ELSEVIER

Image and Vision Computing 20 (2002) 397–414

image AND vision COMPUTING

www.elsevier.com/locate/imavis

The dynamics of linear combinations: tracking 3D skeletons of human subjects

Eng-Jon Ong*, Shaogang Gong

Department of Computer Science, Queen Mary and Westfield College, University of London, London E14NS, UK

Received 16 October 2000; accepted 18 December 2001

Abstract

We propose a general framework for addressing three fundamental issues using linear combinations: (1) the properties of the examples to linearly combine, (2) the constraints, and (3) the method for estimating the linear combinations coefficients for reconstructing an object based on noisy and incomplete visual observations. To this end, we synthesise the necessary examples from known data using principal component analysis. Crucially, the dynamics of the object is dealt with by learning spatio-temporal constraints on the coefficients of the linear combinations. The CONDENSATION framework was adopted to estimate the coefficients for legitimate and plausible linear combinations. Finally, we apply the linear combinations framework to track 3D skeletons of human subjects using a hybrid representation. © 2002 Published by Elsevier Science B.V.

Keywords: Dynamics of linear combinations; CONDENSATION; 3D skeleton tracking

1. Introduction

Linear Combinations of Examples [2] have been used to tackle various computer vision problems. A set of examples for an object, which has undergone different transformations, is linearly combined together to make a novel example. As a result, the dynamics of the object are modelled implicitly by this method. However, there are three fundamental issues about the Linear Combinations of Examples method.

1. The properties of the examples to linearly combine. How many examples are needed to reconstruct any valid instance of an object of interest? What are the examples required for this linear combination?
2. Constraints on the linear combinations. How do we constrain the linear combinations such that only plausible results are acquired?
3. Estimating the valid coefficients for the linear combinations to accurately reconstruct the object of interest, given incomplete and noisy observations.

The aim of this paper is to address these issues in the general context of linearly combining examples of an *arbitrary representation*. To this end, we propose a framework under which we can learn the sufficient examples, determine how many are needed, learn the constraints for the linear combination, and later incorporate this together with temporal information to estimate the valid coefficients for reconstructing some object via Linear Combinations of Examples. The components of the framework and their relations to each other is shown in Fig. 1.

We also present details on applying this framework to the challenging task of tracking the 3D skeleton of a human subject. In this context, the linear combinations method acts as a form of inference engine, whereby the 3D skeleton is inferred (reconstructed) from incomplete information (visual cues) using knowledge in the form of a set of examples and spatio-temporal constraints on how the examples can be combined together over time.

1.1. Background and related work

The earliest use of the Linear combination method to combine prototypes of some sort was for recognition on 3D objects [2]. In this paper, 3D objects undergoing rigid transformations were recognised by attempting to reconstruct its projection using a linear combination of different views of the same object. This was extended for matching line drawings by linearly combining known prototypes pictures. Another area, which has made use of the linear combinations method, was in modelling and tracking deformable models using Point Distributed Models (PDMs). One use was for tracking humans across a scene

* Corresponding author. Fax: +44-0207-882-6533.

0262-8856/02/$ - see front matter © 2002 Published by Elsevier Science B.V.
PII: S0262-8856(02)00011-2

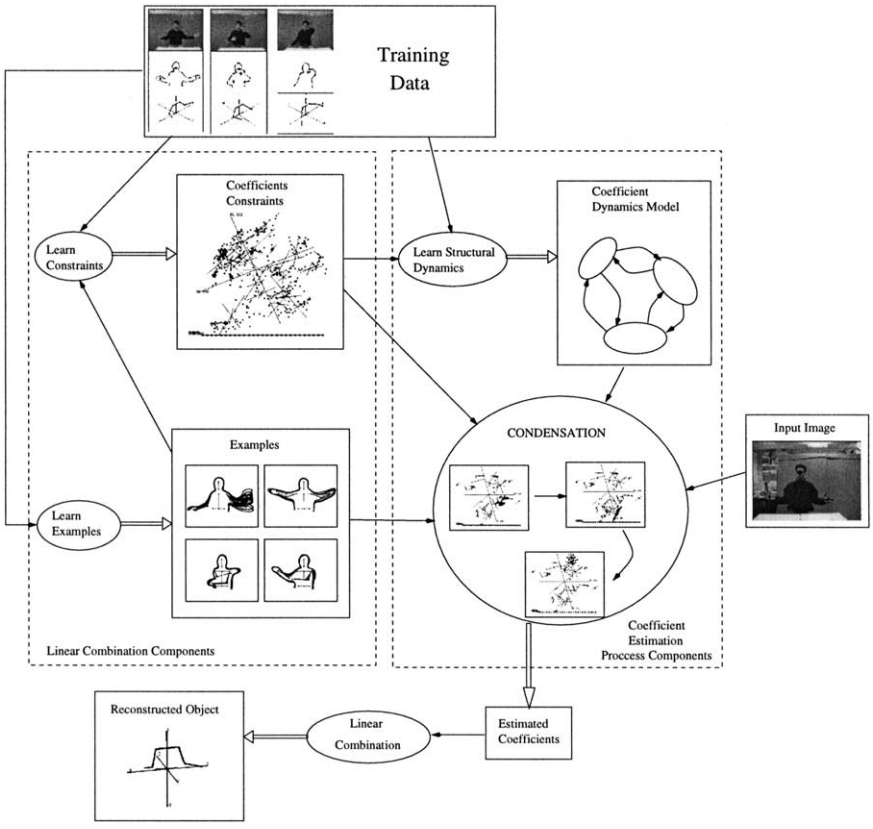


Fig. 1. This figure shows our dynamical linear combinations framework. The ellipses represent functions while the rectangles represent data. The thin arrows represent data being fed into the functions. The results of the functions are denoted by the thick arrowed lines. The framework's components are collected into two major parts. The first part consists of the linear combination method's components: the examples, coefficient constraints and the methods for obtaining them. The second part performs *estimating* the coefficients of the linear combination for reconstructing an object of interest in input images.

by modelling shapes of their body silhouettes [5]. A robust search method for recovering the PDM shapes was then proposed in Ref. [6]. Another use was for tracking PDM models of hands with the addition of constraints on the possible linear combinations, allowing more accurate and robust tracking [7]. This approach was taken further in Ref. [9] for tracking 3D skeletons of human subjects by replacing the PDM model with a hybrid representation combining different modes of information together, (3D skeleton and 2D PDM shapes). Another area which utilises this method is for reconstructing faces using Principal Component Analysis (PCA) [10] to learn the examples to linearly combine. The reconstruction has been used for both high-compression coding of face images for video compression [11,12] and face recognition [13].

1.2. Overview of the paper

The following sections are organised into two main parts. The first part consists of details on the framework for learning the parameters for linear combinations and later, reconstructing objects. An overview of this framework is given in Section 2. The details of this framework are provided in Sections 3 and 4.

The second part concerns the use of this framework for tracking 3D skeletons of human subjects. We have adopted the representation introduced by Bowden et al. [9] for this task. The details of this representation and the methods for associating its components with observable visual cues in the input images will be described in Section 5. Results of tracking the 3D skeleton of a human subject are shown and discussed in Section 5.4. In Section 5.5 we tackle the effects of self-occlusions in our representation by introducing a method for modelling the ambiguities in hybrid representation data. This ambiguity model also provides us with a mechanism for tracking across multiple views by view selection amongst views in a multi-camera setup, as described in Section 5.6. The results of the view selection process is shown in Section 5.7 before we conclude in Section 6.

2. An overview on a dynamical linear combinations framework

We now provide an overview on our dynamical linear combinations framework shown in Fig. 1. The framework is split into two main parts. The first part consists of the components for the linear combinations of examples: the examples and coefficient constraints. Given a set of training

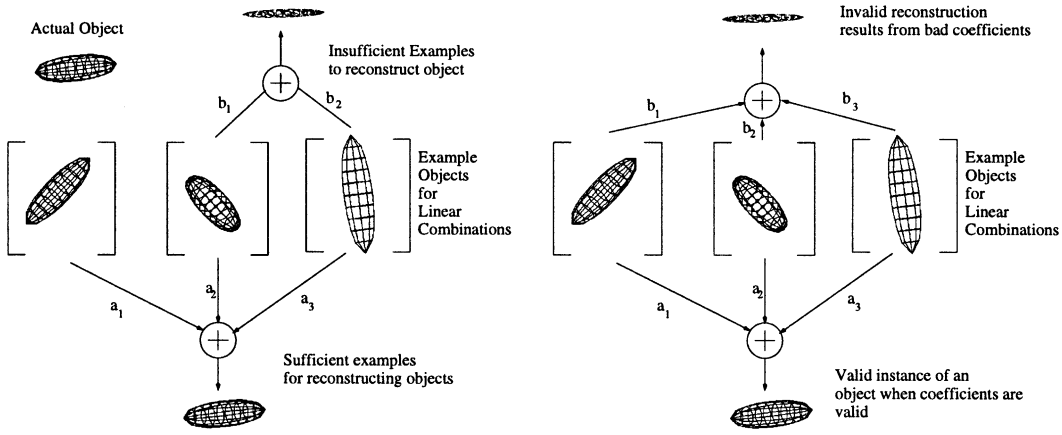


Fig. 2. An illustration of needing sufficient examples for reconstructing objects on the left and the need for constraints on the coefficients for linear combinations on the right.

data, we first tackle the issue of learning the linear combinations examples using PCA. Second, we project the training data onto the *global eigenspace* and model it with piecewise localised clusters. These clusters will act as the constraints for the coefficients. The examples and coefficient constraints will serve as spatial constraints in the estimation process. The full details of the components of this part are given in Section 3.

The other major components fall in the coefficient estimation process part. The structural dynamics of the coefficients is learnt to provide temporal constraints. The spatial and temporal information is then integrated together into the CONDENSATION framework for estimating the coefficients for reconstructing an object based on observable visual cues from input images. The estimated coefficients are then used to linearly combine the examples to reconstruct the object of interest in each input image. The details of this part are given in Section 4.

3. Representation for linear combinations and learning

In this section, we address two issues concerning the method of linear combinations of examples. The first involves obtaining a sufficient amount of examples which captures enough variations in the configurations of the object we wish to reconstruct. The issue then arises of constraining the linear combinations to generate *only* valid objects. Both issues are shown in Fig. 2.

