Human Motion "Following" system using Hidden Markov Models

and application to dance performance

Rémy Muller
Remy.Muller@ircam.fr

March–June 2004

Abstract

This work presents a movement “following” system based on “Hidden Markov Models” and motion descriptors extracted from video. The primary application is in performing arts such as dance but the methodology is general enough to be applied in other fields as long as appropriate descriptors are available.

Different motion features, segmentations, decoding methods and data analysis such as Principal Components analysis have been performed and compared in order to show how they affect the “following” performances. Both contemporary dance performances and synthetic animations have been used in order to evaluate the system.

Contents

Chapter 1

Introduction

Performing arts often require a good synchronization between performers, music, video and lighting. The simplest solution is to ask performers to follow the music and technicians to follow the performers. But when a finelyt timed temporal interaction is required and lots of different events have to be generated in response to what's happening on the stage, the task becomes quickly un manageable, even for teams of skilled professionals. Tradeoffs have to be made in order to make the performance realizable.

We aim at using computer vision an machine–learning techniques to delegate to the computer the task of doing human motion following.

Though machine–learning techniques have been used for a long time in applications such as Robotics, Speech recognition or DNA classification, their use for motion and gesture recognition has been made possible only recently because of the memory capacity requirements and CPU power required by the algorithms. Most of the R&D has been focused on hand-sign recognition [27] and gait characterization [21] for applications such as communication, medical rehabilitation and surveillance.

An overview of Human motion analysis and gesture recognition can be found in [12], [9] and [22] where the common problems such as detection, classification, recognition are exposed as well as the tools commonly used to solve them. One can also note that finite-state machines [16], Markov chains [3], Hidden Markov Model[2], [20] or their derivatives [6], [10] are heavily used to model the temporal dimension of motion or, more generally, time series.

The use of these techniques for artistic applications is still emergent. One important tool called Eyesweb [1] has been made available with the European project

MEGA[1] with a strong involvement toward multimodal analysis of expressive gestures [26]. It has been very helpful to us for extracting the motion descriptors, since all that we needed was already included in it we were thus able to concentrate on the interpretation.

Meanwhile, Ircam has been involved in “following systems” with “score–following” in the works of Barry Vercoe and Miller Puckette [24], [25], [17]. It was refactored later on by Nicolas Orio who introduced the use of HMM to perform the “following” [13], [14], and a master thesis [15] has been done last year to evaluate the possibility to extend it to perform text following in the case of theater applications and to help following singing voice.

The work in this paper is part of a new research project being developed at Ircam dedicated to performing Arts whose scope and goals are described in [8] and we have focused it particulary to dance performance. It has been realized inside the real–time application team.

The works described in [11] and [7] are technically similar to our work with respect to the methods used but differ from ours since we are more concerned by the following–synchronization problem than the classification of elementary gestures for dance notation and because we wanted our system to be able to work in real–time.

Willing to apply this research to artistic context will always impose us the following constraints:

- The number of available examples used to train our model will always be limited and thus the validity of the statistics hypothesis too.
- We won’t be able to use non causal or CPU[2] intensive algorithms such as *Viterbi* or the *forward–backward* procedure. Please refer to [18] for explanations about these methods.

We will first expose the system we have used to perform the “following” with the definition of the motion descriptors, the definition of the Hidden Markov Model, the segmentation methods and the decodingstrategies. Then we will describe different results related to each of these aspects and we will finally present the future directions that might be explored buikding on this work.

[1]Multisensory Expressive Gesture Applications.
[2]Computer Processor Unit

Chapter 2

Methodology

2.1 Video Material

We have used short[1] dances videos with a 300x200 resolution. These were gathered from choreographer Hervé Robbe and were recorded at the Centre National de Dance du Havre with DV cameras. Each dance has been performed twice by two different dancers, which gives us 4 different examples per dance.

To evaluate our work, we've also generated dance animations based on motion capture from the Vicon optical motion capture system. The animations were generated in order to have different time–lines.

Figure 2.1: snapshots from one of the video dances

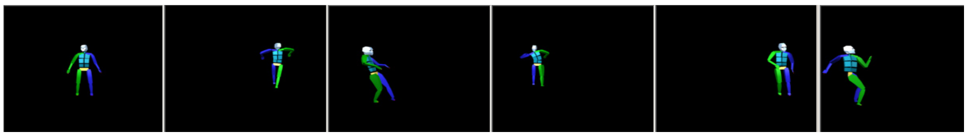
Figure 2.2: snapshots from one of the animation

[1]less than one minute long.

2.2 Hypothesis

During all this work, we have supposed that:

- We have a 3D motion projected on a 2D surface
- the camera is fixed and the focal plan does not change
- the lighting is constant
- there is a single single dancer in the scene
- the only motion is the human one over a constant background
- the size of the character is comparable with the one of the image.

