This paper appeared in CVIU, Volume 115, Number 3, pp 290-299, March 2011

3D Human Pose Recovery from Monocular Images via Efficient Visual Feature Selection

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

[a]IDIAP Research Institute, Martigny, Switzerland
[b]College of Computer Science, Zhejiang University, Hangzhou, China
[c]University of Texas, Arlington, USA

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 weighted set of candidates to impose temporal coherence and recover a continuous pose sequence. We made two contributions within this framework. First, we propose to use an efficient feature selection algorithm to select the optimal visual feature components. The feature selection task is formulated as a trace-ratio objective which measures the score of the selected feature component subset, and the objective is efficiently optimized to get the global optimum. The selected components are used instead of the original visual feature 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 labeled samples in the database. Sparse representation ensures that semantically similar poses have larger probability to be retrieved. The effectiveness of our approach has been validated quantitatively through extensive evaluations on both synthetic data and real data, and qualitatively by inspecting the results of the real time system we have implemented.

Key words: pose recovery, feature selection, motion understanding, sparse representation

1. Introduction

Recovering 3D human pose from marker-free images is a very important research subject in the computer vision community. The success of pose recovery directly benefits many applications such as video understanding, 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. Perhaps the greatest difficulty comes from the fact that the mapping from visual features to poses is very complex and multi-modal. If poses are inferred only from monocular images, the challenge is even greater, as 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 Figure 1. The examplar database contains visual features and corresponding ground-truth poses. For each novel frame, selected visual features are used to retrieve a set of pose candidates. Finally, the optimal pose sequence is estimated from the pose candidates using dynamic programming, which relies on both the feature cue and temporal cue to ensure the smoothness of the recovered poses.

Within the above framework, we make two contributions to improve the performance of pose recovery, as described below.

Visual Feature Selection. Visual feature plays an important role in pose recovery. A proper visual feature can alleviate the 2D-3D ambiguity to a great extent. A lot of features have been proposed using cues from silhouettes, edges and so on. Intuitively, we would like the visual features to be as much as possible discriminative with respect to 3D poses.

Typically, each feature type contains multiple components. While most previous work uses the complete components of a particular feature type, in this paper we propose an algorithm that efficiently selects the optimal subset of feature components for pose recovery. First, the feature selection objective is formulated in trace-ratio form that measures the consistency to the intrinsic data relationship. Then, optimization is performed efficiently to find the global optimal component subset. The selected components are used instead of the original feature in pose recovery to improve both accuracy and efficiency:

- Accuracy. Different visual feature components behaves differently in pose discrimination. For example, for Fourier descriptor [4], it has been shown that pose understanding may depend more on some frequencies than on others [5]. By performing feature selection, we are able to discard irrelevant (or even misleading) components, and thus achieve better pose recovery 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 feature components from images. On the other hand, 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. [7] and Chen et al. [6] employ Adaboost to select effective features. However, Adaboost re-

Email addresses: Cheng.Chen@idiap.ch (Cheng Chen), odobez@idiap.ch (Jean-Marc Odobez)

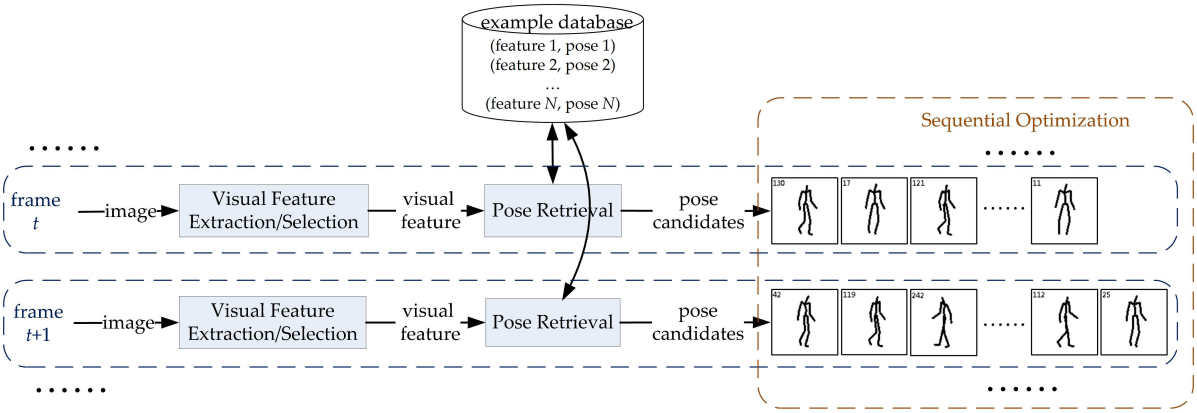

Figure 1: Our examplar based pose recovery framework.

quires a large amount of computation, and it cannot guarantee convergence to the global optimum. On the other hand, our feature selection method guarantees to generate the globally optimal feature subset in a much faster way.

Pose Retrieval via Sparse Representation: An important step in our example-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 are retrieved for each frame. Traditionally, this is often performed by nearest neighbor (NN) method [6][7][12]. In this paper we propose to use sparse representation [37] to perform pose retrieval. Specifically, the visual feature of a novel frame should be reasonably approximated using only a small percentage of features in the example database. This can be solved using sparse representation, and in this way we retrieve relevant poses. Compared to NN, sparse representation tends to select more semantically relevant poses. Another advantage is that its neighborhood size is adaptive rather than fixed.

We perform extensive experiments on both synthetic data 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 our feature selection method in Section 3. Section 4 presents pose retrieval via sparse representation and dynamic programming for sequential optimization. Experiments are given in Section 5 and conclusions are in Section 6.

2. Related Work

2.1. Image based pose recovery

There have been numerous publications on image based human pose recovery [1][2][3]. Broadly speaking, the approaches can be divided as synthetic (model based) and discriminative (learning based). Synthetic 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 [9], or by sampling posterior pose probabilities [8][10].

On the other hand, discriminative methods directly learn a mapping from visual features to pose parameters. The mapping is often approximated using regression models [11]. Alternatively, one can directly rely on a dense training database, leading to the so-called "examplar-based" methods [7][12][13]. These methods are fast, and given proper training data, the recovery accuracy can be satisfactory.

