Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2012)
P. Kry and J. Lee (Editors)

Quaternion Space Sparse Decomposition for Motion Compression and Retrieval

Mingyang Zhu[†][1,2], Huaijiang Sun[‡][1] and Zhigang Deng[§][2]

[1] Nanjing University of Science and Technology, China
[2] University of Houston, Houston, Texas, United States

Abstract

Quaternion has become one of the most widely used representations for rotational transformations in 3D graphics for decades. Due to the sparse nature of human motion in both the spatial domain and the temporal domain, an unexplored yet challenging research problem is how to directly represent intrinsically sparse human motion data in quaternion space. In this paper we propose a novel quaternion space sparse decomposition (QSSD) model that decomposes human rotational motion data into two meaningful parts (namely, the dictionary part and the weight part) with the sparseness constraint on the weight part. Specifically, a linear combination (addition) operation in Euclidean space is equivalently modeled as a quaternion multiplication operation, and the weight of linear combination is modeled as a power operation on quaternion. Besides validations of the robustness, convergence, and accuracy of the QSSD model, we also demonstrate its two selected applications: human motion data compression and content-based human motion retrieval. Through numerous experiments and quantitative comparisons, we demonstrate that the QSSD-based approaches can soundly outperform existing state-of-the-art human motion compression and retrieval approaches.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Human motion has become a hot subject of study in numerous fields for decades including, but not limited to: animation, computer vision , HCI, biomechanics, etc. For example, in computer animation community, researchers and industry practitioners use various motion capture techniques to acquire natural and subtle human motion with high fidelity and further process it for a variety of animation applications.

Concretely, human motion is often computationally represented as the combination of the root information (trajectory and orientations) and rotations of the hierarchically constructed, human skeleton joints. Besides 3D transformation matrices, Quaternion that was introduced into graphics community decades ago by Shoemake [Sho85], has become one of the most widely used representations for rotational transformations in 3D graphics. On the other hand, human motion is sparse by its nature in both the spatial domain and the temporal domain [LFAJ10]. Existing sparse coding techniques approximate or reconstruct the original data through sparse linear combination. Thus, besides their linear nature, they also implicitly assume that the data are in Euclidean space. By contrast, joint rotations in the human motion data, represented as transformation matrices or quaternions, are not in Euclidean space; therefore, naively applying existing sparse coding techniques to process rotational motion data would lead to less optimal outcomes or even strange results [BJJ03].

Therefore, an unexplored yet challenging research problem is *how to directly represent intrinsically sparse human motion data in quaternion space*. Indeed, due to the wide use and fundamental importance of quaternion in 3D graphics, a sparse representation of human motion in quaternion space (instead of the traditional Euclidean space) would see

† zmy8475@gmail.com
‡ sunhuaijiang@njust.edu.cn
§ zdeng@cs.uh.edu

DOI: 10.2312/SCA/SCA12/183-192

its wide use in many graphics and animation applications including but not limited to human motion compression and content-based human motion retrieval. Inspired by the need and challenge, in this paper we propose a novel quaternion space sparse decomposition (QSSD) model that decomposes rotational human motion data into two meaningful parts (namely, the *dictionary* part and the *weight* (or called *coefficient*)) part with the sparseness constraint on the weight part. Specifically, in our QSSD model, a linear combination (addition) operation in Euclidean space is equivalently modeled as a quaternion multiplication operation, and the weight of linear combination is modeled as a power operation on quaternion. Through extensive validations (via both simulation data and real human motion capture data) as well as direct comparisons with the original K-SVD algorithm, we validate the robustness, convergence, and accuracy of the introduced QSSD model.

To demonstrate the usefulness and effectiveness of the QSSD model, without loss of generality, we apply it to two selected applications: *human motion data compression* and *content-based human motion retrieval*. Through numerous quantitative comparisons and qualitative evaluation experiments, we demonstrate that the QSSD-based approaches can soundly outperform existing state-of-the-art human motion compression and retrieval approaches as well as the approaches that are based on the original K-SVD algorithm [AEB06].

The main contributions of this work are: (1) It introduces a novel sparse decomposition (called QSSD) to encode human motion data by directly decomposing the rotational motion data in quaternion space, instead of the traditional Euclidean space, while minimizing the quaternion space reconstruction error with the L_0 sparseness constraint. (2) The usefulness and effectiveness of the proposed QSSD model are demonstrated and validated via two selected applications: human motion compression and retrieval.

2. Related Work

Sparse representation has gained intense attentions in signal processing and machine learning community. In general, sparse representation techniques decompose given signals into meaningful parts. Its process consists of two steps. The first step is *sparse coding* that calculates corresponding coefficients based on the given signals and a group of basis vectors (called dictionary). This step can be solved by pursuit algorithms such as matching pursuit (MP) [MZ93] and orthogonal matching pursuit (OMP) [CBL89, DMZ94]. These algorithms select dictionary items and compute corresponding coefficients using greedy algorithms. Another well-known pursuit method is the basis pursuit (BP) [CDS01]. Essentially, it replaces L_0 norm with L_1 norm so that the objective function becomes convex. The second step is *dictionary learning* that learns dictionary atoms (i.e., basis vectors) based on the given signals and the sparse coding coefficients. Method of optimal directions (MOD) [ERKD99] typically updates the dictionary as a whole by minimizing the mean squared reconstruction error, which can be formulated as a least squares problem. Another widely used algorithm is K-SVD [AEB06]. Different from MOD, the K-SVD approach updates the dictionary atoms one by one by SVD so that the basis vectors and coefficients are updated together.

Human motion retrieval has been a hot topic in recent years with the increasing availability of large-scale human motion capture databases [CMU11, MRC*07]. Over the years various motion retrieval approaches have been proposed [FRM94, CCW*04, LZWM05, CS11]. For example, Liu *et al.* [LZWP03] cluster human motions based on the extracted keyframes using a hierarchical tree. In this way, each motion can be transformed to a sequence of clusters. Muller and colleagues [MRC05, MR06] proposed to use geometric features that encode the spatial relationship among different joints for motion retrieval. In their method, motion sequences are represented as geometric features and a fuzzy algorithm is employed to retrieve similar motions. In addition, match web structure [KG04], weighted PCA [FF05], and hierarchical motion patterns [DGL09] have also been explored for motion retrieval.

Human motion compression has been studied by many researchers recently [LM06, CBL07, BPvdP07]. For example, Arikan [Ari06] proposed a compression scheme for large motion database. In his work, motion clips are represented as Bezier curves, and the dimensionality of motion data is reduced by clustered PCA with negligible visual loss. Gu *et al.* [GPD09] proposed a pattern indexing scheme for motion capture data compression. They extract motion patterns for different body parts by segmenting and clustering the original motion sequences. Arithmetic coding technique is used to further compress the resultant motion pattern indices. Tournier *et al.* [TWC*09] use a principal geodesics analysis (PGA) based inverse kinematics technique to restore the motion, given five end-joints. In addition, lifting scheme is used for further compressing the end-joint trajectories by omitting n levels of details.

