the tensor element $\left(i_1, i_2, \ldots, i_N\right)$ corresponds to the matrix element $\left(i_n, j\right)$, where

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} \left(i_k - 1\right) J_k, \quad \text{with} \quad J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m.$$

***Example 1*** The mode-1, mode-2, and mode-3 matricizations of a third-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ using Matlab notation are, respectively,

$$\begin{aligned}
\boldsymbol{X}^{(1)} &= \left[\mathbf{x}_{:11}, \ldots, \mathbf{x}_{:I_2 1}, \mathbf{x}_{:12} \ldots, \mathbf{x}_{:I_2 2}, \ldots, \mathbf{x}_{:1 I_3}, \ldots, \mathbf{x}_{:I_2 I_3}\right] \in \mathbb{R}^{I_1 \times I_2 I_3},\\
\boldsymbol{X}^{(2)} &= \left[\mathbf{x}_{1:1}, \ldots, \mathbf{x}_{I_1:1}, \mathbf{x}_{1:2} \ldots, \mathbf{x}_{I_1:2}, \ldots, \mathbf{x}_{1:I_3}, \ldots, \mathbf{x}_{I_1:I_3}\right] \in \mathbb{R}^{I_2 \times I_1 I_3},\\
\boldsymbol{X}^{(3)} &= \left[\mathbf{x}_{11:}, \ldots, \mathbf{x}_{I_1 1:}, \mathbf{x}_{12:} \ldots, \mathbf{x}_{I_1 2:}, \ldots, \mathbf{x}_{1 I_2:}, \ldots, \mathbf{x}_{I_1 I_2:}\right] \in \mathbb{R}^{I_3 \times I_1 I_2}.
\end{aligned}$$

## 2.2 CP decomposition

The CP decomposition factorizes multidimensional data, or tensors, into a sum of rank-one tensors. For an $N$-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, we have

$$\mathcal{X} \approx \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} = [\![\boldsymbol{\Lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!], \tag{8}$$

where for any $n \in \{1, \ldots, N\}$, $\boldsymbol{a}_r^{(n)} \in \mathbb{R}^{I_n}$ stand for the factor (or loading) vector and $\lambda_r$ are weight parameters. The matrix $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ called the $n$-th factor (or loading) matrix and refers to the combination of the vectors from the rank-one components, i.e., $\boldsymbol{A}^{(n)} = \left[\boldsymbol{a}_1^{(n)}, \boldsymbol{a}_2^{(n)}, \ldots, \boldsymbol{a}_R^{(n)}\right]$, $\boldsymbol{\Lambda} = \operatorname{diag}\left[\lambda_1, \ldots, \lambda_R\right] \in \mathbb{R}^{R \times \cdots \times R}$ is a super-diagonal tensor, and $R$ is the tensor rank also known as CP rank.

**Definition 9** (*The CP rank*) The CP rank of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is the minimum number of rank-1 tensors

$$\operatorname{rank}(\mathcal{X}) = \min_R \left\{ R \mid \mathcal{X} = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \right\}. \tag{9}$$

## 2.3 The hierarchical alternating least squares (HALS) algorithm

The Alternating Least Squares (ALS) algorithm is a commonly used minimization algorithm to obtain the CP decomposition of a given tensor $\mathcal{X}$. The idea is to consider the factor matrices and minimize the objective function

$$\min_{\boldsymbol{\Lambda}, \{\boldsymbol{A}^{(n)}\}_{n=1}^{N}} \frac{1}{2} \|\mathcal{X} - [\![\boldsymbol{\Lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|_F^2. \tag{10}$$

The ALS algorithm iteratively optimizes the factor matrices $\boldsymbol{A}^{(n)}$ leading to an easy implementation. One major limitation of the algorithm is that it may suffer from slow convergence [40, 41]. An alternative method is to use local cost functions whose simultaneous (one-by-one) minimization leads to a simple ALS algorithm, called the Hierarchical ALS (HALS) algorithm [42]. Instead of updating the factor matrices $\{\boldsymbol{A}^{(n)}\}_{n=1}^{N}$, the algorithm simultaneously updates each vector of $\{\{\boldsymbol{a}_r^{(n)}\}_{n=1}^{N}\}_{r=1}^{R}$ per iteration. The algorithm is a particular case of the Block Coordinates Descent (BCD) algorithm [43]. It is developed to deal with the nonnegative matrix/tensor factorization models by proposing a class of optimized local algorithms [44]. For the CP decomposition, the HALS operates by converting the global objective function for the rank-$R$ tensor to local objective functions for $R$ rank-one tensors

$$\begin{aligned} &\|\mathcal{X} - \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \|_F^2 \\ &\quad = \|\mathcal{X}_r - \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \|_F^2, \end{aligned} \tag{11}$$

where

$$\mathcal{X}_r = \mathcal{X} - \sum_{\substack{k=1 \\ k \neq r}}^{R} \lambda_k \boldsymbol{a}_k^{(1)} \circ \boldsymbol{a}_k^{(2)} \circ \cdots \circ \boldsymbol{a}_k^{(N)} \tag{12}$$

is the residue for $r \in \{1, ..., R\}$. The algorithm minimizes the above objective function by iteratively updating the following subproblems for $r \in \{1, ..., R\}$

$$\begin{aligned} \{\lambda_r, \boldsymbol{a}_r^{(n)}\} &= \arg \min_{\lambda_r, \boldsymbol{a}_r^{(n)}} \frac{1}{2} \|\mathcal{X}_r - \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \|_F^2, \\ & n \in \{1, ..., N\}. \end{aligned} \tag{13}$$

## 3 The completion algorithms

In this section, we describe the proposed MoCap completion algorithms. We first present a tensor representation of human MoCap data and then address the MoCap gap-filling problem using the three tensor completion frameworks.

### 3.1 MoCap tensor representation

Motion data is composed of a sequence of frames (poses) and each frame is characterized by the positions of markers placed on specific parts of the body or joints. Each marker provides 3D coordinates usually denoted by $[x, y, z]$. In all the existent completion methods for MoCap data, the captured motion sequence is represented as an $m \times n$ matrix $\boldsymbol{M}$ where $n = 3p$ is the number of position coordinates for all markers or joints, $m$ corresponds to the total number of frames in the motion sequence. Thus, each row of the matrix $\boldsymbol{M}$ corresponds to a frame in the sequence

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \vdots \\ \boldsymbol{f}_m \end{bmatrix} = \begin{bmatrix} x_{1,1} & y_{1,1} & z_{1,1} & \cdots & x_{1,p} & y_{1,p} & z_{1,p} \\ x_{2,1} & y_{2,1} & z_{2,1} & \cdots & x_{2,p} & y_{2,p} & z_{2,p} \\ & \vdots & & & & \vdots & \\ x_{m,1} & y_{m,1} & z_{m,1} & \cdots & x_{m,p} & y_{m,p} & z_{m,p} \end{bmatrix}. \tag{14}$$

Notably, MoCap data is recorded in three dimensions. Thus, MoCap data can naturally be modeled as a third-order tensor $\mathcal{M} \in \mathbb{R}^{m \times p \times 3}$ where the dimensions correspond to time (frames), joints (markers), and spatial coordinates (3D coordinates), respectively

$$\mathcal{M} = [\boldsymbol{M}_x, \boldsymbol{M}_y, \boldsymbol{M}_z],$$

where $\boldsymbol{M}_x$, $\boldsymbol{M}_y$, and $\boldsymbol{M}_z \in \mathbb{R}^{m \times p}$ represent the components of the MoCap data along the $x$, $y$, and $z$ axes, respectively.

### 3.2 CP decomposition for MoCap completion

Given an incomplete MoCap tensor $\mathcal{M}$ which has gaps due to the missing markers problem, we seek a complete tensor $\mathcal{X}$ where the missing markers are recovered as accurately as possible. In this context, the goal of the CP decomposition is to find a set of $R$ normalized rank-one tensors $\{\boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}\}_{r=1}^{R}$ that best approximates $\mathcal{X}$. Normalizing the loading factors in CP decomposition offers certain advantages, such as improving the robustness to noise, and preventing the minimization process from assigning excessively large values to some components.

We propose the following MoCap tensor completion using CP decomposition

$$\begin{aligned} &\min_{\mathcal{X}, \{\mathcal{A}_r\}_{r=1}^{R}} H(\mathcal{X}, \mathcal{A}_1, ..., \mathcal{A}_R) \\ &\quad = \min_{\mathcal{X}, \{\mathcal{A}_r\}_{r=1}^{R}} \frac{1}{2} \|\mathcal{X} - \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \|_F^2, \end{aligned} \tag{15}$$

such that $\mathcal{P}_\Omega(\mathcal{X}) = \mathcal{P}_\Omega(\mathcal{M})$,

$\|\boldsymbol{a}_r^{(n)}\|_2 = 1, \ n \in \{1, ..., N\}, r \in \{1, ..., R\}$,

where $\mathcal{A}_r = \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}$, $\Omega$ is the index set of observed entries, and $\mathcal{P}_\Omega$ is the projection of $\mathcal{X}$ on the observed set $\Omega$ defined by

$$\mathcal{P}_\Omega(\mathcal{X}) = \begin{cases} x_{i_1, i_2, \ldots, i_N} & \text{if } (i_1, i_2, \ldots, i_N) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

We consider the indicator function $\iota(\cdot)$ given by

$$\iota(\mathcal{X}) = \begin{cases} 0 & \text{if } \mathcal{P}_\Omega(\mathcal{X}) = \mathcal{P}_\Omega(\mathcal{M}), \\ \infty & \text{otherwise.} \end{cases}$$

Then, problem (16) can be written as the following convex minimization problem

$$\begin{aligned} &\min_{\mathcal{X}, \{\mathcal{A}_r\}_{r=1}^{R}} H(\mathcal{X}, \mathcal{A}_1, ..., \mathcal{A}_R) + \iota(\mathcal{X}), \\ &\text{such that} \quad \|\boldsymbol{a}_r^{(n)}\|_2 = 1, \quad n \in \{1, ..., N\}, r \in \{1, ..., R\}. \end{aligned} \tag{16}$$