Generally speaking, synthetic methods are more accurate, but the computation is often expensive. On the other hand, discriminative methods are more efficient. There also exists some hybrid methods [14][15].

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 features are based on silhouettes, such as Fourier descriptor [4], shape contexts [16][17], geometric signature [18], Hu moments [19], Poisson Features [20] and so on. Apart from silhouette, there are also features based on edges or gradients, such as histogram of gradients [21], relational edge distribution [22] and various SIFT-like features [23][24]. Hierarchical features have also been proposed, such as HMAX [25], Vocabulary Tree [27], Hyperfeatures [26] or Spatial Pyramid matching [28]. Space-time interest points [29][30] [31] are also proposed.

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, comparable studies are conducted to evaluate the performance of different feature types. For example, Poppe et al. [32] compare Fourier descriptor, shape contexts and Hu moments in 3D pose recovery. Chen et al. [33] 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. [7] select silhouette feature components from a huge pool of

local rectangle features. Chen et al. [6] perform feature component selection using Adaboost.

Another frequently used technique is subspace learning (also known as dimension reduction) which aims to find the optimal transformation from the original feature space to a low dimensional subspace. Though the dimension in the learned subspace is usually much lower than the input feature dimension, for novel data the full feature set still has to be extracted to obtain its subspace embedding. For feature selection, however, given a novel data sample, we only need to extract the selected feature components, making the feature extraction process 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.

3. Visual Feature Selection for Pose Discrimination

3.1. Formulation

Suppose we have a set of N data samples $\Phi = \{\phi_1, \phi_2, ..., \phi_N\}$ 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 $C = \{c_1, c_2, ...c_d\}$ be the set of feature components, where c_i is the i^{th} component. Our goal is to select a feature component subset $\tilde{C} \in 2^C$ consisting of $d' = \left|\tilde{C}\right|$ ($d' < d$) components (dimensions), where 2^C is the power set of C, and $|.|$ is the cardinality of a set.

For each possible $\tilde{C}$, there is a corresponding feature selection function:

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

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. Naturally, we would like the selected subset $\tilde{C}$ to be as effective as possible in discriminating 3D poses.

3.2. Objective

For image based pose recovery, the desired property on the visual feature is that it should encode data relationship. First, similar poses should correspond to numerically close features. Second, dissimilar poses should correspond to faraway features. These are two guidelines in designing the feature selection objective.

To guide the feature selection, we generate a set $\mathcal{P}$ containing M data pairs $\mathcal{P} = \{(\mathbf{x}_{i1}, \mathbf{x}_{j1}), ..., (\mathbf{x}_{iM}, \mathbf{x}_{jM})\}$, where $\mathbf{x}_{ip} \in \Phi$ and $\mathbf{x}_{jp} \in \Phi (i < p < M)$ are two different data from the training set.

We define a function $f_{\text{a}}(\mathbf{y}_i, \mathbf{y}_j)$ that reflects the pairwise pose similarity. That is, $f_{\text{a}}(\mathbf{y}_i, \mathbf{y}_j)$ is large if and only if $\mathbf{y}_j$ and $\mathbf{y}_j$ are similar. For similar poses, i.e. f_{a} is large, we would expect their visual features to be close as well. Thus, our first objective is defined as:

$$\min_{\tilde{C}\in 2^C, |\tilde{C}|=d'} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{P}} \left(\left\| g_{\tilde{C}}(\mathbf{x}_i) - g_{\tilde{C}}(\mathbf{x}_j) \right\|^2 \times f_{\text{a}}(\mathbf{y}_i, \mathbf{y}_j) \right). \tag{2}$$

Qualitatively, Equation (2) incurs a large penalty if two poses are similar, but the distance between the selected features is large.

Similarly, we can also define a function $f_{\text{b}}(\mathbf{y}_i, \mathbf{y}_j)$ that reflects the pairwise data dissimilarity. That is, $f_{\text{b}}(\mathbf{y}_i, \mathbf{y}_j)$ is large if and only if $\mathbf{y}_i$ and $\mathbf{y}_j$ are dissimilar. For dissimilar samples, i.e. f_{b} is large, we would expect their corresponding features to be far away. Thus, a second objective would be:

$$\max_{\tilde{C}\in 2^C, |\tilde{C}|=d'} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{P}} \left(\left\| g_{\tilde{C}}(\mathbf{x}_i) - g_{\tilde{C}}(\mathbf{x}_j) \right\|^2 \times f_{\text{b}}(\mathbf{y}_i, \mathbf{y}_j) \right). \tag{3}$$

On one hand, we want to minimize (2). On the other hand, we want to maximize (3). Combining these two goals, we can formulate our problem as the maximization of their ratio, that is:

$$\max_{\tilde{C}\in 2^C, |\tilde{C}|=d'} \frac{\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{P}} \left(\left\| g_{\tilde{C}}(\mathbf{x}_i) - g_{\tilde{C}}(\mathbf{x}_j) \right\|^2 \times f_{\text{b}}(\mathbf{y}_i, \mathbf{y}_j) \right)}{\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{P}} \left(\left\| g_{\tilde{C}}(\mathbf{x}_i) - g_{\tilde{C}}(\mathbf{x}_j) \right\|^2 \times f_{\text{a}}(\mathbf{y}_i, \mathbf{y}_j) \right)}. \tag{4}$$

3.3. Pairwise pose (dis)similarity functions

Now we detail how to define the pairwise functions $f_{\text{a}}(\mathbf{y}_i, \mathbf{y}_j)$ and $f_{\text{b}}(\mathbf{y}_i, \mathbf{y}_j)$. As mentioned above, $f_{\text{a}}(\mathbf{y}_i, \mathbf{y}_j)$ is large if and only if poses $\mathbf{y}_i$ and $\mathbf{y}_j$ are similar, and $f_{\text{b}}(\mathbf{y}_i, \mathbf{y}_j)$ is large if and only if poses $\mathbf{y}_i$ and $\mathbf{y}_j$ are dissimilar. Therefore, the functions 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.
A simple definition of the pairwise functions is:

$$\begin{aligned} f_a(\mathbf{y}_i, \mathbf{y}_j) &= \begin{cases} 0, & \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) > \tau \\ 1, & \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) \leqslant \tau \end{cases} \\ f_b(\mathbf{y}_i, \mathbf{y}_j) &= \begin{cases} 1, & \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) > \tau \\ 0, & \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) \leqslant \tau \end{cases} \end{aligned}. \tag{5}$$

That is, if the distance between poses is within a threshold τ, then the data samples are considered as similar, and vice versa.