3. Methodology

3.1. Sparse Representation Basics

Sparse representation techniques learn an overcomplete dictionary $D \in \mathbb{R}^{n\times K}$ and a sparse weight matrix $X \in \mathbb{R}^{K\times N}$ that contains no more than L (typically $L \ll K$) non-zero elements in each column to maximally reconstruct the given data $Y \in \mathbb{R}^{n\times N}$, where n denotes the dimension of the data, K denotes the number of dictionary atoms, and N denotes the number of training data samples. Therefore, the objective function of sparse representation problem can be formulated as Eq. (1), where $\|.\|_0$ is the L_0 norm constraint, counting the non-zero entries of a vector. In this writing, we use X_{i*} and X_{*j} to denote the i-th row and j-th column of matrix X, respectively.

$$\min_{D,X} \left\{ \|Y - DX\|_F^2 \right\} \ s.t.\ \forall j, \|X_{*j}\|_0 \leq L \tag{1}$$

Algorithm 1 K-SVD Algorithm

```
Require: $Y \in \mathbb{R}^{n\times N}, L \in \mathbb{N}^+, K \in \mathbb{N}^+$
 1: Initialize $D$
 2: repeat
 3:   /*Sparse coding stage*/
 4:   for $j = 1, 2, \cdots, N$ do
 5:     Compute $X_{*j}$ by any pursuit algorithm to solve:
 6:     $\min_{X_{*j}} \left\{ \|Y_{*j} - DX_{*j}\|_2^2 \right\}\ s.t.\ \|X_{*j}\|_0 \leq L$
 7:   end for
 8:   /*Dictionary learning stage*/
 9:   for $k = 1, 2, \cdots, K$ do
10:     $\Omega = \{i | 1 \leq i \leq N, X_{ki} \neq 0\}$
11:     $E = Y - \sum_{j\neq k} D_{*j} X_{j*}$
12:     $E_{*\Omega} = U \Delta V^T$
13:     $D_{*k} = U_{*1}, X_{k\Omega} = \Delta_{11} V_{*1}$
14:   end for
15: until Convergence OR a maximum of iterations are
    reached
16: return $D$ and $X$
```

Over the years various algorithms have been proposed to solve the above sparse decomposition equation. It has been demonstrated that simultaneously solving D and X is complicated and non-convex [AEB06]; however, solving either D or X while fixing the other is convex. As mentioned in Section 2, in general, there are two stages to solve the sparse representation problem: sparse coding and dictionary learning. Pursuit algorithms [MZ93] can be used for the sparse coding step. The dictionary D can be solved as a whole by a least-square solver [ERKD99] or solved column by column by the K-SVD approach [AEB06]. Since our approach uses a similar dictionary learning scheme as the K-SVD approach [AEB06], we will get into more details about this algorithm in this section. As shown in Algorithm 1, we first compute the overall error E of the examples (i.e., training data), which uses dictionary atom D_{*k} by removing it from the linear combination. Then, SVD is used to decompose E, and D_{*k} and X_{k*} are updated accordingly.

3.2. Quaternion Combination

Quaternion is one of the popular ways to represent rotational transformations [Sho85]. However, since quaternions are not located in Euclidean space, directly applying linear combination in Eq. (1) to quaternions will be less optimal or even meaningless. In this work, we introduce *quaternion combination* to solve the quaternion space sparse decomposition problem.

For convenience, we first introduce some notations and basic knowledge of quaternions. A rotational transformation can be representated as a unit quaternion, and we denote the unit quaternion space as S^3. A quaternion, $q = \left[\cos\frac{\theta}{2}; \mathbf{n}\sin\frac{\theta}{2}\right]$ ($\mathbf{n}$ is a vector), represents a rotation of angle θ about axis $\mathbf{n}$. Its power operation by scalar t is defined as $q^t = \left[\cos\frac{t\theta}{2}; \mathbf{n}\sin\frac{t\theta}{2}\right]$ [DKL98].

If $Q \in \mathbb{R}^{4n\times N}$ (called *quaternions array*) contains $n \times N$ quaternions, we use $\hat{Q}_{ij}$ to denote the i-th quaternion in the j-th column of Q. We also extend various quaternion operations on Q, including $\ln Q$, e^Q, and Q^t (t is a real scalar value), to act on each quaternion in Q. In addition, if $S \in \mathbb{R}^{4n\times N}$ is another quaternions array, we extend quaternion multiplication to QS to denote element-wise multiplication of quaternions between Q and S.

In order to let sparse decomposition algorithm function on quaternions, we also define a meaningful combination between quaternions, d, and a set of weights, x, as follows: $d \otimes x = \hat{d}_1 \odot x_1 \oplus \hat{d}_2 \odot x_2 \oplus \cdots \oplus \hat{d}_m \odot x_m$, where $x \in \mathbb{R}^m$ is the weight vector, $d \in \mathbb{R}^{4m}$ contains m quaternions. According to the linear combination of transformations [Ale02], quaternion power $\hat{d}_i^{x_i}$ is a good candidate for $\hat{d}_i \odot x_i$. The $\oplus$ operation combines two quaternions together to form a new quaternion, that is, combining two rotational transformations. In this work, quaternion multiplication $\hat{d}_i\hat{d}_j$ is used to denote $\hat{d}_i \oplus \hat{d}_j$. Since the quaternion multiplication is non-commutative, the combination order should be given. Eq. (2) summarizes the defined quaternion combination.

$$d \otimes x = \hat{d}_1^{x_{o1}} \hat{d}_2^{x_{o2}} \cdots \hat{d}_m^{x_{om}} \tag{2}$$

Here the combination order, $o1$, $o2$,...,oi,...,om, denotes the order of selected bases in our algorithm. Also, assume $D \in \mathbb{R}^{4n\times K}$ is the quaternion dictionary and each column has n quaternions, and $X \in \mathbb{R}^{K\times N}$ is the weight matrix. In this work, the quaternion combination of D and X is defined in Eq. (3).