This problem is minimized using the BCD algorithm, where we iteratively update the $R$ rank-one tensors $\{\mathcal{A}_r\}_{r=1}^{R}$ and the tensor $\mathcal{X}$. Each rank-one tensor $\mathcal{A}_r$ is updated using the HALS algorithm by minimizing local objective functions

$$\begin{aligned}&\min_{\lambda_r,\{\boldsymbol{a}_r^{(n)}\in\boldsymbol{C}_n\}_{n=1}^N} F(\mathcal{X}_r,\lambda_r,\boldsymbol{a}_r^{(n)})\\&\quad=\min_{\lambda_r,\{\boldsymbol{a}_r^{(n)}\in\boldsymbol{C}_n\}_{n=1}^N}\frac{1}{2}\|\mathcal{X}_r-\lambda_r\boldsymbol{a}_r^{(1)}\circ\boldsymbol{a}_r^{(2)}\circ\cdots\circ\boldsymbol{a}_r^{(N)}\|_F^2,\end{aligned}\tag{17}$$

where $\mathcal{X}_r$ is the residue and $\boldsymbol{C}_n=\{\boldsymbol{a}_r^{(n)}\in\mathbb{R}^{I_n}\mid\|\boldsymbol{a}_r^{(n)}\|_2=1\}$ is the set of unit vectors. At each iteration of the HALS algorithm, the objective function is minimized for a single vector $\boldsymbol{a}_r^{(n)}$ while fixing all other vectors. The completion process consists of two iterative steps. Initially, the algorithm approximates the CP decomposition of the MoCap tensor. This step involves optimizing the factor vectors hierarchically, where each factor vector is updated while keeping the others fixed

$$\{\mathcal{A}_r\}_{r=1}^R=\begin{cases}\boldsymbol{a}_r^{(n)}=\operatorname*{argmin}_{\boldsymbol{a}_r^{(n)}\in\boldsymbol{C}_n} F(\mathcal{X}_r,\lambda_r,\boldsymbol{a}_r^{(n)}), & n\in\{1,\ldots,N\},\\ \lambda_r=\operatorname*{argmin}_{\lambda_r} F(\mathcal{X}_r,\lambda_r,\boldsymbol{a}_r^{(n)}),\\ \mathcal{X}_r\leftarrow\mathcal{E}+\lambda_r\boldsymbol{a}_r^{(1)}\circ\boldsymbol{a}_r^{(2)}\circ\cdots\circ\boldsymbol{a}_r^{(N)},\end{cases}\tag{18}$$

where

$$\begin{aligned}\mathcal{E}&=\mathcal{X}-\sum_{r=1}^R\lambda_r\boldsymbol{a}_r^{(1)}\circ\boldsymbol{a}_r^{(2)}\circ\cdots\circ\boldsymbol{a}_r^{(N)}\\&=\mathcal{X}_r-\lambda_r\boldsymbol{a}_r^{(1)}\circ\boldsymbol{a}_r^{(2)}\circ\cdots\circ\boldsymbol{a}_r^{(N)}\end{aligned}$$

is the tensor decomposition error. The second step involves completing the gaps in the given tensor based on the obtained CP decomposition by minimizing the convex problem

$$\begin{aligned}\mathcal{X}=\operatorname*{argmin}_{\mathcal{X}}\ &\frac{1}{2}\|\mathcal{X}\\&-\sum_{r=1}^R\lambda_r\boldsymbol{a}_r^{(1)}\circ\boldsymbol{a}_r^{(2)}\circ\cdots\circ\boldsymbol{a}_r^{(N)}\|_F^2+\imath(\mathcal{X}).\end{aligned}\tag{19}$$

### 3.2.1 Tensor decomposition step

This step focuses on fitting the CP decomposition using the HALS algorithm. In this process, each rank-one tensor $\mathcal{A}_r$, undergoes hierarchical updates to its factor vectors $\boldsymbol{a}_r^{(n)}$ and the associated parameter $\lambda_r$.

***Updating the factor vectors*** $\{\boldsymbol{a}_r^{(n)}\}_{n=1}^N$

To update each factor vector $\boldsymbol{a}_r^{(n)}$, we use the mode-$n$ matricization version of problem (17)

$$\begin{aligned}\boldsymbol{a}_r^{(n)}&=\operatorname*{argmin}_{\boldsymbol{a}_r^{(n)}\in\boldsymbol{C}_n} F(\mathcal{X}_r,\lambda_r,\boldsymbol{a}_r^{(n)})\\&=\operatorname*{argmin}_{\boldsymbol{a}_r^{(n)}\in\boldsymbol{C}_n}\frac{1}{2}\|\mathcal{X}_r-\lambda_r\boldsymbol{a}_r^{(1)}\circ\boldsymbol{a}_r^{(2)}\circ\cdots\circ\boldsymbol{a}_r^{(N)}\|_F^2,\\&=\operatorname*{argmin}_{\boldsymbol{a}_r^{(n)}\in\boldsymbol{C}_n}\frac{1}{2}\|\boldsymbol{X}_r^{(n)}-\lambda_r\boldsymbol{a}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}}^T\|_F^2,\end{aligned}\tag{20}$$

where $\{\boldsymbol{a}_r\}_{\odot_{-n}}^T=[\boldsymbol{a}_r^{(N)}]^T\odot\ldots\odot[\boldsymbol{a}_r^{(n+1)}]^T\odot[\boldsymbol{a}_r^{(n-1)}]^T\odot\ldots\odot[\boldsymbol{a}_r^{(1)}]^T$, $(\cdot)^T$ refers to the transpose operator, and $\boldsymbol{X}_r^{(n)}$ is the mode-$n$ matricization of the tensor $\mathcal{X}_r$. Minimizing the objective function for each vector $\boldsymbol{a}_r^{(n)}$, $n\in\{1,\ldots,N\}$, which is a differentiable function requires setting the gradient at zero

$$\begin{aligned}\nabla_{\boldsymbol{a}_r^{(n)}}F(\mathcal{X}_r,\lambda_r,\boldsymbol{a}_r^{(n)})&=\lambda_r\boldsymbol{X}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}}-\lambda_r^2\boldsymbol{a}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}}^T\{\boldsymbol{a}_r\}_{\odot_{-n}}\\&=\lambda_r\boldsymbol{X}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}}-\lambda_r^2\boldsymbol{a}_r^{(n)}=0,\end{aligned}\tag{21}$$

where $\{\boldsymbol{a}_r\}_{\odot_{-n}}^T\{\boldsymbol{a}_r\}_{\odot_{-n}}=\{\boldsymbol{a}_r^T\boldsymbol{a}_r\}_{\odot_{-n}}=1$. Thus, the closed-form solution is given as

$$\boldsymbol{a}_r^{(n)}=\boldsymbol{P}_{\boldsymbol{C}_n}\left(\frac{\boldsymbol{X}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}}}{\lambda_r}\right),\tag{22}$$

where $\boldsymbol{P}_{\boldsymbol{C}_n}(\cdot)$ is the projection mapping onto $\boldsymbol{C}_n$, i.e., the normalization defined by $\boldsymbol{P}_{\boldsymbol{C}_n}(\boldsymbol{u})=\boldsymbol{u}/\|\boldsymbol{u}\|_2$. The term $\boldsymbol{X}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}}$ can also be expressed using either the tensor vector product

$$\begin{aligned}\boldsymbol{X}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}}&=\mathcal{X}_r\bar{\times}_1\boldsymbol{a}_r^{(1)}\cdots\bar{\times}_{n-1}\boldsymbol{a}_r^{(n-1)}\bar{\times}_{n+1}\boldsymbol{a}_r^{(n+1)}\\&\quad\cdots\bar{\times}_N\boldsymbol{a}_r^{(N)}=\mathcal{X}_r\bar{\times}_{-n}\{\boldsymbol{a}_r\},\end{aligned}$$

where $\{\boldsymbol{a}_r\}=\{\boldsymbol{a}_r^{(1)},\ldots,\boldsymbol{a}_r^{(N)}\}$ and $\bar{\times}_{-n}$ is a short notation of the multiplication of tensor $\mathcal{X}_r$ by all the vectors but $\boldsymbol{a}_r^{(n)}$. Or by using the tensor matrix product

$$\begin{aligned}\boldsymbol{X}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}}&=\mathrm{Vec}\big(\mathcal{X}_r\times_1[\boldsymbol{a}_r^{(1)}]^T\cdots\times_{n-1}[\boldsymbol{a}_r^{(n-1)}]^T\times_{n+1}[\boldsymbol{a}_r^{(n+1)}]^T\cdots\times_N[\boldsymbol{a}_r^{(N)}]^T\big)\\&=\mathrm{Vec}\big(\mathcal{X}_r\times_{-n}\{\boldsymbol{a}_r\}^T\big),\end{aligned}$$

where Vec is the operator that stacks the columns of a tensor into a column vector.

***Updating the parameter*** $\lambda_r$ After updating the factor vectors $\{\boldsymbol{a}_r^{(n)}\}_{n=1}^N$, we update the parameter $\lambda_r$ by minimizing the following strictly quadratic function

$$\begin{aligned}\lambda_r &= \underset{\lambda_r}{\operatorname{argmin}}\; F(\mathcal{X}_r, \lambda_r, \boldsymbol{a}_r^{(n)}) \\ &= \underset{\lambda_r}{\operatorname{argmin}}\; \frac{1}{2}\|\mathcal{X}_r - \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}\|_F^2, \\ &= \underset{\lambda_r}{\operatorname{argmin}}\; \frac{\lambda_r^2}{2} - \lambda_r \langle \mathcal{X}_r, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}\rangle.\end{aligned} \tag{23}$$

Setting the gradient of the function with respect to $\lambda_r$ to zero

$$\begin{aligned}&\nabla_{\lambda_r}\left(\frac{\lambda_r^2}{2} - \lambda_r \langle \mathcal{X}_r, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}\rangle\right) \\ &\quad = \lambda_r - \langle \mathcal{X}_r, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}\rangle = 0,\end{aligned} \tag{24}$$

yields to the following closed-form solution

$$\lambda_r = \langle \mathcal{X}_r, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}\rangle. \tag{25}$$

### 3.2.2 Tensor completion step

Once we update the CP decomposition, we turn into the second step to recover the MoCap tensor $\mathcal{X}$ by solving the minimization problem (19) which has the following solution

$$\mathcal{X} = \mathcal{P}_{\Omega^c}\left(\sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}\right) + \mathcal{P}_{\Omega}(\mathcal{M}), \tag{26}$$

where $\Omega^c$ is the complement of $\Omega$.