For tackling both the issues described earlier, we have adopted the Hierarchical Principal Component Analysis (HPCA), a non-linear PCA [8,9]. Briefly, this method consists of two main parts:

1. To reconstruct all the training data using the linear combination method, we only need examples which together span the subspace occupied by the training data. To do this, PCA is used to recover the necessary and sufficient examples for linear combinations. The result is a lower dimensional global eigenspace, on to which all the training data will be projected. The details are described in Section 3.2.
2. However, the projection of data into the global eigenspace often occupies non-linear regions. Thus, the second part serves to model these non-linear regions using piecewise clusters. We will also find that these clusters will serve as the constraints on the possible linear combinations which can take place. Section 3.3 will provide more details.

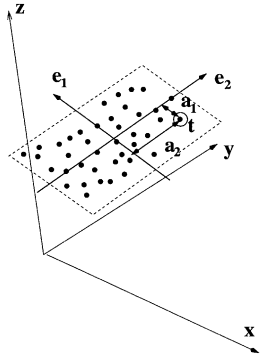


Fig. 3. An illustration of sufficient example vectors for reconstructing all the training data (shown as black circles) using the linear combinations method. The plane upon which all the training data falls is shown with dashed lines.

3.1. Linear combination of examples

First, we provide details on the linear combination of examples, given an arbitrary N-dimensional representation (i.e. a system with N variables for tracking). Suppose we have, a set of (E) example instances, $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_E\}$, of this representation, we can reconstruct a novel instance ($\mathbf{n}$) by a linear combination

$$\mathbf{n} = \sum_{i=1}^{E} a_i \mathbf{e}_i \tag{1}$$

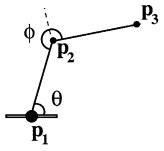


Fig. 4. An illustration of the articulated object with three vertices.

where the set of coefficients for the linear combination is $\{a_1, a_2, ..., a_E\}$.

3.2. Learning the examples for linear combinations

We now describe the method for synthesising the sufficient number of examples for reconstructing novel instances of an arbitrary representation. First, we provide a few definitions. We define an instantaneous state of a N-variable system with $\mathbf{s} = \{s_1, s_2, ..., s_N\}$. We also define a training data set as $\mathbf{T} = \{t_1, t_2, ..., t_{N_T}\}$, where N_T denotes the total number of training examples and $\mathbf{t}_j = \{t_{j,1}, t_{j,2}, ..., t_{j,N}\}$.

We now provide a simple example of what constitutes sufficient examples for reconstructing the training data with linear combinations. Suppose we have a system with three variables, $\{x, y, z\}$ (see Fig. 3). The state of the system can be defined by a three-dimensional vector. We also have a set of 3D training data, all of which falls onto a plane. In order to reconstruct all the training data, only two linearly independent vectors, $\{\mathbf{e}_1, \mathbf{e}_2\}$, lying on the training data plane are required. To reconstruct the training example, $\mathbf{t}$, we linearly combine the $\mathbf{e}_1$ and $\mathbf{e}_2$ as follows

$$\mathbf{t} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2$$

where $\{a_1, a_2\}$ are the coefficients for the linear combination. This is shown in Fig. 3, where these coefficients are just the projections of $\mathbf{t}$ onto $\mathbf{e}_1$ and $\mathbf{e}_2$.

We extend the above example to a more general case, where the number of examples is given by, E, and the basis examples for linear combinations defined as $\{\mathbf{e}_1, ..., \mathbf{e}_E\}$. In the general case of an N-variable system with the training set, $\mathbf{T}$, we often find that the training data only occupies a lower dimensional subspace. We use PCA to determine this subspace, providing us with a set of orthonormal basis examples, $\{\mathbf{e}_1, ..., \mathbf{e}_N\}$. Each basis example lies on the directions of maximal variations in the training data. Associated with each basis example, $\mathbf{e}_j$, is its eigenvalue, λ_j, representing the variance of the training data along its respective direction. The number of sufficient basis examples (E) for reconstructing the training data can be found by retaining only those with a significant eigenvalue. Basis examples with small eigenvalues are rejected on the assumption that they are caused by noise in the training data.

These basis examples can then be linearly combined to generate novel hybrid vectors since they span the subspace of the training data. Thus, PCA has provided us with a method for synthesising the required examples for linear combination. Obtaining the coefficients, $\{a_1, a_2, ..., a_E\}$ for a training example, $\mathbf{t}$, is done simply by projecting it into the normalised global eigenspace

$$a_j = \sum_{i=1}^{N} \frac{e_{j,i}t_i}{\lambda_j} \tag{2}$$

where $j \in \{1, ..., E\}$, λ_j is the eigenvalue of the jth example, $e_{j,i}$ and t_i are the ith component of the jth basis example and training example, respectively. For this reason, from now on, we will identify the eigenspace as the *coefficient space*. However, while PCA has provided us with example bases for generating novel vectors, these may also include *invalid* reconstructions.

3.3. Learning constraints on the linear combination coefficients for validity

Having shown how to obtain the necessary examples for linear combinations, we now address the problem of learning the constraints on the linear combination coefficients. The need for placing constraints on the coefficients is illustrated with a simple example of a 2D hierarchical articulated object (see Fig. 4). The articulated object has three 2D vertices, p_1, p_2 and p_3.

A hierarchical structure is imposed on the vertices of this object. The vertex, p_1, is the parent of all the vertices. The second vertex, p_2, is linked to p_1, and the third vertex p_3, is linked to p_2. Therefore, both p_2 and p_3 are affected by any transformations on p_1. Additionally, any transformation on p_2 affects p_3. Next, we make a 'transformation-unified' representation for the articulated object by concatenating all its vertices into a six-dimensional vector $\mathbf{o} = (p_{1,x}, p_{1,y}, p_{2,x}, p_{2,y}, p_{3,x}, p_{3,y})$. The dynamics of this articulated object is defined by the kinematics equations, $(f_1(\theta, \phi), ..., f_6(\theta, \phi))$, defined in Appendix A, which produces the values for the components $(p_{1,x}, p_{1,y}, p_{2,x}, p_{2,y}, p_{3,x}, p_{3,y})$, respectively. An illustration of the articulated object and its kinematics parameters can be seen in Fig. 4.

Suppose we have a set of training data generated by different combinations of the kinematics parameters (θ, ϕ). Using PCA as described in Section 3.2, we obtain the basis vectors $\{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_E\}$ where $E \leq 6$. These basis vectors will be the axes of the E-dimensional coefficient space. The equations, which all the coefficients must satisfy, can be found by projecting the kinematics equations onto the normalised eigenspace giving the constraints

$$c_i = \frac{1}{\lambda_i}(e_{i,1}f_1(\theta, \phi) + e_{i,2}f_2(\theta, \phi) + e_{i,3}f_3(\theta, \phi) + e_{i,4}f_4(\theta, \phi) + e_{i,5}f_5(\theta, \phi) + e_{i,6}f_6(\theta, \phi)) \tag{3}$$

where $i \in \{1, ..., E\}$, $\mathbf{e}_i = \{e_{i,1}, e_{i,2}, ..., e_{i,6}\}$ and λ_i is the eigenvalue of $\mathbf{e}_i$. For all values of θ and ϕ, the ith coefficient must satisfy Eq. (3) to reconstruct a valid form of the articulated object. The constraint surface for all the valid coefficient sets is visualised in Fig. 5.

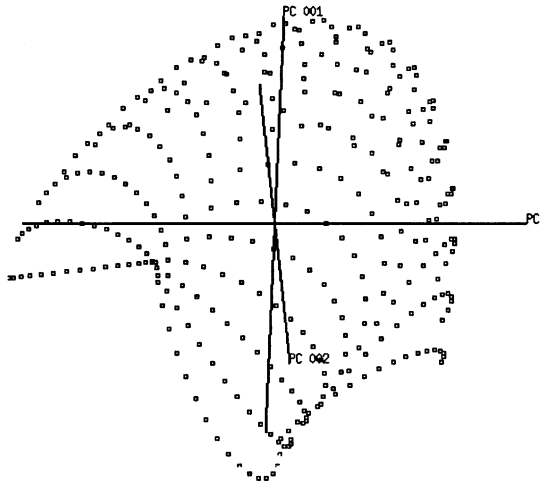


Fig. 5. Visualisation of the constraint surface for Eqs. (A1)–(A6). Displayed here are points with coordinates (c_1, c_2, c_3), produced using different parameters θ and ϕ for the articulated object shown in Fig. 4.

From this, it is clear that the articulated object's valid configurations can only be reconstructed by choosing coefficient sets which lie on a non-linear surface in the coefficient space. The shape of the constraint surface (or high dimensional volume in a more complex case) is determined by the constraint equations, for example, Eq. (3) in this case. Often, these constraint equations can become very complex. Therefore, it may not be realistic to explicitly model the constraint equations. Moreover, in many cases, the underlying dynamics are unknown.

Seeking a more general solution, we model the constraints using a set of piecewise linear clusters. Given a set of training data, we first project all the examples in this set into the coefficient space. We then use a predetermined number (N_{clust}) of clusters, $\{\mathbf{c}_1, \ldots, \mathbf{c}_{N_{\text{clust}}}\}$, to model the projected training data. Each cluster ($\mathbf{c}_i$) consists of a mean position (μ_i), a set of eigenvectors ($\mathbf{P}_i$) and its corresponding eigenvalues ($\mathbf{\Lambda}_i$); $\mathbf{c}_i = (\mu_i, \mathbf{P}_i, \mathbf{\Lambda}_i)$. The mean positions of the clusters are determined using k-means clustering [14]. The covariance matrix of each cluster is found using the training data in its partition. The covariance matrices of all the clusters are then replaced by their eigenvectors and corresponding eigenvalues.

In this section, we have described how we can create enough examples for performing linear combinations and illustrated the importance of placing constraints on the coefficient space. We have also described the means to learn the constraints using a set of piecewise linear clusters in the coefficient space.