$$D \otimes X = \begin{bmatrix} \hat{D}_{11} & \hat{D}_{12} & \cdots & \hat{D}_{1K} \\ \hat{D}_{21} & \hat{D}_{22} & \cdots & \hat{D}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{D}_{n1} & \hat{D}_{n2} & \cdots & \hat{D}_{nK} \end{bmatrix} \otimes \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1N} \\ X_{21} & X_{22} & \cdots & X_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ X_{K1} & X_{K2} & \cdots & X_{KN} \end{bmatrix}$$
$$= \begin{bmatrix} \hat{D}_{1*} \otimes X_{*1} & \hat{D}_{1*} \otimes X_{*2} & \cdots & \hat{D}_{1*} \otimes X_{*N} \\ \hat{D}_{2*} \otimes X_{*1} & \hat{D}_{2*} \otimes X_{*2} & \cdots & \hat{D}_{2*} \otimes X_{*N} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{D}_{n*} \otimes X_{*1} & \hat{D}_{n*} \otimes X_{*2} & \cdots & \hat{D}_{n*} \otimes X_{*N} \end{bmatrix} \tag{3}$$

3.3. QSSD Algorithm

The goal of QSSD is to decompose data matrix $Y \in \mathbb{R}^{4n\times N}$ to quaternion combination $D \otimes X$ with the sparseness constraint on X, and each column of Y contains n quaternions. Typically, the dot product between two unit quaternions is used to to measure their distance [BJJ03]. In order to distinguish quaternion dot product from quaternion multiplication, in this writing, we explicitly use $\cdot$ to denote the dot product of quaternions. Therefore, the cost function of the QSSD problem is defined by the following Eq. (4) (derived from Eq. (1)).

$$\max_{D,X} \sum_{i}^{N} \sum_{j}^{n} \hat{Y}_{ji} \cdot (\hat{D}_{j*} \otimes X_{*i}), s.t. \forall i, \|X_{*i}\|_0 \leq L \tag{4}$$

Our core idea of optimizing the above objective function, Eq. (4), is similar to the K-SVD approach [AEB06]. Basically, it alternately optimizes the two parts, D and X. As such, our algorithm has two main steps: the first is called *quaternion sparse coding* step and the other is *quaternion dictionary learning* step.

In the quaternion sparse coding step, we compute the weight matrix X, by fixing D, via a two-phase protocol: (1) identifying a particular dictionary atom (abbreviated as *atom* in the remaining writing) that can maximally reduce the residual, and the chosen atom is called *the best atom*; (2) adding the chosen best atom to the list and computing corresponding coefficients based on all the selected atoms.

In the first phase, we conduct a logarithm operation on residual vector and each atom, and then calculate the projection between them. To the end, the atom with the largest projection value is chosen as the best atom in this step. After the best atom is selected, we need to compute the coefficient of this atom. Eq. (5) mathematically describes this process, where R denotes the residual that contains n quaternions, d is the selected dictionary atom, and x is the unknown coefficient. After straightforward mathematical derivation, the coefficient x can be obtained by solving a least squares system shown in Eq. (6).

When there is more than one selected atom, we calculate all the coefficients by block coordinate descent algorithm [Ber99]. Basically, the method first divides the set of variables into blocks. Then, the objective function is optimized alternately in an iterative manner. In each iteration, it solves one block while keeping other blocks fixed. In our algorithm, we set each coefficient as a block and the result calculated from Eq. (6) is set as the initial value for the new selected atom. The quaternion sparse coding algorithm (Algorithm 2) repeats until the residual is small enough or the sparseness constraint is satisfied.

$$\begin{gathered}\max_{x} \sum_{i}^{n} \hat{R}_i \cdot \hat{d}_i^x \\ R = [\cos\frac{\theta_1}{2}; \mathbf{u}_1 \cdot \sin\frac{\theta_1}{2}; \cdots; \cos\frac{\theta_n}{2}; \mathbf{u}_n \cdot \sin\frac{\theta_n}{2}] \\ d^x = [\cos\frac{x\phi_1}{2}; \mathbf{v}_1 \cdot \sin\frac{x\phi_1}{2}; \cdots; \cos\frac{x\phi_n}{2}; \mathbf{v}_n \cdot \sin\frac{x\phi_n}{2}]\end{gathered} \tag{5}$$

$$x \cdot \begin{bmatrix} \frac{\phi_1}{2} \\ \vdots \\ \frac{\phi_n}{2} \end{bmatrix} = \begin{bmatrix} \arctan(\mathbf{u}_1 \cdot \mathbf{v}_1 \tan\frac{\theta_1}{2}) \\ \vdots \\ \arctan(\mathbf{u}_n \cdot \mathbf{v}_n \tan\frac{\theta_n}{2}) \end{bmatrix} \tag{6}$$

Algorithm 2 describes the process of this quaternion sparse coding step. It is noteworthy that since the non-commutativity of quaternion multiplication, the selection order of the dictionary atoms matters. However, the dictionary atoms are selected by a greedy strategy, assuming that the coefficients of the selected atoms are ordered in a descending manner. Therefore, we add this constraint as a stopping rule in the block coordinate descent algorithm so that the magnitude of the coefficient of a dictionary atom indicates its order of selection.

Algorithm 2 Quaternion Sparse Coding

```
Require: $Y \in \mathbb{R}^{4n \times N}, L \in \mathbb{N}^+, D \in \mathbb{R}^{4n \times K}$
 1: for $i = 1, 2, \cdots, N$ do
 2:   $R = Y_{*i}$
 3:   repeat
 4:     $proj = (\ln R) \cdot (\ln D)$
 5:     $k = \arg\max\{proj\}$, add $k$ to $\mathcal{P}$
 6:     Compute $X_k$ based on Eq. (6)
 7:     Compute $X_{\mathcal{P}}$ by block coordinate descent algorithm
 8:     $R = Y_{*i}(D_{*\mathcal{P}} \otimes X_{\mathcal{P}})^{-1}$
 9:   until Convergence OR the number of elements in $\mathcal{P}$ $\geq L$
10: end for
11: return $X$
```

We now turn to the second step, process of updating the dictionary together with the non-zero coefficients in quaternion space. Its core idea is to make both X and D fixed, except removing the k-th column in D and its corresponding row in X. Based on the K-1 fixed dictionary atoms and their coefficients, we could easily calculate the residual, denoted as E, but it needs to be careful with left multiplication or right multiplication. Therefore, the optimization problem becomes how to find the closest solution to best approximate E, as described in Eq. (7). Because there is no quaternion multiplication introduced, it equals to the linear system (Eq. (7)). To the end, Eq. (8) can be derived from the objective function of the sub-problem.

We use SVD algorithm to decompose $\ln E = U \Delta V^T$. The first column of U is defined as the solution to $\ln D_{*k}$, which can be transformed back to quaternion space through an exponential operation. The solution to the coefficient vector X_{k*} is defined as the first column of V multiplied by the largest singular value $\Delta(1,1)$. Algorithm 3 shows the process of this quaternion dictionary learning step.

