AUTHOR QUERY FORM

ELSEVIER

Journal: YCVIU

Article Number: 1703

Please e-mail or fax your responses and any corrections to:

E-mail: corrections.essd@elsevier.sps.co.in

Fax: +31 2048 52799

Dear Author,

Please check your proof carefully and mark all corrections at the appropriate place in the proof (e.g., by using on-screen annotation in the PDF file) or compile them in a separate list.

For correction or revision of any artwork, please consult http://www.elsevier.com/artworkinstructions.

Any queries or remarks that have arisen during the processing of your manuscript are listed below and highlighted by flags in the proof. Click on the 'Q' link to go to the location in the proof.

Location in article	Query / Remark: click on the Q link to go Please insert your reply or correction at the corresponding line in the proof
Q1	Please check whether the designated corresponding author and e-mail address are correct, and amend if necessary.
Q2	Please provide a complete postal address for affiliation(s) 'a, b and c'.

Thank you for your assistance.

Research highlights

▶ An examplar-based framework for 3D human pose recovery from images. ▶ Improving the discriminant power by efficient visual feature selection. ▶ Sparse representation used to improve the pose retrieval accuracy. ▶ Both feature selection and sparse representation benefit the method.

Computer Vision and Image Understanding xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

3D human pose recovery from image by efficient visual feature selection

Cheng Chen [a,*], Yi Yang [b], Feiping Nie [c], Jean-Marc Odobez [a]

[a] IDIAP Research Institute, Martigny, Switzerland
[b] University of Queensland, Australia
[c] University of Texas, Arlington, USA

ARTICLE INFO

Article history:
Received 3 March 2010
Accepted 8 November 2010
Available online xxxx

Keywords:
Pose recovery
Feature selection
Motion understanding
Sparse representation

ABSTRACT

In this paper we propose a new examplar-based approach to recover 3D human poses from monocular images. Given the visual feature of each frame, pose retrieval is first conducted in the examplar database to find relevant pose candidates. Then, dynamic programming is applied on the pose candidates to recover a continuous pose sequence. We make two contributions within this framework. First, we propose to use an efficient feature selection algorithm to select effective visual feature components. The task is formulated as a trace-ratio criterion which measures the score of the selected feature component subset, and the criterion is efficiently optimized to achieve the global optimum. The selected components are used instead of the original full feature set to improve the accuracy and efficiency of pose recovery. As second contribution, we propose to use sparse representation to retrieve the pose candidates, where the measured visual feature is expressed as a sparse linear combination of the examplars in the database. Sparse representation ensures that semantically similar poses have larger probability to be retrieved. The effectiveness of our approach is validated quantitatively through extensive evaluations on both synthetic and real data, and qualitatively by inspecting the results of the real time system we have implemented.

1. Introduction

Recovering 3D human poses from marker-free images is a very important research subject in computer vision community. The success of pose recovery can directly benefit many applications such as video motion capture, natural human–computer interaction, and potentially many more.

Pose recovery aims at inferring the hidden pose parameter from the observed visual feature. This is challenging because the mapping from visual features to 3D poses is very complex and multi-modal. If poses are inferred only from monocular images, the 2D–3D ambiguity is more severe.

In this paper, we propose a new examplar based 3D pose recovery approach. The framework is shown in Fig. 1. The examplar database contains visual features and corresponding ground-truth poses. For each novel frame, the observed visual features are used to retrieve a set of pose candidates. Then, the optimal pose sequence is estimated from the pose candidates using dynamic programming, which exploits both feature cue and temporal cue to ensure the smoothness of the recovered poses. Within the above framework, we make two contributions, as described below.

1.1. Visual feature selection

Visual feature plays an important role in pose recovery, and we would like the visual features to be discriminative with respect to 3D poses as much as possible. To this end, a lot of features have been proposed using cues from silhouettes, edges and so on. Typically, each feature type contains multiple components. While most previous work uses all components of a particular feature type, in this paper we propose to select the optimal subset of feature components for pose recovery. First, the feature selection criterion is formulated in a trace-ratio form that measures the consistency of the intrinsic data relationships. Then, an efficient optimization step is performed to find the global optimal component subset. By using the selected components instead of all components, we can improve both accuracy and efficiency.

- *Accuracy*. Different visual feature components behave differently with regard to pose discrimination. For example, for Fourier descriptor [1], it has been shown that pose understanding depends more on some frequencies than on others [2]. By performing feature selection, we are able to discard irrelevant (or even misleading) components, achieving better accuracy.
- *Efficiency*. By performing feature selection, we are able to work with only a small proportion of the complete feature set. This is more efficient, because we only need to extract the selected

* Corresponding author.
E-mail address: cchen@idiap.ch (C. Chen).

1077-3142/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2010.11.007

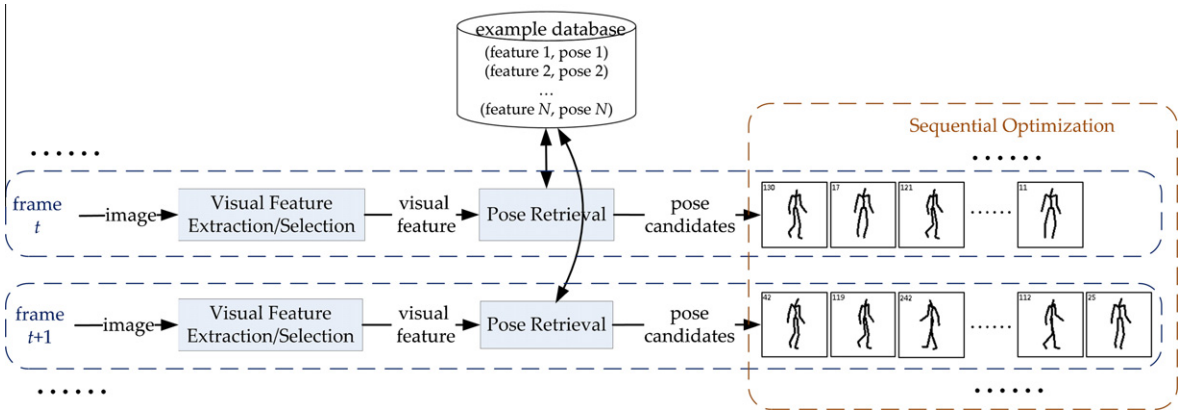

Fig. 1. Our examplar based pose recovery framework.

feature components from images. In addition, because the feature dimension is reduced, subsequent procedures can be accelerated.

There exists some other work on feature selection for pose recovery. For example, Ren et al. [3] and Chen et al. [4] employ Adaboost to select effective features. However, Adaboost requires a large amount of computation, and it cannot guarantee that the global optimum is reached. On the other hand, our feature selection approach identifies the globally optimal feature subset in a much faster way.

1.2. Pose retrieval via sparse representation

An important step in our examplar-based approach is pose retrieval. Because the mapping from visual features to poses is multi-modal, it is not desirable to simply use the topmost match. Instead, a set of pose candidates is retrieved for each frame. Traditionally, this is often performed by nearest neighbor (NN) method [3–5]. In this paper we propose to use sparse representation [6,7]. Specifically, the visual feature vector of a novel frame is reasonably approximated using only a small percentage of features in the examplar database. Compared to NN, sparse representation tends to select poses that are more semantically relevant. Another advantage is that the neighborhood size for sparse representation is adaptive rather than fixed.

We perform extensive experiments on both synthetic and real data, using various types of visual features. The experimental results show that compared to previous methods, our method achieves higher accuracy in pose recovery and significantly reduces the computation time.

The paper is organized as follows. After summarizing the research background in Section 2, we introduce visual feature selection in Section 3. Section 4 presents pose retrieval via sparse representation and sequential optimization by dynamic programming. Experiments are given in Section 5 and conclusions are in Section 6.

2. Related work

This paper performs pose recovery by feature selection, and thus is related to the two fields summarized below.

2.1. Image based pose recovery

There have been numerous publications on image based human pose recovery [8–10]. The approaches can be divided as generative (model based) and discriminative (learning based). Generative methods exploit the fact that although the mapping from visual features to poses is hidden and complex, the reverse mapping is often well-posed. Therefore, pose recovery is tackled by optimizing an object function that encodes the pose-feature correspondence [11], or by sampling posterior pose probabilities [12,13]. On the other hand, discriminative methods directly learn a mapping from visual features to pose parameters. The mapping is often approximated using regression models [14]. Alternatively, one can directly rely on a dense training database, leading to the so-called "examplar-based" methods [15,3,5,16].