## 3.3 Modified CP decomposition for MoCap completion

The modified CP decomposition for the completion problem is given as

$$\begin{aligned}&\min_{\mathcal{X}, \{\mathcal{A}_r\}_{r=1}^{R}} H(\mathcal{X}, \mathcal{A}_1, \ldots, \mathcal{A}_R) \\ &\qquad + \imath(\mathcal{X}) + \sum_{r=1}^{R}\sum_{n=1}^{N} \alpha_n \mathcal{R}(\boldsymbol{a}_r^{(n)}), \\ &\text{such that} \;\; \|\boldsymbol{a}_r^{(n)}\|_2 = 1,\; n \in \{1, \ldots, N\}, r \in \{1, \ldots, N\},\end{aligned} \tag{27}$$

where $\mathcal{R} : \mathbb{R}^N \to \mathbb{R}$ refers to the regularization function and $\{\alpha_n\}_{n=1}^{N}$ are parameters controlling the level of the constraint on each factor vector. Then, under the framework of the HALS algorithm, problem (27) is formulated as

$$\begin{aligned}&\min_{\mathcal{X}, \lambda_r, \{\boldsymbol{a}_r^{(n)} \in \boldsymbol{C}_n\}_{n=1}^{N}} \frac{1}{2}\|\mathcal{X}_r - \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}\|_F^2 \\ &\qquad + \imath(\mathcal{X}) + \sum_{n=1}^{N} \alpha_n \mathcal{R}(\boldsymbol{a}_r^{(n)}), \\ &\text{such that} \;\; \mathcal{X}_r = \mathcal{X} - \sum_{\substack{k=1 \\ k \neq r}}^{R} \lambda_k \boldsymbol{a}_k^{(1)} \circ \boldsymbol{a}_k^{(2)} \circ \cdots \circ \boldsymbol{a}_k^{(N)}.\end{aligned} \tag{28}$$

Similar to (16), problem (28) is carried out in two steps: tensor decomposition and tensor completion steps. The completion step remains the same, only the first step changes entailing the solution of the following minimization problem

$$\begin{aligned}&\{\lambda_r, \boldsymbol{a}_r^{(n)}\} = \underset{\lambda_r, \boldsymbol{a}_r^{(n)} \in \boldsymbol{C}_n}{\operatorname{argmin}} \; F(\mathcal{X}_r, \lambda_r, \boldsymbol{a}_r^{(n)}) + \alpha_n \mathcal{R}(\boldsymbol{a}_r^{(n)}), \;\; n \in \{1, \ldots, N\}, \\ &\qquad \text{such that} \;\; \mathcal{X}_r = \mathcal{X} - \sum_{\substack{k=1 \\ k \neq r}}^{R} \lambda_k \boldsymbol{a}_k^{(1)} \circ \boldsymbol{a}_k^{(2)} \circ \cdots \circ \boldsymbol{a}_k^{(N)}.\end{aligned} \tag{29}$$

To update the factor vector $\boldsymbol{a}_r^{(n)}$ for $n \in \{1, \ldots, N\}$ in problem (29), we use the Projected Gradient Descent (PGD) method.[1] The PGD is an iterative process that performs a descent gradient/subgradient step and then projects the result to satisfy the constraints. To compute the projection effectively, the constraints are required to be sufficiently simple. Let $f(x)$ be a differentiable (not necessarily convex) continuous function over a convex set $Q$, for $k = 1, 2, \ldots, K_{\max}$, the PGD method consists of gradient descent and projection steps

$$x^{k+1} = \boldsymbol{P}_Q(x^k - \gamma \nabla f(x^k)), \tag{30}$$

where $\gamma$ is a positive stepsize, $\nabla f(x^k)$ is the gradient of the function $f$ at $x^k$, and $\boldsymbol{P}_Q(\cdot)$ denotes projection on the convex set $Q$. In the case where the function $f$ is convex and nondifferentiable at some points, the subgradient of the function $f$ is used instead of the gradient

$$x^{k+1} = \boldsymbol{P}_Q(x^k - \gamma \partial f(x^k)), \tag{31}$$

where $\partial f(x^k)$ is any subgradient of $f$ at $x^k$.

### 3.3.1 Smooth CP decomposition

The smoothness prior has been considered in the context of matrix MoCap data completion due to the inherent spatial and temporal smoothness in motion sequences. Among the various smoothness constraints, total variation stands out as a successful choice. Thus, we are interested in the following

[1] In this work, the abbreviation PGD refers to both the projected gradient and subgradient descent methods.

completion problem where the $l_2$-total variation regularization is imposed on the factor vectors

$$\{\lambda_r, \boldsymbol{a}_r^{(n)}\} = \underset{\lambda_r, \boldsymbol{a}_r^{(n)} \in \boldsymbol{C}_n}{\operatorname{argmin}} \ \frac{1}{2}\|\boldsymbol{X}_r^{(n)} - \lambda_r \boldsymbol{a}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}}^T\|_F^2 + \alpha_n\|\boldsymbol{a}_r^{(n)}\|_{\mathrm{TV}}^2, \quad n \in \{1, ..., N\}, \tag{32}$$

where $\|\boldsymbol{x}\|_{\mathrm{TV}}$ refers to the $l_2$-total variation norm of a vector $x \in \mathbb{R}^m$ defined as

$$\|\boldsymbol{x}\|_{\mathrm{TV}} = \left(\sum_{i=1}^{m-1} |x_i - x_{i+1}|^2\right)^{\frac{1}{2}} = \|D_m \boldsymbol{x}\|_2, \text{ where}$$

$$D_m = \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(m-1)\times m}.$$

In contrast to [28], the factors $\lambda_k$ are not involved in the penalty level of the smoothness prior in (32). While involving $\lambda_k$ in the penalty term provides adaptability, in cases where the data exhibits a relatively constant smoothness level, the adaptability of $\lambda_k$ may not provide a significant advantage. Thus, the levels of smoothness of different components can be enforced by setting adequate values of regularization parameters $\{\alpha_n\}_{n=1}^N$. In (32), minimizing each $\boldsymbol{a}_r^{(n)}$-subproblem involves a differentiable objective function. Thus we use the PGD algorithm with gradient descent step (30). Therefore, the HALS-based PGD yields the following iterative scheme

$$\begin{cases} \text{for } n = 1, ..., N \begin{cases} \boldsymbol{b}^k & = \lambda_r \boldsymbol{X}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}} - \lambda_r^2 \boldsymbol{a}_r^{(n),k} + \alpha_n D_n^T D_n \boldsymbol{a}_r^{(n),k}, \\ \boldsymbol{a}_r^{(n),k+1} & = \boldsymbol{a}_r^{(n),k} - \gamma \boldsymbol{b}^k, \\ \boldsymbol{a}_r^{(n),k+1} & = \boldsymbol{a}_r^{(n),k+1} / \|\boldsymbol{a}_r^{(n),k+1}\|_2, \end{cases} \\ \lambda_r = \langle \mathcal{X}_r, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \rangle, \\ \mathcal{X}_r \leftarrow \mathcal{E} + \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}, \end{cases} \tag{33}$$

where $\boldsymbol{b}^k$ is the gradient of the objective function (32) with respect to $\boldsymbol{a}_r^{(n)}$ at iteration $k$ and $\gamma$ is the gradient descent stepsize.

### 3.3.2 Sparse CP decomposition

We exploit the sparsity constraint over the factor vectors of the CP decomposition to encourage the robustness of the decomposition by focusing on the most informative features. Thus, we consider the $l_1$-norm minimization problem

$$\{\lambda_r, \boldsymbol{a}_r^{(n)}\} = \underset{\lambda_r, \boldsymbol{a}_r^{(n)} \in C_n}{\operatorname{argmin}} \ \frac{1}{2}\|\boldsymbol{X}_r^{(n)} - \lambda_r \boldsymbol{a}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}}^T\|_F^2 + \alpha_n\|\boldsymbol{a}_r^{(n)}\|_1, \quad n \in \{1, ..., N\}, \tag{34}$$

where $\|\boldsymbol{x}\|_1 = \sum_{i=1}^m |x_i|$ is the $l_1$-norm. The objective functions of the $\{\boldsymbol{a}_r^{(n)}\}_{n=1}^N$ subproblems are convex. Thus, minimizing each $\boldsymbol{a}_r^{(n)}$-subproblem is carried out by the PGD algorithm (31) with a subgradient descent step. The set of all subgradients of the $l_1$-norm denoted by $\partial(\|\cdot\|_1)$ is given by

$$\partial(\|\boldsymbol{x}\|_1) = \{v \in \mathbb{R}^m \mid v_i = x_i/|x_i| \text{ if } x_i \neq 0, \text{ and } v_i \in [-1, 1] \text{ if } x_i = 0\}.$$

The signum function of a vector $x \in \mathbb{R}^m$ denoted as $\mathrm{Sgn}(\boldsymbol{x}) = [\mathrm{sgn}(x_1), \mathrm{sgn}(x_2), \dots, \mathrm{sgn}(x_m)]^T$ is a subgradient of the $l_1$-norm and it is defined as

$$(\mathrm{Sgn}(\boldsymbol{x}))_i = \mathrm{sgn}(x_i) = \begin{cases} x_i/|x_i| & \text{if } x_i \neq 0, \\ 0 & \text{if } x_i = 0. \end{cases} \quad i = 1, \dots, m.$$

Therefore, the SparseCP decomposition is obtained by the following iterative scheme

$$\begin{cases} \text{for } n = 1, ..., N \begin{cases} \boldsymbol{b}^k & = \lambda_r \boldsymbol{X}_r^{(n)}\{\boldsymbol{a}_r\}_{\odot_{-n}} - \lambda_r^2 \boldsymbol{a}_r^{(n),k} + \alpha_n \mathrm{Sgn}(\boldsymbol{a}_r^{(n),k}), \\ \boldsymbol{a}_r^{(n),k+1} & = \boldsymbol{a}_r^{(n),k} - \gamma \boldsymbol{b}^k, \\ \boldsymbol{a}_r^{(n),k+1} & = \boldsymbol{a}_r^{(n),k+1} / \|\boldsymbol{a}_r^{(n),k+1}\|_2, \end{cases} \\ \lambda_r = \langle \mathcal{X}_r, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \rangle, \\ \mathcal{X}_r \leftarrow \mathcal{E} + \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}. \end{cases} \tag{35}$$

The three MoCap data completion algorithms are arranged together in Algorithm 1.

## 3.4 Rank estimation

A required task in the low-rank tensor factorization problem is to specify the ranks $R \in \{1, \dots, R_{\max}\}$ since in most cases, we do not know the true rank of the given tensor. Therefore, different rank-adjusting schemes have been proposed in [24, 31, 45–47]. We can distinguish between two main rank-adjusting schemes, namely, rank-decreasing and rank-increasing schemes. In this work, we use a rank-increasing technique for the three algorithms. The rank-increasing starts by $R$ such that $R \leq R_{\max}$. Then, we attempt to increase the rank by 1 if both, the rank $R$ has not reached the maximum rank and the following criterion is met

$$\frac{|v - v_{\mathrm{old}}|}{|v|} \leq \epsilon, \tag{36}$$

where $v = \|\mathcal{E}_\Omega^t\|_F^2$ denotes the mean squared error of the tensor decomposition at the current iteration and $\epsilon$ is a threshold value. When the criterion (36) is reached, it means that the current rank is not capturing the complexity or details of the data sufficiently. Thus, increasing the rank allows the model to be adapted to the complexity of the data and improve its performance.