2.3 Motion features

A 300x200 RGB image would represent 180000 parameters/dimensions per frame. Using a dimension reduction technique such as PCA directly on the sequence of images would have been very interesting, but we couldn't try it because of memory limitations. It was thus mandatory upon us to choose a restricted number of parameters to represent the motion occurring in the image and leave the information concerning the rest of the image.

I'll present here the set of motion descriptors we have used during this work that were all provided by Eyesweb [1] image processing and analysis software.

The blob analysis uses the silhouette which is a binary image divided into background (black 0) and silhouette (white 1). This is the most simple way of computing it.

$$S_k = threshold(Image_k - Background) \tag{2.1}$$

It's also possible to use a better estimate as defined in [26]

$$S_k = threshold(Image_k - Background, Background) \tag{2.2}$$

Where the threshold function uses the histogram of the background.

In the case of a changing background, it would still be possible to update the background with an adaptive low-pass filter or have a more sophisticated and robust segmentation. but this is out of the scope of this work.

2.3.1 Skeleton

This is a particular aspect of blob analysis. Eyesweb has a module trying to match a face human skeleton to the image silhouette by dividing the blob in multiple areas and computing the centroid of each area. It gives the following data:

- Bounding rectangle x,y,width,height
- Head x,y
- Center of gravity x,y
- Left–Right Hand x,y
- Left–Right elbow x,y
- Left–Right shoulder x,y
- Left–Right knee x,y
- Left–Right foot x,y

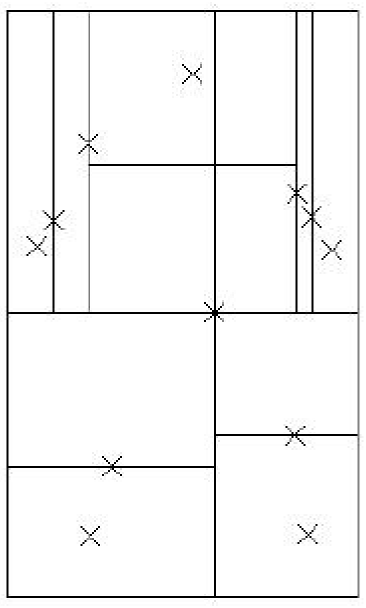
Figure 2.3: blob centroids used to compute the skeleton

Most often, the dancer doesn't face the camera, therefore the computed points hardly match the human body parts but they still are interesting to be considered as a multi–resolution description of the blob, reminiscent of quad–tree decomposition and with a small (22) set of points.

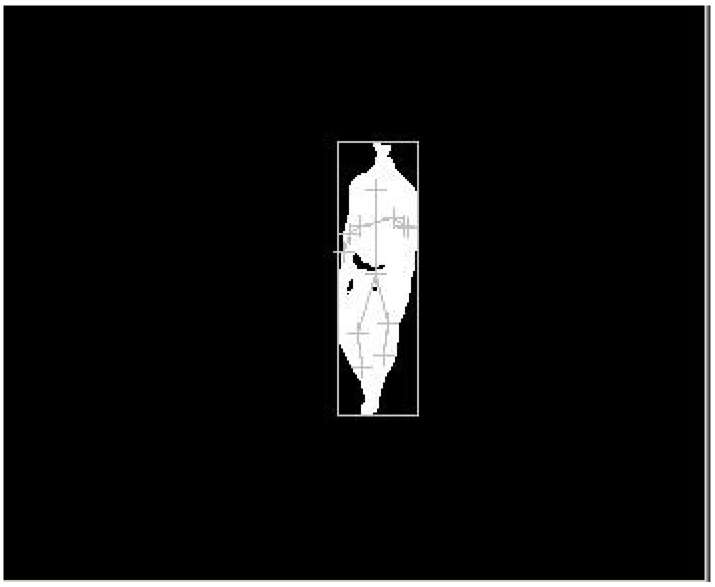

Figure 2.4: skeleton on a real silhouette

2.3.2 Image moments

Another way of characterizing the silhouette, is to use moments.

Cartesian spatial moments

The first that can be considered are the cartesian moments:

$$m_{pq} = \sum_{x=1}^{M} \sum_{y=1}^{N} x^p y^q I(x, y) \tag{2.3}$$

Eyesweb only provides moments $m_{00}, m_{01}, \ldots, m_{30}, m_{03}$. One can note that m_{00} gives the surface of the blob, m_{01} and m_{10} gives its centroid. And m_{22}, m_{11} and m_{02} gives its variance and its orientation.

