Learning a 3D Human Pose Distance Metric from Geometric Pose Descriptor

Cheng Chen, Yueting Zhuang, Feiping Nie, Yi Yang, Fei Wu, Jun Xiao

Abstract—Estimating 3D pose similarity is a fundamental problem on 3D motion data. Most previous work calculates $L2$-like distance of joint orientations or coordinates, which does not sufficiently reflect the pose similarity of human perception. In this paper we present a new pose distance metric. First, we propose a new rich pose feature set called *Geometric Pose Descriptor* (*GPD*). GPD is more effective in encoding pose similarity by utilizing features on geometric relations among body parts, as well as temporal information such as velocities and accelerations. Based on GPD, we propose a semi-supervised distance metric learning algorithm called *Regularized Distance Metric Learning with Sparse Representation* (*RDSR*), which integrates information from both unsupervised data relationship and labels. We apply the proposed pose distance metric to applications of motion transition decision and content based pose retrieval. Quantitative evaluations demonstrate that our method achieves better results with only a small amount of human labels, showing that the proposed pose distance metric is a promising building block for various 3D motion related applications.

Index Terms—human motion, character animation, pose features, distance metric, semi-supervised learning.

1 Introduction

In the past few years, motion capture technique has been used extensively. Since 3D pose is the fundamental element of motion data, distance metrics on 3D poses have attracted much research interest [1][2][3][4][5].

A proper distance metric on 3D poses serves as a fundamental building block for many 3D motion related applications. For example, animators often need to retrieve relevant motions scattered in the 3D motion dataset from examples [6][7]. In this case the motion similarity is often estimated based on the distances between poses (probably with time warping techniques [8]). In the well-known motion graph algorithm [9][10][11], pose similarity is used to detect satisfactory transition points. In some other applications such as motion segmentation [12][13], compression [14][15] and classification [16], a preliminary step is often to represent each pose as a feature vector and then pose distance is calculated in the feature space. Many human-computer interaction systems also need to estimate pose distance. For example, the Tai Chi training system in [17] evaluates the difference between the student's pose and the teacher's pose. Also, in computer vision, image based pose recovery algorithms evaluate the performance by the distance between the recovered poses and ground-truth [18]. In short, working with 3D motion data naturally requires a discriminative pose feature set and a proper distance metric in the feature space.

- *C. Chen is with Idiap Research Institute, Martigny, Switzerland. E-mail: cchen@idiap.ch*
- *Y. Zhuang, F. Wu and J. Xiao are with Zhejiang University, Hangzhou, China.*
- *F. Nie is with University of Texas, Arlington, USA.*
- *Y. Yang is with ITEE, The University of Queensland, Australia.*

A lot of previous work uses joint orientations or coordinates straightforwardly (optionally with velocities) as pose features [2][11][19]. However, there is a substantial gap between perceptual pose distance and the coordinates or orientations of individual joints. It has been indicated that pose discrimination relies heavily on the relational configuration between body parts [3][4][5][20]. Also, human does not put equal emphasis on different body parts when understanding poses.

In this paper we propose a new collection of pose features, referred to as *Geometric Pose Descriptor* (*GPD*). Geometric features have been used in graphics and vision communities [21][22][23], and here we propose a new rich pose feature set exploiting geometric properties and relations between different body parts. GPD emphasizes relational body part configurations, which is more consistent with human perception [5].

Given the pose feature set, another problem is the distance metric design. The simplest metric is $L2$. However, $L2$ does not sufficiently encode pose semantics. Some recent work [1][3][5] tries to learn the distance metric from training data. These distance metrics have shown superior performance compared with $L2$. However, one limitation is that extensive manual labeling is required. For example, [5] uses 12000 pose pairs labeled by 30 human supervisors. Labeling such amount of data is expensive and tedious.

On the other hand, because unlabeled 3D poses are easy to obtain, they can be used to alleviate the extensive need for labels. In this paper we propose a semi-supervised distance metric learning algorithm, namely *Regularized Distance Metric Learning with Sparse Representation* ($RDSR$) to learn an optimal Mahalanobis distance metric based on GPD features. RDSR gracefully integrates information of the unsupervised data relationship with label information, and has an efficient procedure to

Digital Object Indentifier 10.1109/TVCG.2010.272

perform global optimization. Compared with previous algorithms, RDSR shows better performance with a relatively small amount of labels.

This paper makes contributions in pose features and pose distance metric learning, and we show that both bring improvements. We conduct experiments on motion transition decision and content based pose retrieval. Various evaluations show that GPD is better than other features, RDSR outperforms other distance metric learning algorithms, and the combination of GPD and RDSR gives the best result.

In the following, after summarizing the related work in Section 2, we explain GPD feature set in Section 3. Section 4 presents the RDSR distance metric learning algorithm. Experiments on motion transition decision and content based pose retrieval are detailed in Sections 5 and 6, respectively. Section 7 gives the conclusion.

2 Related Work

2.1 Pose Features

It has been known for long that $L2$ distance on raw joint coordinates or orientations does not sufficiently reveal pose similarity. Kovar et al. [24] address this problem by implicitly exploiting the neighborhood graph of the motion manifold. To retrieve logically similar motions given an example, they first retrieve a small amount of numerically similar motions, and then the retrieved motions are used as intermediate queries from which more motions are retrieved. Lee et. al [11] use weighted joint orientation angles and velocities to represent poses, where the weights for each joint are set by hand. Wang et. al [1] improve the method by learning the weights from human labeled data. Our work can be viewed as a further extension in three aspects: we propose a richer feature set, the weights associated with individual joints are extended to more flexible Mahalanobis distance, and a new semi-supervised learning method is proposed to reduce the number of human labels required.

Recently, several new pose feature types have been proposed. For example, Muller et al. [20][25] define 31 Boolean features for retrieving topologically similar motions very efficiently. However, the feature set needs to be manually selected for different motion types. Also, Boolean features are aimed at efficiency, and are too coarse for accurate pose distance estimation. Tang et al. [3] propose Joint Relative Distance that utilizes distances between joint pairs. Chen et al. [5] construct a feature pool that enumerates all possible relational features, and then the relevant features are selected by Adaboost from a large number of labeled pose pairs. Onuma et al. [26] propose kinetic energy based features. This type of feature is defined on an entire motion sequence and is not suitable for accurate similarity measurement on a pose-wise level. Ho et al. [4] propose tangle based features. Tangles successfully encode the twisted contact of two characters, but they are not suitable to encode pose similarity of a single character.

2.2 Semi-supervised Distance Metric Learning

Given pose features, another important issue is how to define the distance metric. Distance metric learning has attracted much research interest [27] [28] [29] [30] [31] [32]. These research efforts have shown that a carefully learned distance metric can yield substantial improvements in many applications. To deal with semantic gap, a practical solution is to incorporate labels [1] [3] [5]. Although labels do help, they require a lot of human labor. Meanwhile, some unsupervised distance metric learning algorithms are proposed [27] [32]. However, due to the lack of label information, the performance of unsupervised algorithms may be unsatisfactory. On the other hand, semi-supervised distance learning algorithms [33][34] learn with both labeled and unlabeled data. In this paper we propose a new semi-supervised metric learning algorithm named RDSR. For a detailed discussion of our RDSR method in the context of semi-supervised learning, please refer to Section 4.4.

3 Geometric Pose Descriptor

3.1 Pose Data Format and Human Skeleton Model

Usually, a 3D pose is encoded as a collection of joint coordinates (e.g. trc files) or orientations (e.g. bvh, asf/amc files). Converting from orientation based format to coordinate based counterpart is straightforward, while the reverse is tricky and ambiguous. Two poses expressed by joint orientations cannot be directly compared unless they have the same skeleton parameterization. To avoid unnecessary complication and for better generality, we use joint coordinates as the raw data of poses (we will compare with rotation based formats in Section 5.3 and find no significant performance difference).

In this paper we use a 16-joint skeleton model shown in Fig. 1(a). The skeleton is a tree structure, where the joints are nodes. Note that in graphics applications, there are often more complex skeletons, such as those containing fingers or toes. Here we assume that pose distance is only related to the configuration of main body joints. Therefore, the information from fingers/toes is not exploited even if it is available.

All joint coordinates and the pose features described below are expressed in the figure's local coordinate frame, which is translated relative to the *Hip* joint and rotated according to the body's yaw angle. This is because pose similarity is independent of global translation and rotation around the vertical axis.

We define a set of joints, lines and planes on the human skeleton as shown in Fig. 1. There are 16 joints, 30 lines and 5 planes in total, as explained below.

- Joint. Each joint J is encoded with its coordinate (J_x, J_y, J_z). There are 16 joints in total.

- Line. $L_{J_1 \to J_2}$ is the line from joint J_1 to J_2, if one of the following three constraints is satisfied:

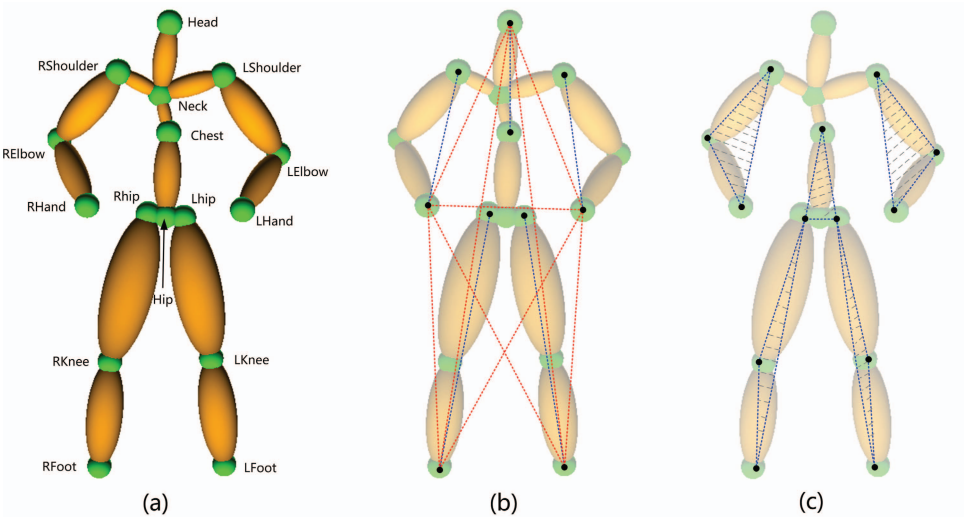

Fig. 1. (a) Skeleton model. Green spheres represent joints and orange ellipsoids represent limbs. (b) Lines. Limbs are already lines. Blue and red dash lines are additional lines of two types. (c) Planes.