4. Modelling the dynamics of coefficients for linear combinations

Having addressed the issues concerning the examples and constraints on the linear combinations method in Section 3, we now describe our method for reconstructing objects. Representing the object with a N-dimensional vector, the problem at hand is to estimate the coefficients ($\{a_1, a_2, \ldots, a_E\}$) for the linear combination of the known examples ($\{\mathbf{e}_1, \ldots, \mathbf{e}_E\}$) such that an object ($\mathbf{n}$) of interest is reconstructed accurately. In practice, we do not know the *actual* values of the object ($\mathbf{n}$). However, we may have an approximation ($\hat{\mathbf{n}}$) acquired from using visual observations (e.g. by deforming an initial model to some visual observations representing the object). This approximation ($\hat{\mathbf{n}}$) may be corrupted by noise in the observations and ambiguities in the deformation process. Therefore, the task now is to remove the corrupting elements from the deformed object. This entails reconstructing the closest object to ($\hat{\mathbf{n}}$) using linear combinations. Mathematically, this involves obtaining the coefficient set ($\{a_1, a_2, \ldots, a_E\}$), which minimises the magnitude of the 'approximation residuals' vector ($\text{res}_1, \ldots, \text{res}_N$),

$$\text{res}_1 = n_1 - a_1 e_{1,1} + a_2 e_{2,1} + \cdots + a_E e_{E,1}$$

$$\text{res}_2 = n_2 - a_1 e_{1,2} + a_2 e_{2,2} + \cdots + a_E e_{E,2}$$

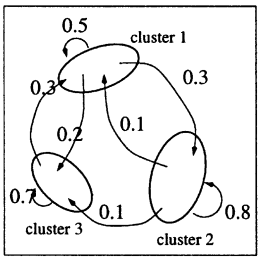
$$\vdots$$

$$\text{res}_N = n_N - a_1 e_{1,N} + a_2 e_{2,N} + \cdots + a_E e_{E,N}$$

where $\mathbf{n} = (n_1, n_2, \ldots, n_N)$ represents the object's current state. The ith example vector's components are denoted by $(e_{i,1}, e_{i,2}, \ldots, e_{i,N})$.

Typical methods for minimising these equations (i.e. obtaining the appropriate coefficients) involves some form of least-squares minimization [2,15] or other optimisation procedures [6,16]. We have adopted the CONDENSATION [17] framework, which estimates the coefficients by tracking them over time. That is, this algorithm allows spatio-temporal knowledge to be used, allowing for a more robust tracking. The spatial knowledge consists of the learnt coefficient space (global eigenspace) and constraints (piecewise clusters) as described in the Section 3. The temporal knowledge is modelled with two methods. For modelling the 'structural dynamics' of coefficients, we adopt a Markov model of transition probabilities between the different piecewise clusters, as described in Section 4.1. On the finer scale within the clusters, we simply use Brownian motion (random displacements). An overview of the CONDENSATION framework is given in Section 4.2. The full details of the framework can be found in Ref. [17].

In this context, the CONDENSATION framework consists of an algorithm working on a set of samples. Here, a sample is a coefficient set for a single linear combination. The algorithm basically propagates the samples in the coefficient space based on learnt dynamics of the coefficients. The samples are rated on how well they fit the observable data which is used in the next iteration's propagation step. Here, the use of multiple samples serves as a mechanism for having multiple hypotheses, allowing recovery from failure in tracking.



Transition Matrix

	1	2	3
1	0.5	0.3	0.2
2	0.1	0.8	0.1
3	0.3	0.0	0.7

Fig. 6. The transition matrix is illustrated in this diagram. The left image illustrates the clusters and the transition probabilities to other clusters.

4.1. Learning the dynamics of coefficients

We have described the method for modelling the examples and constraints for the linear combinations in Section 3. However, these do not provide any information on the dynamics of the state vectors and how the linear combination's coefficients will evolve over time. Knowing the dynamics of the coefficients can add further constraints to reduce ambiguities while estimating the coefficients. This provides the advantages of increasing the robustness as well as the computational efficiency of the estimation process, and avoiding invalid reconstructions. The rest of this section provides details on modelling the structural dynamics of the linear combinations coefficients using a Markov process [7]. The states of the Markov model are the piecewise clusters described in Section 4.1. To account for potential large discontinuous behaviours in the dynamics, each state is fully connected to all the other states. Thus, a point belonging to one cluster has the ability to move to any other clusters at the next time step. The probability of it jumping to another cluster is given in the transition matrix (**U**) of the Markov model. This is shown in Fig. 6.

The ith row vector of the transition matrix (**U**) consists of the transitional probabilities of the ith cluster. The transition matrix can be constructed as follows. Starting with a zero transition matrix, for every frame of a training sequence, the memberships of the current and next frames' state vector's clusters are found. If the current frame belongs to ith cluster and the next frame to the jth cluster, the element of the ith row and jth column of the transition matrix is incremented. This is done for all the sequences. Finally, the values in each row vector of the transition matrix are normalised.

4.2. CONDENSATION for propagating coefficients

We now briefly outline the CONDENSATION algorithm. First, we define a *sample* as a point in the coefficient space (i.e. a coefficient set). Associated with each sample is a probability measure indicating how well it represents the real value. Initially, all the samples are assigned equal probabilities and their components are randomly distributed within their constraints. The algorithm then iterates over the following steps:

1. The first step involves the selection of future samples based on their probabilities.
2. The selected samples are then propagated based on some model of their dynamics. This propagation step has the equivalent effect of predicting the future state of the samples.
3. The accuracy of the prediction is then determined in the next stage of measuring how well each sample 'fits' with the observation data. The probabilities of the samples are updated in proportion to its fitness value.

In adapting the algorithm for estimating the linear combinations coefficients, the sample prediction step is modified to make use of the transitional probability matrix (**U**) and the coefficients constraints (clusters) as described in Sections 4.1 and 3.3, respectively. This allows the tracker to propagate samples across different subspaces for coping with any discontinuities in the state space.

4.2.1. Propagating the samples

The samples prediction step is split into two steps: (1) the first step involves finding out the new cluster membership, labelled as b, for a sample at time t, $\mathbf{s}_n^t$, based on the transitional probabilities given by the ath row vector of the transition matrix, where a is the sample's current cluster membership; (2) the new position of the sample $\mathbf{s}_n^{(t+1)}$ is determined by displacing it linearly in the directions of the principal components of the clusters

$$\mathbf{s}_n^{(t+1)} = \begin{cases} \mathbf{s}_n^{(t)} + \mathbf{P}_b\mathbf{\Lambda}_b\mathbf{\Omega}, & a = b \\ \boldsymbol{\mu}_b + \mathbf{P}_b\mathbf{\Lambda}_b\mathbf{\Omega}, & a \neq b \end{cases} \tag{4}$$

where $\boldsymbol{\mu}_b$, $\mathbf{P}_b$ and $\mathbf{\Lambda}_b$ are the mean, matrix of the principal axes, and eigenvalues of the bth cluster, respectively. $\mathbf{\Omega}$ consists of the vector whose elements are unit Gaussian distributed random values [7].

The position of the new sample, $\mathbf{s}_n^{(t+1)}$, is then constrained to lie within the bounds of its newly assigned cluster $\mathbf{c}_b$. This is done by projecting the sample as follows:

$$\mathbf{r} = \mathbf{s}_n^{(t+1)\mathrm{T}}\mathbf{P}_b\mathbf{\Lambda}_b^{-1}. \tag{5}$$

The restriction is made to obtain a plausible reconstruction for this sample. All the elements of $\mathbf{r}$ are then limited to lie within the range of -1 to $+1$. Any element outside the bounds of this range is set to 1 or -1, respectively.

The sample's elements are then reconstructed by taking a linear combination of the principal components of cluster $\mathbf{c}_b$:

$$\mathbf{s}_n^{(t+1)} = \mathbf{\Lambda}\mathbf{P}_b^{\mathrm{T}}\mathbf{r}. \tag{6}$$

The estimated state vector (**v**) is reconstructed from the components of a given sample in the same manner, by taking a linear combination of N_{gev} number of *global*

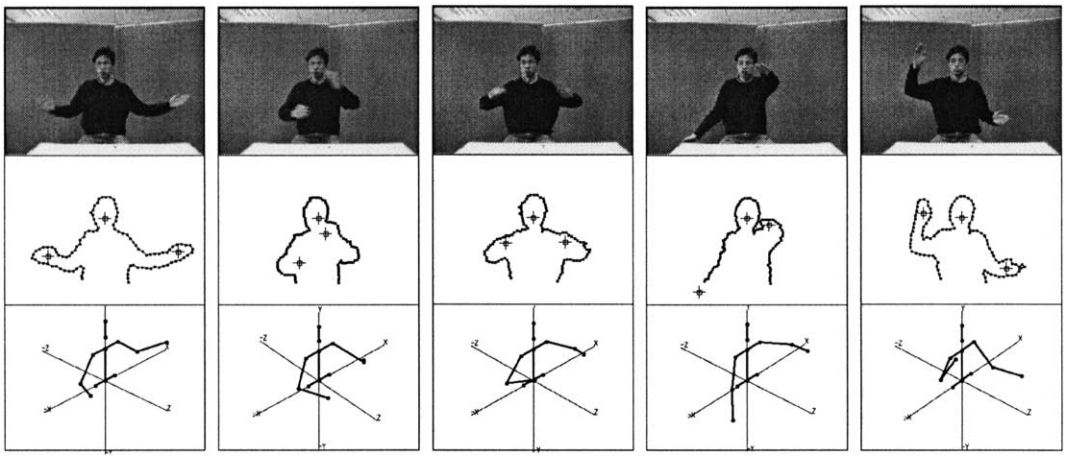


Fig. 7. An illustration of different instances of the state vector. The top row corresponds to the input images. The middle row corresponds to the contours ($\mathbf{v}_S$) and body parts positions ($\mathbf{v}_C$). The bottom row shows the corresponding skeleton ($\mathbf{v}_T$).

eigenvectors ($\mathbf{g}_1, \ldots, \mathbf{g}_{N_{gev}}$) :

$$\mathbf{v} = \sum_{i=1}^{N_{gev}} s_i \mathbf{g}_i. \tag{7}$$

4.2.2. Measuring the samples' fitness

The accuracy of the reconstruction, $\mathbf{v}$ is determined by $P(\mathbf{Z}^t|\mathbf{s}_n^t)$, which calculates the probability of this sample state generating the observable data, $\mathbf{Z}^t$. This function has the effect of 'relating' the estimated reconstruction of, $\mathbf{s}_n^t$, to the input measurements. For example, when $\mathbf{v}$ represents the contour of a shape and the input measurements are edge features, $P(\mathbf{Z}^t|\mathbf{s}_n^t)$, can be a function which calculates how much each vertex of the shape's control points has to move to its nearest edge. The sample vector with the highest fitness value is selected and its reconstruction is used.

5. An application: tracking the 3D skeleton of a human subject

In this section, we describe an application of the dynamical linear combinations method for tracking the pose of a person. We have chosen to represent the pose with a rough 3D skeleton model of a human subject. Related methods for tracking 3D models in individual views include the use of inverse kinematics for estimating the rotational parameters [18]. Other common methods for tracking 3D models uses primitive solids such as superquadrics [19,20] or cylinders [21]. Alternatively, a procedure together with a similarity measurement can be used to compute the parameters of a 3D skeleton [19,21].