In sum, our QSSD algorithm repeats the above two steps until it converges or a maximum number of iterations are reached. A full description of our algorithm is given in Algorithm 4.

$$E_{*i} \approx D_{*k}^{X_{ki}} \Leftrightarrow \ln E_{*i} \approx (\ln D_{*k}) \cdot X_{ki} \tag{7}$$

$$\min_{D_{*k}, X_{k*}} \|\ln E - (\ln D_{*k}) \cdot X_{k*}\|_F^2 \tag{8}$$

Algorithm 3 Quaternion Dictionary Learning

```
Require: $Y \in \mathbb{R}^{4n\times N}, K \in \mathbb{N}^+, D \in \mathbb{R}^{4n\times K}, X \in \mathbb{R}^{K\times N}$
 1: for $k = 1,2,\cdots,K$ do
 2:     for $j = 1,2,\cdots,N$ do
 3:         $\Phi_1 = \{m|1 \le m \le K, X_{mj} > X_{kj}\}$
 4:         $\Phi_2 = \{m|1 \le m \le K, X_{mj} < X_{kj}\}$
 5:         $E_{*j} = (D_{*\Phi_1} \otimes X_{\Phi_1 j})^{-1} Y (D_{*\Phi_2} \otimes X_{\Phi_2 j})^{-1}$
 6:     end for
 7:     $\Omega = \{i|1 \le i \le N, X_{ki} \ne 0\}$
 8:     $\ln E_{*\Omega} = U\Delta V^T$
 9:     $D_{*k} = e^{U_{*1}}, X_{k\Omega} = \Delta(1,1)V_{*1}$
10: end for
11: return $D, X$
```

Algorithm 4 QSSD Algorithm

```
Require: $Y \in \mathbb{R}^{4n\times N}, L \in \mathbb{N}^+, K \in \mathbb{N}^+$
 1: Initialize quaternion dictionary $D$ by randomly selecting
    columns from $Y$
 2: repeat
 3:     Update $X$ based on Algorithm 2
 4:     Update each column of $D$ and its corresponding row
        in $X$ based on Algorithm 3
 5: until Convergence OR a maximum number of iterations
    are reached
 6: return $D, X$
```

4. Validations of the QSSD Model

4.1. Validation via Simulation Data

We validated the proposed QSSD algorithm via simulation data to test whether it can converge and recover the original dictionary that is used to generate the quaternion dataset. To generate the simulation data, we first randomly generated a $4n \times K$ matrix, where n denotes the number of quaternions in one data sample and K denotes the number of dictionary atoms. We refined D to let each column contain n unit quaternions. D is called the *original simulation dictionary*. A $K \times N$ coefficient matrix X (N is the number of data samples) was generated randomly, and X is a sparse matrix with only L non-zero values in each column. The data samples Y were created by quaternion combination of L different dictionary atoms. As mentioned in section 3.3, the order of quaternion combination in our algorithm is the descending order of the coefficients. To validate this, when we generate Y, we first sort the columns of X in a descending order based on their absolute values, and then combine the corresponding dictionary atoms by quaternion multiplication, by following the column order. In our experiment, we set $n = 20$, $K = 20$, $L = 5$, and $N = 500$.

We applied our QSSD algorithm to the above generated simulation dataset: the dictionary was initialized by randomly selecting 100 samples, and the maximum number of iterations was set to 100. We use Eq. (9) to compute the reconstruction error and its unit is degree. The computed dictionary was compared with the original simulation dictionary by the dictionary distance measure described in [AEB06]. It can be summarized as follows: based on each column of the computed dictionary, the closest column in the original simulation dictionary is identified. Then, L_2 distance is computed to measure the difference/distance between the two columns (i.e., dictionary atoms), as shown in Eq. (10).

$$E = \frac{180.\sum_{i=1}^{N}\sum_{j=1}^{n} |\arccos(\hat{Y}_{ji} \cdot (\hat{D}_{j*} \otimes X_{*i}))|}{\pi} \tag{9}$$

$$DictDist = 1 - |D_{*i} \cdot \tilde{D}_{*i}| \tag{10}$$

where D_{*i} is the recovered/computed dictionary atom and $\tilde{D}_{*i}$ denotes the identified, corresponding dictionary atom in the original simulation dictionary.

We consider one dictionary atom is recovered correctly if its corresponding distance (away from the original simulation dictionary atom) is less than 0.01. In our experiments, *Restoration Ratio* is computed to indicate the ratio of the correctly recovered dictionary atoms. Fig. 1 shows the error curves and restoration ratios of the simulation data with different noise levels (noiseless, 40dB and 100dB). As indicated in this figure, it is clear that the proposed QSSD algorithm not only can converge soundly, but also can robustly recover (i.e., inversely compute) almost all the original simulation dictionary atoms with different levels of noise.

4.2. Validation via Real Human Motion Data

Human motion data preprocessing: At the data preprocessing step, we first choose the reference pose (often the first frame of a motion clip). All the joints' rotations are converted to quaternion representation, and then we compute the angular displacements from the reference pose to all other poses. Note that the same motion data preprocessing step is also applied to the selected applications (refer to Section 5.1 and Section 5.2).

Validation experiments: We carried out several experiments on real human motion capture data to validate the training and test errors. In the experiments, we selected two sets of human motion data from the CMU mocap database [CMU11]. The first dataset (called the walking motion dataset) only contains walk motions with various styles, and the other set (called the mixed motion dataset) contains various motion types including walk, jump, dangle, climb, and tai chi. Given the two human motion datasets, angular displacements are extracted using the above motion data preprocessing procedure. Then, we randomly selected 80% data as the training data and the remaining 20% as the test data.

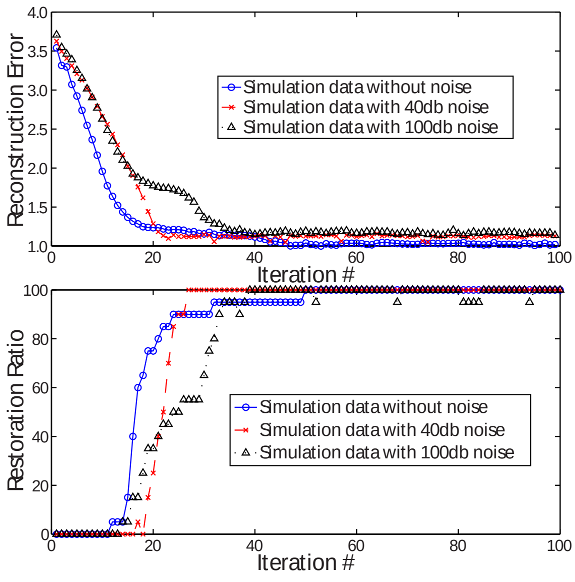


Figure 1: The simulation data based validation results by the proposed QSSD algorithm. (**Top**): The reconstruction error curves of our QSSD algorithm based on the simulation data with different levels of noise (noiseless, 40dB and 100dB). (**Bottom**): The computed restoration ratios by our QSSD algorithm for the same set of simulation data.