The functions in (5) 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 more important roles in the calculation. In order to address this consideration, we propose to exploit soft pairwise functions:

$$\begin{aligned} f_{\text{a}}\left(\mathbf{y}_i, \mathbf{y}_j\right) &= \exp\left(-d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) \big/ \sigma^2\right) \\ f_{\text{b}}\left(\mathbf{y}_i, \mathbf{y}_j\right) &= \begin{cases} 0, \text{if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) \leqslant \tau \\ \left(d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) \big/ d_{\max}\right)^2, \text{ if } d_{\text{pose}}(\mathbf{y}_i, \mathbf{y}_j) > \tau \end{cases} \end{aligned}, \tag{6}$$

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

3.4. Optimization

Now we describe the optimization of the feature selection objective in (4). For the ease of presentation, we note that the feature selection function can be written in matrix form as:

$$\tilde{\mathbf{x}} = g_{\tilde{C}}(\mathbf{x}) = \mathbf{S}_{\tilde{C}}{}^T \mathbf{x}, \tag{7}$$

where $\mathbf{S}_{\tilde{C}} \in \mathbb{R}^{d \times d'}$ is the selection matrix corresponding to the feature subset $\tilde{C}$. Suppose the complete component set is $C = \{c_1, c_2, ...c_d\}$ and the selected subset is $\tilde{C} = \{c_{I(1)}, c_{I(2)}, ...c_{I(d')}\}$, where I is a permutation of $1, 2, ..., d$. Then $\mathbf{S}_{\tilde{C}}$ is defined as:

$$\mathbf{S}_{\tilde{C}} = [\mathbf{s}_{I(1)}, \mathbf{s}_{I(2)}, ..., \mathbf{s}_{I(d')}]. \tag{8}$$

The k^{th} column $\mathbf{s}_{I(k)}$ is defined as:

$$\mathbf{s}_{I(k)} = [\mathbf{0}_{I(k)-1}, 1, \mathbf{0}_{d-I(k)}]^T, \tag{9}$$

where $\mathbf{0}_n$ is the row vector of n zeros. That is, all except the $I(k)^{\text{th}}$ components of $\mathbf{s}_{I(k)}$ are zero.

Substituting (7) into (4), the feature selection objective can be written as:

$$\begin{aligned} &\max_{\tilde{C}} \frac{\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in P} \left((\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}_{\tilde{C}} \mathbf{S}_{\tilde{C}}^T (\mathbf{x}_i - \mathbf{x}_j) \times f_{\text{b}}(\mathbf{y}_i, \mathbf{y}_j) \right)}{\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in P} \left((\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}_{\tilde{C}} \mathbf{S}_{\tilde{C}}^T (\mathbf{x}_i - \mathbf{x}_j) \times f_{\text{a}}(\mathbf{y}_i, \mathbf{y}_j) \right)}, \\ &\text{s.t. } \tilde{C} \in 2^C \text{ and } \left|\tilde{C}\right| = d' \end{aligned} \tag{10}$$

To simply presentation, (10) can be written compactly as:

$$\max_{\tilde{C}} \frac{Tr\left(\mathbf{S}_{\tilde{C}}{}^T \mathbf{U}_B \mathbf{S}_{\tilde{C}}\right)}{Tr\left(\mathbf{S}_{\tilde{C}}{}^T \mathbf{U}_A \mathbf{S}_{\tilde{C}}\right)}, \text{ s.t. } \tilde{C} \in 2^C \text{ and } \left|\tilde{C}\right| = d', \tag{11}$$

where $Tr(.)$ is the trace operator, and $\mathbf{U}_B$ and $\mathbf{U}_A$ are defined as:

$$\begin{aligned} \mathbf{U}_B &= \sum_{(i,j) \in \mathcal{P}} \left((\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T f_{\text{b}}(\mathbf{y}_i, \mathbf{y}_j) \right) \\ \mathbf{U}_A &= \sum_{(i,j) \in \mathcal{P}} \left((\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T f_{\text{a}}(\mathbf{y}_i, \mathbf{y}_j) \right) \end{aligned}. \tag{12}$$

Brutal-force search for the optimal subset $\tilde{C}$ in (11) is clearly prohibitive, because the searching space is factorial on the number of feature components. We propose an efficient algorithm that searches for the global optimal $\tilde{C}$ iteratively. Here we briefly present the method (See [34] for details).

Let us denote:

$$\lambda^* = \frac{Tr\left(\mathbf{S}_{\tilde{C}^*}{}^T \mathbf{U}_B \mathbf{S}_{\tilde{C}^*}\right)}{Tr\left(\mathbf{S}_{\tilde{C}^*}{}^T \mathbf{U}_A \mathbf{S}_{\tilde{C}^*}\right)} = \max_{\tilde{C}} \frac{Tr\left(\mathbf{S}_{\tilde{C}}{}^T \mathbf{U}_B \mathbf{S}_{\tilde{C}}\right)}{Tr\left(\mathbf{S}_{\tilde{C}}{}^T \mathbf{U}_A \mathbf{S}_{\tilde{C}}\right)}. \tag{13}$$

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

$$f(\lambda) = \max_{\tilde{C}} Tr\left(\mathbf{S}_{\tilde{C}}{}^T \left(\mathbf{U}_B - \lambda \mathbf{U}_A\right) \mathbf{S}_{\tilde{C}}\right). \tag{14}$$

Next, we define the function $\mathbf{g} : \mathbb{R} \rightarrow 2^C$ which returns the subset $\tilde{C}$ at which $f(\lambda)$ is reached. For a given λ, we can show (see [34]) that the set $\mathbf{g}(\lambda)$ can be obtained by simply computing the score of each feature component $\mathbf{s}_i (1 \leq i \leq d)$: $sc_i = \mathbf{s}_i^T \left(\mathbf{U}_B - \lambda \mathbf{U}_A\right) \mathbf{s}_i$. Then we sort the components with descending score, and $\mathbf{g}(\lambda)$ is composed of the first d' components.