1) J_1 and J_2 are directly adjacent in the kinetic chain. This produces 15 lines.
2) If one of J_1 and J_2 is end site (*Head*, *L*(*R*)*Hand* or *L*(*R*)*Foot*), then the other one can be two steps away on the same kinetic chain (i.e. one joint is an ancestor of the other and the difference between their depths in the tree is two). For example, $L_{LShoulder \to LHand}$ and $L_{RHip \to RToe}$ are two valid lines. This produces 5 lines. Incorporating these lines is because the kinetic chains towards end sites are important in pose perception.
3) If both J_1 and J_2 are end sites, then $L_{J_1 \to J_2}$ is a line. This produces 10 lines. This line category is incorporated because the relations between end sites play an important role in pose identification.

• Plane. $P_{J_1 \to J_2 \to J_3}$ is the plane determined by the triangle with vertices J_1, J_2 and J_3. Because planes are more complex and only a small number of major planes tend to be noticed in pose perception, only 5 planes are considered, namely: $P_{Chest \to Neck \to Head}$, $P_{LShoulder \to LElbow \to LHand}$, $P_{RShoulder \to RElbow \to RHand}$, $P_{LHip \to LKnee \to LFoot}$, and $P_{RHip \to RKnee \to RFoot}$. These correspond to the planes of torso, arms and legs.

3.2 Pose Features

We define nine types of GPD features (as in Fig. 2), including eight static features and one temporal feature. Static features encode the configuration in one pose, and temporal features represent the variation in time.

3.2.1 Static features

• Joint Coordinate $f_{\text{J\_c}}(J)$:
This is the 3D coordinate of the joint J.

$$f_{\text{J\_c}}(J) = (J_x, J_y, J_z) \tag{1}$$

Note that *Hip*'s coordinate is excluded as it is always $(0, 0, 0)$. On the other hand, the y coordinate of *Hip* in the world coordinate frame reflects the absolute height of body and is informative in some cases (e.g. discerning jumping in the air), and hence is included. Therefore, the total dimension of joint coordinates is $15 \times 3 + 1 = 46$.

• Joint-Joint Distance $f_{\text{JJ\_d}}(J_1, J_2)$:
This is the Euclidean distance from joint J_1 to J_2.

$$f_{\text{JJ\_d}}(J_1, J_2) = \left\| \overrightarrow{J_1 J_2} \right\| \tag{2}$$

• Joint-Joint Orientation $f_{\text{JJ\_o}}(J_1, J_2)$:
This is the orientation from joint J_1 to J_2, represented by vector $\overrightarrow{J_1 J_2}$ normalized to unit length.

$$f_{\text{JJ\_o}}(J_1, J_2) = \text{unit}(\overrightarrow{J_1 J_2}) \tag{3}$$

where function unit() scales a vector into unit length.

• Joint-Line Distance $f_{\text{JL\_d}}(J, L_{J_1 \to J_2})$:
This is the distance from joint J to line $L_{J_1 \to J_2}$.

$$f_{\text{JL\_d}}(J, L_{J_1 \to J_2}) = 2S_{\triangle JJ1J2} / f_{\text{JJ\_d}}(J_1, J_2) \tag{4}$$

where $S_{\triangle JJ1J2}$ is the area of triangle $\triangle JJ_1J_2$. Since we have already calculated the pairwise distances between J, J_1 and J_2 as feature $f_{\text{JJ\_d}}$ in (2), the calculation in (4) can be accelerated by employing Helen formula.

• Line-Line Angle $f_{\text{LL\_a}}(L_{J_1 \to J_2}, L_{J'_1 \to J'_2})$:
This is the angle (0 to π) from line $L_{J_1 \to J_2}$ to $L_{J'_1 \to J'_2}$.

$$\begin{aligned} &f_{\text{LL\_a}}(L_{J_1 \to J_2}, L_{J'_1 \to J'_2}) \\ &= \arccos\left(f_{\text{JJ\_o}}(J_1, J_2) \odot f_{\text{JJ\_o}}(J'_1, J'_2)\right) \end{aligned} \tag{5}$$

where $\odot$ is the dot product operator on two vectors.

• Joint-Plane Distance $f_{\text{JP\_d}}(J, P_{J_1 \to J_2 \to J_3})$:

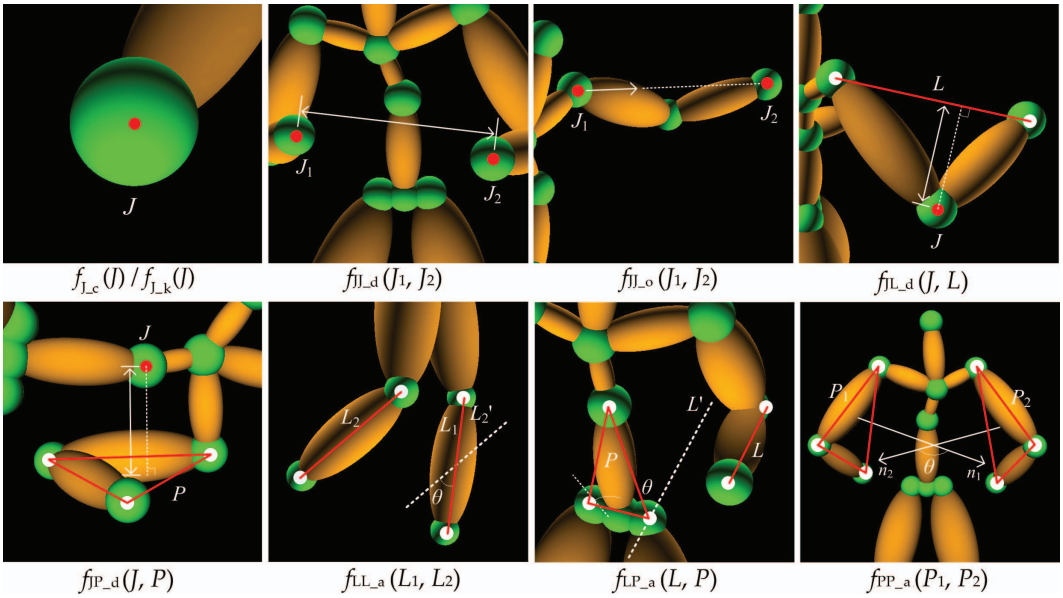

Fig. 2. Nine feature types. Note that for each feature only the relevant joints, lines and planes are drawn in red.

TABLE 1
Summary of pose features

Type	$f_{\text{J\_c}}$	$f_{\text{JJ\_d}}$	$f_{\text{JJ\_o}}$	$f_{\text{JL\_d}}$	$f_{\text{LL\_a}}$	$f_{\text{JP\_d}}$	$f_{\text{LP\_a}}$	$f_{\text{PP\_a}}$	$f_{\text{J\_k}}$	Total
Count	16	120	120	420	65	435	135	10	16	1337
Dimension	46	120	360	420	65	435	135	10	92	1683

This is the distance from joint J to plane $P_{J_1 \to J_2 \to J_3}$:

$$\begin{aligned} &f_{\text{JP\_d}}(J, P_{J_1 \to J_2 \to J_3}) \\ =&f_{\text{JJ\_o}}(J_1, J) \odot \text{unit}\left(f_{\text{JJ\_o}}(J_1, J_2) \otimes f_{\text{JJ\_o}}(J_1, J_3)\right) \end{aligned} \tag{6}$$

where $\otimes$ is the cross product operator on two 3D vectors.

• Line-Plane Angle $f_{\text{LP\_a}}(L_{J_1 \to J_2}, P_{J_1' \to J_2' \to J_3'})$:
This is the angle (0 to π) between line $L_{J_1 \to J_2}$ and the normal vector of plane $P_{J_1' \to J_2' \to J_3'}$:

$$\begin{aligned} &f_{\text{LP\_a}}(L_{J_1 \to J_2}, P_{J_1' \to J_2' \to J_3'}) \\ =&\arccos\left(f_{\text{JJ\_o}}(J_1, J_2)\right. \\ &\left.\odot \text{unit}\left(f_{\text{JJ\_o}}(J_1', J_2') \otimes f_{\text{JJ\_o}}(J_1', J_3')\right)\right) \end{aligned} \tag{7}$$

• Plane-Plane Angle $f_{\text{PP\_a}}(P_{J_1 \to J_2 \to J_3}, P_{J_1' \to J_2' \to J_3'})$:
This is the angle (0 to π) between the normal vectors of planes $P_{J_1 \to J_2 \to J_3}$ and $P_{J_1' \to J_2' \to J_3'}$:

$$\begin{aligned} &f_{\text{PP\_a}}(P_{J_1 \to J_2 \to J_3}, P_{J_1' \to J_2' \to J_3'}) \\ =&\arccos\left(\text{unit}\left(f_{\text{JJ\_o}}(J_1, J_2) \otimes f_{\text{JJ\_o}}(J_1, J_3)\right)\right. \\ &\left.\odot \text{unit}\left(f_{\text{JJ\_o}}(J_1', J_2') \otimes f_{\text{JJ\_o}}(J_1', J_3')\right)\right) \end{aligned} \tag{8}$$

3.2.2 Temporal features

• Joint Kinetics $f_{\text{J\_k}}(J)$:
This is the velocity and acceleration of joint J's coordinate in the temporal domain.

$$f_{\text{J\_k}}(J) = \left(\dot{J}_x, \dot{J}_y, \dot{J}_z, \ddot{J}_x, \ddot{J}_y, \ddot{J}_z\right) \tag{9}$$

where $\dot{a}$ and $\ddot{a}$ are the first-order and second-order derivatives of variable a in the time axis.

In temporal features, we only consider the velocity and acceleration of the joint coordinates. In theory, we could also include the derivatives of all the other static features. However, the physical meanings of the other static features' derivatives are not so obvious. Also, we observe that including derivatives of all static features does not seem to improve the results, but significantly increases the computational burden, as the feature dimension will be nearly tripled (from around 1600 dimensions to around 5000 dimensions).

3.2.3 Feature enumeration

Combining the joints, lines and planes with the nine feature types, and removing duplicated features due to symmetry or degeneration[1], 1337 features are generated in total, and the entire feature set falls into 1683 dimensions as summarized in Table 1.

Note that in this paper we assume the poses are drawn from motion clips, and the neighboring poses are known and the temporal features can be calculated. In practice, if the poses are static and independent, then we can use only the eight static features with the same methodology.