We have adopted the hybrid representation introduced by Bowden et al. [9]. Different modes of information are combined into a unified representation. In using the linear combinations framework with this representation, it was found that the examples used also encode the correlations between different modes of information. The different information combined includes shape, body parts positions and the corresponding 3D skeleton. The details of the hybrid representation are given in Section 5.1. In order to measure how well an instance of the hybrid representation represents the subject's actual pose, a representation–image measurement function is introduced in Section 5.2. We then describe the results of learning the examples and their constraints for tracking using linear combinations in Section 5.3. Examples of tracking are shown and discussed in Section 5.4.

To make the tracking more reliable, we introduce mechanisms to allow this method to be used in a multiple camera setup. We start by modelling the ambiguities present in the representation to be described in Section 5.5. We then use this ambiguity model to select the least ambiguous view amongst those provided by different cameras. Finally, we show some results of view selection in Section 5.7.

5.1. Hybrid representation for tracking 3D skeletons

The hybrid representation is the combination of two types of information; observable data and data to be inferred from the former components. The observable data are 2D features, including the 2D shape of a person's body and the positions of some body parts, both of these features can be directly measured from the image. The contour of a person's silhouette is represented by a PDM. It consists of N_C 2D vertices, $\mathbf{v}_S = (x_1, y_1, \ldots, x_{N_C}, y_{N_C})$, distributed evenly across the entire contour. The body parts are represented by the positions of the left hand (x_L, y_L), the right hand (x_R, y_R) and the head (x_H, y_H). The positions of left and right hands are concatenated into a vector, $\mathbf{v}_C$. The head is used as an alignment point for both the body parts and the contour. Therefore, the coordinates of the contour vertices and hand positions are all made relative to the position of the head. Finally, the corresponding 3D data, which consists of N_T 3D vertices of a skeleton, is similarly concatenated into a vector ($\mathbf{v}_T$). Here, the 3D data were obtained manually. This was achieved by firstly determining the length of the

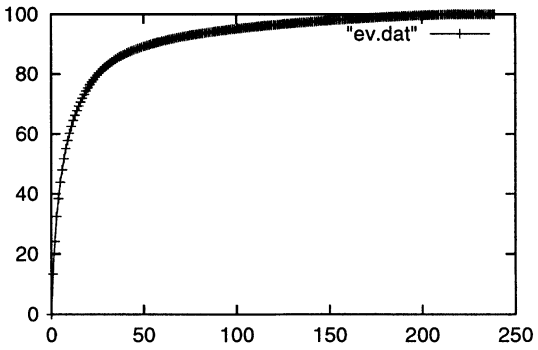


Fig. 8. The weight distribution of eigenvectors in the global eigenspace used to represent the hybrid 2D–3D state vectors.

skeleton bones using an image of a subject with his limbs stretched out. Following this, the 3D skeleton information of the subject at different poses was determined in two steps. First, the 2D image positions of the body joints of a subject were located. Second, using the predetermined bone lengths and 2D joint positions, spherical coordinates can be used to recover the depth information of a joint. A state vector, $\mathbf{v} = (\mathbf{v}_S, \mathbf{v}_C, \mathbf{v}_T)$, therefore is a hybrid concatenation of three 2D–3D components: 2D shape contour, 2D body parts positions and their 3D skeleton vertices (see Fig. 7).

5.2. Measuring the hybrid representation against the observation data

Given a hybrid representation vector $\mathbf{v} = (\mathbf{v}_S, \mathbf{v}_C, \mathbf{v}_T)$, the accuracy of both the shape contour ($\mathbf{v}_S$) and the body parts ($\mathbf{v}_C$) are measured individually before combined to yield a final fitness value. More precisely, a prediction accuracy (f_S) for the contour can be computed as follows:

1. Assign the prediction accuracy, $f_S = 0$.
2. For N_C vertices of a contour:
   (a) find the distance (s) from each 2D shape vertex position to the pixel of greatest intensity gradient by searching along its normal;
   (b) compute $f_S = f_S + s$.

We now consider how to measure the accuracy of the state vector's predictions for the body parts' positions, (x_{p1}, y_{p1}) and (x_{p2}, y_{p2}). In each frame, three skin-coloured regions of interests are tracked corresponding to positions of the hands and the face [9,22]. Two of these three positions are taken and ordered into a four-dimensional vector $\mathbf{m}_b = (x_{m1}, y_{m1}, x_{m2}, y_{m2})$ where (x_{m1}, y_{m1}) and (x_{m2}, y_{m2}) are the coordinates of the first and second position, respectively. An accurate prediction of the body parts positions would contain values of (x_{p2}, y_{p2}) which are close to $\mathbf{m}_b$. Thus, to quantify such a closeness value, it would be useful to have a measure which decreases, as the predicted and observed body parts positions become closer. To this end, the sum of the Euclidean distances between the hypothesised and observed body parts' positions was used. Formally, this can be defined as:

$$f_C = \sqrt{(x_{p1} - x_{m1})^2 + (y_{p1} - y_{m1})^2} + \sqrt{(x_{p2} - x_{m2})^2 + (y_{p2} - y_{m2})^2}. \quad (8)$$

A final fitness value (f_n) for $\mathbf{v}$ of the nth sample, ($\mathbf{s}_n^{(t+1)}$), is then given by the individual fitness measurements as follows

$$f_n = O \exp\left(\frac{-f_C}{2P}\right) + R \exp\left(\frac{-f_S}{2Q}\right) \quad (9)$$

where O and R are scale constants that can be employed to give different reliability weighting to the fitness measurements, f_C and f_S, respectively. Here, we have heuristically set both O and R to be 0.5, giving both the body parts' positions and contour fitness measurements equal importance. A more elegant solution to determine the values of O and R could perhaps be to employ a Bayesian network [1] to determine the reliability weightings of different types of information. Constants P and Q represents the amount of variance or tolerance allowed between the predicted and observed values for the body parts and the shape contour, respectively.

5.3. Learning the hybrid representation

Initially, our single view-based skeleton model was used to track upper torso of people performing a series of different gestures. The 2D shape contour was represented using 100 image feature points. The 3D skeleton consisted of 12 vertices for the upper torso. The hands and head were tracked. Here, only the positions of the hands were used in the state vector. The coordinates of both the hands and contour feature points were made relative to the head. This resulted in a state vector of 240 dimensions. Sixteen continuous sequences were captured and used as training data. Background subtraction and connected components analysis [23] were used to extract the shape contour from the sequence images. The body parts positions were located by firstly finding pixels which were skin-coloured [22] which were partitioned into three clusters using the k-means algorithm [9]. In each sequence, a subject was requested to perform random gestures.

It was found that 30 eigenvectors accounted for roughly 80 percent of the variance in the global eigenspace (see Fig. 8). This number was found to be sufficient for tracking. Additionally, the eigenvectors with smaller variances may only represent the noise in the training data.

In order to approximate the space occupied by the projected training examples, a set of cluster models with different numbers of clusters were built. In total, a total of 10 cluster models, or more specifically, cluster models with 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 clusters were built. A visual example of a cluster model with 40 clusters can be

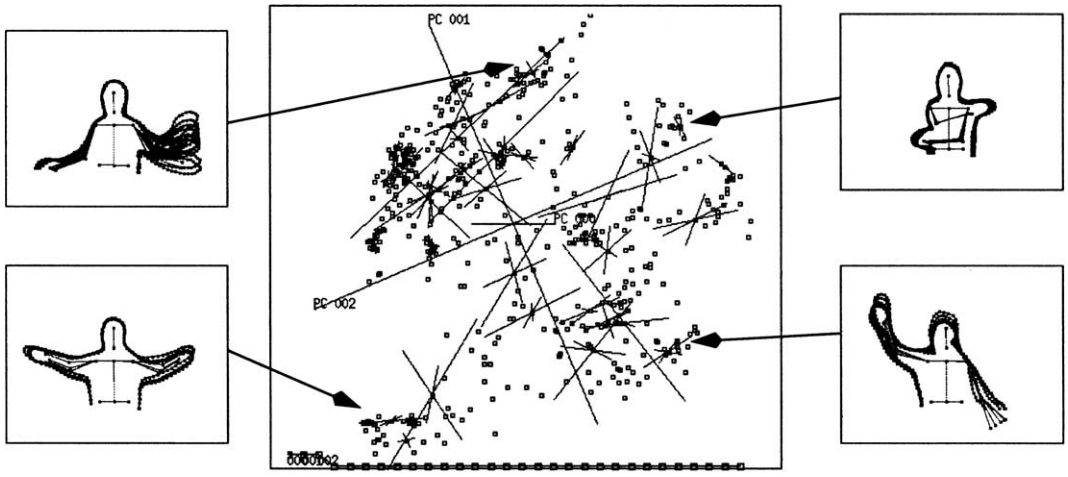


Fig. 9. An illustration of the projections of state vectors to the three largest global eigenvectors. Training example vectors are shown with the local principal components of different clusters. The side figures show the reconstruction of moving points along the largest local principal components from the mean of a cluster.

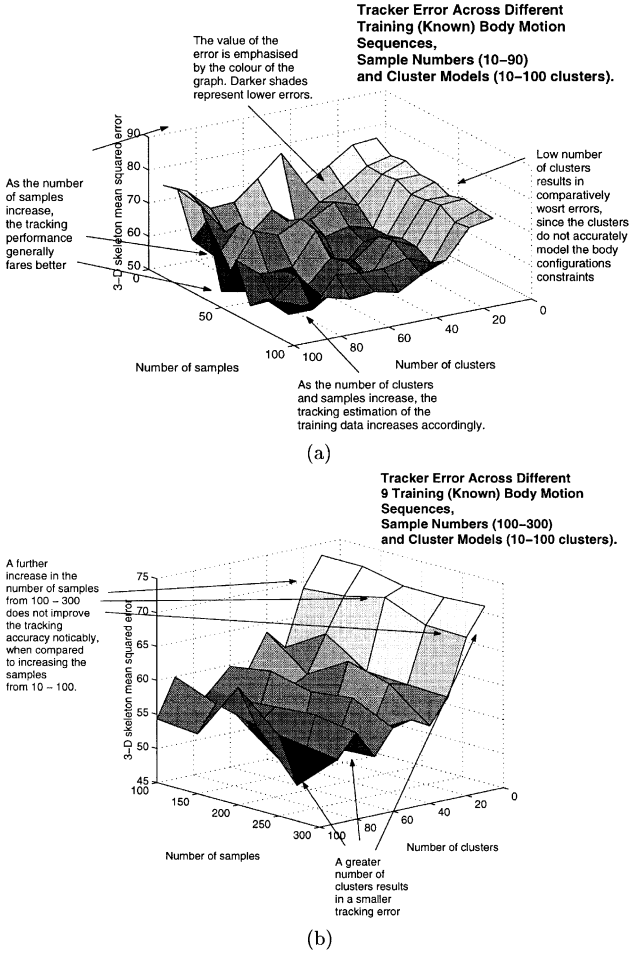


Fig. 10. The error surfaces of the tracking experiments carried out on training data when different number of samples and cluster models are used.

seen in Fig. 9. Once the parameters of each of the cluster models were determined, the training data obtained from the 16 continuous sequences were used to build their corresponding transition matrices.