All angular displacements are converted to a quaternion array, and we applied both our introduced QSSD algorithm and the K-SVD approach [AEB06] to the same quaternion array. The training data is used to learn the dictionary, and then given the constructed dictionary, we apply the sparse coding step on the test data. In the experiments, the same parameters were used in both our QSSD algorithm and the K-SVD approach. Specifically, we set $L = 15$, $K = 2000$, and set the maximum number of iterations to 50. We ran the procedure 10 times, and at each time different portions of the data were used as the training data and the test data. Table 1 shows the average errors of both the methods on the two datasets. The error is calculated based on Eq. (11), where Y and $\tilde{Y}$ represent the original and reconstructed data, respectively. The error unit is degree per joint per frame.

$$E = \frac{180.\sum_{i=1}^{N}\sum_{j=1}^{n}|\arccos(\hat{Y}_{ji}\cdot\hat{\tilde{Y}}_{ji})|}{nN\pi} \qquad (11)$$

As shown in Table 1, with the same parameter setting, our QSSD algorithm can achieve lower fitting errors than the K-SVD approach (i.e., directly applying the linear combination based sparse representation technique [AEB06] onto quaternion data). The results validate the effectiveness of our QSSD algorithm on real quaternion-represented human motion data.

Since in our QSSD, we replace linear combination with quaternion combination, it runs slower than the K-SVD approach [AEB06]. Table 2 shows the average running time per iteration (the unit is second per iteration) of our QSSD algorithm and the K-SVD approach, with various parameter settings.

Dataset	Walking	Mixed
Training Error (QSSD)	0.0030	0.0052
Test Error (QSSD)	0.0053	0.0105
Training Error (K-SVD)	0.0135	0.0148
Test Error (K-SVD)	0.0271	0.0293

Table 1: Comparison of the average training and test errors by our QSSD algorithm and the K-SVD approach [AEB06] for two different motion datasets.

N	1000				5000			
K	50		100		50		100	
L	5	10	5	10	5	10	5	10
K-SVD	0.8	1.8	0.9	2.0	3.6	9.1	3.7	9.5
QSSD	1.8	3.1	2.1	4.2	6.6	13.3	8.2	14.9

Table 2: Comparison of the average running time (seconds) per iteration by our QSSD algorithm and the K-SVD approach [AEB06] with various parameter settings. The used computer's configuration is: 2.8GHz Intel Core i7 Processor and 4GB memory.

5. Selected Applications

To demonstrate the usefulness and effectiveness of the proposed QSSD model, we apply it to two selected applications: human motion compression and content-based human motion retrieval.

5.1. Human Motion Compression

Our proposed QSSD model can also be directly used for lossy human motion compression. It works as follows: (1) At the compression step, we preprocess human motion data using the same protocol described in Section 4.2 to extract the reference pose, the root information (trajectory and orientations), and all the joints' rotations. Then, the root information and all the joints' rotations are compressed in different ways. Basically, we first employ the multiscale representation algorithm [TWC*09] to initially compress both the root information and all the joints' rotations. After that, we decompose the initially compressed joints' rotations using the QSSD algorithm to obtain the dictionary and weights. Finally, an arithmetic coding technique [Say05] is used to further compress the dictionary and weights, the initially compressed root information, and the reference pose. (2) At the decompression step, the dictionary and weights, as well as the reference pose, are first decoded by the arithmetic coding technique [Say05]. Then, all the joints' rotations are recovered by quaternion combination based on the obtained dictionary and weights, followed by the multiscale representation decompression algorithm [TWC*09]. Finally, the decoded root information is added to recover the original motion sequence. Fig. 2 shows the pipeline of our QSSD-based human motion compression process (decompression is its inverse process).

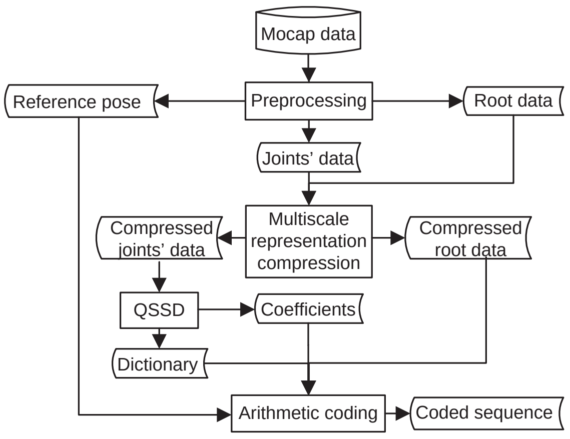


Figure 2: The pipeline of our QSSD-based human motion compression approach

In our experiment, we omitted three levels of details in the multiscale representation step. The parameters used in our QSSD algorithm are: $L = 15$, $K = 2000$, and the maximum number of iterations is set to 50. It is noteworthy that we do not keep the order of dictionary atoms during the compression procedure, but sort them in a descending order during the decompression procedure.

We extracted two test datasets from the CMU motion capture database [CMU11]: the first mixed motion dataset contains various types of motions including jump, climb, walk, dangle, tai-chi, etc (subjects #1, #2, #5, #6, #9, #12 in the CMU mocap database), and the second dataset only contains walk motions with various styles. Table 3 details the statistics of the used test datasets.

A sound error metric is required to quantitatively evaluate the quality of the recovered (decompressed) motion data by any lossy data compression algorithms. In our experiment, we use two error metrics that are employed in previous work: the *ARMS* metric [GPD09] and the distortion rate [Kar04]. For the sake of completeness, the original definitions of the two metrics are described in Eqs. (12) and (13), respectively.

$$ARMS = \frac{\sum_{j=1}^{Mkr\#} \sqrt{(\sum_{i=1}^{Mkr\#} \|P_i - \hat{P}_i\|^2)/Mkr\#}}{Frame\#} \tag{12}$$

$$DistortionRate = 100\frac{\|P - \hat{P}\|}{\|P - E(P)\|} \tag{13}$$

Here P represents the joints' trajectories in the original data, $\hat{P}$ represents the joints' trajectories in the recovered data, and $E(P)$ represents the mean pose. The unit of *ARMS* is centimeter.