**Algorithm 1** The three MoCap completion algorithms.

```
Input: The $N$-order tensor $\mathcal{M}$, the set of observed indexes $\Omega$, the parameters
  $\{\alpha_n\}_{n=1}^{N}$ and $\gamma$. The number of iterations $\mathrm{Iter}_{\max}$ and $K_{\max}$.
Output: The recovered tensor $\mathcal{X}$, the CP factors $\left\{\left\{\boldsymbol{a}_r^{(n)}\right\}_{n=1}^{N}\right\}_{r=1}^{R}$, and the core
tensor $\boldsymbol{\Lambda}$.
Initialization: $R = 1$, Randomly set and normalize to unit length:
$\left\{\left\{\boldsymbol{a}_r^{(n)}\right\}_{n=1}^{N}\right\}_{r=1}^{R}$, $\lambda_r = \left\langle \mathcal{X}, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_R^{(N)} \right\rangle$,
$\mathcal{E}_\Omega = [\mathcal{X} - \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}]_\Omega, \nu_{\text{old}} = \|\mathcal{E}_\Omega\|_F^2$, $\mathcal{P}_\Omega(\mathcal{X}) = \mathcal{P}_\Omega(\mathcal{M})$.
repeat
  Step 1: Tensor decomposition
    Choose the decomposition: $\{$CP; SmoothCP; SparseCP$\}$
    for $r = 1 : R$ do
      for $n = 1 : N$ do
        Algorithm 1.1: CP
          $\begin{cases} \boldsymbol{a}_r^{(n)} = \frac{1}{\lambda_r} \boldsymbol{X}_r^{(n)} \{\boldsymbol{a}_r\}_{\odot_{-n}}, \\ \boldsymbol{a}_r^{(n)} = \boldsymbol{a}_r^{(n)} / \|\boldsymbol{a}_r^{(n)}\|_2. \end{cases}$
        Algorithm 1.2: SmoothCP
        for $k = 1 : K_{\max}$ do
          $\begin{cases} \boldsymbol{b}^k & = \lambda_r \boldsymbol{X}_r^{(n)} \{\boldsymbol{a}_r\}_{\odot_{-n}} - \lambda_r^2 \boldsymbol{a}_r^{(n),k} + \alpha_n D_n^T D_n \boldsymbol{a}_r^{(n),k}, \\ \boldsymbol{a}_r^{(n),k+1} & = \boldsymbol{a}_r^{(n),k} - \gamma \boldsymbol{b}^k, \\ \boldsymbol{a}_r^{(n),k+1} & = \boldsymbol{a}_r^{(n),k+1} / \|\boldsymbol{a}_r^{(n),k+1}\|_2. \end{cases}$
        end
        Algorithm 1.3: SparseCP
        for $k = 1 : K_{\max}$ do
          $\begin{cases} \boldsymbol{b}^k & = \lambda_r \boldsymbol{X}_r^{(n)} \{\boldsymbol{a}_r\}_{\odot_{-n}} - \lambda_r^2 \boldsymbol{a}_r^{(n),k} + \alpha_n \operatorname{Sgn}(\boldsymbol{a}_r^{(n),k}), \\ \boldsymbol{a}_r^{(n),k+1} & = \boldsymbol{a}_r^{(n),k} - \gamma \boldsymbol{b}^k, \\ \boldsymbol{a}_r^{(n),k+1} & = \boldsymbol{a}_r^{(n),k+1} / \|\boldsymbol{a}_r^{(n),k+1}\|_2. \end{cases}$
        end
      end
      The parameters $\lambda_r$: $\lambda_r = \langle \mathcal{X}_r, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \rangle$,
      The residue $\mathcal{X}_r$: $\mathcal{X}_r = \mathcal{E} + \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}$,
      The tensor error $\mathcal{E}_\Omega$: $\mathcal{E}_\Omega = [\mathcal{X} - \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}]_\Omega$.
  end
  Step 2: Tensor completion
    $$\mathcal{X} = \mathcal{P}_{\Omega^c}\left(\sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}\right) + \mathcal{P}_\Omega(\mathcal{M}).$$
  Step 3: Rank estimation
    Compute $\nu = \|\mathcal{E}_\Omega\|_F^2$,
    if $\frac{|\nu - \nu_{\text{old}}|}{|\nu|} \leq \epsilon$ then
      update the rank: $R = R + 1$, $\nu_{\text{old}} = \nu$,
      Randomly initialize and normalize to unit length: $\left\{\left\{\boldsymbol{a}_r^{(n)}\right\}_{n=1}^{N}\right\}_{r=1}^{R}$,
      16
      $\lambda_r = \left\langle \mathcal{X}, \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_R^{(N)} \right\rangle$, $\mathcal{E}_\Omega = [\mathcal{E} - \lambda_r \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}]_\Omega$.
  end
until convergence is achieved or $\mathrm{Iter}_{\max}$ is reached;
```

**Table 1** Motion files from CMU used in the experiments

| Motion | File name | Description | Frames | Markers | Frequency |
|---|---|---|---|---|---|
| M1 | 02_01 | Walking | 343 | 41 | 120 |
| M2 | 64_02 | Playing golf: swing | 493 | 44 | 120 |
| M3 | 85_02 | Breakdance: jump twist | 810 | 41 | 120 |

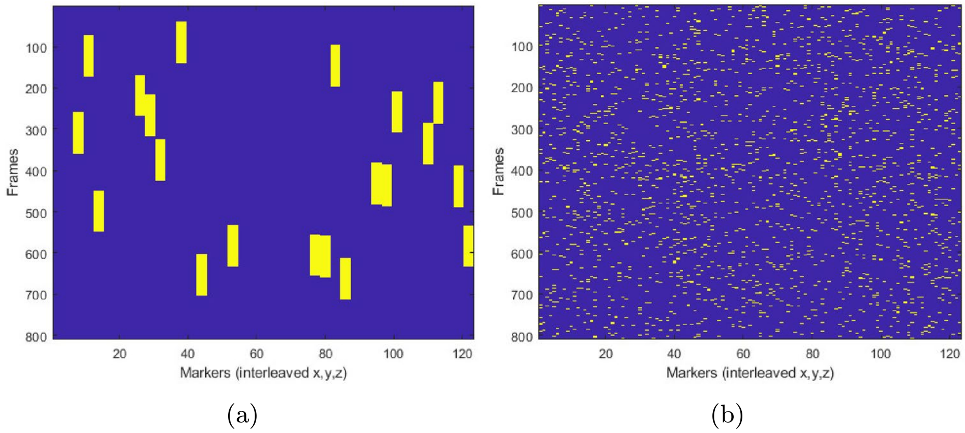


**Fig. 1** Masks: **a** random gaps. **b** Random missing entries (uniformly distributed)

**Table 2** The RMSE of SmoothCP and SparseCP versus different values of $\alpha = [\alpha_1, \alpha_2, \alpha_3]$

| Methods | [.1, .1, .1] | [.01, .01, .01] | [.001, .001, .001] | [.1, .01, 0] | [.1, .01, .01] | [.001, .001, .01] |
|---|---|---|---|---|---|---|
| SmoothCP | 1.991 | 2.341 | 2.369 | 2.639 | 2.182 | 2.354 |
| SparseCP | 2.448 | 2.240 | 2.575 | 2.425 | 2.350 | 2.274 |

**Table 3** The RMSE of SmoothCP and SparseCP versus different values of $\gamma$

| Methods | $\gamma = 10^{-1}$ | $\gamma = 10^{-2}$ | $\gamma = 10^{-3}$ | $\gamma = 10^{-4}$ | $\gamma = 10^{-5}$ | $\gamma = 10^{-6}$ |
|---|---|---|---|---|---|---|
| SmoothCP | 3.120 | 3.000 | 3.012 | 2.971 | 3.341 | 2.907 |
| SparseCP | 2.871 | 3.350 | 2.991 | 3.208 | 3.050 | 3.086 |

## 3.5 Convergence analysis

**The convergence of HALS and HALS-based PGD**

To guarantee the convergence of Algorithm 1, we first need to ensure the convergence of *Algorithm 1.1, 1.2, and 1.3* used to fit the CP, SmoothCP, and SparseCP decompositions respectively. For *Algorithm 1.1*, the HALS algorithm is used to fit the CP decomposition. The HALS is a special case of the BCD algorithm, and its convergence can be obtained from [48, 49]. Briefly, the objective function $F$ in problem (17) is a continuously differentiable and convex function, and each subproblem of (17) has a closed-form solution. Thus, the sequence generated by the HALS algorithm converges to a stationary point, see [49, Proposition 7.2.1]. For both *Algorithm 1.2, and 1.3*, HALS-based PGD is used to fit the SmoothCP and SparseCP decompositions. The HALS-based PGD method can be viewed as the Block Successive Upper bound Minimization (BSUM) algorithm [50], and it converges since (32) and (34) satisfy the assumptions of [50, Theorem 2].

**The convergence of Algorithm 1**

The convergence can be shown using [48, Theorem 4.1(b)]. To do this, first, note that the objective functions in (16) and (27) are the sum of convex and differentiable functions. Thus, they are continuous and have a compact level set, see [49, Proposition B.9], and by [48, Lemma 3.1], are regular. Second, the uniqueness of the solution of subproblems of (17), (32), and (34) is guaranteed by the convergence of HALS and HALS-based PGD algorithms. Therefore, according to [48, Theorem 4.1(b)], every limit point of the sequence generated by each algorithm of Algorithm 1 is a stationary point of its associated optimization problem.