Generally speaking, generative methods are more accurate, but the computation is often expensive. On the other hand, discriminative methods are more efficient. There also exist some hybrid methods [17,18].

Our work in this paper is based on the examplar-based pose recovery framework, with contributions in visual feature selection and pose retrieval.

2.2. Visual features for human motion analysis

A lot of features have been proposed for human motion analysis. Many are based on silhouettes, such as Fourier descriptor [1], shape contexts [19,20], geometric signature [21], Hu moments [22], Poisson features [23] and so on. There are also features based on edges or gradients, such as histogram of oriented gradients [24], relational edge distribution [25] and various SIFT-like features [26,27]. Hierarchical features have also been proposed, such as HMAX [28], Vocabulary Tree [30], Hyperfeatures [29] or Spatial Pyramid matching [31]. Space-time interest points [32–34] are also exploited.

Since there are many feature choices, a question naturally arises: how to determine the most suitable features for pose recovery? This can be answered in two ways. On one hand, comparative studies are conducted to evaluate the performance of different feature types. For example, Poppe and Poel [35] compare Fourier descriptor, shape contexts and Hu moments in 3D pose recovery. Chen et al. [36] compare various silhouette-based features. On the other hand, another step is to select effective feature components via machine learning. For example, Ren et al. [3] select silhouette feature components from a huge pool of Harr-like features. Chen et al. [4] perform feature component selection using Adaboost.

Another approach related to feature selection is subspace learning (also known as dimension reduction). It aims to find the transformation from the original feature space to a low dimensional subspace that retains most of the discriminative information. Though the dimension in the learned subspace is usually much lower than the input feature dimension, the full feature set still

Please cite this article in press as: C. Chen et al., 3D human pose recovery from image by efficient visual feature selection, Comput. Vis. Image Understand. (2010), doi:10.1016/j.cviu.2010.11.007

has to be extracted for novel data to obtain its subspace embedding. In feature selection framework, however, given a novel data sample, we only need to extract the selected feature components, making feature extraction and subsequent procedures much faster. Moreover, the time complexity of most subspace learning algorithms is $O(d^3)$, where d is the dimension of input feature space. Our feature selection algorithm, however, is faster ($O(d\log(d))$).

3. Visual feature selection for pose discrimination

3.1. Formulation

Suppose we have a set of N data samples $\Phi = \{\phi_1, \phi_2, \ldots, \phi_N\}$ in the examplar database. Each data sample is $\phi_i = (\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the image feature vector and $\mathbf{y}_i \in \mathbb{R}^p$ is the ground-truth 3D pose parameter. Let $\mathcal{C} = \{c_1, c_2, \ldots c_d\}$ be the set of feature components, where c_i is the ith component. Our goal is to select a feature component subset $\widetilde{\mathcal{C}} \subset \mathcal{C}$ consisting of $d' = |\widetilde{\mathcal{C}}|$ ($d' < d$) components (dimensions), where $|.|$ represents the cardinality of a set. For each possible $\widetilde{\mathcal{C}}$, there is a corresponding feature selection function:

$$\tilde{\mathbf{x}} = g_{\widetilde{\mathcal{C}}}(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the original visual feature vector, and $\tilde{\mathbf{x}} \in \mathbb{R}^{d'}$ is the selected feature.

The principle of our approach is to select features such that close data in the pose space are also close in the feature space, and vice versa. To encode the data relationships, we randomly generate a set $\mathcal{P}$ containing M data pairs $\mathcal{P} = \{(\mathbf{x}_{i1}, \mathbf{x}_{j1}), \ldots, (\mathbf{x}_{iM}, \mathbf{x}_{jM})\}$, where $\mathbf{x}_{ip} \in \Phi$ and $\mathbf{x}_{jp} \in \Phi$ are two different data points from the training set. In addition, we define a function $f_a(\mathbf{y}_i, \mathbf{y}_j)$ that reflects the pairwise pose similarity. That is, $f_a(\mathbf{y}_i, \mathbf{y}_j)$ is large if and only if $\mathbf{y}_j$ and $\mathbf{y}_j$ are similar. Similarly, we also define a function $f_b(\mathbf{y}_i, \mathbf{y}_j)$ encoding pairwise data dissimilarity. That is, $f_b(\mathbf{y}_i, \mathbf{y}_j)$ is large if and only if $\mathbf{y}_i$ and $\mathbf{y}_j$ are dissimilar.

Using the above notation, our principle can be formulated in two ways. First, similar poses (i.e. $f_a(\mathbf{y}_i, \mathbf{y}_j)$ is large) should correspond to close features, and hence we want to minimize $\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}} \left(\|g_{\widetilde{\mathcal{C}}}(\mathbf{x}_i) - g_{\widetilde{\mathcal{C}}}(\mathbf{x}_j)\|^2 f_a(\mathbf{y}_i, \mathbf{y}_j) \right)$. On the other hand, dissimilar poses (i.e. $f_b(\mathbf{y}_i, \mathbf{y}_j)$ is large) should correspond to faraway features, and hence we want to maximize $\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}} \left(\|g_{\widetilde{\mathcal{C}}}(\mathbf{x}_i) - g_{\widetilde{\mathcal{C}}}(\mathbf{x}_j)\|^2 f_b(\mathbf{y}_i, \mathbf{y}_j) \right)$. One way to combine the two above goals into a single criterion is to define an objective function as the ratio between the two:

$$\max_{\widetilde{\mathcal{C}}} \frac{\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}} \left(\left\| g_{\widetilde{\mathcal{C}}}(\mathbf{x}_i) - g_{\widetilde{\mathcal{C}}}(\mathbf{x}_j) \right\|^2 f_b(\mathbf{y}_i, \mathbf{y}_j) \right)}{\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}} \left(\left\| g_{\widetilde{\mathcal{C}}}(\mathbf{x}_i) - g_{\widetilde{\mathcal{C}}}(\mathbf{x}_j) \right\|^2 f_a(\mathbf{y}_i, \mathbf{y}_j) \right)},$$

$$\text{s.t. } \widetilde{\mathcal{C}} \subset \mathcal{C} \quad \text{and } |\widetilde{\mathcal{C}}| = d'. \tag{2}$$

3.2. Pairwise pose (dis)similarity functions

Now we detail how to define the pairwise functions $f_a(\mathbf{y}_i, \mathbf{y}_j)$ and $f_b(\mathbf{y}_i, \mathbf{y}_j)$. They should be defined using $d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j)$, which is the distance function in the pose parameter space. $d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j)$ is generally calculated by the difference of 3D marker coordinates or rotational joint angles, depending on the motion data format. (Please see Section 5.1.2 for a explanation of pose distance used in the experiments of this paper.)

A simple definition is by thresholding:

$$f_a(\mathbf{y}_i, \mathbf{y}_j) = \begin{cases} 0, & \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) > \tau, \\ 1, & \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) \leqslant \tau, \end{cases}$$
$$f_b(\mathbf{y}_i, \mathbf{y}_j) = \begin{cases} 1, & \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) > \tau, \\ 0, & \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) \leqslant \tau, \end{cases} \tag{3}$$

where τ is the threshold on pose distance.

The functions in (3) consider each similar or dissimilar pose pair with the same weight, and we call them hard pairwise functions. In fact, this has some disadvantages. For example, within the similar pairs, some pairs may be more "similar" than others, and they should play a more important role. In order to address this consideration, we propose the following soft pairwise functions:

$$f_a(\mathbf{y}_i, \mathbf{y}_j) = \exp\left(-d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j)/\sigma^2\right),$$
$$f_b(\mathbf{y}_i, \mathbf{y}_j) = \begin{cases} 0, & \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) \leqslant \tau, \\ \left(d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j)/d_{\max}\right)^2, & \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) > \tau, \end{cases} \tag{4}$$

where $d_{\max}$ is the maximum value of all pose pairs in $\mathcal{P}$.