1. For example, $f_{\text{JJ\_d}}(J_1, J_2)$ is symmetric to $f_{\text{JJ\_d}}(J_2, J_1)$, and $f_{\text{JL\_d}}(J, L_{J_1 \to J_2})$ degenerates to zero if J is the same as J_1 or J_2.

4 POSE DISTANCE LEARNING

With the GPD features defined in Section 3, each pose is represented by a vector in $\mathbb{R}^{1683}$. Now we need to learn a distance metric that matches perceptual pose similarity.

One problem of pose distance learning is the effort needed to annotate a large amount of data. On the other hand, it is easier to get unlabeled poses. Motivated by the success of semi-supervised learning [33], we propose a new semi-supervised distance metric learning algorithm named RDSR to learn an optimal pose distance metric. RDSR simultaneously utilizes pairwise label information as well as the relationship between both labeled and unlabeled data, and thus takes the advantages of both supervised and unsupervised learning.

4.1 Problem Definition

We are given the training pose set $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, where $\mathbf{x}_i \in \mathbb{R}^{1683}$ and N is the number of poses. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N] \in \mathbb{R}^{1683\times N}$. Additionally, we have some positive labels $\mathcal{P} = \{(\mathbf{x}_k, \mathbf{x}_l) | \mathbf{x}_k \text{ and } \mathbf{x}_l \text{ are similar}\}$, where $\mathbf{x}_k, \mathbf{x}_l \in \mathcal{X}$. RDSR learns a Mahalanobis distance metric formulated as:

$$d(\mathbf{x}_i, \mathbf{x}_j)_{\mathbf{M}} = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)} \tag{10}$$

where $\mathbf{M} \in \mathbb{R}^{1683\times 1683}$. Therefore, learning the distance metric is equivalent to determining $\mathbf{M}$. Note that by setting $\mathbf{M} = \mathbf{I}$, (10) gives the $L2$ distance.

4.2 The objective function

First of all, to make (10) a valid distance metric, $\mathbf{M}$ should be symmetric and positive semi-definite so that non-negativity and triangle inequality hold. Therefore, $\mathbf{M}$ can be written as $\mathbf{M} = \mathbf{W}\mathbf{W}^T$ for some $\mathbf{W} \in \mathbb{R}^{d\times d'}$ with $d' < d$. Hence, (10) can be reformulated as:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W}\mathbf{W}^T (\mathbf{x}_i - \mathbf{x}_j)} \tag{11}$$

The task is then to find the optimal $\mathbf{W}$ according to some criteria encoded in an objective function. We use three criteria: $E_{\text{supervision}}$ the consistency with the labels, $E_{\text{relationship}}$ the consistency with the data relationship, and a regularizor $E_{\text{regularization}}$. Generally speaking, $E_{\text{supervision}}$ enforces that labeled similar poses are close under the learned distance metric. $E_{\text{relationship}}$ enforces that the relations among all data (both labeled and unlabeled) should be retained. $E_{\text{regularization}}$ prevents overfitting.

4.2.1 Using label information

$\mathcal{P}$ contains labeled similar poses, and we expect that these similar poses are close in the learned distance metric. Therefore, we define a criterion $E_{\text{supervision}}$ as the total squared distance between the labeled similar poses in the learned distance metric:

$$\begin{aligned} &E_{\text{supervision}} \\ =&\sum_{(\mathbf{x}_k, \mathbf{x}_l)\in\mathcal{P}} (\mathbf{x}_k - \mathbf{x}_l)^T \mathbf{W}\mathbf{W}^T (\mathbf{x}_k - \mathbf{x}_l) \\ =&Tr\left(\sum_{(\mathbf{x}_k, \mathbf{x}_l)\in\mathcal{P}} \left(\mathbf{W}^T (\mathbf{x}_k - \mathbf{x}_l) (\mathbf{x}_k - \mathbf{x}_l)^T \mathbf{W}\right)\right) \\ =&Tr\left(\mathbf{W}^T \mathbf{S}_W \mathbf{W}\right) \end{aligned} \tag{12}$$

where $\mathbf{S}_W$ is the within-class scatter matrix as:

$$\mathbf{S}_W = \sum_{(\mathbf{x}_k, \mathbf{x}_l)\in\mathcal{P}} (\mathbf{x}_k - \mathbf{x}_l) (\mathbf{x}_k - \mathbf{x}_l)^T \tag{13}$$

Thus, we have the following objective:

$$\min Tr\left(\mathbf{W}^T \mathbf{S}_W \mathbf{W}\right), \text{s.t.} \mathbf{W}^T\mathbf{W} = \mathbf{I} \tag{14}$$

The constraint $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ in (14) and hereafter is imposed to avoid arbitrary scaling.

4.2.2 Exploiting unsupervised data relationship

We use sparse representation (SR) to exploit the unsupervised relationship of data. SR has been used in computer vision[35][36], and it has been shown to be more effective than nearest neighbor methods in image analysis [37] and face recognition [38].

Formally, each pose $\mathbf{x}_i$ in $\mathcal{X}$ can be approximately reconstructed as a combination of other poses:

$$\mathbf{x}_i \approx a_{i,1}\mathbf{x}_1 + ... + a_{i,N}\mathbf{x}_N = \mathbf{X}\mathbf{a}_i, \text{ s.t. } a_{i,i} = 0 \tag{15}$$

where $a_{i,1}$, ..., $a_{i,N}$ are reconstructing weights of $\mathbf{x}_i$ (with $a_{i,i} = 0$ to avoid trivial self-reconstruction), and $\mathbf{a}_i = [a_{i,1}, ..., a_{i,N}]^T$ is the reconstruction weight vector.

According to SR [37][38], $\mathbf{a}_i$ in (15) should be sparse, i.e. only a small proportion of training poses should be used to reconstruct $\mathbf{x}_i$. If no constraint is enforced on the sparsity, then $\mathbf{x}_i$ can be approximated with very small error by dense weights that are not informative on the real data structure. Following [38], $\mathbf{a}_i$ can be derived by:

$$\mathbf{a}_i = \underset{a_{i,j}(1\le j\le N)}{\arg\min} \|\mathbf{x}_i - \mathbf{X}\mathbf{a}_i\|_2 + \gamma\|\mathbf{a}_i\|_1 \tag{16}$$

where $\|.\|_2$ and $\|.\|_1$ are $L2$ norm and $L1$ norm, respectively. $\mathbf{a}_i$ reflects the discriminative information, i.e. relations between $\mathbf{x}_i$ and other poses. A reasonable assumption is that such information be retained under the learned distance metric. Or saying in another way, the weights $\mathbf{a}_i$ should be discriminative in approximating the data in the learned distance metric. Therefore, we define a criterion $E_{\text{relationship}}$ as the residual error of approximation in the learned distance metric:

$$
\begin{aligned}
&E_{\text{relationship}} \\
=&\sum_{i=1}^{N}\left(\mathbf{x}_i-\mathbf{X}\mathbf{a}_i\right)^T\mathbf{W}\mathbf{W}^T\left(\mathbf{x}_i-\mathbf{X}\mathbf{a}_i\right) \\
=&Tr\left(\mathbf{W}^T\left(\sum_{i=1}^{N}\left(\mathbf{x}_i-\mathbf{X}\mathbf{a}_i\right)\left(\mathbf{x}_i-\mathbf{X}\mathbf{a}_i\right)^T\right)\mathbf{W}\right) \\
=&Tr\left(\mathbf{W}^T\mathbf{X}\left(\mathbf{I}_N-\mathbf{A}\right)\left(\mathbf{I}_N-\mathbf{A}\right)^T\mathbf{X}^T\mathbf{W}\right) \\
=&Tr\left(\mathbf{W}^T\mathbf{X}\mathbf{S}_L\mathbf{X}^T\mathbf{W}\right)
\end{aligned}
\tag{17}
$$

where $\mathbf{I}_N$ is the $N\times N$ identity matrix, $\mathbf{A}=[\mathbf{a}_1,...,\mathbf{a}_N]$, and $\mathbf{S}_L$ is the $N\times N$ Laplacian matrix:

$$
\mathbf{S}_L=\left(\mathbf{I}_N-\mathbf{A}\right)\left(\mathbf{I}_N-\mathbf{A}\right)^T \tag{18}
$$

Thus, we have the following objective:

$$
\min\ Tr\left(\mathbf{W}^T\mathbf{X}\mathbf{S}_L\mathbf{X}^T\mathbf{W}\right),\text{s.t.}\mathbf{W}^T\mathbf{W}=\mathbf{I} \tag{19}
$$

4.2.3 Regularization

To prevent overfitting, care should be taken that the learned distance metric does not go too far away from the original distance metric. For this, we introduce a regularizor. Let d_{ij} be the $L2$ distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, and d'_{ij} be the learned Mahalanobis distance. The deviation from the original data relationship is measured by:

$$
\sum_{i,j}\left|d_{ij}-d'_{ij}\right| \tag{20}
$$

Because $\mathbf{W}\in\mathbb{R}^{d\times d'}$ and $\mathbf{W}^T\mathbf{W}=\mathbf{I}$, there exists a columnly orthogonal matrix $\hat{\mathbf{W}}\in\mathbb{R}^{d\times(d-d')}$ such that $\mathbf{W}\mathbf{W}^T+\hat{\mathbf{W}}\hat{\mathbf{W}}^T=\mathbf{I}$. Thus we have:

$$
\begin{aligned}
&d_{ij}-d'_{ij} \\
=&(\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_i-\mathbf{x}_j)-(\mathbf{x}_i-\mathbf{x}_j)^T\mathbf{W}\mathbf{W}^T(\mathbf{x}_i-\mathbf{x}_j) \\
=&(\mathbf{x}_i-\mathbf{x}_j)^T\hat{\mathbf{W}}\hat{\mathbf{W}}^T(\mathbf{x}_i-\mathbf{x}_j)\geq 0
\end{aligned}
\tag{21}
$$

Therefore, (20) can be rewritten as:

$$
\begin{aligned}
&\sum_{i,j}\left|d_{ij}-d'_{ij}\right|=\sum_{i,j}\left(d_{ij}-d'_{ij}\right) \\
=&\sum_{i,j}(\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_i-\mathbf{x}_j)-Tr\left(\mathbf{W}^T\mathbf{X}\mathbf{S}_C\mathbf{X}^T\mathbf{W}\right)
\end{aligned}
\tag{22}
$$

where $\mathbf{S}_C$ is the centering matrix defined as:

$$
\mathbf{S}_C=\mathbf{I}_N-\frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T \tag{23}
$$

where $\mathbf{1}_N=[1,1,...,1]^T$.