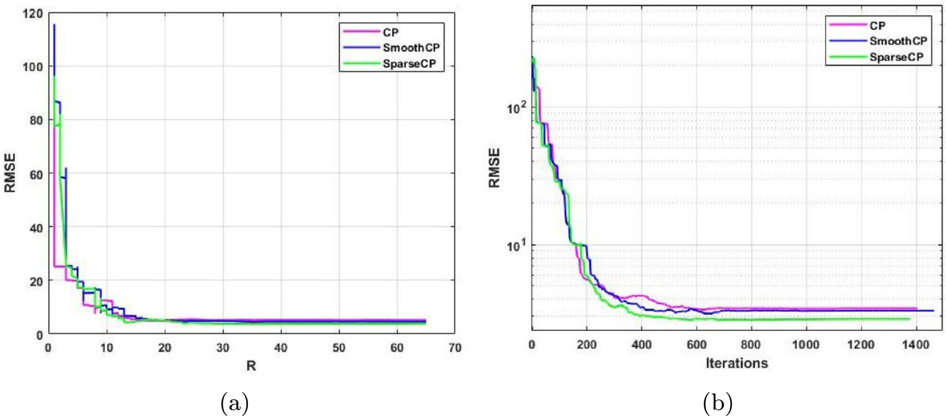


**Fig. 2** **a** The RMSE versus *R*. **b** The convergence of RMSE of the three proposed algorithms

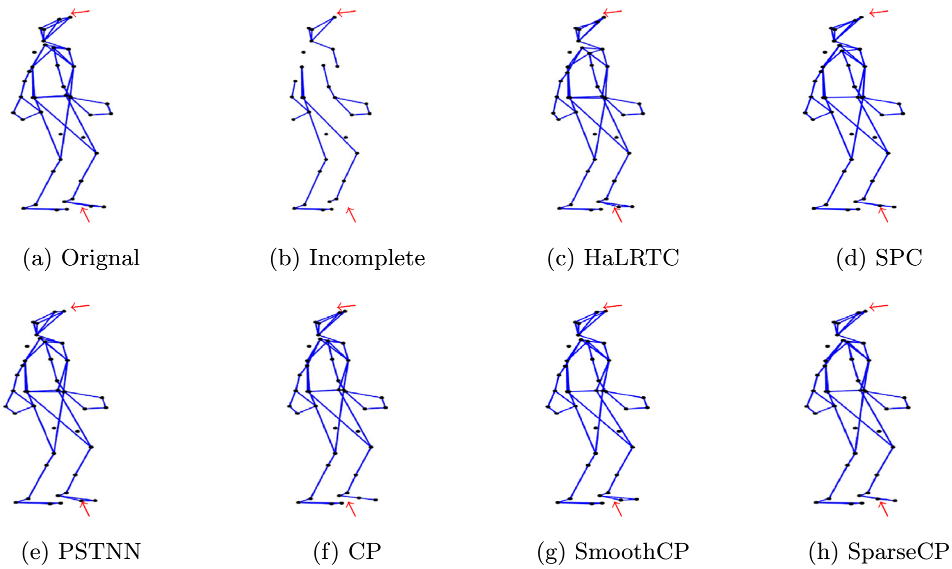


**Fig. 3** The visual comparison of the completion results for the sequence M1 (Walking) with the length of gaps set to 50 and 30 missing markers. Red arrows highlight small inaccuracies in the recovered markers

## 4 Experiments

In this section, we evaluate the results of the three suggested completion algorithms. We first introduce the settings of the experiments, including the used data, the gap-filling setting, the evaluation metrics, and the parameter setting.Then, we illustrate and discuss the results of several tests conducted on different motion sequences. We compare the results of the proposed algorithms with three recently developed state-of-the-art tensor completion methods including the tensor trace norm-based LRTC (HaLRTC) [25], the smooth Parafac (SPC) [28], and the partial sum of the tubal nuclear norm (PSTNN) [51]. The MoCap toolbox[2] was employed for these tests. All experiments were executed in MATLAB (R2018b) on a system equipped with an Intel Core i5-10210U CPU @ 1.60GHz (2.11 GHz) and 8 GB of RAM.

**Data:** The data used in our experiments is from the online CMU human motion database.[3] The database contains various human motions of several categories including, human interaction, physical activities, and sports. We select three motion sequences given in Table 1. These motion sequences cover different types of motion with varying complexity. In all the tests, the MoCap data is represented as a third-order tensor $\mathcal{S} \in \mathbb{R}^{m \times p \times 3}$.

**Gap-filling setting:** The completion of MoCap data involves recovering gaps that depend on factors such as the length of gaps (the number of consecutively missing frames), the number of missing markers, and the complexity of the motion sequence. Thus, the challenge of gap-filling differs significantly from the conventional missing data problem studied in other applications such as image processing as illustrated in Fig. 1. To effectively address the gap-filling issue in MoCap data, it is important to replicate real-world

[2] https://github.com/mocaptoolbox.

[3] http://mocap.cs.cmu.edu/subjects.php.

**Table 4** The comparative evaluation of RMSE, AvE, and STD for completing the motion data M1 (walking) with varied numbers of missing markers and the length of gaps

| Missing markers | Methods | Length of gaps | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 20 | | | 50 | | | 100 | | |
| | | RMSE | AvE | STD | RMSE | AvE | STD | RMSE | AvE | STD |
| | HaLRTC | 1.752 | 1.53e−1 | 1.75 | 3.153 | 4.75e−1 | 3.15 | 4.157 | 8.42e−1 | 3.37 |
| | SPC | 0.748 | 6.86e−2 | 0.74 | 1.799 | 2.53e−1 | 1.79 | 4.648 | 7.59e−1 | 4.64 |
| 10 | PSTNN | **0.434** | **3.96e −2** | **0.43** | **1.008** | **1.41e −1** | **1.00** | *3.083* | *5.59e −1* | *3.07* |
| | CP | *0.733* | *6.72e −2* | *0.73* | 1.618 | 2.46e−1 | 1.61 | 3.378 | 6.59e−1 | 3.37 |
| | SmoothCP | 0.790 | 7.19e−2 | 0.79 | 1.652 | *2.24e −1* | 1.65 | 3.703 | 6.74e−1 | 3.70 |
| | SparseCP | 0.873 | 7.77e−2 | 0.87 | *1.570* | 2.26e−1 | *1.57* | **2.973** | **5.66e −1** | **2.97** |
| | HaLRTC | 1.704 | 2.17e−1 | 1.70 | 3.100 | 6.26e−1 | 3.09 | 9.160 | 2.24 | 9.06 |
| | SPC | 1.338 | 1.71e−1 | 1.33 | 2.417 | 4.68e−1 | 2.41 | 5.739 | 1.60 | 5.73 |
| 20 | PSTNN | **0.711** | **8.46e −2** | **0.71** | **2.212** | **4.32e −1** | **2.21** | *4.581* | *1.18* | *4.57* |
| | CP | 1.152 | 1.46e−1 | 1.15 | *2.249* | *4.32e −1* | *2.24* | 4.583 | **1.27** | 4.58 |
| | SmoothCP | 1.129 | 1.38e−1 | 1.12 | 2.422 | 4.91e−1 | 2.42 | 5.453 | 1.48 | 5.45 |
| | SparseCP | *0.919* | *1.24e −1* | *0.92* | 2.478 | 5.09e−1 | 2.47 | **4.567** | 1.30 | **4.56** |
| | HaLRTC | 2.641 | 3.76e−1 | 2.64 | 4.403 | 9.82e−1 | 4.40 | 15.398 | 4.62 | 15.25 |
| | SPC | 1.410 | 2.09e−1 | 1.41 | *3.393* | 7.94e−1 | 3.38 | 7.969 | 2.75 | 7.96 |
| 30 | PSTNN | **1.130** | **1.50e −2** | **1.13** | **3.258** | **6.79e −1** | **3.25** | 8.294 | 2.49 | 8.28 |
| | CP | 1.530 | 2.26e−1 | 1.53 | 3.396 | *7.87e −1* | *3.38* | 7.859 | 2.69 | 7.85 |
| | SmoothCP | *1.375* | *1.99e −1* | *1.37* | 3.607 | 8.23e−1 | 3.60 | **7.023** | **2.42** | **7.01** |
| | SparseCP | 1.670 | 2.431e−1 | 1.67 | 3.556 | 8.27e−1 | 3.55 | *7.093* | *2.45* | *7.08* |

The results of the six algorithms HaLRTC, SPC, PSTNN, CP, SmoothCP, and SparseCP are provided for each setting. The best two results are highlighted in bold and italic, respectively

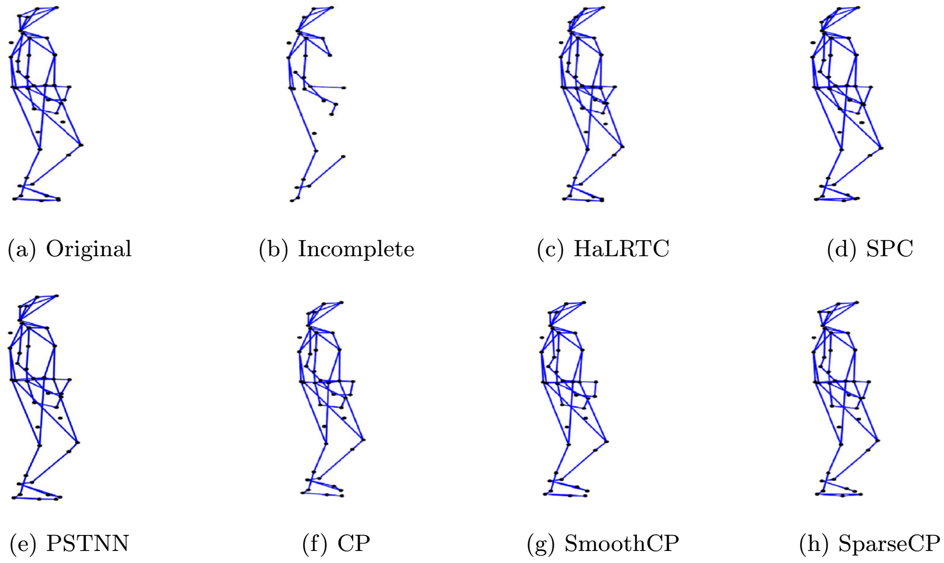

(a) Original (b) Incomplete (c) HaLRTC (d) SPC
(e) PSTNN (f) CP (g) SmoothCP (h) SparseCP

**Fig. 4** The visual comparison of the completion results for the sequence M1 (Walking) with the length of gaps set to 100 and 30 missing markers

complexities accurately. Therefore, we intentionally introduce gaps in multiple markers at random locations. The gaps in MoCap data are represented as 'Not a Number' (NaN) values, making them easily identifiable. We test different scenarios where we increase the missing markers and the gap length to evaluate their impact on the three algorithms.

**Evaluation metrics:** To analyze the performance of each algorithm, we use three basic measures: the Root Mean Square Error (RMSE), the Average Error (AvE), and the Standard Deviation (STD)

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{3}|x_{ijk}-s_{ijk}|^2},$$

$$\mathrm{AvE} = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{3}|x_{ijk}-s_{ijk}|,$$

$$\mathrm{STD} = \sqrt{\frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{3}\left((x_{ijk}-s_{ijk})-\bar{e}\right)^2},$$

$$\text{where } \bar{e} = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{3}(x_{ijk}-s_{ijk}).$$

where $x_{ijk}$ and $s_{ijk}$ are the $(i, j, k)$-th element of the recovered motion sequence $\mathcal{X}$ and the ground truth motion sequence $\mathcal{S}$, respectively, and $n = m \times p \times 3$.