A major distinction between our QSSD-based compression approach and existing human motion compression approaches (e.g., the work of [Ari06, GPD09]) is that our approach directly compresses the rotational motion data in quaternion space, while existing compression approaches [Ari06, GPD09] typically compress the markers' 3D trajectory data. In order to perform a fair comparison, in our QSSD-based motion decompression process, we compute the joints' trajectories based on the reconstructed rotation data and then calculate the *ARMS* metric and the distortion rate.

Table 3 shows comparison of the compression results including compression ratio, reconstruction error, and compression/decompression time. Since the same algorithm parameters are used, our QSSD-based motion compression approach achieves the same compression ratios as the K-SVD based approach [AEB06]. However, the former achieves much lower reconstruction errors than the latter. Also, as shown in Table 3, our QSSD-based approach significantly outperforms the two selected state-of-the-art motion compression approaches [Ari06, GPD09] in terms of both compression ratio and reconstruction quality. In terms of algorithm efficiency, although the decompression efficiencies of the QSSD-based approach, the K-SVD based approach, and [GPD09] are sufficiently close, but all the three are faster than [Ari06]. Also, our QSSD-based approach consumes more computational time to compress motion data than the other three approaches.

Fig. 3 shows the compression ratio curves of our QSSD-based approach, [Ari06] and [GPD09] on motion datasets with various sizes. Our QSSD-based approach significantly outperforms the other two comparative approaches in terms of compression ratio. Also, we can observe a clear trend that our QSSD-based approach achieves an increasingly higher compression ratio when the size of the motion dataset is larger, compared with the other two approaches [Ari06, GPD09]. Fig. 4 shows side-by-side comparisons between the original character poses (blue skeleton) and the reconstructed character poses (green skeleton) by our QSSD-based approach. As shown in this figure, the visual difference between the original poses and the reconstructed poses is negligible.

5.2. Content-based Human Motion Retrieval

The goal of content-based motion retrieval is to find similar motion sequences in a database given a query motion. Therefore, a sound similarity measure is crucial to the success of any motion retrieval algorithms. Linear subspace decomposition methods (e.g., the work of [FF05]) project human motion data to a low dimensional subspace in order to cluster similar motions. However, human motion data lie on a nonlinear manifold [EL04]; as such, straightforward linear methods would lead to less accurate motion retrieval outcomes. By contrast, our QSSD algorithm replaces linear combination with quaternion combination, which can de-

Dataset	Parameter	QSSD-based	K-SVD based	[GPD09]	[Ari06]
Walk motions (Original size: 111MB Total frame #: 145828 Total clip #: 121)	Compressed size(MB)	1.8	1.8	2.4	3.1
	Compression ratio	62	62	46	36
	ARMS (*cm*)	1.33	5.86	4.12	4.31
	Distortion rate	2.64	6.22	5.43	5.87
	Compression time (ms/frm)	42.5	23.9	2.7	0.9
	Decompression time (ms/frm)	0.8	0.7	0.7	1.0
Mixed motions (Original Size: 93.5MB Total frame #: 122846 Total clip #: 75)	Compressed size(MB)	1.7	1.7	2.5	2.7
	Compression ratio	55	55	37	35
	ARMS (*cm*)	1.92	6.25	5.32	5.87
	Distortion rate	2.91	8.70	6.36	6.55
	Compression time (ms/frm)	48.4	25	3.1	1.0
	Decompression time (ms/frm)	0.7	0.6	0.7	1.2

Table 3: Comparison of motion compression results among our QSSD-based approach, the K-SVD [AEB06] based compression approach, and the two selected state-of-the-art human motion compression approaches [Ari06, GPD09]. Here the used walk/mixed motion datasets are the same as those used in Table 1. The experiments were conducted on an off-the-shelf desktop computer (2.8GHz Intel Core i7 Processor and 4GB memory).

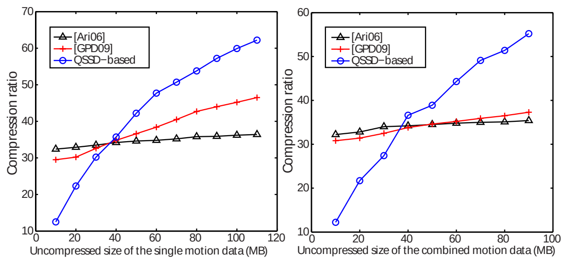


Figure 3: Compression ratio curves by our QSSD-based approach, [Ari06] and [GPD09] on motion datasets with various sizes. The used datasets are: walk motion (left) and mixed motion (right).

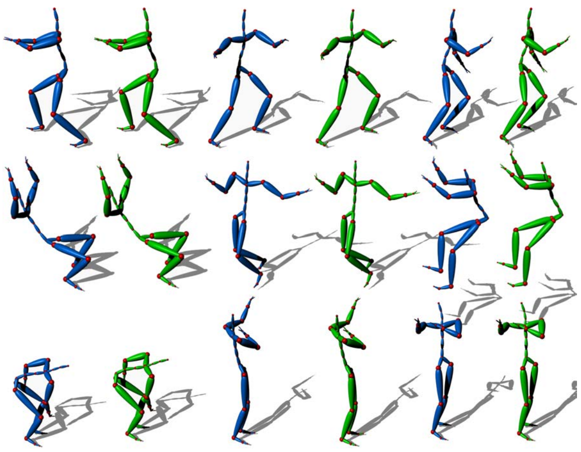


Figure 4: Side-by-side comparison between the original (the blue skeleton, odd columns) and the compressed (the green skeleton, even columns) character poses by our QSSD-based motion compression approach. The three motion clips (From top to bottom: 12_04 (tai chi), 01_13 (climb, go under, jump down), 02_10 (wash self)) are selected from the CMU mocap database.

compose and represent the high dimensional human motion data in a more meaningful way.

Our QSSD-based motion retrieval approach works as follows: (1) At the offline processing stage, angular displacement data is first extracted through motion data preprocessing (refer to Section 4.2). Then, each motion is decomposed to a dictionary and corresponding weights by the QSSD algorithm. (2) At the runtime stage, given a query motion, we first decompose it to the dictionary part and the weight part using the QSSD algorithm. After that, we compute the distance between the dictionary decomposed from the query motion and the dictionaries decomposed from existing motions in the database. Finally, based on the computed dictionary distances, we can find and rank the similar motions. Fig. 5 shows the pipeline of our QSSD-based motion retrieval approach.