We would like to minimize (22) to prevent large deviation from the original distance metric. Note that the first term of (22), $\sum_{i,j}(\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_i-\mathbf{x}_j)$, is a constant. Therefore, if we define $E_{\text{regularization}}$ as the second term:

$$
E_{\text{regularization}}=Tr\left(\mathbf{W}^T\mathbf{X}\mathbf{S}_C\mathbf{X}^T\mathbf{W}\right) \tag{24}
$$

then the objective becomes maximizing $E_{\text{regularization}}$:

$$
\max\ Tr\left(\mathbf{W}^T\mathbf{X}\mathbf{S}_C\mathbf{X}^T\mathbf{W}\right),\text{s.t.}\mathbf{W}^T\mathbf{W}=\mathbf{I} \tag{25}
$$

4.2.4 The complete objective function

Combining (14), (19) and (25), we get:

$$
\begin{aligned}
\mathbf{W}^*&=\underset{\mathbf{W}^T\mathbf{W}=\mathbf{I}}{\arg\max}\frac{E_{\text{regularization}}}{E_{\text{relationship}}+E_{\text{supervision}}} \\
&=\underset{\mathbf{W}^T\mathbf{W}=\mathbf{I}}{\arg\max}\frac{Tr\left(\mathbf{W}^T\left(\mathbf{X}\mathbf{S}_C\mathbf{X}^T\right)\mathbf{W}\right)}{Tr\left(\mathbf{W}^T\left(\mathbf{X}\mathbf{S}_L\mathbf{X}^T+\alpha\mathbf{S}_W\right)\mathbf{W}\right)}
\end{aligned}
\tag{26}
$$

where $\mathbf{S}_W$, $\mathbf{S}_L$ and $\mathbf{S}_C$ are given in (13), (18) and (23), respectively, and α is the weighting parameter.

After solving (26), the optimal $\mathbf{W}^*$ is incorporated into (11) to make the desired pose distance.

4.3 Optimization

Eq. (26) is a trace ratio optimization. By defining $\mathbf{A}=\mathbf{X}\mathbf{S}_C\mathbf{X}^T$ and $\mathbf{B}=\mathbf{X}\mathbf{S}_L\mathbf{X}^T+\alpha\mathbf{S}_W$, the problem becomes:

$$
\mathbf{W}^*=\underset{\mathbf{W}^T\mathbf{W}=\mathbf{I}}{\arg\max}\frac{Tr\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)}{Tr\left(\mathbf{W}^T\mathbf{B}\mathbf{W}\right)} \tag{27}
$$

Let $\eta^*=\max\limits_{\mathbf{W}^T\mathbf{W}=\mathbf{I}}\frac{Tr(\mathbf{W}^T\mathbf{A}\mathbf{W})}{Tr(\mathbf{W}^T\mathbf{B}\mathbf{W})}$, where $\mathbf{W}\in\mathbb{R}^{d\times d'}$. It has been proved in [39] that η^* is bounded by:

$$
\eta_{\text{lower}}\leq\eta^*\leq\eta_{\text{upper}} \tag{28}
$$

where η_{lower} and η_{upper} are given by:

$$
\eta_{\text{lower}}=Tr\left(\mathbf{A}\right)/Tr\left(\mathbf{B}\right) \tag{29}
$$

$$
\eta_{\text{upper}}=\sum\nolimits_{i=1}^{d'}\alpha_i\Big/\sum\nolimits_{i=1}^{d'}\beta_i \tag{30}
$$

where $\alpha_i(1\leq i\leq d')$ are the first d' largest eigenvalues of $\mathbf{A}$ and $\beta_i(1\leq i\leq d')$ are the first d' smallest eigenvalues of $\mathbf{B}$. Following [39], we define a function:

$$
f(\eta)=\max_{\mathbf{W}^T\mathbf{W}=\mathbf{I}}Tr\left(\mathbf{W}^T(\mathbf{A}-\eta\mathbf{B})\mathbf{W}\right) \tag{31}
$$

Given a value $\eta=\eta_1$, the corresponding matrix $\mathbf{W}_1$ where $f(\eta_1)$ is reached is given by:

$$
\mathbf{W}_1=\underset{\mathbf{W}^T\mathbf{W}=\mathbf{I}}{\arg\max}Tr\left(\mathbf{W}^T(\mathbf{A}-\eta_1\mathbf{B})\mathbf{W}\right) \tag{32}
$$

which can be solved by eigenvalue decomposition.

The key to solving (27) is the following observation:

$$
\begin{aligned}
&\text{if } f(\eta_1)>0,\text{then } \eta^*>\eta_1 \\
&\text{if } f(\eta_1)<0,\text{then } \eta^*<\eta_1
\end{aligned}
\tag{33}
$$

The proof of (33) is given in Appendix A. From (28) and (33), we can employ binary search as in [39] to get the globally optimal η and then solve for $\mathbf{W}$ using (32).

The procedure of RDSR is summarized in Fig. 3.

Input:
unlabeled poses $\{\mathbf{x}_i\}(1 \leq i \leq N\}, \mathbf{x}_i \in \mathbb{R}^{1683}$
constraints $P = \{(\mathbf{x}_k, \mathbf{x}_l)|\mathbf{x}_k \text{ and } \mathbf{x}_l \text{ are similar}\}$
Output:
pose distance function $d(\mathbf{x}_i, \mathbf{x}_j)$

Procedure:
1) Compute $\mathbf{S}_W$, $\mathbf{S}_L$ and $\mathbf{S}_C$ according to (13), (18) and (23), respectively.
2) Define $\mathbf{A} = \mathbf{X}\mathbf{S}_C\mathbf{X}^T$ and $\mathbf{B} = \mathbf{X}\mathbf{S}_L\mathbf{X}^T + \alpha\mathbf{S}_W$.
3) Compute η_{lower} and η_{upper} according to (29) and (30), respectively.
4) **while** $\eta_{\text{upper}} - \eta_{\text{lower}} > \varepsilon$
   $\eta = 0.5 \times (\eta_{\text{upper}} + \eta_{\text{lower}})$
   if $\max_{\mathbf{W}^T\mathbf{W}=\mathbf{I}} Tr(\mathbf{W}^T(\mathbf{A} - \eta\mathbf{B})\mathbf{W}) > 0$
   $\eta_{\text{lower}} = \eta$
   else
   $\eta_{\text{upper}} = \eta$
   endif
   endwhile
5) Solve for optimal $\mathbf{W}^*$ given η by (32).
6) Return distance function as
$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W}^*\mathbf{W}^{*T} (\mathbf{x}_i - \mathbf{x}_j)}$.

Fig. 3. Summary of RDSR procedures.

4.4 Discussion

Our RDSR algorithm has some relations and differences with regard to other conventional algorithms.

4.4.1 Measure of data relationship

In many machine learning algorithms, a proper measure encoding data relationship is very important. Many well known distance metric learning methods can be formulated as optimization based on some similarity measure on training data points, which is typically expressed as a similarity graph or matrix [40].

For supervised methods, the similarity measure can be directly derived from labels. For unsupervised methods, the measure can only be inferred from the data itself. For semi-supervised methods, both supervised and unsupervised measures are used. The former enforces that the learned distance should be consistent with the labels, while the latter often exploits the structure of the data and enforces the consistency throughout the data manifold. This is the basic idea of many semi-supervised algorithms (including our RDSR). For example, by incrementing the criterion of LDA with a smoothness term derived from $k$nn, we get SDA (Semi-supervised Discriminant Analysis) [34].

A conventional method to build unsupervised measure of data relationship, such as in ISOMAP [41], LLE [42] and SDA [34], is $k$nn. Another common choice is to use some simple non-linear similarity functions such as:

$$M_{ij} \propto \exp\left(\|\mathbf{x}_i - \mathbf{x}_j\| \Big/ \sigma^2\right) \quad (34)$$

where σ is the bandwidth parameter. One problem of these conventional methods is that they assume the data relationship measure is solely dependent on the numeric $L2$ distance, which does not necessarily encode the intrinsic data similarity. Another disadvantage is that they are sensitive to the parameters [43].

In this paper, RDSR employs sparse representation (SR) to build the unsupervised similarity measure. The advantage of SR lies in two aspects. First, it is more discriminative. Second, SR allows adaptive neighborhood.

4.4.2 Trace ratio criterion

Many distance metric learning algorithms try to simultaneously maximize a term $Tr\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)$ and minimize another term $Tr\left(\mathbf{W}^T\mathbf{B}\mathbf{W}\right)$. LDA is an example, where $\mathbf{A}$ and $\mathbf{B}$ are the between-class and within-class scatter matrix, respectively. For these methods, a natural choice for the complete objective would be to maximize the ratio between the two traces, i.e. maximize $\frac{Tr\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)}{Tr\left(\mathbf{W}^T\mathbf{B}\mathbf{W}\right)}$. Conventionally, this trace-ratio criterion is approximated by ratio-trace $Tr\left(\frac{\mathbf{W}^T\mathbf{A}\mathbf{W}}{\mathbf{W}^T\mathbf{B}\mathbf{W}}\right)$, or determinant-ratio $\frac{\left|\mathbf{W}^T\mathbf{A}\mathbf{W}\right|}{\left|\mathbf{W}^T\mathbf{B}\mathbf{W}\right|}$ (where $|.|$ is the matrix determinant), because the latter two can be solved in closed-form by generalized eigenvalue decomposition [44]. However, as pointed out by [45], this kind of approximation deviates from the original objectives and may have negative effects on the results. In this paper, we directly optimize the trace-ratio objective.

5 Experiments on Motion Transition Decision

In this section we perform experiments on determining optimal motion transitions, which is important for many motion synthesis applications [9][10][11]. Given a set of motion sequences, new motions can be synthesized by linking different motion segments, where the "goodness" of the transition is judged by the distance between the two linking poses. In the following, Section 5.1 describes the data and evaluation methodology. Sections 5.2 to 5.5 analyse the results in different respects.

5.1 Experiment Setup

5.1.1 Training data

All data (training and testing) in this section is from CMU motion capture dataset. For the training data, we select some motion clips performed by several subjects. The selected motion clips contain almost 30 minutes data in total, and belong to a variety of types, including walking, running, jumping and modern dancing.

Note that because the motions are performed by different subjects, for each clip, we uniformly scale all the joint coordinates according to the body height (this is done for both training and testing clips)[2]. This helps to reduce the effects introduced by different builds.