3.3. Optimization

Now we describe the optimization of the feature selection criterion in (2). For the ease of presentation, we note that the feature selection function can be written in matrix form as:

$$\tilde{\mathbf{x}} = g_{\widetilde{\mathcal{C}}}(\mathbf{x}) = \mathbf{S}_{\widetilde{\mathcal{C}}}^T \mathbf{x}, \tag{5}$$

where $\mathbf{S}_{\widetilde{\mathcal{C}}} \in \mathbb{R}^{d \times d'}$ is the selection matrix corresponding to the feature subset $\widetilde{\mathcal{C}}$. Suppose the selected subset is defined as $\widetilde{\mathcal{C}} = \{c_{I_1}, c_{I_2}, \ldots, c_{I_{d'}}\}$, where $I_1, \ldots, I_{d'}$ are the indices of selected feature components. Then $\mathbf{S}_{\widetilde{\mathcal{C}}}$ is defined as:

$$\mathbf{S}_{\widetilde{\mathcal{C}}} = \left[\mathbf{s}_{I_1}, \mathbf{s}_{I_2}, \ldots, \mathbf{s}_{I_{d'}}\right], \tag{6}$$

where the kth column $\mathbf{s}_{I_k}$ is defined as: $\mathbf{s}_{I_k} = [\mathbf{0}_{I_k - 1}, 1, \mathbf{0}_{d - I_k}]^T$. where $\mathbf{0}_n$ is the row vector of n zeros. That is, all components of $\mathbf{s}_{I_k}$ except the I_k^{th} one are zero.

Substituting (5) into (2), the criterion can be written as:

$$\max_{\widetilde{\mathcal{C}}} \frac{\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}} \left((\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}_{\widetilde{\mathcal{C}}} \mathbf{S}_{\widetilde{\mathcal{C}}}^T (\mathbf{x}_i - \mathbf{x}_j) f_b(\mathbf{y}_i, \mathbf{y}_j) \right)}{\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}} \left((\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}_{\widetilde{\mathcal{C}}} \mathbf{S}_{\widetilde{\mathcal{C}}}^T (\mathbf{x}_i - \mathbf{x}_j) f_a(\mathbf{y}_i, \mathbf{y}_j) \right)},$$
$$\text{s.t.} \quad \widetilde{\mathcal{C}} \subset \mathcal{C} \quad \text{and } |\widetilde{\mathcal{C}}| = d', \tag{7}$$

which can be written more compactly as:

$$\max_{\widetilde{\mathcal{C}}} \frac{Tr\left(\mathbf{S}_{\widetilde{\mathcal{C}}}^T \mathbf{U}_b \mathbf{S}_{\widetilde{\mathcal{C}}}\right)}{Tr\left(\mathbf{S}_{\widetilde{\mathcal{C}}}^T \mathbf{U}_a \mathbf{S}_{\widetilde{\mathcal{C}}}\right)}, \quad \text{s.t. } \widetilde{\mathcal{C}} \subset \mathcal{C} \quad \text{and } |\widetilde{\mathcal{C}}| = d', \tag{8}$$

where $Tr(.)$ is the trace operator. $\mathbf{U}_a$ in the above equation is defined as:

$$\mathbf{U}_a = \sum_{(i,j) \in \mathcal{P}} \left((\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T f_a(\mathbf{y}_i, \mathbf{y}_j) \right), \tag{9}$$

and $\mathbf{U}_b$ is defined similarly.

The Brute force search for the optimal subset $\widetilde{\mathcal{C}}$ in (8) is clearly prohibitive, because the searching space is factorial on the number of feature components. We propose to use an efficient algorithm that searches for the global optimal $\widetilde{\mathcal{C}}$ iteratively [37]. The procedure is summarized in Fig. 2. For details, please refer to Appendix A or [37]. It is easy to deduce from Fig. 2 that the computation complexity of the algorithm is $O(d\log(d))$.

Please cite this article in press as: C. Chen et al., 3D human pose recovery from image by efficient visual feature selection, Comput. Vis. Image Understand. (2010), doi:10.1016/j.cviu.2010.11.007

Input:

Data samples $\Phi = \{\phi_1, \phi_2, ..., \phi_N\}$, $\phi_i = (\mathbf{x}_i, \mathbf{y}_i)$

Pairs $\mathcal{P} = \{(\mathbf{x}_{i1}, \mathbf{x}_{j1}), ..., (\mathbf{x}_{iM}, \mathbf{x}_{jM})\}$

Output: Feature component subset $\tilde{C} \in 2^C$, $|\tilde{C}| = d'$.

Procedure:

1. Compute $\mathbf{U}_a$ and $\mathbf{U}_b$ according to (9).
2. Initialize $\tilde{C}$ randomly, and set $\mathbf{S}_{\tilde{C}}$ according to (6).
3. Calculate $\lambda = Tr\left(\mathbf{S}_{\tilde{C}}{}^T \mathbf{U}_b \mathbf{S}_{\tilde{C}}\right) \Big/ Tr\left(\mathbf{S}_{\tilde{C}}{}^T \mathbf{U}_a \mathbf{S}_{\tilde{C}}\right)$
4. For i=1 to d, calculate the scores $sc_i = \mathbf{s}_i^T(\mathbf{U}_b - \lambda \mathbf{U}_a)\mathbf{s}_i$
5. Sort components according to sc_i in descent order. Update $\tilde{C}$ with the first d' components.
6. Update $\mathbf{S}_{\tilde{C}}$ according to (6).
7. Repeat steps 3 to 6 until convergence.

Fig. 2. Visual feature selection algorithm.

4. Pose recovery via sparse representation and dynamic programming

Using the feature selection algorithm introduced in the previous section, we get the best feature component subset for pose recovery. Therefore, pose retrieval in Fig. 1 is conducted using the vector of selected feature components instead of the full feature set. Because the mapping from visual features to poses is highly complicated and multi-modal, we cannot expect that a single topmost match will always correspond to the correct pose. Instead, for each frame, we retrieve a set of pose candidates using the sparse representation approach. Then, sequential optimization is performed by dynamic programming to generate a continuous pose sequence based on the candidates of each frame.

4.1. Pose retrieval via sparse representation

We employ sparse representation (SR) [6,38] to find the pose candidates for each frame. It is reported in [6] that SR generally outperforms nearest neighbor method. Another advantage of SR is that the neighborhood size (number of candidates) is adaptive rather than fixed.

Let $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_N] \in \mathbb{R}^{d' \times N}$ denote the matrix of all visual features in the examplar database, and let $\tilde{\mathbf{x}}_t \in \mathbb{R}^{d'}$ be the novel visual feature of frame t whose 3D pose candidates are to be retrieved. $\tilde{\mathbf{x}}_t$ can be approximated as a combination of examplar features:

$$\tilde{\mathbf{x}}_t \approx w_{t,1}\tilde{\mathbf{x}}_1 + w_{t,2}\tilde{\mathbf{x}}_2 + \cdots + w_{t,N}\tilde{\mathbf{x}}_N = \tilde{\mathbf{X}}\mathbf{w}_t, \tag{10}$$

where $w_{t,1}, \ldots, w_{t,N} \geqslant 0$ are the positive reconstruction weights of $\tilde{\mathbf{x}}_t$, and $\mathbf{w}_t = [w_{t,1}, \ldots, w_{t,N}]^T$ is the reconstruction weight vector. The residual error is $\|\tilde{\mathbf{x}}_t - \tilde{\mathbf{X}}\mathbf{w}_t\|$.

On one hand, we want the residual error to be small. On the other hand, we expect the reconstruction weight vector $\mathbf{w}_t$ to be sparse, i.e. $\tilde{\mathbf{x}}_t$ should be reasonably approximated by only a small subset of examplar visual features. If no constraint was enforced on the sparsity, then $\tilde{\mathbf{x}}_t$ would be approximated with a very small residual error by dense weights that are not informative on the intrinsic data relationship. Therefore, our goal is to solve the following problem:

$$\mathbf{w}_t = \arg\min\left(\left\|\tilde{\mathbf{x}}_t - \tilde{\mathbf{X}}\mathbf{w}_t\right\| + \gamma|\mathbf{w}_t|_1\right), \quad \text{s.t. } w_{t,1}, \ldots w_{t,N} \geqslant 0, \tag{11}$$

where $|.|_1$ is the L_1 norm. The optimization in (11) can be solved efficiently by Lasso [39]. In this way, for each frame, we retrieve a set of pose candidates as the items in the examplar database with non-zero weights.