5.4. Tracking the 3D skeleton

The tracker was implemented on a PC equipped with a Pentium 200 MHz processor. All the input images were captured at a resolution of 320 by 240 pixels. The tracking processing time was between 1 and 3 s for each frame. The majority of the computation time was spent in obtaining the measurements probabilities for the samples.

A series of different experiments on tracking the 3D skeletons using a single view was performed. In order to quantify the accuracy of the tracking, the mean squared error of the 3D skeleton vertices was used. For each experiment, a series of tracking tests using different cluster models and sample numbers were carried out. For the cluster models, all 10 cluster models described in Section 5.3 were used. Additionally for each cluster model, the lack of sufficient training examples resulted in a transitional probability matrix that was not an accurate representation of its real values, slowing down the transition of the samples across different clusters. It was found that iterating the CONDENSATION process for a number of times over the same frame (five times was sufficient for our experiments) allowed the samples to converge on the correct subspace.

We also investigate how differing number of samples can influence the tracking accuracy. To do this, tracking tests using each of the 10 learnt cluster model was carried with a tracker configured with sample numbers of 10–90 in increments of 10 and 100–300 in increments of 50.

To determine how accurately the training poses were learnt in the HPCA model, the tracker was made to track the 3D skeleton of the subject in the training sequences. The

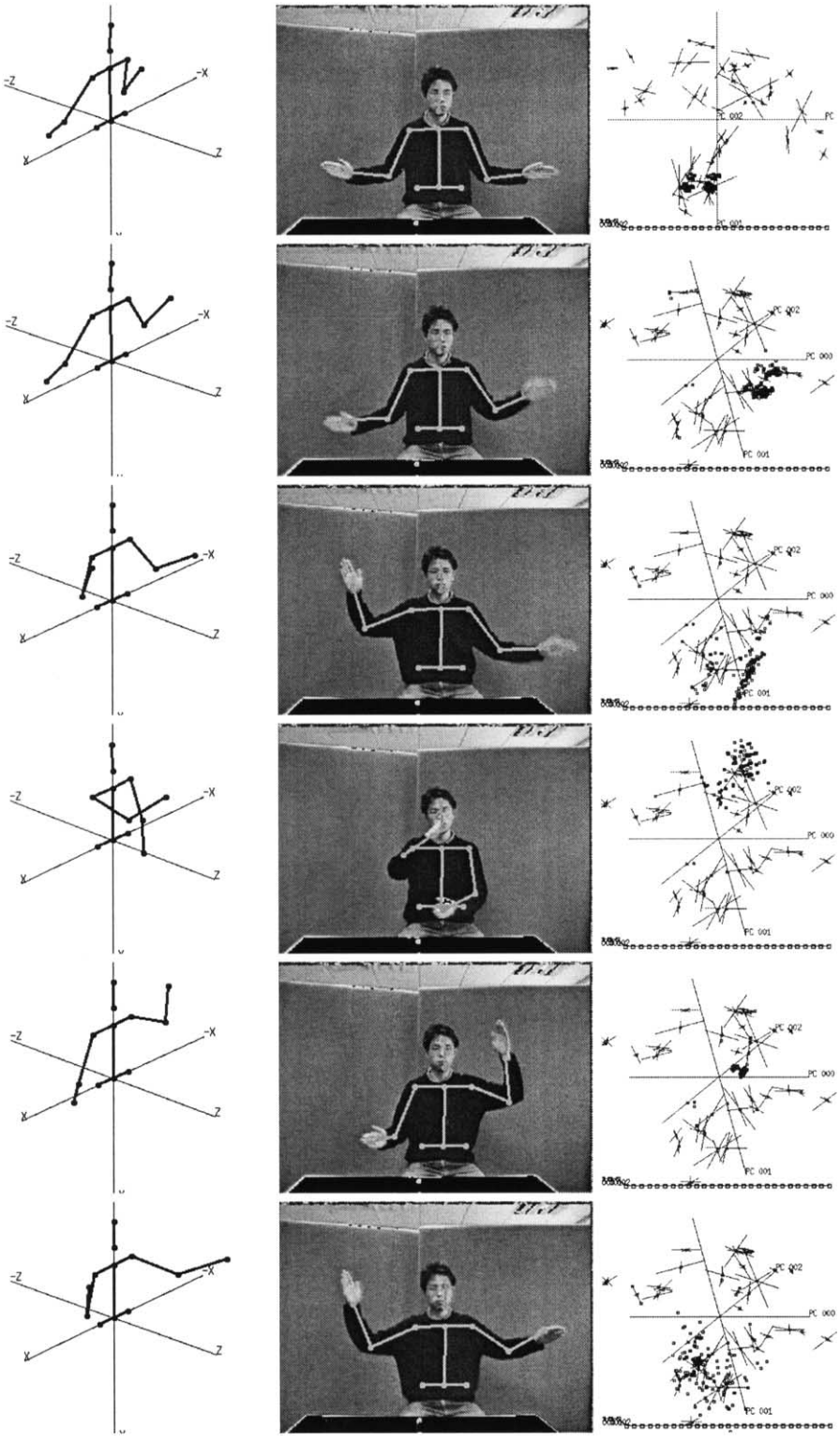


Fig. 11. This figure shows the tracking of 3D skeletons using the CONDENSATION algorithm on the training sequences. Every 10th frame is shown. For each frame, the left window shows the 3D skeleton, the middle window shows the 3D skeleton projected onto the image and the right window shows the global eigenspace together with the localised principal components and the samples tracked using CONDENSATION.

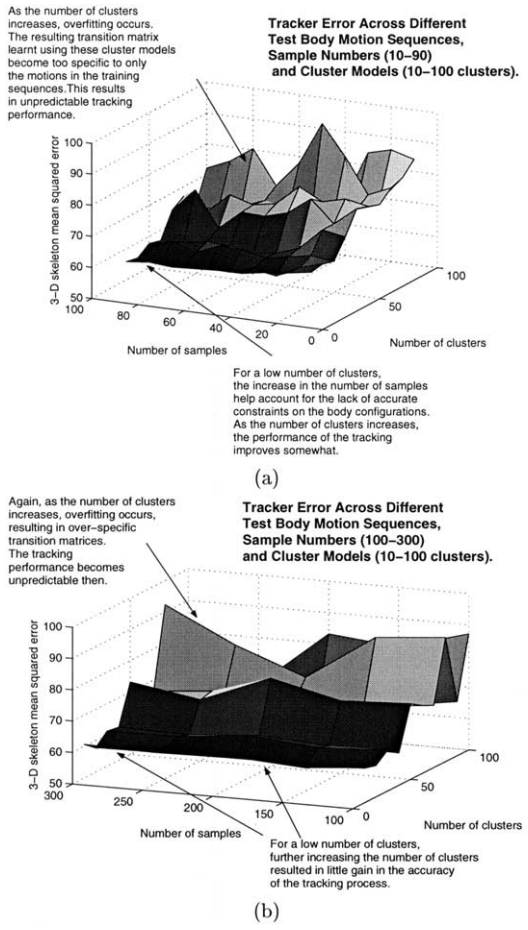


Fig. 12. The error surfaces of the tracking experiments carried out on test sequences of novel body movements with a different number of samples and cluster models with different number of clusters.

mean squared error of each of the 3D skeleton's vertices for different combinations of cluster and sample numbers can be seen in Fig. 10. Additionally, a visual illustration of an example of the tracking results can be seen in Fig. 11.

It was found that as the number of clusters and samples increased, there was also a general trend for improvement in the tracking accuracy. The increasing number of cluster allowed more accurate and realistic body configuration information to be produced. However, as only training sequences are used for this experiment, it is not clear as to how well the tracker can generalise to novel body motions using the different parameters and cluster models.

The subsequent experiments were used to evaluate the tracker's ability to generalise and track novel poses. In order to investigate the tracking accuracy under fairly controlled conditions, a blue screen was placed behind the subject to provide a homogeneous background. The background was then removed to experiment on tracking a subject with the presence of a cluttered background. Finally, to evaluate the tracker's performance in generalising to novel subjects, the fourth experiment was performed with a subject which is not present in any of the training sequences.

The overall error surfaces for the three experiments are shown in Fig. 12. From the results, it was observed that the tracking performance deteriorates and becomes more unpredictable as the number of clusters increase. Such results bear the implication that cluster models with too large a number of clusters have over-fitted the data. The resulting transition matrix for the cluster models would account only for the training motion patterns. A novel sequence may contain transitions between clusters that were not modelled using the training data.

A visual example of tracking with a homogeneous background can be seen in Fig. 13. Having a homogeneous background allowed one for a fairly accurate segmentation of the subject in the input image. The tracker was then able to accurately compare the contour components of its samples to the edges of the segmented image. In the presence of a cluttered background, it was found that the positions of the hands were important in disambiguating the poses in the presence of contours matched inaccurately to spurious edges. A visual example of tracking with a cluttered background results can be seen in Fig. 14. Finally, Fig. 15 shows a visual example of the tracking of a novel subject.

5.5. Exploiting multi-view information

Observations are ambiguous when more than one projections of a 3D skeleton can be matched. Fig. 16 shows an example of this phenomenon. We consider that a state vector's observation information is ambiguous if there are many other state vectors which have *similar* observations but *dissimilar* underlying 3D skeleton models. The following method measures the degree of ambiguities in the observations of a state vector.