From (14) and the definition of $\mathbf{g}(\lambda)$, $f(\lambda)$ can be written as:

$$f(\lambda) = Tr\left(\mathbf{S}_{\mathbf{g}(\lambda)}{}^T \left(\mathbf{U}_B - \lambda \mathbf{U}_A\right) \mathbf{S}_{\mathbf{g}(\lambda)}\right), \tag{15}$$

where the max operator in (14) has been removed.

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:

$$\frac{df(\lambda)}{d\lambda} = -Tr\left(\mathbf{S}_{\mathbf{g}(\lambda)}{}^T \mathbf{U}_A \mathbf{S}_{\mathbf{g}(\lambda)}\right). \tag{16}$$

That is, $f(\lambda)$ is monotonically decreasing. From (13) and (14) 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 5 iterations).

3.5. Summary

The visual feature selection method is summarized below.

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 $\tilde{C} \in 2^C$, $\left|\tilde{C}\right| = d'$.

Procedure:

1. Compute $\mathbf{U}_A$ and $\mathbf{U}_B$ according to (12).
2. Initialize $\tilde{C}$ randomly; Initialize $\mathbf{S}_{\tilde{C}}$ according to (8).
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 score $sc_i = \mathbf{s}_i^T (\mathbf{U}_B - \lambda \mathbf{U}_A) \mathbf{s}_i$
5. Sorting components according to the scores sc_i in decent order. Update $\tilde{C}$ with the first d' components.
6. Update $\mathbf{S}_{\tilde{C}}$ according to (8).
7. Repeat steps 3 to 6 until convergence.

It is easy to deduce that the computation complexity of the above algorithm is $O(d \log(d))$.

4. Pose Recovery via Sparse Representation and Dynamic Programming

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

4.1. Pose Retrieval via Sparse Representation

We employ sparse representation [37] [38] to find the pose candidates for each frame. It has been reported in [37] that if the training dataset is sufficiently large, then sparse representation generally outperforms nearest neighbor method. Another advantage of sparse representation is that the neighborhood size (number of candidates) is adaptive rather than fixed.

Let $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, ..., \tilde{\mathbf{x}}_N] \in \mathbb{R}^{d' \times N}$ denote the matrix of all visual features in the example 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:

$$\tilde{\mathbf{x}}_t \approx w_{t,1}\tilde{\mathbf{x}}_1 + w_{t,2}\tilde{\mathbf{x}}_2 + ... + w_{t,N}\tilde{\mathbf{x}}_N = \tilde{\mathbf{X}}\mathbf{w}_t, \quad (17)$$

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

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 constraints was enforced on the sparsity of $\mathbf{w}_t$, then $\tilde{\mathbf{x}}_t$ would be approximated using (17) with a very small residual error by dense weights that are not informative on the intrinsic data relationship. Therefore, we have the following objective:

$$\mathbf{w}_t = \arg\min \left(\left\|\tilde{\mathbf{x}}_t - \tilde{\mathbf{X}}\mathbf{w}_t\right\| + \gamma|\mathbf{w}_t|_1 \right), \text{s.t.} w_{t,1}, ...w_{t,N} \geq 0, \quad (18)$$

where γ is the regularizer, and $|.|_1$ is the L_1 norm. The optimization in (18) can be solved efficiently by Lasso [39]. In this way, for each frame, a set of pose candidates is constructed by sparse representation with the items in the example database with non-zero weights.

4.2. Sequential Optimization via Dynamic programming

The next step is to conduct sequential optimization from pose candidates of each frame to get a continuous pose sequence. We use a graph model depicted in Figure 2 to illustrate the problem. Node $\mathbf{y}_{t,c}$ represents the c^{th} pose candidate of frame t, and $w_{t,c}$ is the corresponding weight derived from (18). Nodes from successive frames are connected by edges with weights. Under this model, sequential optimization is equivalent to finding a best path $H = h(1)h(2)...h(T)$.

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

$$f_{h(t)} = w_{t,h(t)} \Big/ \sum_c w_{t,c} \quad (19)$$

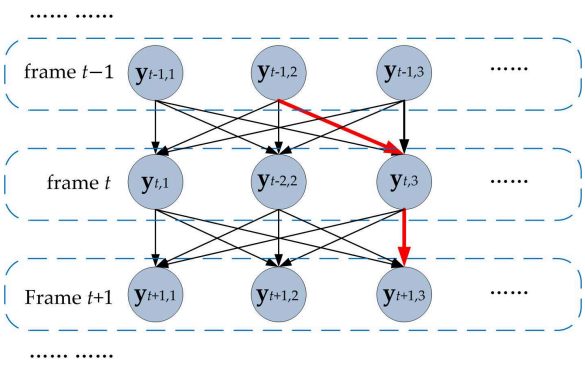

Figure 2: Graph model for sequential optimization.

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:

$$o_{h(t),h(t+1)} = \exp\left(-d_{\text{pose}}\left(\mathbf{y}_{t,h(t)}, \mathbf{y}_{t+1,h(t+1)}\right) \Big/ \sigma^2\right) \quad (20)$$

Under the above model, the score of a path $H = h(1)h(2)...h(T)$ is the total feature scores of all nodes it passes by, plus the total transition scores of all edges it traverses:

$$score(H) = \sum_{t=1}^{T} f_{h(t)} + \alpha \sum_{t=1}^{T-1} o_{h(t),h(t+1)}, \quad (21)$$

where α is the weighting parameter. The global optimal pose path can be efficiently found using dynamic programming.

Details have to be discussed depending on the system type.