**Parameter setting:** For all algorithms, the number of iterations $\mathrm{Iter}_{\max}$ is fixed at 1500. The algorithms also use the relative change of two successive reconstructed tensors $\|\mathcal{X} - \mathcal{X}^{old}\|/\|\mathcal{X}^{old}\| < \hat{\epsilon}$ with a given tolerance $\hat{\epsilon} = 10^{-6}$ as a stopping rule. In the case of SmoothCP and SparseCP, the number of iterations $K_{\max}$ of the PGD algorithm is set to 1000. The step size $\gamma$ of the gradient descent and the regularization parameters $\alpha_n$ for $n = 1, 2, 3$ are critical for the algorithm's performance. Tables 2 and 3 illustrate the behavior of the SmoothCP and SparseCP algorithms for different values of $\alpha_n$ and $\gamma$.

### 4.1 Results and discussion

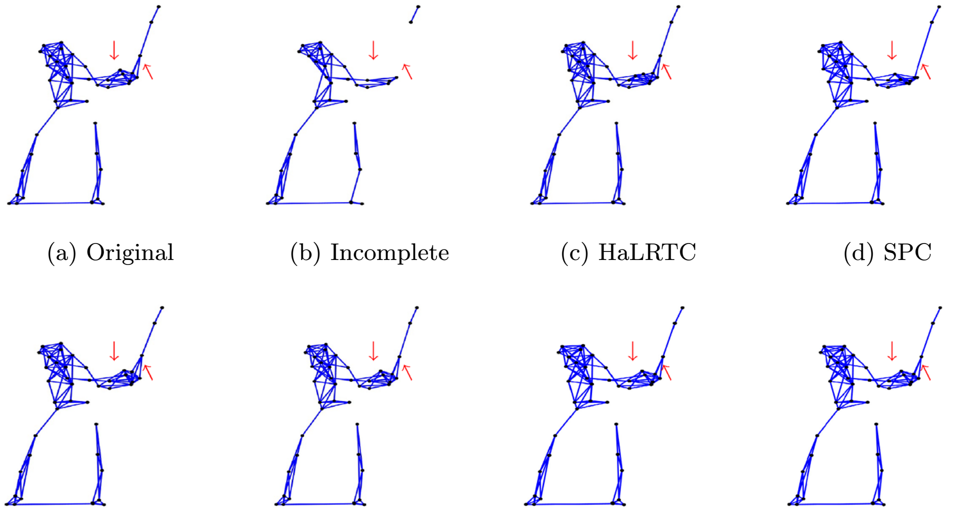
In Tables 4, 5, and 6 we summarize the quantitative results obtained from the completion of motion sequences (M1, M2, M3) by the proposed and the comparative algorithms. We evaluate the performance using RMSE, AvE, and STD metrics across varying lengths of the gaps (20, 50, 100) and numbers of missing markers (10, 20, 30). From the results, we observe that when the number of missing frames is small (gap length=20 or 50), increasing the number of missing markers does not drastically affect the recovery results. In contrast, when increasing the length of gaps, the RMSE, AvE, and STD increase significantly, especially for motions M2 and M3. Increasing the gaps' length and the number of missing markers significantly affects the RMSE, AvE, and STD. The gap-filling problem appears more sensitive to missing markers' duration than quantity.

For all algorithms, the recovery results of the walking sequence (M1) are more robust than those of the playing golf (swing) or breakdance (jump twist) sequences (M2 and M3). This can be explained by the fact that simple motions, like walking, have repetitive and predictable patterns. Breakdance, on the other hand, tends to be more complex and challenging to recover due to its dynamic nature and sudden changes in direction and speed. These dynamic movements can result in more jerky changes in the MoCap data. Thus, it is important to note that the performance of the completion process depends on motion complexity.

When comparing the performance of the six algorithms, the PSTNN algorithm demonstrates robust reconstruction results across all three motions for short gap lengths, with motion M1 exhibiting particularly favorable outcomes. This effectiveness can be attributed to PSTNN's approach of directly minimizing the partial sum of the tubal multi-rank, which proves more precise in handling missing markers with short durations. However, as gap lengths increase, CP-based algorithms show superior performance over the tensor rank minimization method. For instance, SparseCP and SmoothCP consistently yield more accurate results when gap lengths extend to 100 and 30 missing markers across all three motions.

When comparing the three proposed algorithms, the CP algorithm shows favorable results for walking motion M1 and when the gap lengths are relatively small for motions M2 and M3. Furthermore, the modified CP algorithms demonstrate better performance, particularly for more complex motions (M2 and M3). For motion M2, the SparseCP model exhibits lower RMSE, AvE, and STD, especially as the number of gaps increases. In the case of motion M3, there is no clear distinction in performance between the two algorithms. Therefore, we conclude that the CP algorithm effectively captures the low-rank property of the data and reconstructs the missing markers for regular motions. Conversely, for more varied and complex motions, the modified CP decompositions handle irregularities more accurately. Thus, we can conclude that incorporating additional priors over the CP decomposition is valuable and significantly impacts the accuracy of data completion.

Additionally, we conduct a visual comparison of the completion results of the six algorithms in Figs. 3, 4, 5, 6, 7, and 8 for the three motion sequences (M1, M2, M3), respectively. We present the recovered results of different frames of each motion sequence with 30 missing markers and (50, and 100) consecutive missing frames, respectively. We



(a) Original (b) Incomplete (c) HaLRTC (d) SPC

(e) PSTNN (f) CP (g) SmoothCP (h) SparseCP

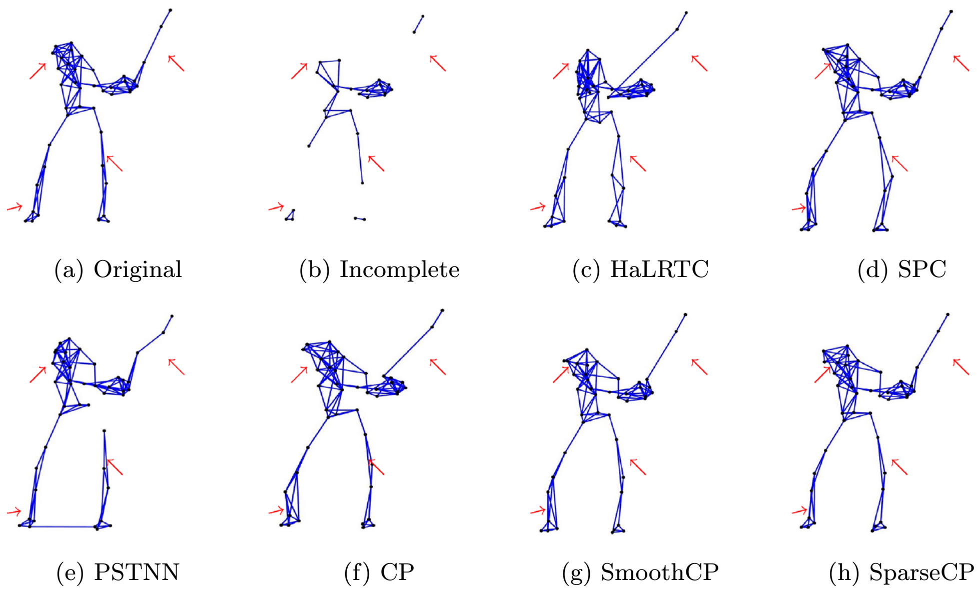
**Fig. 5** The visual comparison of the completion results for the sequence M2 (Playing golf) with the length of gaps set to 50 with 30 missing markers. Red arrows highlight inaccuracies in the recovered markers

**Table 5** The comparative evaluation of RMSE, AvE, and STD for completing the motion data M2 (Playing golf) with varied numbers of missing markers and length of gaps

| Missing markers | Methods | Length of gaps | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 20 | | | 50 | | | 100 | | |
| | | RMSE | AvE | STD | RMSE | AvE | STD | RMSE | AvE | STD |
| 10 | HaLRTC | 1.354 | 8.97e−2 | 1.35 | 2.889 | 2.61e−1 | 2.88 | **7.482** | *8.86e −1* | **7.47** |
| | SPC | 0.692 | 4.26e−2 | 0.69 | *2.266* | *1.92e −1* | *2.26* | 8.936 | 9.61e−1 | 8.93 |
| | PSTNN | **0.122** | **8.54e −3** | **0.12** | **2.225** | **1.76e −1** | **2.22** | 8.762 | 8.70e−1 | 8.75 |
| | CP | 0.639 | 3.61e−2 | 0.63 | 2.950 | 2.39e−1 | 2.95 | 10.503 | 1.43 | 15.03 |
| | SmoothCP | 0.673 | 3.88e−2 | 0.67 | 2.604 | 2.32e−1 | 2.60 | 9.652 | 1.04 | 9.65 |
| | SparseCP | *0.539* | *3.20e −2* | *0.53* | 2.363 | 1.94e−1 | 2.36 | *7.529* | **8.21e −1** | *7.52* |
| 20 | HaLRTC | 1.071 | 1.00e−1 | 1.07 | 3.213 | 4.17e−1 | 3.21 | 29.719 | 4.96 | 29.62 |
| | SPC | 0.810 | 27.31e−2 | 0.81 | *2.853* | *3.55e −1* | *2.85* | **10.587** | **1.81** | **10.58** |
| | PSTNN | **0.349** | **2.95e −2** | **0.34** | 3.139 | 3.25e−1 | 3.13 | 21.628 | 3.82 | 21.54 |
| | CP | 0.767 | 6.09e−2 | 0.67 | 2.929 | 3.67e−1 | 2.92 | 15.484 | 2.03 | 15.48 |
| | SmoothCP | 0.693 | 5.72e−2 | 0.69 | 3.435 | 4.21e−1 | 3.43 | *12.523* | *1.95* | *12.51* |
| | SparseCP | *0.692* | *6.08e −2* | *0.69* | **2.760** | **3.37e −1** | **2.76** | 13.109 | 2.03 | 13.10 |
| 30 | HaLRTC | 2.567 | 2.12e−1 | 2.56 | 11.749 | 1.59 | 11.73 | 39.634 | 8.39 | 39.47 |
| | SPC | 2.553 | 2.07e−1 | 2.55 | 10.610 | 1.48 | 10.60 | 22.890 | 4.71 | 22.88 |
| | PSTNN | **0.793** | **6.61e −2** | **0.79** | 10.282 | 1.31 | 10.28 | 33.435 | 22.42 | 33.43 |
| | CP | *2.302* | *1.77e −1* | *2.30* | 12.544 | 1.63 | 12.54 | 26.490 | 5.47 | 26.49 |
| | SmoothCP | 2.994 | 2.20e−1 | 2.99 | *10.095* | **1.41** | *10.09* | **18.516** | **3.97** | **18.50** |
| | SparseCP | 2.425 | 1.94e−1 | 2.24 | **10.068** | *1.42* | **10.06** | *22.800* | *4.36* | *22.79* |