Let us define an *observation subspace* of a state vector ($\mathbf{v}_C$, $\mathbf{v}_S$, $\mathbf{v}_T$) to be $\mathbf{v}_S$ and $\mathbf{v}_C$. There are N_C number of vertices for a shape contour ($\mathbf{v}_S$) while the number of tracked body parts positions is two. Thus, the observation subspace spans from dimension 0 through dimension $2N_C + 4$ in the state vector. We also define a *skeleton subspace* to be the dimensions of a state vector for all the 3D skeleton vertices. There are N_T number of 3D vertices for a 3D skeleton. In order to measure how close the 2D observations in two state vectors are, we firstly assume that all the observation subspace state vectors are initially aligned spatially in terms of translation, scale and rotation on the image plane [3]. The *observation-distance* (d_{ob}) between two state vectors $\mathbf{x}$ and $\mathbf{y}$ is then defined using the Euclidean distance measure

$$d_{ob}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{2N_C+4} \sqrt{dx_{ob}^2 + dy_{ob}^2} \tag{10}$$

where $dx_{ob} = x_{2i} - y_{2i}$ and $dy_{ob} = x_{2i+1} - y_{2i+1}$. The smaller the observation-distance, the more similar the observation information of the state vectors. That is, from the information extracted from the camera for both state vectors, the features appear the same.

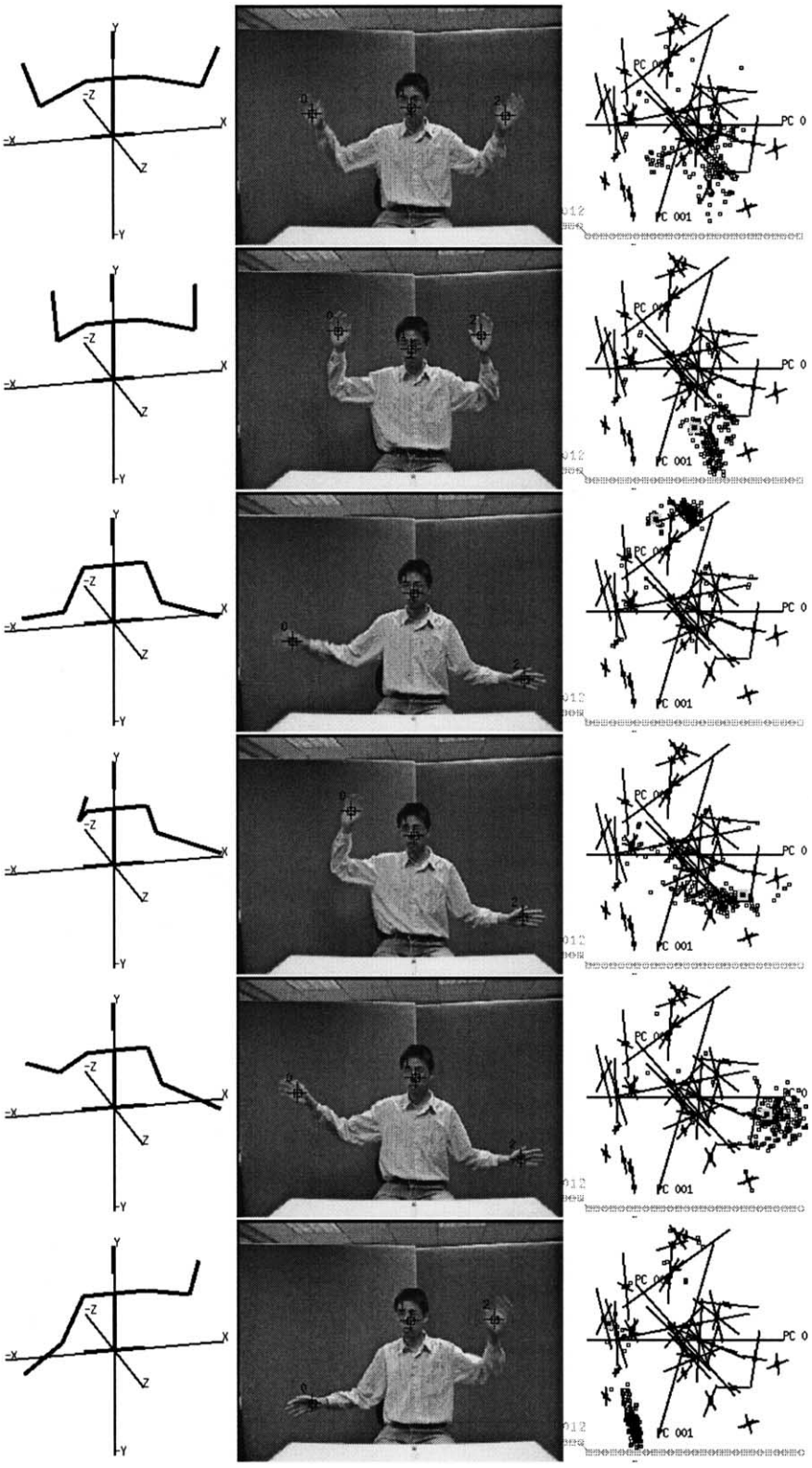


Fig. 13. Single view tracking using the CONDENSATION algorithm. This shows the tracking of a novel gesture in a controlled environment. The 10th, 17th, 30th, 40th, 50th and 76th frame is shown from top to bottom, respectively. Again, the left window shows the tracked 3D skeleton, the middle shows the input image and the right window shows the global eigenspace together with the localised principal components and the samples tracked using CONDENSATION.

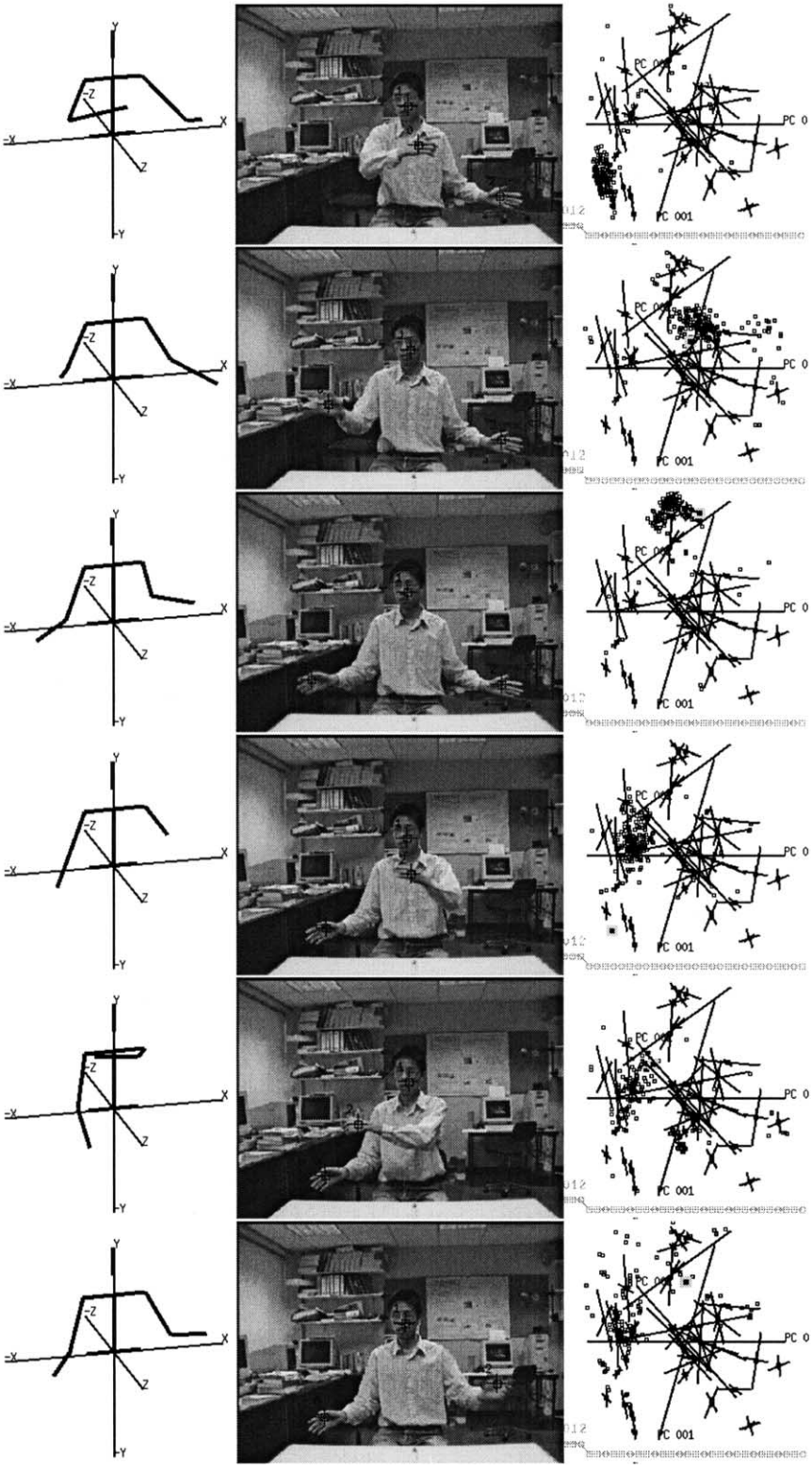


Fig. 14. This figure illustrates the tracker working in the presence of a cluttered background. The 3rd, 10th, 14th, 25th, 30th and 42nd frame of the continuous sequence is shown from top to bottom, respectively. Again for each frame, the 3D skeleton is on the left, the input image on the middle and the tracked samples in the global eigenspace with the localised principal components on the right.