The regularizer γ in Eq. (11) controls the weight of the sparsity constraint and consequently influences the average number of neighbors for each data sample. In this paper we fix $\gamma = 50$, which produces around 10–20 neighbors for each data sample averagely.

4.2. Sequential optimization via dynamic programming

The next step is to conduct sequential optimization from the pose candidates of each frame to get a continuous pose sequence. We use a graph model depicted in Fig. 3 to illustrate the problem. Node $\mathbf{y}_{t,c}$ represents the cth pose candidate of frame t, and $w_{t,c}$ is the corresponding weight derived from (11). Nodes from successive frames are connected by edges with weights. Using this model, we define a path H as:

$$H = h(1)h(2)\ldots h(T), \tag{12}$$

where $h(t)$ is the index of pose candidate at frame t which is selected as the recovered pose. Different cues are used to recover the best path.

- *Feature cue*. $\mathbf{y}_{t,h(t)}$ should be consistent with the visual feature at frame t. This is encoded in the weight $w_{t,h(t)}$. For frame t, the normalized weights are used as the feature score $f_{h(t)}$:

$$f_{h(t)} = w_{t,h(t)} \Big/ \sum_c w_{t,c}, \tag{13}$$

- *Temporal cue*. The recovered poses from successive frames should not change abruptly. That is, $d_{\text{pose}}(\mathbf{y}_{t,h(t)}, \mathbf{y}_{t+1,h(t+1)})$ should be relatively small. This is encoded in edge weights (transition scores). From frame t to $t+1$, the transition score $o_{h(t),h(t+1)}$ is:

$$o_{h(t),h(t+1)} = \exp\left(-d_{\text{pose}}\left(\mathbf{y}_{t,h(t)}, \mathbf{y}_{t+1,h(t+1)}\right)/\sigma^2\right), \tag{14}$$

where σ is a parameter (we set $\sigma = 5$ in this paper). Using the above model, the score of a path H is the total feature scores of all nodes it passes by, plus the total transition scores of all edges it traverses:

$$score(H) = \sum_{t=1}^{T} f_{h(t)} + \alpha \sum_{t=1}^{T-1} o_{h(t),h(t+1)}, \tag{15}$$

where α is the weighting parameter controlling the importance of temporal cue. As extremes, if $\alpha = 0$, then temporal cue is not

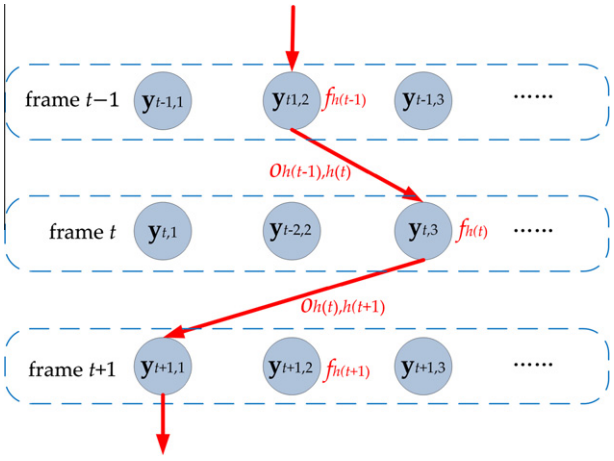

Fig. 3. Graph model for sequential optimization. This figure shows a segment of path $H = \ldots h(t-1)h(t)h(t+1)\ldots$, where $h(t-1) = 2$, $h(t) = 3$ and $h(t+1) = 1$. The feature scores and transition scores are annotated beside the corresponding nodes and edges, respectively.

considered at all, and for each frame we just select the candidate with the highest feature score. If $\alpha \to \infty$, then feature cue is not considered. In this paper we set $\alpha = 1$. The global optimal pose path can be efficiently found by dynamic programming. In practice, there are some differences, depending on whether the image sequence is available all at once or incrementally.

- *Batch system*. If all video frames from 1 to T are known before sequential optimization, the global optimal path can be recovered. Note that under this model it is very convenient to incorporate user defined constraints. For example, user may specify the correct pose for a frame. This can be treated as hard constraints by forcing the path to pass the specified nodes. This makes the maintenance of pose paths very intuitive.
- *Online system*. For online systems, such as real-time human-computer interaction, at each time t, we can derive the optimal path up to frame t, and the traversed pose at frame t is displayed to the user as the recovered pose. Note that when new frames at time $t+1$, $t+2$, ... arrive, the optimal path before time t may change, and this may cause discontinuity in the online pose display. However, the action of "changing the history" occurs only occasionally and can be viewed as automatic correction from errors when additional information is available (see Section 5.4 in the experiments).

5. Experiments

Now we present experimental results. Conceptually, the approach proposed in this paper consists of three parts: visual feature selection, pose retrieval, and sequential optimization. In the following we evaluate each part and show their effectiveness. Specifically, Sections 5.1 and 5.2 evaluate our visual feature selection method. Section 5.3 fixes the visual features and sequential optimization strategy, and evaluates our pose retrieval method via sparse representation. Section 5.4 fixes the visual features and pose retrieval method, and evaluates our sequential optimization strategy. Finally, Section 5.5 demonstrates the effectiveness of the proposed approach as a whole by implementing real systems.

In the following, both Sections 5.1 and 5.2 use HumanEva dataset [40] to evaluate feature selection, where Section 5.1 uses synthetic images to evaluate the impact of unusual appearances, and Section 5.2 uses real images to evaluate in practical situations with noise.

5.1. Evaluating feature selection on synthetic data

5.1.1. Data

HumanEva contains data of five motion types, namely *boxing*, *gestures*, *jog*, *throw-catch* and *walking* performed by three subjects. For each motion type and each subject, there are three trials. Trial 1 contains synchronized video and motion data, and is split as training partition and validation partition. Trial 2 contains only video data and is used for testing. Trial 3 contains only motion data and is used for learning motion priors. In this subsection, we use the motion data (3D poses) from trial 3 of all subjects, and the corresponding images are synthesized by retargeting the poses to 3D characters.

Due to the high frame rate and the repetitive nature of motions, the dataset contains a lot of very similar poses. In order to make pose recovery more efficient, we generate only a subset of the 3D poses. First, we rotate all 3D poses in the data to 0° yaw angle (i.e. in frontal view respect to the camera). Then, k-means is employed on the resulting poses with $k = 300$. For each of these 300 pose configuration, we generate 24 poses by cycling the yaw angle from 0° to 345° with 15° interval. Thus we generate $300 \times 24 = 7200$ poses. Then, each pose is targeted to eight characters by MotionBuilder (see Fig. 4). In this way, we generate $7200 \times 8 = 57{,}600$ images. They typically have unusual appearances (e.g. exaggerated head, helmet, gun in hand).

The ground-truth poses of these 57,600 images are trivially known. The image features are based on silhouettes and are composed of several feature types as follows (see [36] for a more detailed discussion of these features):

- *Occupancy map* [41]: Human body's bounding box is divided evenly into 12×8 cells, and the percentages of foreground pixels in each cell are used as features. This generates $12 \times 8 = 96$ feature components.
- *Contour signature* [21]: It is defined by some geometric quantity measured along the silhouette border, which starts from the topmost point and is followed in a clockwise manner. We use the following three measures: coordinates, distances to centroid, and tangent angles. For each silhouette, we uniformly sample 64 points along the contour. Therefore the total dimension is $128 + 64 + 128 = 320$.
- *Fourier descriptor* [1]: It consists of the normalized Fourier coefficients at different frequencies obtained by applying Discrete Fourier Transform to the contour signatures introduced above. It contains $62 + 32 + 62 = 156$ feature components.
- *Hu moments* [22]: They consist of seven moment based features calculated by treating the shape as a two-dimensional density distribution.
- *Shape contexts* [19,20]: We use the histogram of shape contexts constructed from 12 angular bins and five radial bins, and the size of codebook is 100.
- *Poisson features* [23]: We use 30 moment based features calculated on local Poisson features.