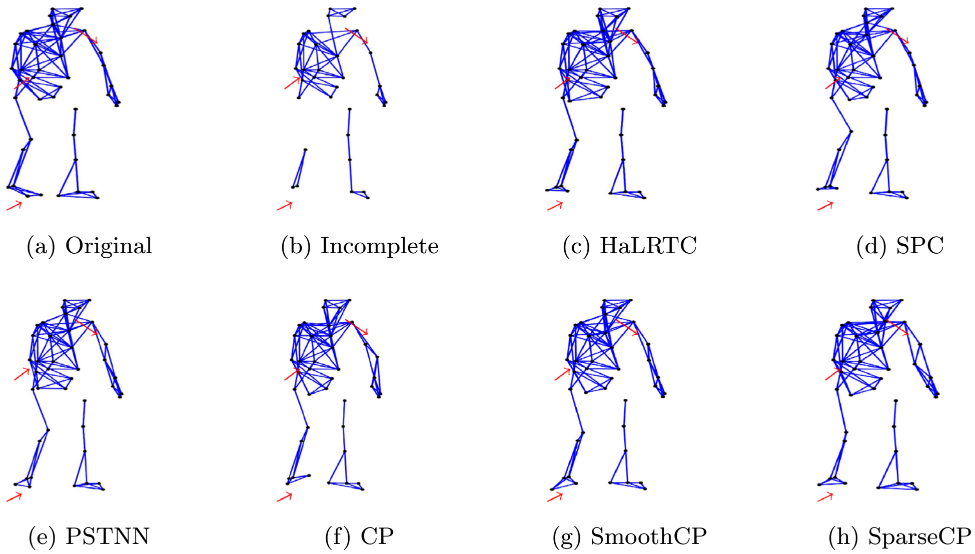
The results of the six algorithms HaLRTC, SPC, PSTNN, CP, SmoothCP, and SparseCP are provided for each setting. The best two results are highlighted in bold and italic, respectively

highlight inaccurately recovered markers using red arrows. As expected, the six algorithms perform well in recovering the missing markers for motion M1, as shown in Fig. 3. However, for motion sequences, (M2, M3), which involve more inconsistent movements, large errors in the recovered markers occur when the length of gaps is high. Therefore, certain motion details are inaccurately predicted especially by HaLRTC and PSTNN, see Figs. 6 and 8. Further, to better


(a) Original (b) Incomplete (c) HaLRTC (d) SPC

(e) PSTNN (f) CP (g) SmoothCP (h) SparseCP

**Fig. 6** The visual comparison of the completion results for the sequence M2 (Playing golf) with the length of gaps set to 100 with 30 missing markers. Red arrows highlight inaccuracies in the recovered markers

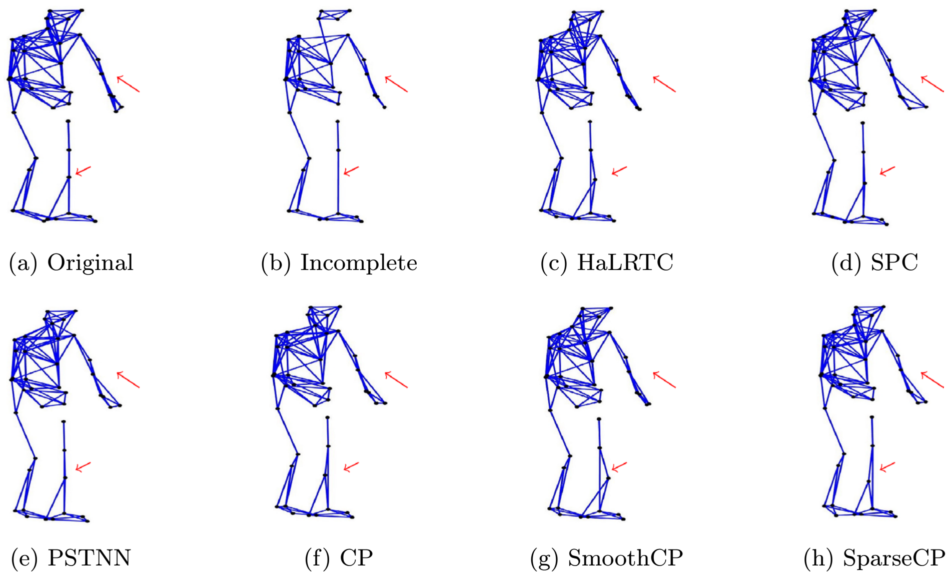

(a) Original (b) Incomplete (c) HaLRTC (d) SPC

(e) PSTNN (f) CP (g) SmoothCP (h) SparseCP

**Fig. 7** The visual comparison of the completion results for the sequence M3 (Breakdance) with the length of gaps set to 50 and 30 missing markers. Red arrows highlight inaccuracies in the recovered markers

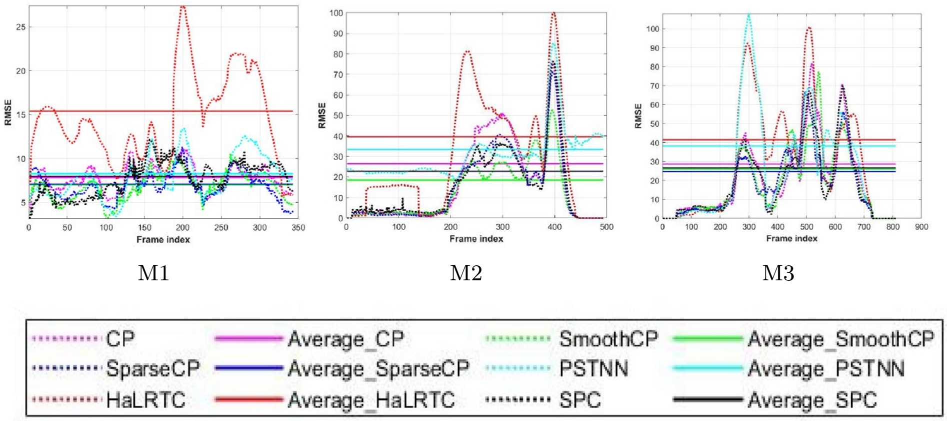
illustrate the performance of our completion algorithms compared with the other methods, we illustrate in Fig. 9 the results obtained from the plots of RMSE versus the frame indices of the three motions. In this experiment, we introduce 30 missing markers and gaps of length 100. The curves depicting RMSE across all frames demonstrate the effectiveness of the proposed algorithms in handling large missing gaps. For example, in the second plot (M2), SmoothCP consistently exhibits low RMSE values across all frames.

Figure 2a presents the plots of RMSE of the three CP-based decompositions versus $R$ and the estimation of rank R versus iterations. The plots reveal that small values of $R$ are insufficient to accurately fit the decomposition, resulting in notably high RMSE values. This is because a low $R$ restricts the model's capacity to capture the underlying structure of the data. To overcome this issue, the experiments were conducted with a very large $R_{max}$. This large initial value of $R_{max}$ ensures that the criterion (36) controls


(a) Original (b) Incomplete (c) HaLRTC (d) SPC

(e) PSTNN (f) CP (g) SmoothCP (h) SparseCP

**Fig. 8** The visual comparison of the completion results for the sequence M3 (Breakdance) with the length of gaps set to 100 and 30 missing markers. Red arrows highlight inaccuracies in the recovered markers



M1 M2 M3

**Fig. 9** The RMSE values of all recovered frames and their average (Average_HaLRTC, Average_SPC, Average_PSTN, Average_CP, Average_SmoothCP, Average_SparseCP) for three MoCap data obtained by the six algorithms. The gap length is set to 100 with 30 missing markers. From left to right, the recovered results are for M1, M2, and M3, respectively

the rank estimation process, incrementally adjusting the rank until the method converges. We illustrate in Fig. 2b the convergence behavior of three algorithms based on RMSE values across iterations. We observe that the RMSE keeps decreasing as the iteration number increases, which demonstrates the numerical stability and the convergence of the proposed algorithm.

Table 7 provides the running times of the six algorithms for recovering the three motion sequences. Notably, the HaLRTC algorithm demonstrates the most efficient performance, emerging as the fastest option among the tested methods. Following HaLRTC, the PSTNN algorithm also exhibits commendable speed. Regarding the CP-based algorithms, the CP algorithm itself shows a relatively quick execution time, and the SparseCP algorithm closely follows with comparable running times.

## 5 Conclusion

This study expands the application scope of tensor completion techniques and offers promising avenues for enhancing data quality in MoCap systems. We address the gap-filling problem in MoCap data by introducing three completion algorithms based on the tensor CP decomposition: the CP, SmoothCP, and SparseCP algorithms. The experimental results demonstrate the significant impact of incorporating priors in the CP decomposition process. Moreover, the results reveal that the SmoothCP and SparseCP algorithms, outperform or achieve comparable results to existing state-of-the-art tensor completion methods. For instance, while the PSTNN method excels in accurately completing small gaps, SmoothCP and SparseCP demonstrate superior accuracy in handling larger gaps, especially

**Table 6** The comparative evaluation of RMSE, AvE, and STD for completing the motion data M3 (Breakdance) with varied numbers of missing markers and length of gaps

| Missing markers | Methods | Length of gaps | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 20 | | | 50 | | | 100 | | |
| | | RMSE | AvE | STD | RMSE | AvE | STD | RMSE | AvE | STD |
| 10 | HaLRTC | *1.440* | *9.22e −2* | *1.43* | **4.484** | **3.48e −1** | **4.47** | 13.204 | 1.37 | 13.19 |
| | SPC | 2.291 | 1.25e−1 | 2.29 | 5.219 | 4.06e−1 | 5.21 | 14.295 | 1.51 | 14.28 |
| | PSTNN | **1.205** | **6.24e −2** | **1.20** | *5.210* | *3.92e −1* | *5.20* | 13.308 | 1.38 | 13.29 |
| | CP | 2.013 | 1.237e−1 | 2.01 | 6.924 | 4.91-1 | 6.91 | 14.989 | 1.56 | 14.98 |
| | SmoothCP | 1.687 | 9.36e−2 | 1.68 | 5.893 | 4.40e−1 | 5.89 | **9.309** | **1.13** | **9.30** |
| | SparseCP | 1.749 | 9.25e−2 | 1.74 | 5.467 | 4.24e−1 | 5.467 | *11.463* | *1.11* | *11.46* |
| 20 | HaLRTC | *1.990* | *1.75e −1* | *1.99* | *7.127* | **7.51e −1** | *7.11* | 30.693 | 4.26 | 30.63 |
| | SPC | 2.214 | 1.87e−1 | 2.21 | 8.120 | 8.87e−1 | 8.11 | 19.356 | 2.72 | 19.35 |
| | PSTNN | **1.678** | **1.22e −1** | **1.67** | 10.071 | 1.03 | 10.05 | 21.307 | 3.11 | 21.27 |
| | CP | 2.481 | 2.00e−1 | 2.48 | 8.313 | 9.09e−1 | 8.11 | 22.944 | 3.12 | 22.93 |
| | SmoothCP | 2.728 | 2.15e−1 | 2.72 | **7.042** | *7.87e −1* | **7.04** | *18.052* | *2.71* | *18.04* |
| | SparseCP | 2.729 | 2.08e−1 | 2.48 | 8.324 | 9.55e−1 | 8.32 | **16.604** | **2.57** | **16.60** |
| 30 | HaLRTC | *2.555* | *3.24e −1* | *2.55* | *9.762* | **1.33** | *9.75* | 41.528 | 7.61 | 41.44 |
| | SPC | 3.324 | 3.26e−1 | 3.32 | 11.835 | 1.51 | 11.83 | 26.635 | 5.09 | 26.63 |
| | PSTNN | **2.550** | **2.28e −1** | **2.55** | 12.472 | 1.70 | 12.46 | 38.151 | 6.84 | 38.04 |
| | CP | 3.536 | 3.48e−1 | 3.32 | 12.508 | 1.62 | 12.50 | 28.738 | 5.34 | 28.70 |
| | SmoothCP | 3.681 | 3.45e−1 | 3.68 | 10.885 | 1.58 | 10.88 | *25.877* | *4.91* | *25.85* |
| | SparseCP | 3.177 | 3.28e−1 | 3.17 | **9.547** | *1.39* | **9.54** | **24.748** | **4.82** | **24.74** |