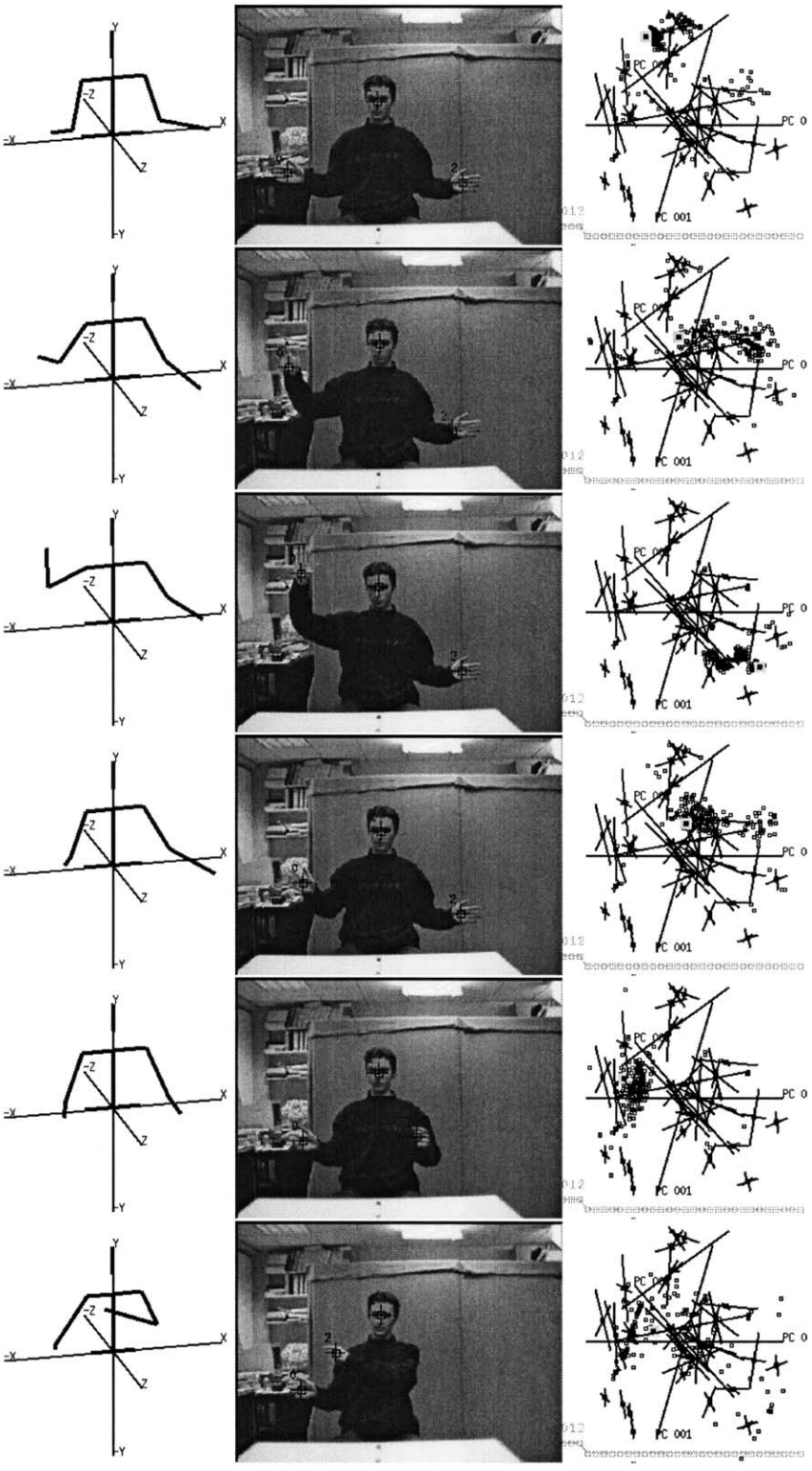


Fig. 15. This figure illustrates the tracker working on a novel subject which is not present in any of the training sequences. The 3rd, 16th, 29th, 50th, 61st and 71st frames of the continuous sequence are shown from top to bottom, respectively. Similar to the previous figures showing the tracking results, the tracked 3D skeleton is shown on the left, the input image in the middle and the tracked samples in the global eigenspace along with the localised principal components on the right.

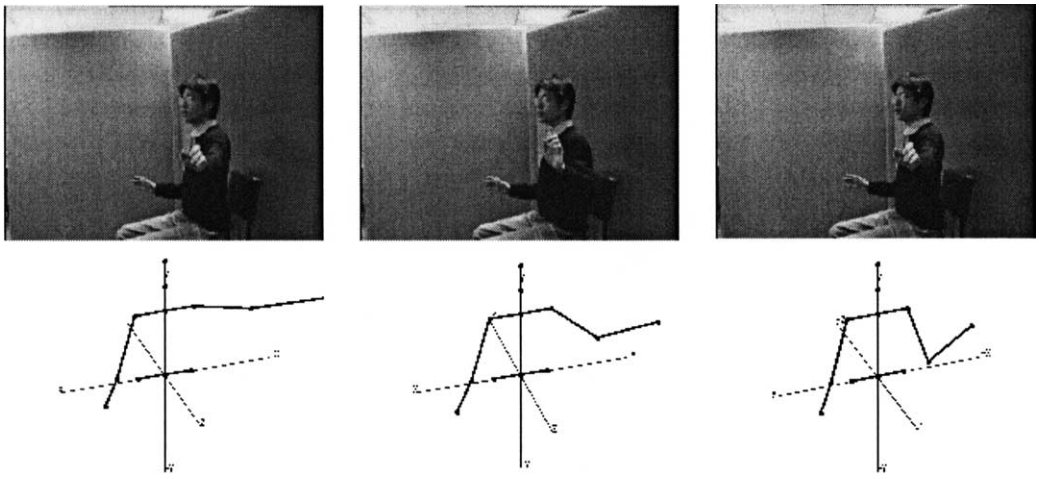


Fig. 16. The ambiguous nature of 2D observations in certain views is shown. Despite the fact that the 3D skeleton is changing, there is little difference in the 2D shape contour and the positions of the hands in the image.

Similarly, to measure the similarity of 3D skeleton components in two state vectors, we define a *skeleton-distance* (d_s) between two state vectors $\mathbf{x}$ and $\mathbf{y}$ as

$$d_s(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{3N_T} \sqrt{dx_s^2 + dy_s^2 + dz_s^2} \tag{11}$$

where $dx_s = x_{3i} - y_{3i}$, $dy_s = x_{3i+1} - y_{3i+1}$ and $dz_s = x_{3i+2} - y_{3i+2}$. Based on a set of K example state vectors of the 2D–3D hybrid representation, $\{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_K\}$, we now define the ambiguity of an example state vector as

$$a = \sum_{j=1}^{K} G(s_{2D}, \sigma) f(s_{3D}, \beta, t) \tag{12}$$

where $s_{2D} = d_{ob}(\mathbf{w}, \mathbf{y}_j)$ and $s_{3D} = d_s(\mathbf{w}, \mathbf{y}_j)$. Eq. (12) formulates our notion of *ambiguity between a 3D skeleton and its 2D observations in a single view*. If two example state vectors have very similar observations, we would like to consider the difference in their 3D skeletons. However, if they are *not* similar, we would not like to compare them, as they are two different observations. This is the role of the one-dimensional Gaussian kernel $G(x, \sigma)$. It returns the similarity between observations of two example state vectors. The smaller the observation-distance is, greater is the similarity. The rate at which this value reduces as the observation-distance increases is determined by the standard deviation (σ) of the Gaussian kernel.

If they both have similar observations and 3D skeletons, there is little or no difference in them, the ambiguity ratio should not be increased, To achieve this computationally, given that the difference between two 3D skeletons is quantified by Eq. (11), a translated Sigmoid function is then used. Formally, it can be defined as:

$$f(x, \beta, t) = \frac{1}{1 + \exp(-\beta(x - t))}. \tag{13}$$

With its monotically increasing property, its role is to weigh down contributions of example state vectors whose ambiguity is currently measured. Additionally, the f in Eq. (13) has the advantage of lying in the range between 0 and 1. Such a property allows it to provide a normalised weighting measure regardless of the scale of the 3D skeletons. Finally, precisely how dissimilar two skeletons must be before they are considered to cause ambiguities is determined by t and β, the Sigmoid's mid-point and scaling parameter, respectively.

5.6. Tracking across multiple views

In order to make use of the ambiguity measurement, a set of K number of example state vectors, $\{\mathbf{y}_l, ..., \mathbf{y}_K\}$ of the 2D–3D hybrid representation is provided. The 3D components of all the example states are made to be viewpoint invariant. That is, the transformation caused by a change in the viewpoint does not affect an example's associated 3D skeleton. For example, if two basis set differ from each other because they were obtained at different viewpoints, their 3D components (associated skeleton) will be the same. Therefore, these examples only capture the changes of the 2D measurements due to different viewpoints for the *same* body pose allowing for the recovery of consistently aligned skeletons despite differences in viewpoints.

The ambiguity, (a_i) for each example, ($\mathbf{y}_i$) is computed using Eq. (12) resulting in the set $\{a_1, ..., a_K\}$. Given a novel state vector ($\mathbf{n}$) its ambiguity can be calculated by assigning it the ambiguity value of its nearest neighbour

$$a_{\mathbf{n}} = a_j | d_e(\mathbf{n}, \mathbf{y}_i) < d_e(\mathbf{n}, \mathbf{y}_j)$$
$$\forall i, \quad i = 1, ..., j-1, j+1, ..., K \tag{14}$$

where $d_e(\mathbf{n}, \mathbf{y})$ gives the Euclidean distance between ($\mathbf{n}$) and ($\mathbf{y}$). This is applied to tracking using multiple views. For each individual view, the 3D skeleton is tracked using the CONDENSATION algorithm as described in Section 4. We