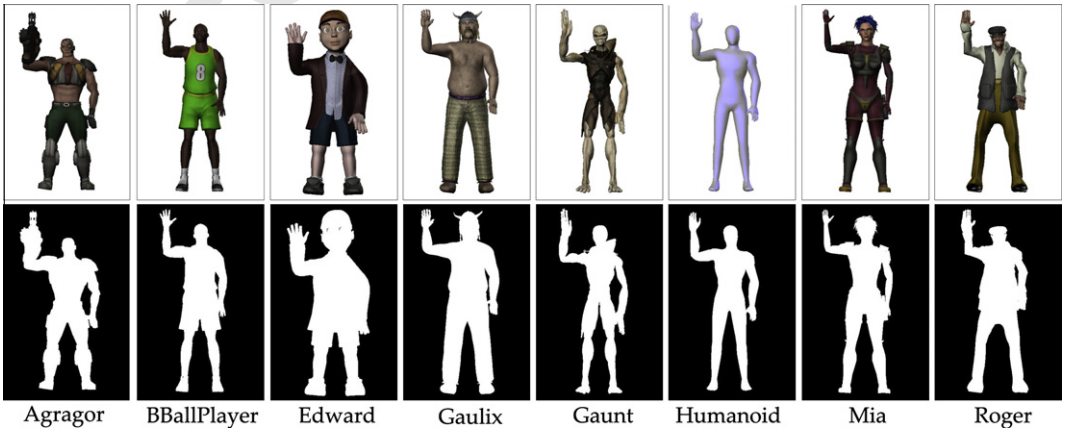

Fig. 4. Characters used in image synthesis.

We end up with 716 feature components in total, from which feature selection is performed.

5.1.2. Evaluation metric

Here we describe the evaluation metric in the context of pose retrieval. First, the images of a given experiment are divided as training and testing. For each testing image, its visual feature vector is calculated, and pose candidates are retrieved from the training data as described in Section 4.1. Then, the retrieval error is evaluated independently for each testing image by measuring the weighted distance between the ground-truth pose and the pose candidates. Suppose the ground-truth pose for a testing image is $\mathbf{y}$. Let $\mathbf{y}'_c (i = c, \ldots, C)$ denote the pose candidates, and let $f_c (i = 1, \ldots, C)$ be the corresponding scores defined as:

$$f_c = w_c \Big/ \sum_c w_c, \tag{16}$$

where w_c is the weights recovered by sparse representation. Then, the retrieval error is defined as:

$$error = \sum_{c=1}^{C} f_c d_{\text{pose}}(\mathbf{y}, \mathbf{y}'_c), \tag{17}$$

where $d_{\text{pose}}(.,.)$ calculates the pose distance. Here we use the built-in pose distance function provided in HumanEva. Specifically, $M = 15$ virtual markers are defined as $\{m_i(\mathbf{y})\}$, $i = 1, \ldots, M$, where $m_i(\mathbf{y}) \in \mathbb{R}^3$ is a function of pose that returns the coordinate of the ith marker in the local coordinate frame centered at the root segment on the body. Then, the distance between two poses is calculated as the average Euclidean distance between corresponding markers:

$$d_{\text{pose}}(\mathbf{y}, \mathbf{y}'_c) = \frac{1}{M} \sum_{i=1}^{M} \|m_i(\mathbf{y}) - m_i(\mathbf{y}'_c)\|. \tag{18}$$

Note that body orientations (yaw angles) are not aligned. Therefore, poses under different yaw angles have large distance. This is reasonable for view-independent pose recovery.

5.1.3. Compared methods

We compare several feature selection methods as below.

- *All components*: All the 716 feature components are used. This is the baseline.
- *Subset selection (hard)*: Feature selection is performed using the method proposed in Section 3, using the hard pairwise functions as defined in Eq. (3), where the cut-off threshold τ is set to 80 mm.
- *Subset selection (soft)*: Feature selection is performed using the soft pairwise functions as in defined in Eq. (4), where τ is set to 80 mm.
- *Adaboost (no weights)*: Feature selection is performed by Adaboost as in [4]. Weights of the selected features are discarded.
- *Adaboost (weights)* Feature selection is performed using Adaboost as described above. Each selected feature component is scaled by the corresponding weight.

5.1.4. Results using a single character

First, we evaluate the methods using data of a single character. In the 57,600 images, there are 7200 images belonging to the standard *Humanoid* subject. These 7200 images are randomly divided into 3600 for training and 3600 for testing. This is a relatively easy scenario, since the same subject appears in both training and testing. From the 3600 training data, we randomly generate 10,000 pairs for feature selection. Then, the evaluation is performed on the testing images. The results are plotted in Fig. 5. It can be seen that our feature selection method outperforms others. Hard and soft pairwise (dis)similarity functions generate comparable results in this case. We can also conclude that for Adaboost, the weights of selected features play an important role. If the weighs are not used, the error is notably larger.

5.1.5. Results using multiple characters

Now we consider the evaluation on all the 57,600 images of the eight characters as displayed in Fig. 4. This is clearly more difficult than the single-character case, as the appearances of different characters differ a lot. We use the 28,800 data samples from characters *Agragor*, *BBallPlayer*, *Edward* and *Gaulix* for training and the remaining 28,800 samples of *Gaunt*, *Humanoid*, *Mia* and *Roger* for testing. That is, no character appears in both the training and testing data. This significantly increases the difficulty. In this way we emphasize on the generality, which is extremely important for image based pose recovery. From the 28,800 training images, 100,000 pairs are randomly generated for feature selection. The other settings are the same as the single-character case presented above.

The results are shown in Fig. 6. It can be seen that our subset selection algorithm generates the best performance. In this case, the soft pairwise (dis)similarity functions notably outperforms the hard functions when the number of selected components is less than 200, indicating that soft functions are more robust in

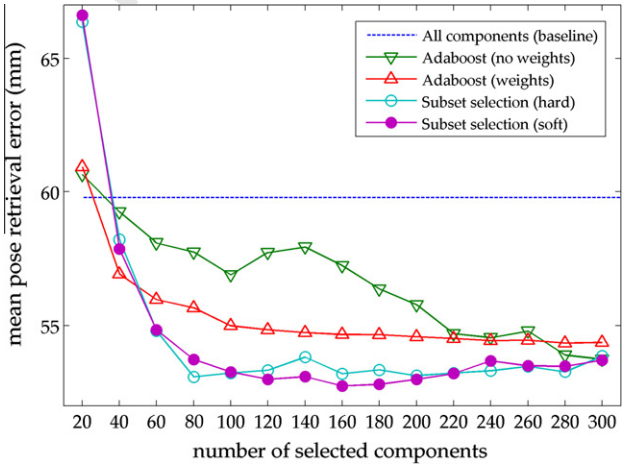

Fig. 5. Evaluation results on synthetic images using single character.

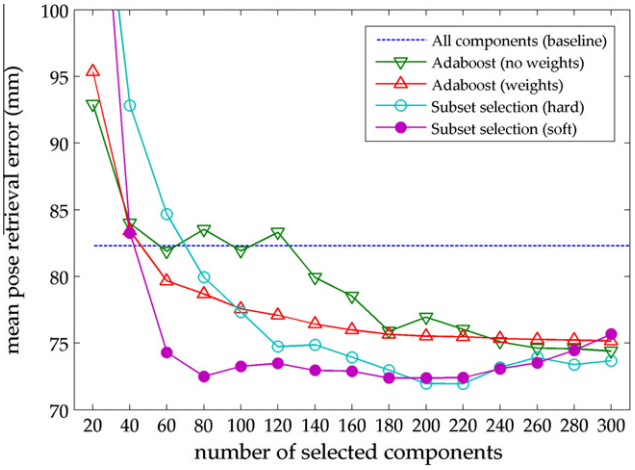

Fig. 6. Evaluation results on synthetic images using multiple characters.