Batch system. For batch systems, all video frames from 1 to T are known before sequential optimization, and thus the global optimal path can be guaranteed. Note that under this model it is very convenient to incorporate user defined constraints. 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 convenient.

Online system. Things are a little different for online systems, such as real-time human-computer interaction. In such cases, 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. One thing we have to pay attention to is that when new frames at time $t+1$, $t+2$... arrive, the path before time t might be changed, and this may cause discontinuity in the online pose display. However, the action of "changing the history" occurs only occasionally and can be seen as automatic correction from errors. Our method gets back from temporary errors once more information is available. For other incremental tracking methods, it is possible that the tracking is permanently drifted from such errors (See Section 5.4 in the experiments). For online systems, the ability to sustain satisfying recovery over a long time is very important.

5. Experiments

In this section we present experimental results. Sections 5.1 and 5.2 show the evaluations of our visual feature selection method using synthetic images and real images, respectively. Both Sections 5.1 and 5.2 are based on HumanEva dataset, where synthetic images allow us to evaluate in unusual appearances and real images allows the evaluation with noise. Section 5.3 evaluates the pose retrieval via sparse representation. Section 5.4 evaluates the sequential optimization strategy.

5.1. Evaluating feature selection on synthetic data

5.1.1. Data

The data used here is from HumanEva dataset. HumanEva contains data of 5 motion types, namely *boxing*, *gestures*, *jog*, *throw-catch* and *walking* performed by 3 subjects. For each motion type and each subject, there are three trials, where trial 1 contains synchronized video and motion data, trial 2 contains only video data, and trial 3 contains only motion data. 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 the evaluations more efficient, we generate a subset of 3D poses. First, we rotate all 3D poses to 0^o yaw angle (i.e. in frontal view respect to the camera). Then, we perform k-means clustering on the poses with k=300. For each pose configuration, we generate 24 poses by cycling the yaw angle from 0^o to 345^o with 15^o interval. Thus we generate $300 \times 24 = 7200$ poses. Then, each pose is targeted to 8 characters by MotionBuilder (See Figure 3). In this way, we generate $7200 \times 8 = 57600$ images. They typically have unusual appearances (e.g. exaggerated head, helmet, gun in hand).

The ground-truth poses of the 57600 images are trivially known because they are synthesized using the ground-truth poses. The image features are based on silhouettes and are composed of several feature types as follows (See [33] for details):

- Occupancy map ([36]): 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 ([18]): Contour signature is defined by some geometric quantity along the silhouette border. We adopt three versions: complex coordinates, centroid distances, 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 ([4]): Fourier descriptor consists of normalized Fourier coefficients at different frequencies after applying DFT to the contour signatures introduced above. It contains $62+32+62 = 156$ feature components.
- Hu moments ([19]): Hu moments consist of 7 moment based features calculated by treating the shape as a two-dimensional density distribution.

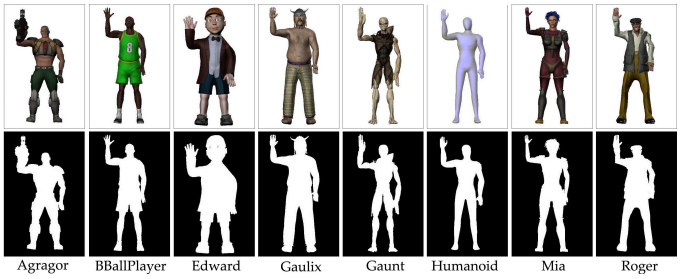

Figure 3: Characters used in image synthesis.

- Shape contexts ([16][17]): We use histogram of shape contexts with codebook size 100, constructed from 12 angular bins and 5 radial bins.
- Poisson features ([20]): We use 30 moment based features calculated on local Poisson features.

There are totally 716 feature components, from which feature selection is performed.

5.1.2. Evaluation metric

The visual feature selection methods are compared by pose retrieval as in Section 4.1. First, the images are divided as training and testing. Then, for each testing image, its visual feature vector is calculated, and pose candidates are retrieved from the training data as in Section 4.1. Since we are evaluating the visual features, sequential optimization introduced in Section 4.2 is not employed. That is, the retrieval error is evaluated independently for each testing image by 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 $\mathbf{y}'_c$ and $f_c (i = 1, ..., C)$ denote the pose candidates and corresponding scores (defined in (19)), the retrieval error is:

$$error = \sum_{c=1}^{C} f_c d_{\text{pose}}(y, y'_c), \tag{22}$$

where $d_{\text{pose}}(.,.)$ calculates the pose distance. Here we use the built-in pose distance function provided in HumanEva, which calculates the mean 3D distance between virtual markers. 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.

All components. The evaluation is performed using all the 716 feature components. This is the baseline.

Subset selection (hard). Feature selection is performed using the method proposed in Section 3, using hard pairwise functions as in (5), where the cut-off threshold τ is set to 80 mm.

Subset selection (soft). Feature selection is performed using the method proposed in Section 3, using soft pairwise functions as in (6), where τ is set to 80 mm..

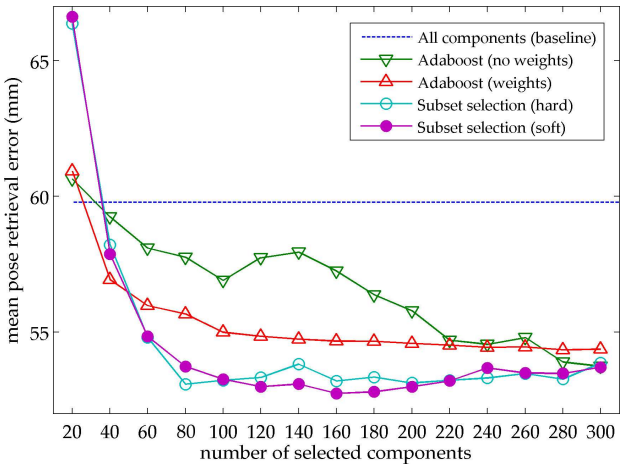

Figure 4: Evaluation results on synthetic images using single character.