2. Since each CMU motion clip has a corresponding T-pose, this normalization is straightforward.

During label generation, a frequent situation in machine learning happens: there are far more negative samples than positive samples, as only a very small proportion of possible pose pairs are suitable for transition. This unbalance is not a problem for RDSR, as RDSR does not rely on negative data. However, for many other methods that utilize both positive and negative data, the positive and negative data should be balanced. Moreover, for evaluation purpose, we also expect a balance between positive and negative data in the testing set.

A typical solution to this unbalance, such as in cascade and bootstrapping framework [47], is to use only a small amount of "difficult" negative samples by filtering out a large amount of "easy" ones by some (often simple) rules. In our case, this is also applicable, as *most negative pose pairs can be easily recognized by some simple criteria*, such as the mean Euclidean distance between corresponding 3D joints, which can be formulated as:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{K} \sum_{k=1}^{K} d_{\text{Euc}}\left(\mathbf{c}_i^k, \mathbf{c}_j^k\right) \tag{35}$$

where $\mathbf{c}_i^k$ is the 3D coordinate of the k^{th} joint in pose $\mathbf{x}_i$.

Specifically, all pose pairs are divided into two groups G_1 and G_0. A pose pair $(\mathbf{x}_i, \mathbf{x}_j)$ is put into G_1 if and only if $d(\mathbf{x}_i, \mathbf{x}_j) < \tau$. We set $\tau = 100$mm, and around 20 percent of the pairs are put in G_1. The pose pairs in G_0 (which are "easy" negative samples) are discarded, and the pairs in G_1 are sent to 5 persons with animation experience for manual labeling[3]. For each pose pair, the corresponding transition is displayed, and each person independently labels it as a good or bad transition. The final labeling is made in such a strategy that only the pairs labeled positively (negatively) by at least 4 persons are used as positive (negative) pairs. Other pairs are excluded. We adopt this relatively strict strategy to reduce the noise in labels. In this way we generate 500 positive (similar) pose pairs for training. Similarly, we also generate 500 negative (dissimilar) pose pairs.

We use $\mathcal{P} = \{(\mathbf{x}_k, \mathbf{x}_l) | \mathbf{x}_k \text{ and } \mathbf{x}_l \text{ are similar}\}$ and $\mathcal{Q} = \{(\mathbf{x}_k, \mathbf{x}_l) | \mathbf{x}_k \text{ and } \mathbf{x}_l \text{ are dissimilar}\}$ to denote the positive and negative pose pairs, respectively. Note that RDSR only uses positive pairs in $\mathcal{P}$. The negative pairs in $\mathcal{Q}$ are used by some other algorithms involved later.

All the original poses contained in the selected motion clips can be used as the unlabeled data. However, the number is too large, making the algorithm inefficient. Therefore, we randomly select 5000 poses as unlabeled training data. In Section 5.5 we will analyze the effect of unlabeled data, and show that the choice of 5000 unlabeled poses is suitable in this scenario.

5.1.2 Testing data

Testing data is generated in a similar way as training data. We select some other motion clips of walking, running, jumping and dancing performed by characters different from those in training data. Then, 500 positive and 500 negative pose pairs are generated for testing by another 5 persons different from those in acquiring labels for training data. We use $\mathcal{P}'$ and $\mathcal{Q}'$ to denote the positive and negative pairs for testing use, respectively.

Note that the motion data used in training and testing are performed by different subjects in CMU dataseet, and that the training and testing data are labeled by different human supervisors. This ensures that the result is not tuned to specific motion subjects or human judgers.

5.1.3 Evaluation Criterion

Given any pose distance function $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$, its performance on motion transition decision can be evaluated using the testing pose pairs $\mathcal{P}'$ and $\mathcal{Q}'$. The number of correctedly decided pairs of a pose distance d_{ij} at a given threshold δ is calculated by:

$$\begin{aligned} n_{d,\delta} = &\|\{(\mathbf{x}_i, \mathbf{x}_j) \,|\, (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}' \text{ and } d_{ij} < \delta\}\| \\ &+ \|\{(\mathbf{x}_i, \mathbf{x}_j) \,|\, (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{Q}' \text{ and } d_{ij} \geq \delta\}\| \end{aligned} \tag{36}$$

The final precision of a pose distance d_{ij} is the best correct proportion over all possible thresholds:

$$p_d = \max_{\delta} \frac{n_{d,\delta}}{\|\mathcal{P}'\| + \|\mathcal{Q}'\|} \tag{37}$$

This precision is determined by exhaustive search for optimal δ value.

5.2 Result of GPD+RDSR

Here we report the evaluation result using the proposed pose distance metric GPD+RDSR, which learns a distance using RDSR from GPD as pose features. Note that GPD features need to be normalized before any further processing. The normalization contains two steps. First, because GPD contains heterogeneous features, the normalization is conducted independently on each dimension by linearly transforming each dimension to span in the range [0, 1]. The transformation can be expressed as:

$$\mathbf{x}_i{}' = \mathbf{U}\mathbf{x}_i + \mathbf{v} \tag{38}$$

where $\mathbf{U} = \text{diag}(u_1, ..., u_d)$, $\mathbf{v} = [v_1, ..., v_d]^T$, and $\mathbf{x}_i{}'$ is the transformed feature vector. u_i and v_i are scaling and shifting constants for the i^{th} dimension, and can be easily determined by the maximum and minimum value on this dimension. After normalizing on each dimension, the feature vector of each pose is normalized to unit length in $\mathbb{R}^d$ space.

The rank of matrix $\mathbf{W}$ in Equation (27) is a parameter. The results at different ranks are plotted in Fig. 4. When the rank is very small (< 20), the precision is low, because such low-rank distance metric is too simple and not informative enough. On the other hand, as the rank becomes very large (> 100), the precision gradually

3. Here we can see another advantage of filtering out "easy" negative samples first. If such filtering is not performed, most samples sent to human labeling will be obviously negative, wasting the human labor.

TABLE 2
Time consumed on each computation step in training.

	Calculate $\mathbf{S}_W$	Calculate $\mathbf{S}_L$	Calculate $\mathbf{S}_C$	Calculate $\mathbf{A}$	Calculate $\mathbf{B}$	Final optimization
Time (sec)	3.6	5260	0.03	6.3	6.3	16.5

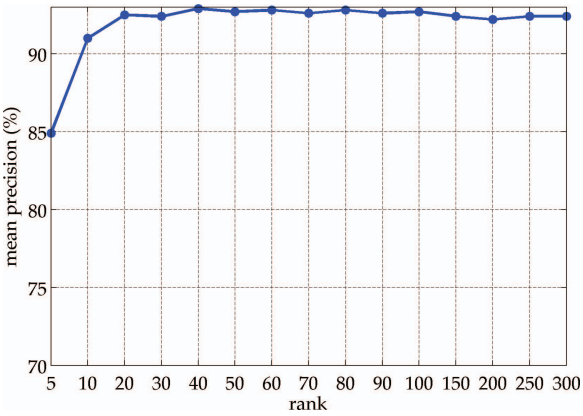

Fig. 4. Performance of GPD+RDSR on different ranks.

drops. This is because the learned metric is forced to be very similar to $L2$. As an extreme, if $\mathbf{W}$ is full ranked, then $\mathbf{W}\mathbf{W}^T$ becomes identity matrix, making the learned metric exactly $L2$. In order to determine the optimal rank, one conventional method is to use some criteria. One common criterion is Fisher criterion. The optimal rank in the experiments of this paper roughly ranges from 30 to 90, depending on the data. However, as the performance is relatively insensitive to the rank as shown in Fig. 4, we fix $rank(\mathbf{W}) = 50$ in all the experiments below.

Now we shift the attention to efficiency. The calculation of GPD feature vector on a pose is within 50 ms on highly non-optimized Matlab code. The optimization of the RDSR objective function (See Fig. 3) has $O(d^3)$ complexity due to the eigenvalue decomposition required to solve (32). Before this optimization, several calculation need to be done, including calculating $\mathbf{S}_W$ in (13), $\mathbf{S}_L$ in (18), $\mathbf{S}_C$ in (23) and calculating matrices $\mathbf{A}$ and $\mathbf{B}$ used in (27). Table 2 lists the actual time consumed in each step recorded on a PC with 3.2GHz CPU using the same data configuration as described in Section 5.1. It can be seen that the calculation of $\mathbf{S}_L$ takes most of the time. This computation needs to be done only once during training stage. In the testing stage, calculating the distance between two poses is very fast (typically takes 1 or 2 ms given GPD).

5.3 Comparing with Other Pose Distance Metrics

In this subsection we take an application-oriented view and compare our method (GPD+RDSR) with some other pose distances in computer animation literature.

Suppose each pose $\mathbf{x}_i$ is encoded by:

$$\mathbf{x}_i = \left(\mathbf{c}_i^0, \mathbf{r}_i^0, \mathbf{c}_i^1, \mathbf{r}_i^1, ..\mathbf{c}_i^m, \mathbf{r}_i^m\right) \tag{39}$$

where $\mathbf{c}_i^0 \in \mathbb{R}^3$ and $\mathbf{r}_i^0 \in \mathbb{S}^3$ are the 3D coordinate and orientation angles of the *Hip* joint, respectively, $\mathbf{c}_i^k \in \mathbb{R}^3$ and $\mathbf{r}_i^k \in \mathbb{S}^3$ are the 3D coordinates and orientation angles of the k^{th} joint, respectively, and $m+1$ is the number of joints (including *Hip*). Based on this representation, the considered pose distances are as follows.

- Orientation based Distances.
  1) Joint Orientation Distance (JOD).

$$d_{ij} = d_{\mathrm{E}}^2\left(\mathbf{c}_i^0, \mathbf{c}_j^0\right) + \sum_{k=0}^{m} d_{\mathrm{R}}^2\left(\mathbf{r}_i^k, \mathbf{r}_j^k\right) + \lambda \sum_{k=0}^{m} d_{\mathrm{E}}^2\left(\dot{\mathbf{r}}_i^k, \dot{\mathbf{r}}_j^k\right) \tag{40}$$

  where $\dot{\mathbf{r}}_i^k$ is the velocity of joint k in pose $\mathbf{x}_i$, $d_{\mathrm{E}}(.,.)$ calculates the Euclidean distance between joint coordinates or between joint velocities, $d_{\mathrm{R}}(.,.)$ calculates the distance of joint orientations in $\mathbb{S}^3$, and λ is the weighting parameter controlling the importance of velocity. It has been reported in [1] that the performance is insensitive to the value of λ and we follow their way by setting $\lambda = 1$.
  2) Weighted Joint Orientation Distance (WJOD).

$$d_{ij} = d_{\mathrm{E}}^2\left(\mathbf{c}_i^0, \mathbf{c}_j^0\right) + \sum_{k=0}^{m} w_k d_{\mathrm{R}}^2\left(\mathbf{r}_i^k, \mathbf{r}_j^k\right) + \lambda \sum_{k=0}^{m} w_k d_{\mathrm{E}}^2\left(\dot{\mathbf{r}}_i^k, \dot{\mathbf{r}}_j^k\right) \tag{41}$$

  This distance is similar to JOD defined in Equation (40), except that now each joint is associated with a weight w_k. This is the distance used in [11]. They set w_k to one for joints on shoulders, elbows, hips, knees, pelvis and spine, and set w_k to zero for other joints. We follow the same way.
  3) Learned Joint Orientation Distance (LJOD).
  This distance defined in [1] is in the same form as WJOD which is defined in (41). The difference is that, other than heuristically setting weights, the weights are learned from training pose pairs by least-squares minimization.