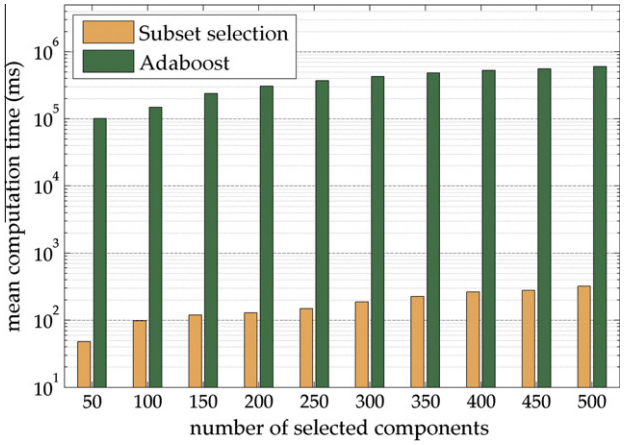

Fig. 7. Comparison of computation time in training stage.

dealing with different human appearances. For Adaboost, the weights of selected features are still important as we observed in Section 5.1.4.

Note that in Figs. 5 and 6, as the number of selected components exceeds 200, the error slightly increases. This is because, as the number gets larger and larger, irrelevant components get selected, impairing the discriminative power. This is consistent with the spirit of feature selection: select discriminative components, and leave behind the components that are ineffective or even misleading. As an extreme, when the number of selected components equals 716, all components are selected, and feature selection simply degenerates to the baseline.

5.1.6. Comparison on computation time

As said in Section 3, our subset selection algorithm is very fast as its complexity is $O(d\log(d))$. On the other hand, Adaboost is much slower. Adaboost requires d' rounds to select d' components. In each round, we have to calculate the error rate of each of the d components, where the computation of each component involves classifying the M data pairs. Therefore, the computation complexity is $O(d \times d' \times M)$.

Fig. 7 compares the mean computation time in training stage in the settings of Section 5.1.5. On average, our method is around 1000 times faster than Adaboost.

5.2. Evaluating feature selection on real data

In this subsection we evaluate our the feature selection method using the real images from HumanEva dataset, where all the subjects S1, S2 and S3 are included. HumanEva utilizes seven cameras. Because we are recovering poses from monocular images, we only use the images from camera C1. We use the training partition of the original dataset as the training data, and the validation partition as the testing data.[1] The original testing data is not used, as the ground-truth pose data is not provided. Note that we exclude the poses in HumanEva which are marked as invalid by the dataset publisher. We generate 4729 training and 4848 testing data samples altogether. Each sample contains the image and the corresponding ground-truth 3D pose.

In contrast to the previous subsection, instead of silhouette-based visual features, here we use HoG [24,16], which is calculated from the original image.[2] Specifically, the bounding box of the person in the image is equally split into $H \times W$ cells. For each cell, we construct K bins representing K directions in the image plane. Then, the gradient of each pixel inside the bounding box is calculated, and the gradient direction of each pixel contributes to the corresponding direction bin of the cell it belongs to. In this way, $H \times W \times K$ feature components are extracted for each image.

We set $K = 9$, i.e. each direction bin corresponds to a range of 20°. The setting of H and W is not so easy. It specifies the resolution at which the gradients are accumulated. Because the aspect ratio of human bounding box is roughly 0.5, we adopt several configurations: $(H,W) = (2,1), (4,2), (8,4), (16,8)$. This produces a total of 1530 hierarchical feature components, on which feature selection is applied.

The evaluation metric is the same as in Section 5.1.2, and Fig. 8 shows the results. When the number of selected components is relatively small ($\leqslant 80$), Adaboost achieves lower error. However, in such cases neither Adaboost nor our method generates satisfactory performance (the error is even higher than baseline). When the number of selected components is larger, our method shows its advantage. The soft pairwise functions outperform the hard ones in most cases, but when the number of components is large, their performances are comparable.

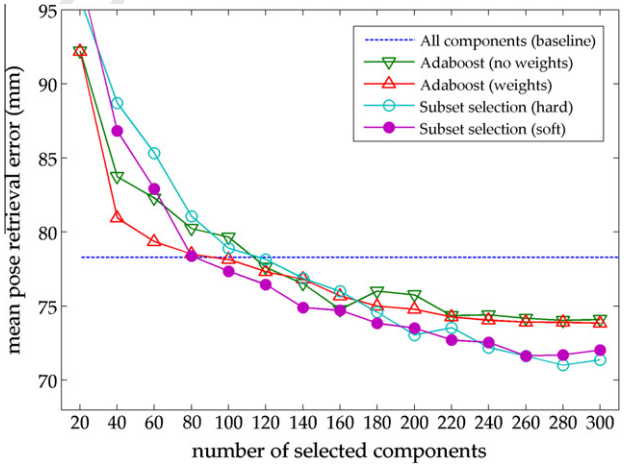

Fig. 8. Evaluation results on real images.

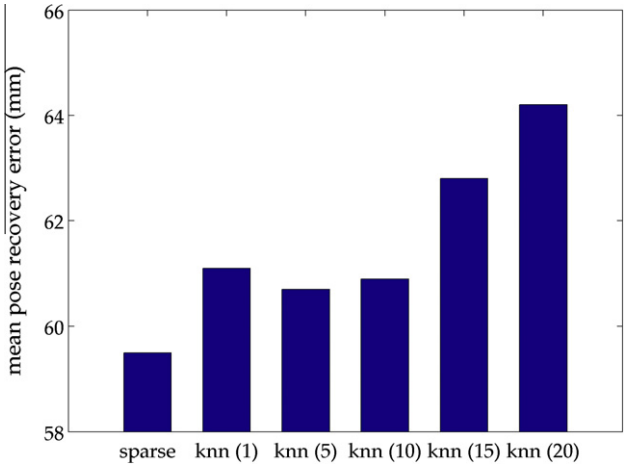

Fig. 9. Comparison of sparse representation and nearest neighbor method.

[1] Many other evaluations [42,36] use the same configuration, where the testing partition is not used because ground-truth is not available.

[2] Actually, we could still use the silhouette-based features as in Section 5.1 and get similar results. In this section we use HoG to demonstrate that our method can be applied to various types of visual features.

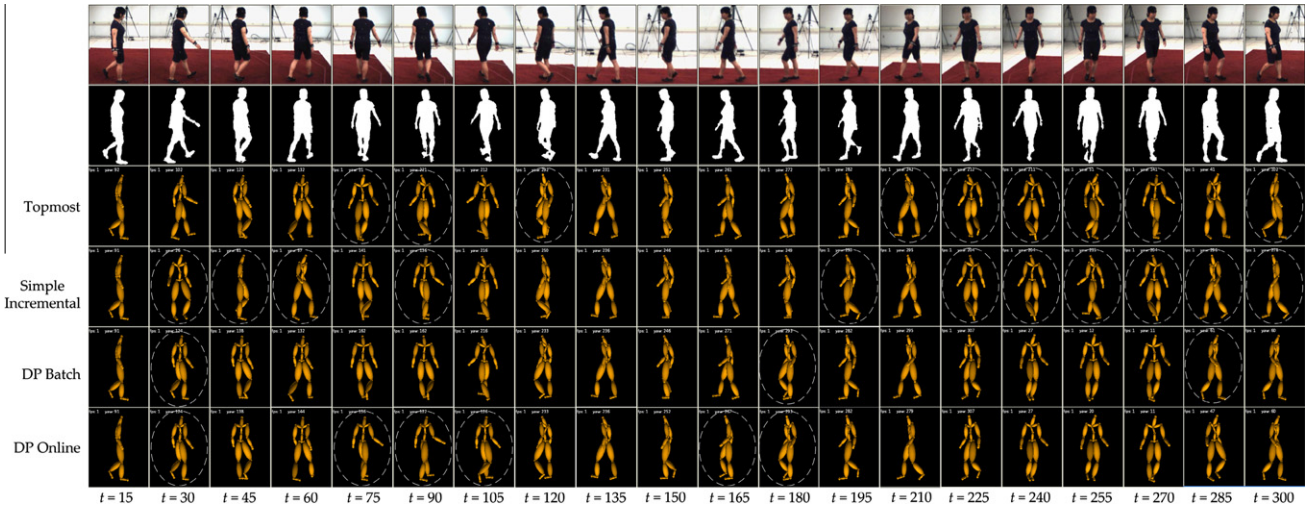

Fig. 10. Evaluating sequential optimization. The first and second rows are images and silhouettes, respectively. The third, fourth, fifth and sixth rows are the recovered poses using four different sequential optimization methods. Significant incorrect poses are marked by dash ellipses.