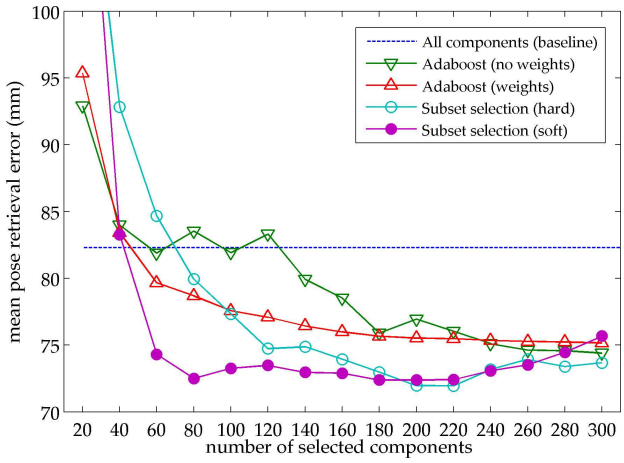

Figure 5: Evaluation results on synthetic images using multiple characters.

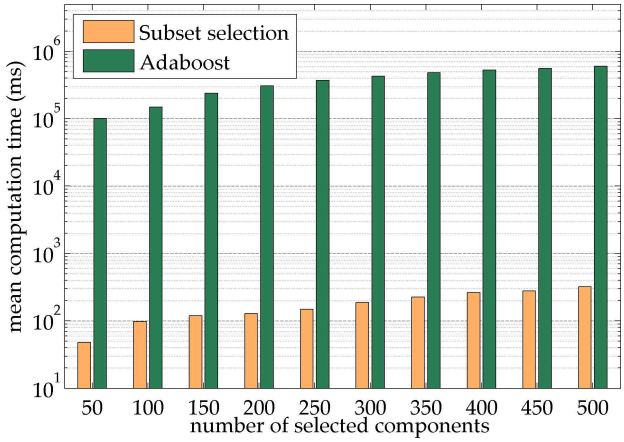

Figure 6: Comparison of computation time in training stage.

Adaboost (no weights). Feature selection is performed by Adaboost as in [6]. Weights of the selected features are discarded.

Adaboost (weights). Feature selection is performed using Adaboost as described above. Weights of the selected features are reserved. That is, each selected feature component is scaled by the corresponding weight.

5.1.4. Results using a single character

First, we evaluate the methods on a single character. In the 57600 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 10000 pairs for feature selection. Then, the evaluation is performed on the testing images. The results are plotted in Figure 4. 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 the selected features play an important role. If the weighs are not used, error is notably larger.

5.1.5. Results using multiple characters

Now we consider the evaluation on the 57600 images of all the eight characters in Figure 3. This is clearly more difficult than the single-character case, as the appearances of different characters differ a lot. We use the 28800 data samples from characters *Agragor*, *BBallPlayer*, *Edward* and *Gaulix* for training and the remaining 28800 samples of *Gaunt*, *Humanoid*, *Mia* and *Roger* for testing. That is, no character appears in both training and testing. This significantly increases the difficulty, because the algorithm has no way to learn the testing characters' appearances from the training data. In this way we emphasize on the generality, which is extremely important for image based pose recovery. From the 28800 training images, 100000 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 Figure 5. It can be seen that our subset selection algorithm generate 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 dealing with different human appearances. For Adaboost, the weights of selected features are still important as we observed in Section 5.1.4.

5.1.6. Comparison on computation time

As said in Section 3.5, 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 $O(d \times d' \times M)$.

Figure 6 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 perform evaluations on the feature selection method using the real images from HumanEva dataset,

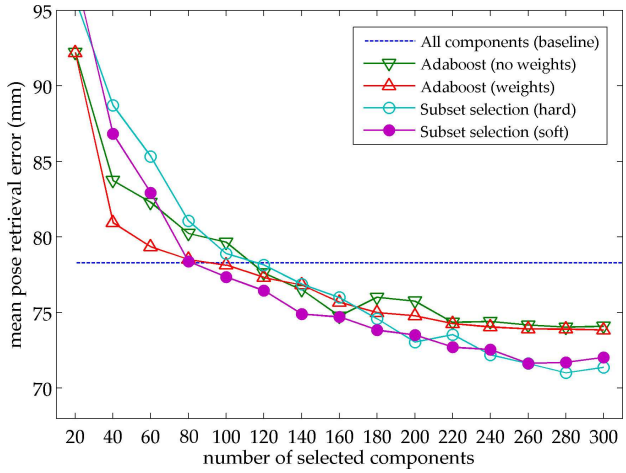

Figure 7: Evaluation results on real images.

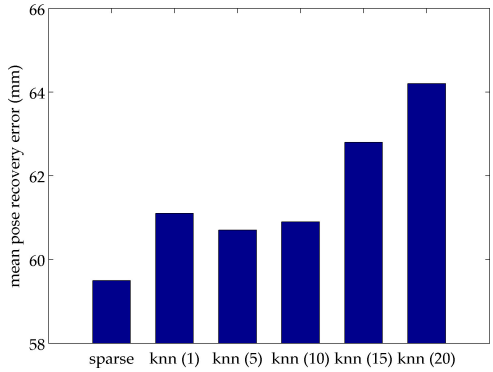

Figure 8: Comparison of sparse representation and nearest neighbor method.

where all subjects S1, S2 and S3 are included in evaluation. 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. The original testing data is not used, as ground-truth motion data is not provided. Moreover, some 3D poses in the original dataset are marked as invalid by the dataset publisher, and are removed from evaluation. In this way we generate a total of 4729 training data and 4848 testing data. Each sample contains the image and corresponding ground-truth 3D pose.

In contrast to the previous subsection, here we do not utilize silhouette-based visual features. Instead, we employ histogram of gradients [21][13] that is calculated from the original image[1]. 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, a set of $H \times W \times K$ feature components are extracted for each image.

We set $K = 9$, i.e. each direction bin corresponds to a range of 20^oThe setting of H and W are not so easy, as they specify 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, from which feature selection is performed.

The evaluation metric is the same as in Section 5.1.2., and Figure 7 shows the results. When the number of selected components is relatively small (≤ 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.