- Coordinate based Distances.
  1) Joint Coordinate Distance (JCD).

$$d_{ij} = \sum_{k=0}^{m} d_{\mathrm{E}}^2\left(\mathbf{c}_i^k, \mathbf{c}_j^k\right) + \lambda \sum_{k=0}^{m} d_{\mathrm{E}}^2\left(\dot{\mathbf{c}}_i^k, \dot{\mathbf{c}}_j^k\right) \tag{42}$$

  This is the coordinate-based counterpart of JOD.
  2) Weighted Joint Coordinate Distance (WJCD).

$$d_{ij} = \sum_{k=0}^{m} w_k d_{\mathrm{E}}^2\left(\mathbf{c}_i^k, \mathbf{c}_j^k\right) + \lambda \sum_{k=0}^{m} w_k d_{\mathrm{E}}^2\left(\dot{\mathbf{c}}_i^k, \dot{\mathbf{c}}_j^k\right) \tag{43}$$

This is the coordinate-based counterpart of WJOD, and we follow the same weight settings as WJOD.

3) Learned Joint Coordinate Distance (LJCD).
This is the coordinate-based counterpart of LJOD by learning the weights using LMS minimization, and we follow the same weight settings as LJOD.

- Feature based Distances.
  1) Joint Relative Features + LMS learning (JRF+LMS).

$$d_{ij} = \sum_{(u,v)} w_{u,v} \left\| d_{\mathrm{E}}\left(\mathbf{c}_i^u, \mathbf{c}_i^v\right) - d_{\mathrm{E}}\left(\mathbf{c}_j^u, \mathbf{c}_j^v\right) \right\| \quad (44)$$

This is the pose distance introduced in [3]. (u, v) are pairs of joints. Thus, (44) considers the weighted difference of Euclidean distances between joint pairs. The weights $w_{u,v}$ are learned from positive and negative pose pairs by least-mean-square minimization, similar to the learning of weights in [1].

2) Relational Geometric Features + Boost (RGF+Boost).

$$d_{ij} = \sum_{f_u \in \mathcal{F}} w_u \left\| f_u\left(\mathbf{x}_i\right) - f_u\left(\mathbf{x}_j\right) \right\|. \quad (45)$$

This is the pose distance introduced in [5]. $f_u \in \mathcal{F}$ denotes a feature in feature set $\mathcal{F}$, which is a huge pool (more than 500000). Adaboost is employed to select a small amount of features that are relevant. The weights w_u for the selected features are set to one and all other weights are zero.

3) GPD+RDSR.
The method proposed in this paper.

Note that JOD, WJOD, JCD and WJCD does not include a learning stage, so they don't utilize training data.

The comparison results are plotted in Figure 5. The first thing to notice is that GPD+RDSR does give the best precision. Comparing JOD, WJOD and LJOD, we can see that WJOD is better than JOD and LJOD is better than WJOD. This means that assigning different weights to joints does help, and that the weights learned from training data is better than those heuristically specified to one or zero. This is consistent with the observation made in [1]. The same trend can also be found for JCD, WJCD and LJCD. Also, notice that the overall performances of orientation based distances and coordinate distances are comparable. Regarding the feature based distances, JRF+LMS is comparable LJOD/LJCD, RGF+Boost is better than JRF+LMS, and GPD+RDSR is better than RGF+Boost.

5.4 Comparing with Other Features/Algorithms

The contribution of the paper lies in two aspects: the pose feature set GPD and the learning algorithm RDSR. The comparison in Section 5.3 demonstrates the advantage of GPD+RDSR. However, it is not clear whether both GPD and RDSR are helpful. In this subsection we answer this question by inspecting the performance of different pose features and distance learning methods. We consider four pose features:

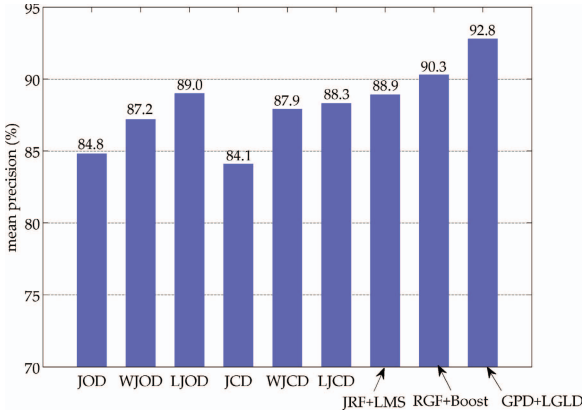

Fig. 5. Comparing different pose distances.

- JO: Joint Orientations.
- JC: Joint Coordinates.
- JRF: Joint Relative Features as defined in [3].
- GPD: Geometric Pose Descriptor proposed in this paper.

On the other hand, we consider five different distance learning algorithms:

- L2: $L2$ is used on corresponding pose feature vectors.
- LMS: This is the weighted $L2$ distance, with weights learned using the same method as in [1].
- Xing: This is the distance metric learning algorithm proposed by Xing et. al [28].
- SDA: This is the Semi-supervised Discriminant Analysis algorithm proposed by Cai et. al [34].
- RDSR: The algorithm proposed in this paper.

Note that among the above algorithms, $L2$ does not perform any learning. During training, LMS, Xing and SDA utilize both positive labels $\mathcal{P}$ and negative labels $\mathcal{Q}$, and RDSR utilizes only $\mathcal{P}$. On the other hand, LMS and Xing are supervised, while SDA and RDSR are semi-supervised.

Combining the pose features and the learning algorithms, the comparison results are shown in Fig. 6. First, comparing the rows in Fig. 6, we can see that RDSR is the best of the learning algorithms. Then, comparing the columns in Fig. 6, we can see that GPD is the best pose feature set.

This experiment shows that both GPD and RDSR make contributions. First, by representing each pose using GPD, we have a discriminative representation. Then, RDSR learns a distance metric based on the GPD feature vectors. The combination of GPD and RDSR gives the best performence among all the compared alternatives.

Note that RGF which is proposed in [5] is not included in this comparison, as its dimension (> 500000) makes it prohibitive for RDSR.

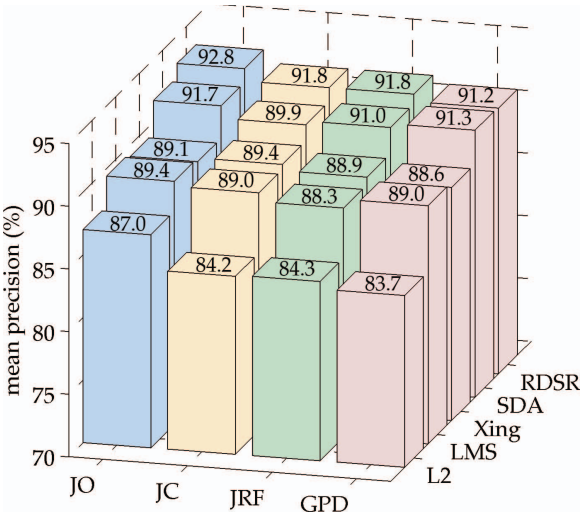

Fig. 6. Comparing with other features/algorithms.

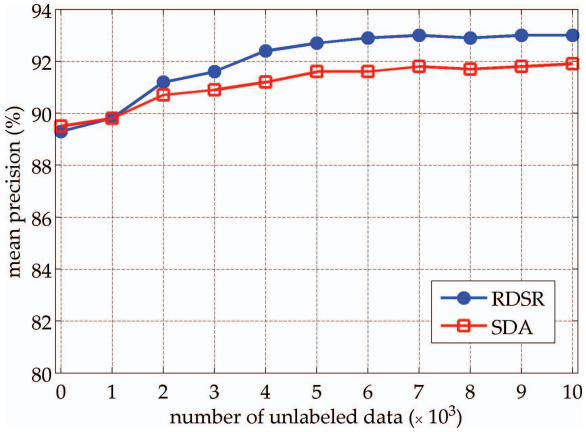

Fig. 7. Performance variation with different numbers of unlabeled data used in training.

5.5 Analyzing the Effect of Unlabeled Data

Fig. 6 gives an illustration that RDSR and SDA outperform other algorithms. Since RDSR and SDA are semi-supervised algorithms, it seems that unlabeled data does help, and a question naturally arises: how does the performance vary along with different amounts of unlabeled data? In this subsection we answer this question by giving an analyze on the effect of unlabeled data.

As mentioned above, 500 labeled pairs are used for RDSR and 1000 labeled pairs are used for SDA. Here, we fix the label data, and change the number of unlabeled data. Specifically, we randomly choose N' poses from the pose repertoire as the unlabeled training data and perform RDSR and SDA. The case of $N' = 0$ (where no unlabeled data is used and the algorithm becomes pure supervised) needs special attention. For SDA, it simply degenerates to traditional LDA [48] if $N' = 0$. For RDSR, $N' = 0$ means that the terms $E_{\text{regularization}}$ and $E_{\text{relationship}}$ are dropped from the objective function (26) and the objective becomes:

$$\mathbf{W}^* = \underset{\mathbf{W}^T\mathbf{W}=\mathbf{I}}{\arg\max} \left(\frac{1}{E_{\text{supervision}}} \right) = \underset{\mathbf{W}^T\mathbf{W}=\mathbf{I}}{\arg\min} Tr\left(\mathbf{W}^T \mathbf{S}_W \mathbf{W}\right) \tag{46}$$

which can be solved by SVD on $\mathbf{S}_W$.

The performance variation is plotted in Fig. 7, with the number of unlabeled data varies from 1000 to 10000. It is easy to see the improvements introduced by exploiting unlabeled data during training.

Fig. 7 can also serve as a support to our selecting 5000 unlabeled data during training. Further increasing the number of unlabeled data will not notably impact the precision but will increase the computational burden.