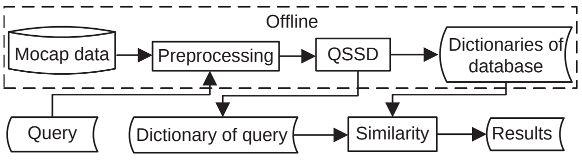


Figure 5: The pipeline of our QSSD-based human motion retrieval approach

In our experiment, we used the publicly accessible, segmented human motion database [MRC*07]. Seven different types of human motions were chosen, including walk, hop, jumping jack (abbreviated as "jjack"), kick, punch, squat, and elbow-to-knee (abbreviated as "etk") exercises. The parameters used in the QSSD-based approach are: $K = 5$ and $L = 2$. To compute the distance between two dictionary atoms (Eq. (10)), we first find the optimal one-to-one map between two dictionaries by minimizing the total distance. For the purpose of comparison, we also performed the same experiment using the original K-SVD approach [AEB06] on the same dataset, with the same parameter setting. In addition, we compared the QSSD-based approach with four state-of-the-art motion retrieval approaches [LZWP03, KG04, FF05, DGL09].

True-Positive Ratio is defined as the percentage of the top N retrieved motions that have the same label as the query motion. In our experiment, N was set to 20. To calculate the average true-positive ratio for each type of motion, every motion clip in the dataset was used as the query motion once. Fig. 6 shows the comparison of the true-positive ratios by all the approaches. For all the test cases, the QSSD-based approach significantly outperforms the original K-SVD approach. Also, for most test cases (except the kick motion), the QSSD-based motion retrieval approach outperforms the other selected comparative approaches in terms of retrieval accuracy (i.e., the true-positive ratio).

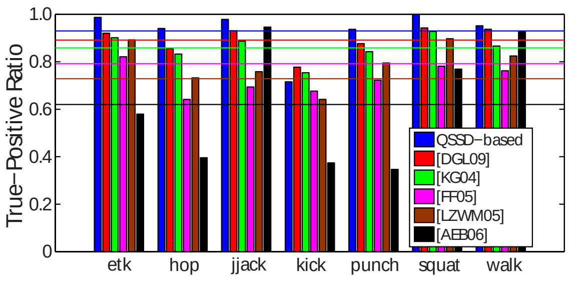


Figure 6: The motion retrieval results by the QSSD-based approach, the K-SVD based approach [AEB06], and four selected human motion retrieval approaches [LZWP03, KG04, FF05, DGL09]. Six horizontal lines illustrate the averaged true-positive ratios by the six approaches.

Confusion matrix is another widely used criterion to evaluate a classification algorithm. It can clearly reveal the confusion level between any two classes. In this work, a similar experiment was conducted to obtain a motion retrieval confusion matrix: besides the number of correctly retrieved motion clips, we also considered the number of mistakenly retrieved motions clips in the top N results.

Fig. 7 shows the obtained retrieval confusion matrices by the original K-SVD approach (left panel) and the QSSD-based human motion retrieval approach (right panel). In this figure, the diagonal elements are the correct retrieval rates of different types of motions, and the off-diagonal elements are the rates of mistakenly retrieved results. For example, for the jjack motion at the right panel, the average true-positive ratio of all the jjack motion queries is 98%, and only at 2% cases, the jjack motions are incorrectly retrieved as punch motions. The results demonstrate that the QSSD-based approach can incur much lower confusions between different types of human motions than naively applying the K-SVD algorithm to human motion data.

6. Discussion and Conclusions

We introduce a novel quaternion space sparse decomposition (QSSD) model that decomposes rotational human motion data into a dictionary part and a weight part. Its core idea is to replace *linear combination*, heavily used in current sparse learning literature, with *quaternion combination* that is first introduced in this work, during the data decomposition procedure. Through extensive validations as well as direct comparisons with the original K-SVD algorithm [AEB06], we validate the robustness, convergence, and accuracy of the introduced QSSD model.

The introduced QSSD model can be potentially used for a variety of human motion related applications. For the sake of demonstration, we apply the QSSD model to two selected applications: human motion compression and content-based human motion retrieval. Through extensive experiments and comparisons, we show that the QSSD-based approaches can significantly outperform the original K-SVD based approaches as well as the state-of-the-art motion compression and retrieval approaches at most cases.

The current QSSD model has two major limitations: (1) Due to the expensive computational cost of quaternion combination (compared with linear combination), the QSSD model takes significant computational time, which is evident in Table 3. As the future work, we plan to alleviate this limitation by exploiting the capability of GPUs or multi-core CPU to accelerate the computation of quaternion combination operations in the QSSD model. (2) Except the sparseness constraint on the weight part, no other constraints (e.g., non-negativity or affinity constraint) is applied to the weight part or to the dictionary part during the data composition process. Consequently, the decomposition results are less intuitive for certain applications such as human motion analysis and character motion editing/synthesis. In the future, we plan to extend the current QSSD model by properly incorporating the non-negativity and/or affinity constraint to the weight part or the sparseness constraint to the dictionary part so that the obtained QSSD dictionary/weights would be more interpretable, intuitive and semantically valid for many animation applications.

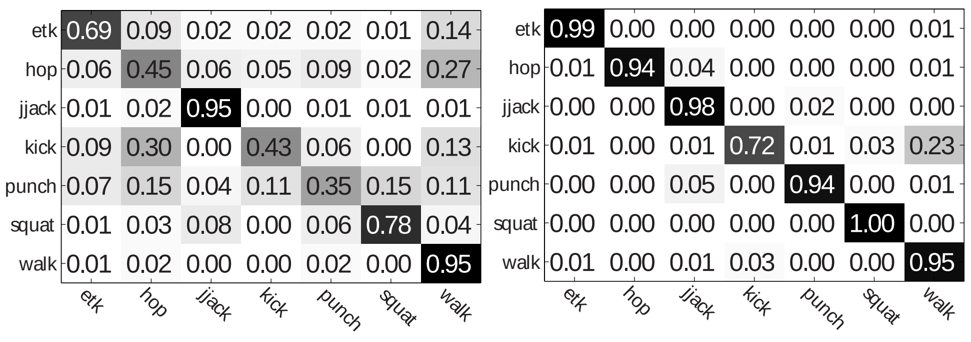
Acknowledgements

This work is supported in part by NSF IIS-0914965, generous unrestricted research gifts from Google and Nokia, and NUST Research Funding, No.2011YBXM79. Mingyang Zhu is supported by a China CSC Fellowship. We would also like to thank Xiaomao Wu for insightful discussion.

References

[AEB06] AHARON M., ELAD M., BRUCKSTEIN A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Sig. Proc. 54*, 11 (2006), 4311–4322.