[1]Actually, we can 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.

5.3. Evaluating Pose Retrieval via Sparse Representation

In this subsection we evaluate performance of sparse representation against nearest neighbor method. The training data is the same as in Section 5.1.4, i.e. 3600 samples belonging to *Humanoid* character. The testing data is a sequence of real images from subject S1 in HumanEva dataset, where she walks in a 360^o circle (see Figure 9). Here we fix the feature to be the 150 components selected in Section 5.1.4 using our subset selection algorithm with soft pairwise functions. Pose retrieval is conducted using sparse representation as well as k-nearest-neighbor, 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-nearest-neighbor, 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 (19) in $k$nn case). For sparse representation, we fix the regularizer γ to be 50. The comparison is plotted in Figure 8. It is clear that sparse representation generates a lower recovery error compared to k-nearest-neighbor method.

5.4. Evaluating sequential optimization

In this subsection we evaluate the sequential optimization via dynamic programming. The training data and testing data are the same as in Section 4.1. We still fix the feature to be the 150 components selected in Section 5.1.4 using our subset selection 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 sequential optimization methods are compared.

Topmost. The topmost match of each frame is used. This does not consider temporal information at all.

Simple Incremental. For frame $t = 1$, topmost match is used. For frame $t = 2, ..., T$, the pose is inferred from the feature cue of the current frame and the recovered pose of the previous frame. Specifically, the index of recovered pose of frame t is:

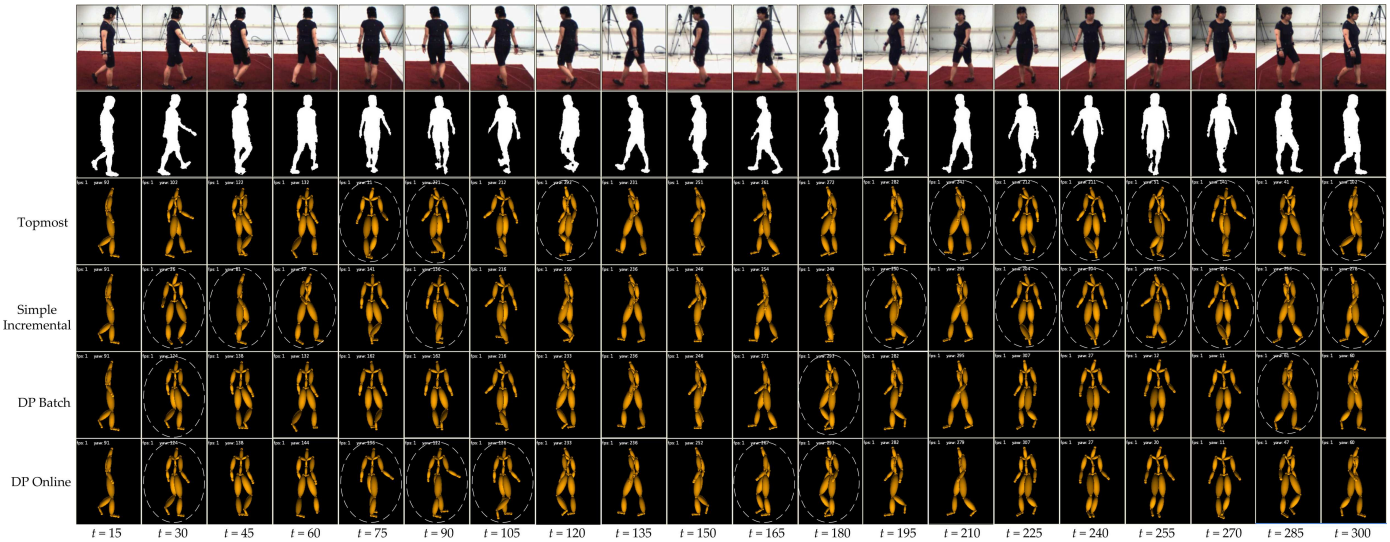

Figure 9: Evaluating sequential optimization. The first and second rows are images and silhouettes, respectively. The third, fourth, fifth and sixth rows are the recovered poses using four different sequential optimization methods. Significant incorrect poses are marked by dash ellipses.

$$c^* = \arg\min_c \left(f_{t,c} + \beta d_{\text{pose}}(\mathbf{y}_{t-1}, \mathbf{y}_{t,c}) \right), \tag{23}$$

where $f_{t,c}$ is the feature cue defined as in (19), $\mathbf{y}_{t-1}$ is the recovered pose in frame $t-1$, and β is the weighting parameter.

DP Batch. This is the dynamic programming method proposed in Section 4.2, used in batch mode.

DP Online. This is the dynamic programming method proposed in Section 4.2, used in online mode.

Table 1 shows the quantitative comparison of mean recovery errors for the four sequential optimization methods. As can be seen, our approach using dynamic programming provides the best results. The performance is lower with the Online approach, which can not use future measures to correct erroneous estimates. From the two worst methods, it is interesting to see that the topmost method performs better than the the incremental one, demonstrating that the increase in performance can not only be obtained through temporal smoothing, but that it is important to maintain multiple hypotheses.

Figure 9 illustrate the results on one example. 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 in Figure 9. "DP Batch" and "DP Online" produces best results. For "Topmost", the recovered poses are not continuous, and many suffer from reflective ambiguity. For "Simple Incremental", the result is not satisfactory either. Also, the tracking is lost due to error 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 when $t = 120$.

5.5. System implementation

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

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

Because HumanEva only contains a limited amount of motion. We also use our own motion capture device to capture some more motion data, such as punching, kicking, various gestures. These data are also used in the example database to recover more motions. See Figure 10 for several illustrations.