6 Experiments on Content Based Pose Retrieval

In this section we demonstrate the effectiveness of the proposed method in content based pose retrieval. Pose/motion retrieval is very important in many animation systems. As motion datasets often lack proper semantic annotations, animators often need to search for similar motions/poses scattered in the dataset given examples. On the other hand, evaluating similarity between motions is often based on evaluating the similarity between poses. Given an appropriate distance metric at pose-wise level, the similarity between two motion clips is typically evaluated at pose-wise level after alignment/wrapping in time axis [20][8]. Therefore, we focus on pose-wise level retrieval in this section. Specifically, given a query pose, the database poses are ranked according to the pose distance, and k nearest poses are returned as results.

6.1 Data

We still use CMU motion capture dataset. In this section we use a subset including motion clips from 15 subjects, which contains nearly 800000 poses.

The goal of content-based pose retrieval is different from deciding optimal transitions. In Section 5, we pay attention to the visual continuity between poses. Here, however, we pay attention to the pose semantics. For example, a moderate crouching pose is semantically similar to a deep crouching pose, but the two poses are negative for transition: linking them will generate significant visual discontinuity. In general, pose semantics put a more relaxed constraint on similarity: two poses can be notably numerically different, but they are still semantically similar. Actually, numerically very similar poses are of no challenge, as we know they should be semantically similar.

Therefore, considering the high frame rate of CMU dataset and repetitive nature of motions, we should only use a small subset of poses that differ from each other. Otherwise, if we used all the 800000 poses, the database would contain many very similar poses and all metrics will get very high precisions, making the comparison non-informative.

Guided by the above principal, for each subject, we select 200 poses by k-means clustering on the poses, using the simple pose distance as in (35). In this way, we get $200 \times 15 = 3000$ poses in total, on which the experiments are performed. This strategy ensures that: 1. The selected poses reasonably cover the diversity of poses; 2. The selected poses differ from each other (because they are k-means clustering centers). Note that all poses are rotated to the same yaw angle before k-means, because pose semantics is independent of the body's vertical rotation.

To acquire label information, some pose pairs are generated from the 3000 poses and are labeled as positive or negative. In this section, 1000 pairwise labels (500 positive and 500 negative) are used as supervision information in training.

6.2 Results and Discussions

Pose retrieval is conducted in such a way that for each query pose, its pose distance to each database pose is calculated and ranked accordingly. For each query, the top s database poses are returned (s is termed as the "scope" of the retrieval). During each case of retrieval, one pose from the 3000 poses is used as query example and the remaining 2999 poses are used as database poses to be retrieved. As there are no ground-truth data available to evaluate the retrieval performance, similar to many retrieval applications where the performance is measured by subjective evaluations, the retrieval results are judged by human. Each retrieved pose is marked as correct or incorrect and the precision is the percentage of correct results in the s returned results.

We perform 500 retrieval cases using four different pose distance metrics[4]: WJOD, JRF+LMS, RGF+Boost and GPD+RDSR, whose definitions are in Section 5.3. The results are shown in Fig. 8. GPD+RDSR outperforms other pose distance metrics in most cases. When scope $s = 5$, the performance of different methods does not vary significantly. This is because for each query there are typically a couple of poses in database that are very similar and easy to find even using a naive method. When the scope becomes larger (≥ 10), the performance difference becomes more notable.

Fig. 9 to Fig. 11 give some examples. In Fig. 9 the query example is a pose of raising the left leg taken from modern dance motion of subject 49. The top ten retrieval results of WJOD, RGF+Boost and GPD+RDSR are shown. Incorrectly returned poses are marked by dash ellipses. Both WJOD and RGF+Boost return several incorrect poses, while using GPD+RDSR all returned poses are correct. In this case, it is understandable that GPD provides more discriminative features than simple joint coordinates or rotations. For example, $f_{\text{LL\_a}}(L_{LHip \to LKnee}, L_{LKnee \to LFoot})$ (the angle between

4. Theoretically, we could perform 3000 cases. As the evaluation involves a lot of human labor, we just perform evaluation on 500 cases.

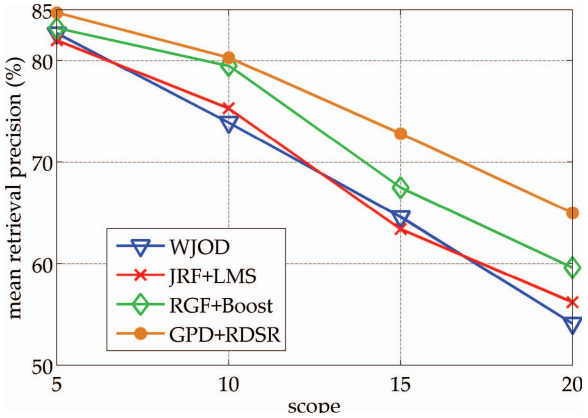

Fig. 8. Retrieval precision comparison of different distance metrics and scopes.

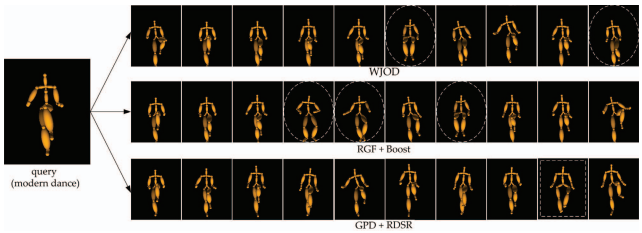

Fig. 9. Retrieval results for a modern dance pose. Incorrectly returned poses are marked by dash ellipses. The poses marked by dash rectangle are correctly returned pose with notable different skeletons.

left thigh and left calf), $f_{\text{JJ\_o}}(LFoot, RFoot)$ (the orientation between two feet) are potential effective features. Also, note that the 9th returned pose of GPD+RDSR (annotated with a dash rectangle) is correctly returned although its skeleton is notably different from the query example (the distances from *Hip* joint to both *LHip* and *RHip* are large).

Fig. 10 is a cartwheel pose of subject 81, where both WJOD and RGF+Boost return three incorrect poses and GPD+RDSR returns one. If we use joint coordinates or rotations, or some simple logical feature, a cartwheel pose might be recognized as a pose supported by the right foot and right hand. However, this simple criterion is not enough, as some incorrectly returned poses are also supported by the right foot and right hand. For GPD, the ambiguity is smaller. For example, the angle made between two forearms, the angle between the left (or right) arm and the torso plane are all potentially informative in this case.

Fig. 11 is another example, where the query is taken from "jumps, flips, breakdance" motion of subject 85. This is a difficult case, where half of the returned poses of WJOD and RGF+L2 are incorrect. GPD+RDSR performs better by returning three incorrect poses.

7 Conclusion

In this paper we have proposed a new pose distance metric on 3D motion data. First, poses are represented by

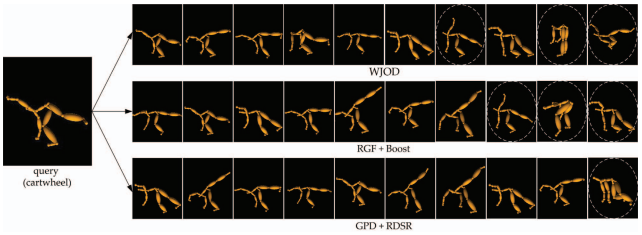

Fig. 10. Retrieval results for a cartwheel pose. Incorrectly returned poses are marked by dash ellipses.

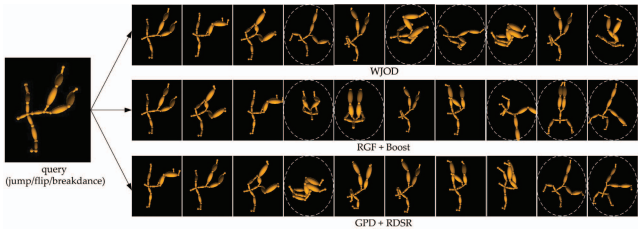

Fig. 11. Retrieval results for a flip/breakdance pose. Incorrectly returned poses are marked by dash ellipses.

GPD (*Geometric Pose Descriptor*) as a rich set of geometric features focusing on relations between body parts. Then, the distance metric is learned from the features by RDSR (*Regularized Distance Metric Learning with Sparse Representation*) by considering both labeled and unlabeled data.

We perform extensive experiments to evaluate our proposed GPD feature and RDSR algorithm on motion transition decision and content based pose retrieval. The proposed method can be applied to various 3D motion applications where evaluating pose similarity is needed, serving as a fundamental building block.

In the future we would like to develop a distance metric between motion clips based on the pose-wise distance proposed in this paper. We also plan to study on pose distance that is suited for identity recognition, i.e. recognizing the subject performing the motion.

APPENDIX A
PROOF OF (33) IN SECTION 4.3

Following notations in Section 4.3, first, we can prove:

$$\begin{aligned}&\frac{Tr\left(\mathbf{W}_1^T\mathbf{A}\mathbf{W}_\mathbf{1}\right)}{Tr\left(\mathbf{W}_1^T\mathbf{B}\mathbf{W}_\mathbf{1}\right)} \leq \max_{\mathbf{W}^T\mathbf{W}=\mathbf{I}} \frac{Tr\left(\mathbf{W}^T\mathbf{A}\mathbf{W}\right)}{Tr\left(\mathbf{W}^T\mathbf{B}\mathbf{W}\right)} = \eta^* \\ \Rightarrow &Tr\left(\mathbf{W}_1^T\mathbf{A}\mathbf{W}_1\right) - \eta^* \times Tr\left(\mathbf{W}_1^T\mathbf{B}\mathbf{W}_1\right) \leq 0 \\ \Rightarrow &Tr\left(\mathbf{W}_1^T(\mathbf{A}-\eta^*\mathbf{B})\mathbf{W}_1\right) \leq 0\end{aligned} \tag{47}$$

On one hand, $f(\eta_1)$ can be rewritten as:

$$\begin{aligned}f(\eta_1) &= \max_{\mathbf{W}^T\mathbf{W}=\mathbf{I}} Tr\left(\mathbf{W}^T(\mathbf{A}-\eta_1\mathbf{B})\mathbf{W}\right) \\ &= Tr\left(\mathbf{W}_1^T(\mathbf{A}-\eta_1\mathbf{B})\mathbf{W}_1\right) \\ &= Tr\left(\mathbf{W}_1^T(\mathbf{A}-\eta_1\mathbf{B}-\eta^*\mathbf{B}+\eta^*\mathbf{B})\mathbf{W}_1\right) \\ &= Tr\left(\mathbf{W}_1^T(\mathbf{A}-\eta^*\mathbf{B})\mathbf{W}_1\right) \\ &\quad + (\eta^*-\eta_1) \times Tr\left(\mathbf{W}_1^T\mathbf{B}\mathbf{W}_1\right)\end{aligned} \tag{48}$$

From (47) and (48), if $f(\eta_1) > 0$, then $(\eta^* - \eta_1) \times Tr\left(\mathbf{W}_1^T\mathbf{B}\mathbf{W}_1\right) > 0$. Considering that $Tr\left(\mathbf{W}_1^T\mathbf{B}\mathbf{W}_1\right) > 0$, we have the following observation:

$$\text{if } f(\eta_1) > 0, \text{then } \eta^* > \eta_1 \tag{49}$$

On the other hand, we have:

$$f(\eta_1) \geq Tr[\mathbf{W}^{*T}(\mathbf{A}-\eta^*\mathbf{B})\mathbf{W}^*] + (\eta^*-\eta_1)Tr(\mathbf{W}^{*T}\mathbf{B}\mathbf{W}^*) \tag{50}$$

Because $Tr\left(\mathbf{W}^{*T}(\mathbf{A}-\eta^*\mathbf{B})\mathbf{W}^*\right) = 0$, we have the following observation:

$$\text{if } f(\eta_1) < 0, \text{then } \eta^* < \eta_1 \tag{51}$$

This concludes the proof.