The results of the six algorithms HaLRTC, SPC, PSTNN, CP, SmoothCP, and SparseCP are provided for each setting. The best two results are highlighted in bold and italic, respectively

**Table 7** The CPU time (in min) of the six algorithms for (M1, M2, M3) sequences

| Motion | Number of frames | HaLRTC | SCP | PSTNN | CP | SparseCP | SmoothCP |
|---|---|---|---|---|---|---|---|
| M1 | 343 | 0.002 | 0.951 | 0.018 | 0.557 | 0.927 | 4.775 |
| M2 | 493 | 0.002 | 1.371 | 0.022 | 0.917 | 1.328 | 7.335 |
| M3 | 810 | 0.005 | 2.102 | 0.031 | 1.175 | 1.709 | 17.639 |

for more complex data scenarios. However, the current algorithms primarily focus on single-subject scenarios and may not perform as well in multi-subject environments. In future work, we aim to extend the proposed algorithms to more challenging scenarios involving multiple individuals, such as couples dancing. Additionally, we will explore applying multiple regularization techniques on the tensor decomposition factors to enhance performance in these more challenging contexts.

**Funding** Open access funding provided by Örebro University. Carl Tryggers Stiftelse supported the first author's work under Grant CTS 22:2196. The Swedish Research Council (VR) supports the second author's work under Grant 2021-05393.

## Declarations

**Conflict of interest** The authors have no conflict of interest.

## References

1. Zhou H, Huosheng H (2008) Human motion tracking for rehabilitation: a survey. Biomed Signal Process Control 3(1):1–18
2. Takeda I, Yamada A, Onodera H (2021) Artificial intelligence-assisted motion capture for medical applications: a comparative study between markerless and passive marker motion capture. Comput Methods Biomech Biomed Eng 24(8):864–873
3. Van der Kruk E, Reijne MM (2018) Accuracy of human motion capture systems for sport applications; state-of-the-art review. Eur J Sport Sci 18(6):806–819
4. Millour G, Velásquez AT, Domingue F (2023) A literature overview of modern biomechanical-based technologies for bike-fitting professionals and coaches. Int J Sports Sci Coach 18(1):292–303
5. Field M et al (2011) Human motion capture sensors and analysis in robotics. Ind Robot Int J 38(2):163–171
6. Schreiter T et al (2022) The magni human motion dataset: accurate, complex, multi-modal, natural, semantically-rich and contextualized. In: arXiv preprint arXiv:2208.14925
7. Schreiter T et al (2024) THÖR-MAGNI: a large-scale indoor motion capture recording of human movement and robot interaction. arXiv preprint arXiv:2403.09285
8. Boschetti G et al (2023) 3D collision avoidance strategy and performance evaluation for human–robot collaborative systems. Comput Ind Eng 179:109225
9. Candes E, Recht B (2012) Exact matrix completion via convex optimization. Commun ACM 55(6):111–119
10. Cai J-F, Candès EJ, Shen Z (2010) A singular value thresholding algorithm for matrix completion. SIAM J Optim 20(4):1956–1982
11. Lai Ranch YQ, Yuen Pong C, Lee Kelvin KW (2011) Motion capture data completion and denoising by singular value thresholding. In: Eurographics (short papers), pp 45–48
12. Tan C-H, Hou J, Chau L-P (2013) Human motion capture data recovery using trajectory-based matrix completion. Electron Lett 49(12):752–754
13. Tan C-H, Hou JH, Chau L-P (2015) Motion capture data recovery using skeleton constrained singular value thresholding. Vis Comput 31:1521–1532
14. Chen B et al (2018) Human motion recovery utilizing truncated schatten p-norm and kinematic constraints. Inf Sci 450:89–108
15. Feng Y et al (2014) Exploiting temporal stability and low-rank structure for motion capture data refinement. Inf Sci 277:777–793
16. Wenyu H et al (2017) Motion capture data completion via truncated nuclear norm regularization. IEEE Signal Process Lett 25(2):258–262
17. Yang J et al (2019) Spatio-temporal reconstruction for 3D motion recovery. IEEE Trans Circuits Syst Video Technol 30(6):1583–1596
18. Cui Q, Chen B, Sun H (2019) Nonlocal low-rank regularization for human motion recovery based on similarity analysis. Inf Sci 493:57–74
19. Peng S-J et al (2015) Hierarchical block-based incomplete human mocap data recovery using adaptive nonnegative matrix factorization. Comput Graph 49:10–23
20. Xia G et al (2018) Nonlinear low-rank matrix completion for human motion recovery. IEEE Trans Image Process 27(6):3011–3024
21. Douglas Carroll J, Chang J-J (1970) Analysis of individual differences in multidimensional scaling via an N-way generalization of “Eckart-Young" decomposition. Psychometrika 35(3):283–319
22. Tucker LR (1966) Some mathematical notes on three-mode factor analysis. Psychometrika 31(3):279–311
23. Oseledets IV (2011) Tensor-train decomposition. SIAM J Sci Comput 33(5):2295–2317
24. Zhao Q et al (2016) Tensor ring decomposition. arXiv preprint arXiv:1606.05535
25. Liu J et al (2012) Tensor completion for estimating missing values in visual data. IEEE Trans Pattern Anal Mach Intell 35(1):208–220
26. Bengua JA et al (2017) Efficient tensor completion for color image and video recovery: low-rank tensor train. IEEE Trans Image Process 26(5):2466–2479
27. Zhao Q, Zhang L, Cichocki A (2015) Bayesian CP factorization of incomplete tensors with automatic rank determination. IEEE Trans Pattern Anal Mach Intell 37(9):1751–1763
28. Yokota T, Zhao Q, Cichocki A (2016) Smooth parafac decomposition for tensor completion. IEEE Trans Signal Process 64(20):5423–5436
29. Heidel G, Schulz V (2018) A Riemannian trust-region method for low-rank tensor completion. Numer Linear Algebra Appl 25(6):e2175
30. Liu X-Y et al (2019) Low-tubal-rank tensor completion using alternating minimization. IEEE Trans Inf Theory 66(3):1714–1737
31. Zhou P et al (2017) Tensor factorization for low-rank tensor completion. IEEE Trans Image Process 27(3):1152–1163
32. Wang W, Aggarwal V, Aeron S (2017) Efficient low-rank tensor ring completion. In: Proceedings of the IEEE international conference on computer vision, pp 5697–5705
33. Chen Y-L, Hsu C-T, Liao H-YM (2013) Simultaneous tensor decomposition and completion using factor priors. IEEE Trans Pattern Anal Mach Intell 36(3):577–591
34. Yuankai W et al (2018) A fused CP factorization method for incomplete tensors. IEEE Trans Neural Netw Learn Syst 30(3):751–764
35. Madathil B, George SN (2018) Twist tensor total variation regularized-reweighted nuclear norm-based tensor completion for video missing area recovery. Inf Sci 423:376–397
36. Mohaoui S, Hakim A, Raghay S (2022) Parallel matrix factorization-based collaborative sparsity and smooth prior for estimating missing values in multidimensional data. Pattern Anal Appl 25(4):963–980
37. Zhao X-L et al (2021) Tensor completion via complementary global, local, and nonlocal priors. IEEE Trans Image Process 31:984–999
38. Bazerque JA, Mateos G, Giannakis GB (2013) Rank regularization and Bayesian inference for tensor completion and extrapolation. IEEE Trans Signal Process 61(22):5689–5703
39. Kolda TG, Bader BW (2009) Tensor decompositions and applications. SIAM Rev 51(3):455–500
40. Comon P, Luciani X, De Almeida ALF (2009) Tensor decompositions, alternating least squares and other tales. J Chemomet J Chemomet Soc 23(7–8):393–405
41. Uschmajew A (2012) Local convergence of the alternating least squares algorithm for canonical tensor approximation. SIAM J Matrix Anal Appl 33(2):639–652
42. Cichocki A, Zdunek R, Amari S (2007) Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. In:

International conference on independent component analysis and signal separation, vol 1(1). Springer, pp 169–176

43. Kim J, He Y, Park H (2014) Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework. J Glob Optim 58:285–319
44. Cichocki A, Phan A-H (2009) Fast local algorithms for large scale nonnegative matrix and tensor factorizations. IEICE Trans Fundam Electron Commun Comput Sci 92(3):708–721
45. Tian X, Xie K, Zhang H (2022) A low-rank tensor decomposition model with factors prior and total variation for impulsive noise removal. IEEE Trans Image Process 31:4776–4789
46. Sedighin F, Cichocki A, Phan A-H (2021) Adaptive rank selection for tensor ring decomposition. IEEE J Sel Top Signal Process 15(3):454–463
47. Yangyang X et al (2015) Parallel matrix factorization for low-rank tensor completion. Inverse Probl Imaging 9(2):601–624
48. Tseng P (2001) Convergence of a block coordinate descent method for nondifferentiable minimization. J Optim Theory Appl 109:475–494
49. Bertsekas DP (1997) Nonlinear programming. J Oper Res Soc 48(3):334–334
50. Razaviyayn M, Hong M, Luo Z-Q (2013) A unified convergence analysis of block successive minimization methods for nonsmooth optimization. SIAM J Optim 23(2):1126–1153
51. Jiang T-X et al (2020) Multi-dimensional imaging data recovery via minimizing the partial sum of tubal nuclear norm. J Comput Appl Math 372:112680

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.