6. Conclusions

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

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

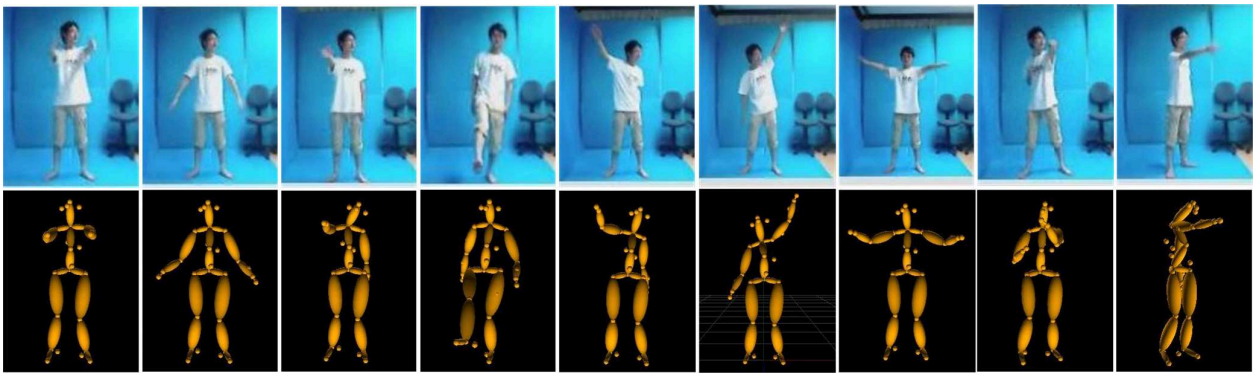

Figure 10: Recovery results for online system.

References

[1] T. B. Moeslund, E. Granum, A survey of computer vision-based human motion capture, Computer Vision and Image Understanding 81 (3) (2001) 231–268.

[2] 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.

[3] R. Poppe, Vision-based human motion analysis: An overview, Computer Vision and Image Understanding 108 (1-2) (2007) 4–18.

[4] C. T. Zahn, R. Z. Roskies, Fourier descriptors for plane closed curves, Transactions on Computers, IEEE c-21 (3) (1972) 269–281.

[5] R. D. de León, L. E. Sucar, Human silhouette recognition with Fourier descriptors, in: ICPR, 2000, pp. 3713–3716.

[6] C. Chen, Y. Zhuang, J. Xiao, F. Wu, Adaptive and compact shape descriptor by progressive feature combination and selection with boosting, in: CVPR, 2008.

[7] L. Ren, G. Shakhnarovich, J. K. Hodgins, H. Pfister, P. A. Viola, Learning silhouette features for control of human motion, ACM Trans. Graph. 24 (4) (2005) 1303–1331.

[8] 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.

[9] X. Zhao, Y. Liu, Generative tracking of 3d human motion by hierarchical annealed genetic algorithm, Pattern Recognition 41 (8) (2008) 2470–2483.

[10] M. W. Lee, R. Nevatia, Human pose tracking in monocular sequence using multilevel structured models, IEEE Trans. Pattern Anal. Mach. Intell. 31 (1) (2009) 27–38.

[11] A. Agarwal, B. Triggs, Recovering 3D human pose from monocular images, IEEE Trans. Pattern Anal. Mach. Intell. 28 (1) (2006) 44–58.

[12] N. R. Howe, Silhouette lookup for monocular 3d pose tracking, Image Vision Comput. 25 (3) (2007) 331–341.

[13] 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.

[14] C. Sminchisescu, A. Kanaujia, D. N. Metaxas, Learning joint top-down and bottom-up processes for 3d visual inference, in: CVPR (2), 2006, pp. 1743–1752.

[15] 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.

[16] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Trans. Pattern Anal. Mach. Intell. 24 (4) (2002) 509–522.

[17] G. Mori, S. J. Belongie, J. Malik, Efficient shape matching using shape contexts, IEEE Trans. Pattern Anal. Mach. Intell. 27 (11) (2005) 1832–1837.

[18] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, J. S. B. Mitchell, An efficiently computable metric for comparing polygonal shapes, IEEE Trans. Pattern Anal. Mach. Intell. 13 (3) (1991) 209–216.

[19] M. K. Hu, Visual pattern recognition by moment invariants, IRE Transactions on Information Theory IT-8 (1962) 179–187.

[20] L. Gorelick, M. Galun, E. Sharon, R. Basri, A. Brandt, Shape representation and classification using the poisson equation, IEEE Trans. Pattern Anal. Mach. Intell. 28 (12) (2006) 1991–2005.

[21] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: CVPR (1), 2005, pp. 886–893.

[22] S. Nayak, S. Sarkar, B. L. Loeding, Distribution-based dimensionality reduction applied to articulated motion recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2009) 795–810.

[23] A. Agarwal, B. Triggs, A local basis representation for estimating human pose from cluttered images, in: ACCV (1), 2006, pp. 50–59.

[24] P. Scovanner, S. Ali, M. Shah, A 3-dimensional sift descriptor and its application to action recognition, in: ACM Multimedia, 2007, pp. 357–360.

[25] T. Serre, L. Wolf, T. Poggio, Object recognition with features inspired by visual cortex, in: CVPR (2), 2005, pp. 994–1000.

[26] A. Agarwal, B. Triggs, Hyperfeatures - multilevel local coding for visual recognition, in: ECCV (1), 2006, pp. 30–43.

[27] D. Nistér, H. Stewénius, Scalable recognition with a vocabulary tree, in: CVPR (2), 2006, pp. 2161–2168.

[28] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: CVPR (2), 2006, pp. 2169–2178.

[29] I. Laptev, On space-time interest points, International Journal of Computer Vision 64 (2-3) (2005) 107–123.

[30] 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.

[31] M. Bregonzio, S. Gong, T. Xiang, Recognising action as clouds of space-time interest points, in: CVPR, 2009.

[32] R. Poppe, M. Poel, Comparison of silhouette shape descriptors for example-based human pose recovery, in: FG, 2006, pp. 541–546.

[33] 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

[34] F. Nie, S. Xiang, Y. Jia, C. Zhang, S. Yan, Trace Ratio Criterion for Feature Selection, in: AAAI, 2008, pp. 671–676.

[35] G. Shakhnarovich, P. Viola, T. Darrell, Fast pose estimation with parameter-sensitive hashing, in: ICCV, 2003, pp. 750–757.

[36] 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.

[37] 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.

[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.