[Ale02] ALEXA M.: Linear combination of transformations. *ACM Trans. Graph. 21*, 3 (July 2002), 380–387.



	etk	hop	jjack	kick	punch	squat	walk
etk	0.69	0.09	0.02	0.02	0.02	0.01	0.14
hop	0.06	0.45	0.06	0.05	0.09	0.02	0.27
jjack	0.01	0.02	0.95	0.00	0.01	0.01	0.01
kick	0.09	0.30	0.00	0.43	0.06	0.00	0.13
punch	0.07	0.15	0.04	0.11	0.35	0.15	0.11
squat	0.01	0.03	0.08	0.00	0.06	0.78	0.04
walk	0.01	0.02	0.00	0.00	0.02	0.00	0.95

	etk	hop	jjack	kick	punch	squat	walk
etk	0.99	0.00	0.00	0.00	0.00	0.00	0.01
hop	0.01	0.94	0.04	0.00	0.00	0.00	0.01
jjack	0.00	0.00	0.98	0.00	0.02	0.00	0.00
kick	0.01	0.00	0.01	0.72	0.01	0.03	0.23
punch	0.00	0.00	0.05	0.00	0.94	0.00	0.01
squat	0.00	0.00	0.00	0.00	0.00	1.00	0.00
walk	0.01	0.00	0.01	0.03	0.00	0.00	0.95

Figure 7: Comparison of the confusion matrices of seven classes of motions between naively using the original K-SVD algorithm [AEB06] on human motion retrieval (left panel), and our QSSD-based motion retrieval approach (right panel).

[Ari06] ARIKAN O.: Compression of motion capture databases. *ACM Trans. Graph. 25*, 3 (July 2006), 890–897.

[Ber99] BERTSEKAS D. P.: *Nonlinear Programming*, 2nd ed. Athena Scientific, Sept. 1999.

[BJJ03] BLUMBERG B. M., JOHNSON M. P., JOHNSON M. P.: *Exploiting Quaternions to Support Expressive Interactive Character Motion*. Tech. rep., MIT, 2003.

[BPvdP07] BEAUDOIN P., POULIN P., VAN DE PANNE M.: Adapting wavelet compression to human motion capture clips. In *Graphics Interface 2007* (May 2007), pp. 313–318.

[CBL89] CHEN S., BILLINGS S. A., LUO W.: Orthogonal least squares methods and their application to non-linear system identification. *Int'l J. of Control 50*, 5 (1989), 1873–1896.

[CBL07] CHATTOPADHYAY S., BHANDARKAR S. M., LI K.: Human motion capture data compression by model-based indexing: A power aware approach. *IEEE TVCG 13*, 1 (2007), 5–14.

[CCW*04] CHIU C. Y., CHAO S. P., WU M. Y., YANG S. N., LIN H. C.: Content-based retrieval for human motion data. *Journal of Visual Communication and Image Representation 15*, 3 (2004), 446–466.

[CDS01] CHEN S. S., DONOHO D. L., SAUNDERS M. A.: Atomic decomposition by basis pursuit. *SIAM Rev. 43*, 1 (Jan. 2001), 129–159.

[CMU11] The CMU motion capture library, http://mocap.cs.cmu.edu, 2011.

[CS11] CHUAN SUN IMRAN N. JUNEJO H. F.: Motion retrieval using low-rank subspace decomposition of motion volumel. *Comput. Graph. Forum 30*, 7 (2011), 1953–1962.

[DGL09] DENG Z., GU Q., LI Q.: Perceptually consistent example-based human motion retrieval. In *I3D'09* (New York, NY, USA, 2009), ACM, pp. 191–198.

[DKL98] DAM E. B., KOCH M., LILLHOLM M.: *Quaternions, interpolation and animation*. Tech. Rep. DIKU-TR-98/5, Institute of Computer Science, University of Copenhagen, Copenhagen Denmark., 1998.

[DMZ94] DAVIS G., MALLAT S., ZHANG Z.: Adaptive time-frequency decompositions. *Optical Engineering 33*, 7 (1994), 2183–2191.

[EL04] ELGAMMAL A., LEE C.-S.: Inferring 3D body pose from silhouettes using activity manifold learning. In *IEEE CVPR'04* (2004), pp. 681–688.

[ERKD99] ENGAN K., RAO B. D., KREUTZ-DELGADO K.: *Frame Design Using Focuss with Method of Optimal Directions (MOD)*, vol. 65. 1999, p. 69.

[FF05] FORBES K., FIUME E.: An efficient search algorithm for motion data using weighted pca. In *SCA'05* (2005), pp. 67–76.

[FRM94] FALOUTSOS C., RANGANATHAN M., MANOLOPOULOS Y.: Fast subsequence matching in time-series databases. In *Proc.of ACM SIGMOD'94* (1994), pp. 419–429.

[GPD09] GU Q., PENG J., DENG Z.: Compression of human motion capture data using motion pattern indexing. *Computer Graphics Forum 28* (2009), 1–12.

[Kar04] KARNI Z.: Compression of soft-body animation sequences. *Computers and Graphics 28*, 1 (2004), 25–34.

[KG04] KOVAR L., GLEICHER M.: Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 559–568.

[LFAJ10] LI Y., FERMÜLLER C., ALOIMONOS Y., JI H.: Learning shift-invariant sparse representation of actions. In *CVPR'10* (june 2010), IEEE, pp. 2630 – 2637.

[LM06] LIU G., MCMILLAN L.: Segment-based human motion compression. In *SCA'06* (2006), pp. 127–135.

[LZWM05] LIU G., ZHANG J., WANG W., MCMILLAN L.: A system for analyzing and indexing human-motion databases. In *Proc. of ACM SIGMOD '05* (2005), pp. 924–926.

[LZWP03] LIU F., ZHUANG Y., WU F., PAN Y.: 3d motion retrieval with motion index tree. *Computer Vision and Image Understanding 92*, 2-3 (2003), 265–284.

[MR06] MÜLLER M., RÖDER T.: Motion templates for automatic classification and retrieval of motion capture data. In *SCA '06* (2006), pp. 137–146.

[MRC05] MÃIJLLER M., RÃŰDER T., CLAUSEN M.: Efficient content-based retrieval of motion capture data. *ACM Transactions on Graphics 24*, 3 (2005), 677.

[MRC*07] MÜLLER M., RÖDER T., CLAUSEN M., EBERHARDT B., KRÜGER B., WEBER A.: *Documentation Mocap Database HDM05*. Tech. Rep. CG-2007-2, Universität Bonn, June 2007.

[MZ93] MALLAT S., ZHANG Z.: Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing 41*, 12 (Dec 1993), 3397–3415.

[Say05] SAYOOD K.: *Introduction to Data Compression*. Morgan Kaufmann Publishers, 2005.

[Sho85] SHOEMAKE K.: Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph. 19*, 3 (July 1985), 245–254.

[TWC*09] TOURNIER M., WU X., COURTY N., ARNAUD E., REVÃL'RET L.: Motion compression using principal geodesics analysis. *Computer Graphics Forum 28*, 2 (2009), 355–364.