5.3. Evaluating pose retrieval via sparse representation

In this subsection we compare performance of our sparse representation approach with the nearest neighbor method in the task of pose retrieval. The training data is the same as in Section 5.1.4, i.e. 3600 samples belonging to *Humanoid* character. The testing data is the real data from HumanEva dataset (same as in Section 8). Fig. 10 gives an example of subject S1, where she walks in a 360° circle. Here we fix the feature to be the 150 components selected in Section 5.1.4 using our algorithm with soft pairwise functions. Pose retrieval is conducted using sparse representation or k-NN, and then sequential optimization by dynamic programming is performed to get the final recovered pose sequence. The mean recovery error is the mean distance between the ground-truth pose and the recovered pose in each frame.

For k-NN, we evaluate different values of $k = 1, 5, 10, 15$ and 20. Also, for feature-cue, we use the normalized distance in the feature space as the weight (which serves as the counterpart of Eq. (13) in k-NN case). The comparison is plotted in Fig. 9. It can be seen that sparse representation generates a lower recovery error compared to k-NN. For k-NN, as k increases from 1 to 20, the error first drops and then increases significantly, implicating that incorrect poses are being retrieved for large k. This is also an evidence that setting the value of k is important, but unfortunately, non-trivial, for k-NN method.

5.4. Evaluating sequential optimization

In this subsection we evaluate the sequential optimization strategies. The training and testing data are the same as in Section 5.3. We still fix the feature to be the 150 components selected in Section 5.1.4 using our algorithm with soft pairwise functions. Pose retrieval is conducted using sparse representation, and then sequential optimization is performed to get the final recovered pose sequence. Several methods are compared.

- *Topmost*: The topmost match of each frame is used.
- *Simple Incremental (S-I)*: At frame $t = 1$, topmost match is used. For frame $t = 2, \ldots, T$, the pose is inferred from the feature-cue of the current frame and the recovered pose of the previous frame. Specifically, the index of the recovered pose at frame t is:

$$c^* = \arg\min_c \left(f_{t,c} + \beta d_{\text{pose}}(\mathbf{y}_{t-1}, \mathbf{y}_{t,c})\right), \tag{19}$$

where $f_{t,c}$ is the feature-cue defined as in Eq. (13), $\mathbf{y}_{t-1}$ is the recovered pose in frame $t-1$, and β is the weighting parameter (we set $\beta = 1$).

- *DP Batch*: The method proposed in Section 4.2, used in batch mode.
- *DP Online*: The method proposed in Section 4.2, used in online mode.

Table 1
Quantitative comparison of sequential optimization.

Method	*Topmost*	*S-I*	*DP Batch*	*DP Online*
Error (mm)	82.3	87.0	66.5	73.9

Fig. 10 illustrates the results on one example, and Table 1 shows the mean recovery error for this case. Note that this is a difficult case because the subject is turning her body as she walks and silhouette based features tend to suffer more from ambiguity in body orientation. Incorrect poses are annotated by dash ellipses. *DP Batch* and *DP Online* produces best results. For *Topmost*, the recovered poses are not continuous, and many suffer from reflective ambiguity. For *S-I*, the tracking is lost at $t = 225$. *DP Online* produces similar results to *DP Batch* at most frames. Errors occur at frames $t = 90$ and $t = 105$ for *DP Online*. However, the error is automatically recovered at $t = 120$.

Generally speaking, our sequential optimization methods provide the best results. As expected, the error of *DB Online* is somewhat higher than *DB Batch*. It is also interesting to note that *Topmost* outperforms *S-I*, implicating that the performance improvement can not only be obtained through temporal smoothing, but also by maintaining multiple hypotheses.

5.5. System implementation

We implement two systems using the approach proposed in this paper: a batch system and an online system.[3] In the batch system, user loads a sequence of images, and the corresponding pose sequence is recovered. To fine-tune the result, user can also specify hard constraints at some frames, i.e. forcing the pose sequence to pass some pose candidates. The online system operates in real-time (15 fps).

[3] Please see Supplementary materials for demonstrations.

Please cite this article in press as: C. Chen et al., 3D human pose recovery from image by efficient visual feature selection, Comput. Vis. Image Understand. (2010), doi:10.1016/j.cviu.2010.11.007

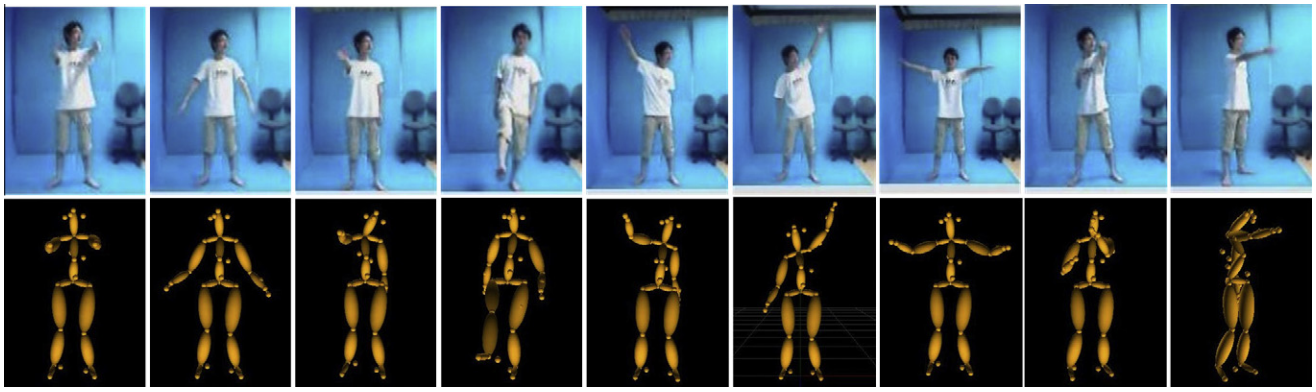

Fig. 11. Recovery results for online system.

Because HumanEva only contains a limited amount of motions. We use our own motion capture device to capture more motion data, such as punching, kicking and various gestures. These data are also used in the examplar database to recover more motions. See Fig. 11 for illustrations.

6. Conclusions

In this paper we proposed a new examplar-based approach to recover 3D human poses from monocular images. We made two contributions. First, we proposed to use an efficient feature selection algorithm to select the globally optimal visual feature components. It was shown that the selected components improve the accuracy and efficiency. Second, we proposed to conduct pose retrieval using sparse representation, which ensures that semantically similar poses have larger probability to be retrieved.

Currently, our examplar database contains up to ten thousand samples. If the database contains many types of motions, it will be much larger. This has two consequences. First, the speed of pose retrieval will be slower. Second, different types of motions will tend to interfere with each other, causing more ambiguity. In the future we would like to study more closely on the performance and optimization of our approach on very large examplar databases.

Appendix A. Solution to the optimization problem in Section 3.3

Following the notation as in Section 3.3, let us denote:

$$\lambda^* = \frac{Tr\left(\mathbf{S}_{\tilde{c}^*}^T \mathbf{U}_B \mathbf{S}_{\tilde{c}^*}\right)}{Tr\left(\mathbf{S}_{\tilde{c}^*}^T \mathbf{U}_A \mathbf{S}_{\tilde{c}^*}\right)} = \max_{\tilde{c}} \frac{Tr\left(\mathbf{S}_{\tilde{c}}^T \mathbf{U}_B \mathbf{S}_{\tilde{c}}\right)}{Tr\left(\mathbf{S}_{\tilde{c}}^T \mathbf{U}_A \mathbf{S}_{\tilde{c}}\right)} \tag{20}$$

and introduce the function $f(\lambda)$:

$$f(\lambda) = \max_{\tilde{c}} Tr\left(\mathbf{S}_{\tilde{c}}^T (\mathbf{U}_B - \lambda \mathbf{U}_A) \mathbf{S}_{\tilde{c}}\right). \tag{21}$$

Next, we define the function $\mathbf{g}$ which returns the subset $\tilde{c}$ at which $f(\lambda)$ is reached. For a given λ, we can show (see [37]) that the set $\mathbf{g}(\lambda)$ can be obtained by simply computing the score of each feature component $\mathbf{s}_i (1 \leqslant i \leqslant d)$: $sc_i = \mathbf{s}_i^T (\mathbf{U}_B - \lambda \mathbf{U}_A) \mathbf{s}_i$. Then we sort the components with descending score, and $\mathbf{g}(\lambda)$ is composed of the first d components.