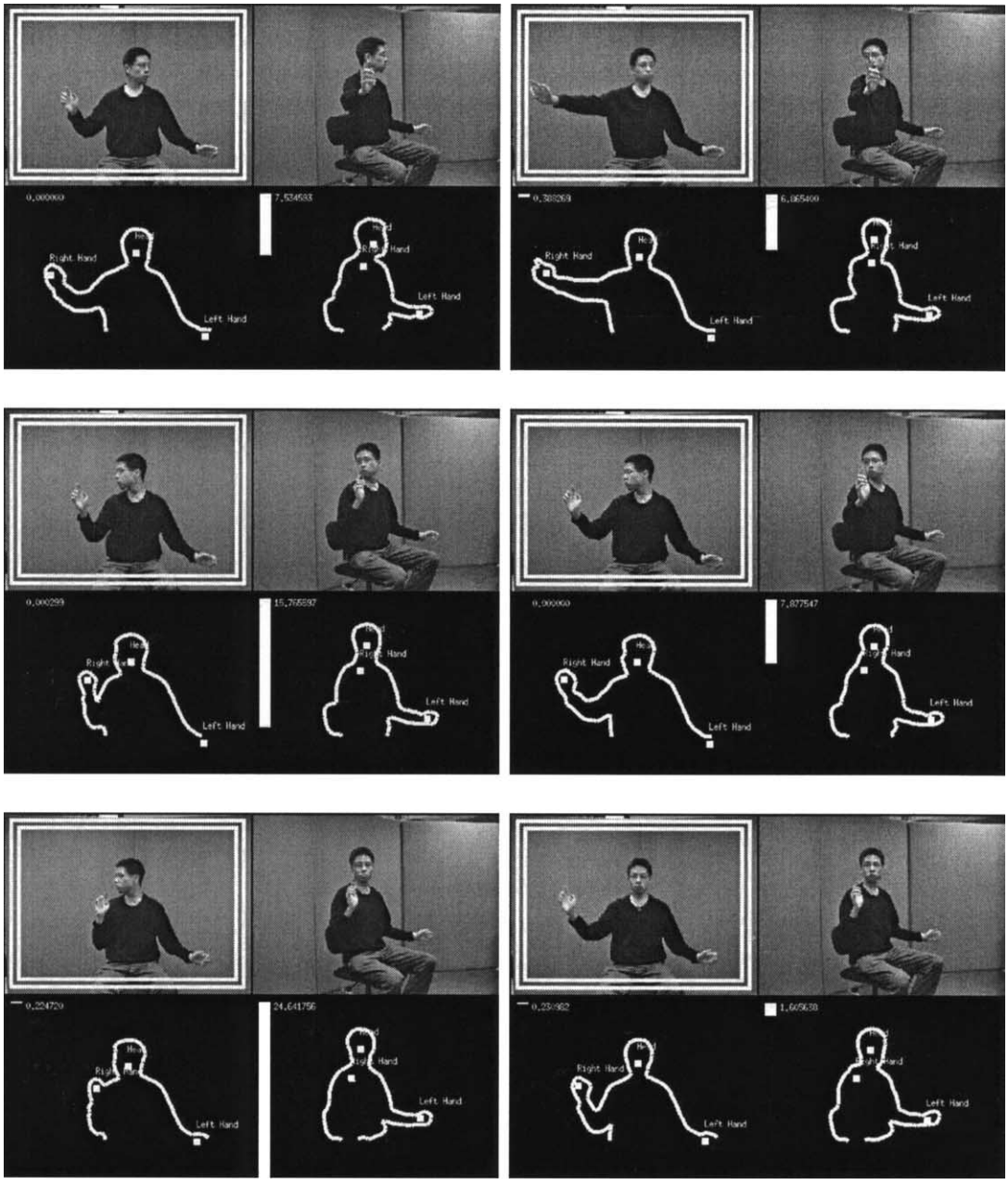


Fig. 17. Example results of the view selection using the ambiguity measurements. The sequence is shown from left to right, top to bottom. Each frame shows two views, one near frontal and one position diagonally to the user. Below the camera views are the 2D measurements extracted from it (i.e. the contour and body parts positions). On the left of the contours for each view is a white bar going from the top to the bottom. The length of the white bar indicates the ambiguity value of the extracted 2D measurements for each respective view. The longer the bar is, more ambiguous the view's extracted information is. The view selected is therefore the view with the shortest ambiguity bar. The selected camera view is indicated by two surrounding white rectangles.

select M samples with the highest fitness values given by Eq. (9). The model state vector ($\mathbf{v}$) is reconstructed from each sample using Eq. (7). Its ambiguity is calculated using Eq. (14). We select the sample which has the highest product of its fitness value given by CONDENSATION together with its ambiguity value. Different views are given the ambiguity measurement using the nearest neighbour. We then select the vector which is least ambiguous and fits the data the best. This can be measured by taking the product of a vector's probability for measurement fitness to the inverse of its ambiguity measurement. Finally, multiple instances of single view trackers are integrated together using the ambiguity measurements.

5.7. View selection results

A total of two views were used. Training sequences of a person performing random gestures were captured simultaneously by two cameras at different viewpoints. For each individual view in each frame, the 2D measurements (i.e.

contour and body parts positions) were extracted. Using Eq. (12), each of the measurement was labelled with its ambiguity value.

Using the same two views used for capturing the training sequence, a test sequence was obtained. For each frame in the training sequence, Eq. (14) was used to determine the ambiguity of the 2D measurements in each view. Example results are shown in Fig. 17.

6. Conclusions

In this article, we have addressed three main issues in the use of the linear combination method: determining the examples, learning the spatio-temporal constraints on the linear combination coefficients and finally a method for estimating the necessary coefficients to reconstruct a valid object of interest over time. Each object's configuration is represented as a high dimensional vector of variables. From this perspective, recovering the examples required for reconstructing the object's different configurations implies modelling its subspace within this high dimensional space. PCA was used to extract the axes of this subspace, where each subspace axis serves as a basis example. Thus, the eigenspace also serves as a 'coefficient space' whose points are coefficient sets for possible linear combinations. However, when an object undergoes non-linear dynamics, only a non-linear portion of the coefficient space can provide coefficients for plausible object reconstructions. This brings about the need to impose spatial constraints in the coefficient space. The problem addressed using a collection of piecewise linear clusters. Additionally, we model the temporal constraints to account for the non-linear dynamics of the linear combinations coefficients using a Markov model. In order to make use of the linear combinations model, we need a way to estimate its coefficients over time. To this end, we adopted the CONDENSATION model for its properties of multiple hypotheses and ability to use the spatio-temporal constraints for robust estimations. Finally we gave details and experimental results on the application of our dynamical framework for linear combinations to track moving 3D skeletons of human subjects.

Acknowledgements

We would like to thank D. Parkinson, J. Ng, J. Sherrah, R. Howarth, T. Chang and Y. Li for their helpful comments.

Appendix A. Kinematics of a simple hierarchical object

The dynamics of the simple articulated object described in Section 3.3 can be defined with the following kinematic equations:

$$f_1(\theta, \phi) = q_x \tag{A1}$$

$$f_2(\theta, \phi) = q_y \tag{A2}$$

$$f_3(\theta, \phi) = r_x \cos(\theta) - r_y \sin(\theta) + q_x \tag{A3}$$

$$f_4(\theta, \phi) = r_x \sin(\theta) + r_y \cos(\theta) + q_y \tag{A4}$$

$$f_5(\theta, \phi) = s_x(\cos(\phi)\cos(\theta) - \sin(\phi)\sin(\theta)) - s_y(\sin(\phi)\cos(\theta) + \cos(\phi)\sin(\theta)) + r_x \cos(\theta) - r_y \sin(\theta) + q_x \tag{A5}$$

$$f_6(\theta, \phi) = s_x(\cos(\phi)\sin(\theta) + \sin(\phi)\cos(\theta)) + s_y(\cos(\phi)\cos(\theta) - \sin(\phi)\sin(\theta)) + r_x \sin(\theta) + r_y \cos(\theta) + q_y. \tag{A6}$$

The articulated object has three 2D vertices, p_1, p_2 and p_3. The kinematics functions $(f_1(\theta, \phi), \ldots, f_6(\theta, \phi))$ produces the values for the components $(p_{1,x}, p_{1,y}, p_{2,x}, p_{2,y}, p_{3,x}, p_{3,y})$, respectively, r and s are the original local coordinates for p_2 and p_3. The position of the object in the world is given by q. The kinematics parameters θ and ϕ are the angles of rotation on p_2 relative to p_1 and rotation angle on p_3 relative to p_2, respectively. An illustration of the articulated object and its kinematics parameters can be seen in Fig. 4.

References

[1] T. Chang, S. Gong, Bayesian modality fusion for tracking multiple people with a multi-camera system, Proceedings of European Workshop on Advanced Video-based Surveillance Systems, September 2001.
[2] S. Ullman, R. Basri, Recognition by linear combinations of models, Pattern Analysis and Machine Intelligence 13 (10) (1991) 992–1005.
[3] T. Cootes, C. Taylor, A mixture model for representing shape variation, BMVC (1997) 110–119 Essex, UK.
[5] A. Baumberg, D. Hogg, Learning flexible models from image sequences, European Conference on Computer Vision, ECCV-94, Stockholm, Sweden, LNCS-Series, vol. 800, Springer, Berlin, 1994, pp. 299–308.
[6] T.F. Cootes, C.J. Taylor, D.H. Cooper, J. Graham, Active shape models—their training and applications, Computer Vision and Image Understanding 61 (1) (1995) 38–59.
[7] T. Heap, Learning deformable shape models for object tracking, PhD thesis, School of Computer Studies, University of Leeds, UK, September 1997.
[8] T. Heap, D. Hogg, Improving specificity in pdms using a hierarchical approach, BMVC (1997) 80–89 Essex, UK.
[9] R. Bowden, T. Mitchell, M. Sarhadi, Reconstructing 3D pose and motion from a single camera view, BMVC (1990) 904–913 Southhampton.
[10] L. Sirovich, M. Kirby, Application of the Karhunen–Loève procedure of the characterization of human faces, IEEE Pattern Analysis and Machine Intelligence 12 (1) (1990) 103–108.
[11] B. Moghaddam, A. Pentland, An automatic system for model-based coding of faces, Proceedings of the IEEE Data Compression Conference, Snowbird, Utah, 1995.
[12] I. Koufakis, B.F. Buxton, Very low bit rate face video compression using linear combination of 2D face views and principal component analysis, Image Vision Computing 17 (14) (1999) 1031–1051.

[13] M. Turk, A. Pentland, Eigenfaces for recognition, Journal of Cognitive Neuroscience 3 (1) (1991) 71–86.
[14] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1996.
[15] T. Vetter, T. Poggio, Linear object classes and image synthesis from a single example image, Pattern Analysis and Machine Intelligence 19 (7) (1997) 733–741.
[16] M.J. Black, A.D. Jepson, Eigen tracking: robust matching and tracking of articulated objects using a view based representation, International Journal of Computer Vision 26 (1) (1998) 63–84.
[17] M. Isard, A. Blake, Condensation—density propagation for visual tracking, International Journal of Computer Vision 29 (1) (1998) 5–28.
[18] J. Regh, Visual analysis of high DOF articulated objects with application to hand tracking, PhD thesis, Carnegie Mellon University, April 1995.
[19] D. Gavrila, L. Davis, 3D model-based tracking of humans in action, in: K. Bowyer, N. Ahuja (Eds.), Advances in Image Understanding, IEEE Computer Society Press, Silver Spring, MD, 1995.
[20] A. Pentland, Automatic extraction of deformable models, International Journal of Computer Vision 4 (1990) 107–126.
[21] D. Hogg, Model based vision: a program to see a walking person, Image Vision Computing 1 (1) (1983) 5–20.
[22] S. McKenna, Y. Raja, S. Gong, Tracking colour objects using adaptive mixture models, Image Vision Computing 17 (1999) 225–231.
[23] M. Sonka, V. Havac, R. Boyle, Image Processing, Analysis and Machine Vision, Thomson Computer Press, 1995.