REFERENCES

[1] J. Wang, and B. Bodenheimer, "An Evaluation of a Cost Metric for Selecting Transitions between Motion Segments", *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 232-238, 2003.

[2] T. Harada, S. Taoka, T. Mori, and T. Sato, "Quantitative evaluation method for pose and motion similarity based on human perception. *Proc. IEEE/RAS International Conference on Humanoid Robots*, pp. 494-512, 2004.

[3] J. Tang, H. Leung, T. Komura, and H. Shum, "Emulating human perception of motion similarity", *Computer Animation and Virtual Worlds*; vol. 19, no. 3-4, pp. 211-221, 2008.

[4] E.S.L. Ho and T. Komura, "Indexing and retrieving motions of characters in close contact", *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 3, pp. 481-492, 2009.

[5] C. Chen, Y. Zhuang, J. Xiao, and Z. Liang, "Perceptual 3D pose distance estimation by boosting relational geometric features", *Computer Animation and Virtual Worlds*, vol. 20, no. 2-3, pp. 267-277, 2009.

[6] F. Liu, Y. Zhuang, F. Wu, Y. Pan, "3D motion retrieval with motion index tree", *Computer Vision and Image Understanding*, vol. 92, no. 2-3, pp. 265-284, 2003.

[7] E. Keogh, T. Palpanas, V. Zordan, D. Gunopulos, and M. Cardle, "Indexing large human-motion databases", *Proc. International Conference on Very Large Data Bases*, pp. 780-791, 2004.

[8] E. Hsu, M. Silva, and J. Popovic, "Guided time warping for motion editing", *Proc. Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (SCA)*, 2007.

[9] O. Arikan and D. Forsyth, "Motion generation from examples", *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 483-490, 2002.

[10] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs", *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 473-482, 2002.

[11] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard, "Interactive control of avatars animated with human motion data", *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 491-500, 2002.

[12] J. Barbic, A. Safonova, J. Pan, C. Faloutsos, J. K. Hodgins and N. S. Pollard, "Segmenting Motion Capture Data into Distinct Behaviors", *Proc. Graphics Interface*, pp. 185-194, 2004.

[13] C. Lu and N. J. Ferrier, "Repetitive Motion Analysis: Segmentation and Event Classification", *IEEE Transactions on Pattern Analysis and Meachine Intelligence*, vol. 26, no. 2, pp. 258-263, 2004.

[14] O. Arikan, "Compression of motion capture databases", *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 890-897, 2006.

[15] S. Chattopadhyay, S.M. Bhandarkar, and K. Li, "Human motion capture data compression by model-based indexing: a power aware approach", *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 1, pp. 5-14, 2007.

[16] O. Arikan, D.A. Forsyth, and J. OBrien, "Motion synthesis from annotations", *ACM Transactions on Graphics*, vol. 33, no. 3, pp. 402-408, 2003.

[17] P. T. Chua, R. Crivella, B. Daly, H. Ning, R. Schaaf, D. Ventura, T. Camill, J. Hodgins, and R. Pausch, "Training for physical tasks in virtual environments: Tai Chi", *Proc. IEEE Virtual Reality*, pp. 87-94, 2003.

[18] C. Chen, Y. Zhuang, J. Xiao, and F. Wu, "Adaptive and compact shape descriptor by progressive feature combination and selection with boosting", *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.
[19] CK-F So and G. Baciu, "Entropy-based motion extraction for motion capture animation: motion capture and retrieval", *Computer Animation and Virtual Worlds*, vol. 16, no. 3-4, pp. 225-235, 2005.
[20] M. Muller, T. Roder, and M. Clausen, "Efficient content-based retrieval of motion capture data", *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 677-685, 2005.
[21] S. Carlsson, "Combinatorial geometry for shape representation and indexing", *Object Representation in Computer Vision*, pp. 53-78, 1996.
[22] J. Sullivan, and S. Carlsson, "Recognizing and tracking human action". *Proc. European Conf. on Computer Vision*, pp. 629-644, 2002.
[23] T. Mukai, K. Wakisaka, and S. Kuriyama, "Generating concise rules for retrieving human motions from large datasets", *Computer Animation and Social Agents 2009 (CASA2009)*, Short Paper, 2009.
[24] L. Kovar and M. Gleicher, "Automated extraction and parameterization of motions in large datasets", *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 559-568, 2004.
[25] M. Muller, and T. Roder, "Motion templates for automatic classification and retrieval of motion capture data", *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 137-146, 2006.
[26] K. Onuma, C. Faloutsos, and J.K. Hodgins, "FMDistance: A fast and effective distance function for motion capture data", *Proc. Eurographics*, 2008.
[27] L. Yang and R. Jin, "Distance metric learning: a comprehensive survey", Technical Report, Michigan State University, 2006.
[28] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information", *Advances in Neural Information Processing Systems 15*, pp. 505-512, 2003.
[29] S. Xiang, "Learning a Mahalanobis distance metric for data clustering and classification", *Pattern Recognition*, vol. 41, no. 12, pp. 3600-3612, 2008.
[30] K.Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classfication", *The Journal of Machine Learning Research*, vol. 10, pp. 207-244, 2009.
[31] M.H. Nguyen and F. Torre, "Metric Learning for Image Alignment", *International Journal of Computer Vision*, 1573-1405, 2009.
[32] Y. Yang, Y. Zhuang, D. Xu, Y. Pan, D. Tao, S. J. Maybank, "Retrieval based interactive cartoon synthesis via unsupervised bi-distance metric learning", *Proc. ACM Multimedia*, pp. 311-320, 2009.
[33] X. Zhu, "Semi-Supervied Learning Literature Survey", Computer Sciences Technical Report, University of Wisconsin-Madison.
[34] D. Cai, X. He, and J. Han, "Semi-supervised Discriminant Analysis", *Proc. IEEE International Conference on Computer Vision*, 2007
[35] R. Tibshirani, "Regression Shrinkage and Selection via the LASSO", *Journal of the Royal Statistical Society*, vol. 58, no. 1, pp. 267-288.
[36] D. Donoho, "For most large underdetermined systems of linear equations the minimal l1-norm solution is also the sparsest solution", *Communications on Pure and Applied Mathematics*, vol. 59, no. 6, pp. 797-829, 2006.
[37] J. Wright, Y. Ma, J. Mairal, G. Spairo, T. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition", *Proc. IEEE International Conference on Computer Vision*, 2009.
[38] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210-227, 2009.
[39] F. Nie, S. Xiang, and C. Zhang, "Neighborhood MinMax Projections", *Proc. International Joint Conferences on Artificial Intelligence*, pp. 993-998, 2007.
[40] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin, "Graph Embedding and Extensions: A General Framework for Dimensionality Reduction", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40-51, 2009.
[41] J. Tenenbaum, V. de Silva, and J. Langford, "A global geometric framework for dimensionality reduction", *Science*, vol. 290, no. 5500, pp. 2319-2323, 2000.
[42] S. Roweis, and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding", *Science*, vol. 290, no. 22, pp.2323-2326, 2000.
[43] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration", *IEEE Transactions on Image Processing*, in press.
[44] R. Duda, P. Hart, and D. Stork, "Pattern classification (2nd edition)", Wiley-Interscience, 2000.
[45] H. Wang, S, Yan, D. Xu, X. Tang, T. Huang, "Trace ratio vs. ratio trace for dimensionality reduction", *Proc. IEEE International Conference on Computer Vision*, 2007.
[46] http://mocap.cs.cmu.edu/
[47] P. Viola, and M. Jones, "Rapid object detection using a boosted cascade of simple features", *Proc. IEEE International Conference on Computer Vision*, 2001.
[48] R. A. Fisher, "The use of multiple measurements in taxonomic problems", *Annals of Eugenics*, vol. 7, pp 179-188, 1936.

PLACE PHOTO HERE

Cheng Chen Cheng Chen received the PhD degree in computer science from Zhejiang University, China, in 2009. Since March 2010, he has been a postdoctoral researcher at Idiap Research Institute (affiliated to the Swiss Federal Institute of Technology at Lausanne, EPFL). His research interests include computer vision, 3D computer animation and pattern recognition.

Yueting Zhuang Yueting Zhuang got the PhD degree in computer science from Zhejiang University, China, in 1998. Now he is the full professor and dean of Computer Science Department, Zhejiang University. His research interests include digital library, computer graphics and multimedia analysis.

Feiping Nie Feiping Nie received his Ph.D. degree in Computer Science from Tsinghua University, China in 2009. His research interests include machine learning and its application fields, such as pattern recognition, data mining, image processing and information retrivial.

Yi Yang Yi Yang got his PhD degree from the Department of Computer Science at Zhejiang University in 2010. He is now a Post-doc research fellow at the University of Queensland, Australia. His research interests include machine learning and data mining and their applications to multimedia content analysis and computer vision.

Fei Wu Fei Wu received his PhD degree in computer science from Zhejiang University, China in 2002. Now he is an associate professor in Computer Science Department, Zhejiang University.

Jun Xiao Jun Xiao got the PhD degree in computer science from Zhejiang University, China in 2007. Now he is an associate professor in Computer Science Department, Zhejiang University.