From (21) and the definition of $\mathbf{g}(\lambda)$, $f(\lambda)$ can be written as:

$$f(\lambda) = Tr\left(\mathbf{S}_{\mathbf{g}(\lambda)}^T (\mathbf{U}_B - \lambda \mathbf{U}_A) \mathbf{S}_{\mathbf{g}(\lambda)}\right), \tag{22}$$

where the max operator in (21) has been removed. $\mathbf{S}_{\mathbf{g}(\lambda)}$ is the selection matrix corresponding to subset $\mathbf{g}(\lambda)$.

It is easy to note that $\mathbf{g}(\lambda)$ is a piecewise constant function, and that $f(\lambda)$ is a piecewise linear function with derivative:

$$\frac{df(\lambda)}{d\lambda} = -Tr\left(\mathbf{S}_{\mathbf{g}(\lambda)}^T \mathbf{U}_A \mathbf{S}_{\mathbf{g}(\lambda)}\right). \tag{23}$$

From (23), $f(\lambda)$ is monotonically decreasing. From (20) and (21) we have $f(\lambda^*) = 0$. Therefore, λ^*, which is the root of $f(\lambda)$, can be efficiently searched for using Newton method. Note that because $f(\lambda)$ is piecewise linear, the iterative optimization is very fast (we observe in experiments that the optimization typically terminates within five iterations).

Appendix B. Supplementary material

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.cviu.2010.11.007.

References

[1] C.T. Zahn, R.Z. Roskies, Fourier descriptors for plane closed curves, IEEE Transactions on Computers c-21 (3) (1972) 269–281.
[2] R.D. de León, L.E. Sucar, Human silhouette recognition with Fourier descriptors, in: ICPR, 2000, pp. 3713–3716.
[3] L. Ren, G. Shakhnarovich, J.K. Hodgins, H. Pfister, P.A. Viola, Learning silhouette features for control of human motion, ACM Transactions on Graphics 24 (4) (2005) 1303–1331.
[4] C. Chen, Y. Zhuang, J. Xiao, F. Wu, Adaptive and compact shape descriptor by progressive feature combination and selection with boosting, in: CVPR, 2008.
[5] N.R. Howe, Silhouette lookup for monocular 3D pose tracking, Image Vision Computing 25 (3) (2007) 331–341.
[6] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 210–227.
[7] J. Wright, Y. Ma, J. Mairal, G. Spairo, T. Huang, S. Yan, Sparse representation for computer vision and pattern recognition, in: ICCV, 2009.
[8] T.B. Moeslund, E. Granum, A survey of computer vision-based human motion capture, Computer Vision and Image Understanding 81 (3) (2001) 231–268.
[9] T.B. Moeslund, A. Hilton, V. Krüger, A survey of advances in vision-based human motion capture and analysis, Computer Vision and Image Understanding 104 (2–3) (2006) 90–126.
[10] R. Poppe, Vision-based human motion analysis: an overview, Computer Vision and Image Understanding 108 (1–2) (2007) 4–18.
[11] X. Zhao, Y. Liu, Generative tracking of 3D human motion by hierarchical annealed genetic algorithm, Pattern Recognition 41 (8) (2008) 2470–2483.
[12] H. Sidenbladh, M.J. Black, D.J. Fleet, Stochastic tracking of 3D human figures using 2D image motion. In: European Conference on Computer Vision, 2000, pp. 702–718.
[13] M.W. Lee, R. Nevatia, Human pose tracking in monocular sequence using multilevel structured models, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (1) (2009) 27–38.
[14] A. Agarwal, B. Triggs, Recovering 3D human pose from monocular images, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (1) (2006) 44–58.
[15] G. Shakhnarovich, P. Viola, T. Darrell, Fast pose estimation with parameter-sensitive hashing, in: ICCV, 2003, pp. 750–757.
[16] R. Poppe, Evaluating example-based pose estimation: experiments on the humaneva sets, in: Online Proceedings of the Workshop on Evaluation of

Articulated Human Motion and Pose Estimation (EHuM) at the International Conference on Computer Vision and Pattern Recognition (CVPR), Minnesota, Minneapolis, 2007, pp. 1–8.
[17] C. Sminchisescu, A. Kanaujia, D.N. Metaxas, Learning joint top-down and bottom-up processes for 3d visual inference, in: CVPR, vol. 2, 2006, pp. 1743–1752.
[18] R. Rosales, S. Sclaroff, Combining generative and discriminative models in a framework for articulated pose estimation, International Journal of Computer Vision 67 (3) (2006) 251–276.
[19] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (4) (2002) 509–522.
[20] G. Mori, S.J. Belongie, J. Malik, Efficient shape matching using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (11) (2005) 1832–1837.
[21] E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem, J.S.B. Mitchell, An efficiently computable metric for comparing polygonal shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (3) (1991) 209–216.
[22] M.K. Hu, Visual pattern recognition by moment invariants, IRE Transactions on Information Theory IT-8 (1962) 179–187.
[23] L. Gorelick, M. Galun, E. Sharon, R. Basri, A. Brandt, Shape representation and classification using the poisson equation, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (12) (2006) 1991–2005.
[24] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: CVPR, vol. 1, 2005, pp. 886–893.
[25] S. Nayak, S. Sarkar, B.L. Loeding, Distribution-based dimensionality reduction applied to articulated motion recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (5) (2009) 795–810.
[26] A. Agarwal, B. Triggs, A local basis representation for estimating human pose from cluttered images, in: ACCV, vol. 1, 2006, pp. 50–59.
[27] P. Scovanner, S. Ali, M. Shah, A 3-dimensional sift descriptor and its application to action recognition, in: ACM Multimedia, 2007, pp. 357–360.
[28] T. Serre, L. Wolf, T. Poggio, Object recognition with features inspired by visual cortex, in: CVPR, vol. 2, 2005, pp. 994–1000.
[29] A. Agarwal, B. Triggs, Hyperfeatures - multilevel local coding for visual recognition, in: ECCV, vol. 1, 2006, pp. 30–43.
[30] D. Nistér, H. Stewénius, Scalable recognition with a vocabulary tree, in: CVPR, vol. 2, 2006, pp. 2161–2168.
[31] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: CVPR, vol. 2, 2006, pp. 2169–2178.
[32] I. Laptev, On space-time interest points, International Journal of Computer Vision 64 (2–3) (2005) 107–123.
[33] J.C. Niebles, H. Wang, F.-F.L. 0002, Unsupervised learning of human action categories using spatial–temporal words, International Journal of Computer Vision 79 (3) (2008) 299–318.
[34] M. Bregonzio, S. Gong, T. Xiang, Recognising action as clouds of space-time interest points, in: CVPR, 2009.
[35] R. Poppe, M. Poel, Comparison of silhouette shape descriptors for example-based human pose recovery, in: FG, 2006, pp. 541–546.
[36] C. Chen, Y. Zhuang, J. Xiao, Silhouette representation and matching for 3D pose discrimination – a comparative study, Image and Vision Computing 28 (4) (2010) 654–667.
[37] F. Nie, S. Xiang, Y. Jia, C. Zhang, S. Yan, Trace ratio criterion for feature selection, in: AAAI, 2008, pp. 671–676.
[38] 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 59 (6) (2006) 797–829.
[39] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, Annals of Statistics 32 (2004) 407–499.
[40] L. Sigal, A. Balan, M.J. Black, HumanEva: synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion, International Journal on Computer Vision 87 (1) (2010) 4–27.
[41] F. Fleuret, J. Berclaz, R. Lengagne, P. Fua, Multicamera people tracking with a probabilistic occupancy map, IEEE Transaction on Pattern Analysis and Machine Intelligence 30 (2) (2008) 267–282.
[42] H. Ning, W. Xu, Y. Gong, T. Huang, Discriminative learning of visual words for 3D human pose estimation, in: CVPR, 2008.

Please cite this article in press as: C. Chen et al., 3D human pose recovery from image by efficient visual feature selection, Comput. Vis. Image Understand. (2010), doi:10.1016/j.cviu.2010.11.007