Normalized moments

they are scale invariant.

$$\nu_{pq} = \frac{m_{pq}}{m_{00}^{\gamma}} \qquad \gamma = \frac{p+q}{2} + 1 \qquad \forall p, q \geq 2 \tag{2.4}$$

Eyesweb only provides $\nu_{00}, \nu_{01}, \ldots, \nu_{30}, \nu_{03}$

Central cartesian moments

$$\mu_{pq} = \sum_{x=1}^{M} \sum_{y=1}^{N} (x - \overline{x})^p (y - \overline{y})^q I(x, y) \tag{2.5}$$

This is a variant which is translation invariant. Eyesweb only provides moments $\mu_{00}, \mu_{01}, \ldots, \mu_{30}, \mu_{03}$

Normalized central moments

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \qquad \gamma = \frac{p+q}{2} + 1 \qquad \forall p, q \geq 2 \tag{2.6}$$

These moments are scale and translation invariant.

Hu moments

$$\begin{aligned}
I_1 &= \eta_{20} + \eta_{02} \\
I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 + 3(\eta_{21} + \eta_{03})^2\right] \\
&\quad +(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \\
I_6 &= (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right. \\
&\quad \left. +4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\right] \\
I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 + 3(\eta_{21} + \eta_{03})^2\right] \\
&\quad +(\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]
\end{aligned}$$

These moments have the good property (if required by the application) to be scale, translation and rotation invariant and to be orthogonal.

So far, we have described *instantaneous* motion descriptors but there are many systems using accumulation of images over a time window to recognize gestures [4] [5]. However we have decided no to show them because they introduce a delay in the "following" system. It's important to keep in mind that 1 frame at a rate of 25 images per second represents 40 milliseconds which is already audibly noticeable, especially if rhythmic musical events have to be synchronized with the dance.

2.4 Principal Components Analysis

Most of the features we use are correlated and can be affected by different kinds of perturbations due to variations in the lighting conditions, DV compression and bad silhouette segmentation. We want to evaluate the advantage of using a Principal Components Analysis decomposition both as a dimension reduction technique and as a perturbation reduction tool (supposing that the perturbation's variance is low compared to the variance of the signal). It can also be a powerfull method of visualising the distribution of the data set in the two or three most significant components of the features–space. Please refer to [23] for more information on Principal Components Analysis.

If we have a sequence of parameters $\{\mathbf{P}_t\}$ that we can represent by a matrix P where $\mathbf{P}_t$ is a vector of parameters.

$$P = [\mathbf{P}_1 \mathbf{P}_2 \ldots \mathbf{P}_T] \tag{2.7}$$

P is of dimension $N \times T$ and N is the number of parameters.
To perform PCA, we deduce the mean of the data set for each parameter in order to have centered data.

$$P' = P - \bar{P} \qquad \bar{P} = \begin{pmatrix} \bar{p_1} \\ \vdots \\ \bar{p_N} \end{pmatrix}$$

then we search a reduced set of R orthogonal vectors e_i which will best describe the data set in a least-squares sense, i.e. the Euclidian projection of the error is minimized. The common method of computing the principal components is to find the eigenvectors of the covariance matrix C of size $N \times N$ such that:

$$C = P' P'^T \tag{2.8}$$

and

$$C e_i = \lambda_i e_i \tag{2.9}$$

It can be done using a singular value decomposition. Then the last step consists to keep the R eigenvectors that have the highest eigenvalues λ_i. We can then project our original sequence in the eigensubspace.

$$P'' = U^T P, \qquad U = [e_i \ldots e_R] \tag{2.10}$$

We then have the sequence $\{\mathbf{P}''_t\}$ that we can use as the observations of our "following" system instead of directly using the parameters given by the video analysis.

We have also considered using Kernel PCA [28] because it can handle inherent nonlinear variation of the data set and transform it to a linear variation thus making the separation of the classes easier with linear tools, but time was short to do it.

2.5 Hidden Markov Model

Hidden Markov Models are probabilistic finite-state automatas, where transitions between states are ruled by probability functions. At each transitions, the new state emits a value with a given probability. Emmissions can be both symbols from a finite alphabet and continuous multidimensional values. In markovian process, transition probabilities are assumed to depend only on a finite number of previous transitions (usually one) and they may be modeled as a Markov chain. The presence of transition with probability equal to zero defines the topology of the model, limiting the number of possible paths. For more in formation on HMM see [18].

2.5.1 Structure

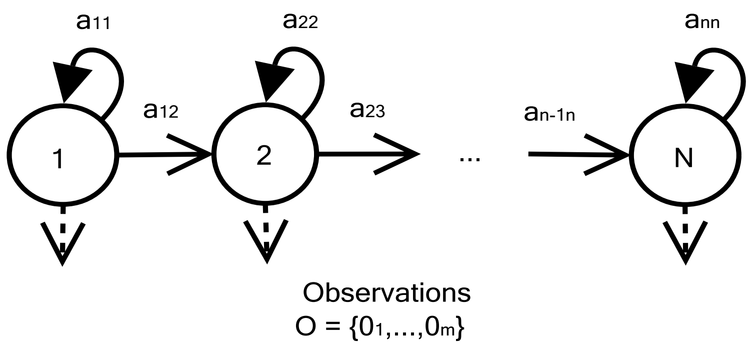

Figure 2.5: left-right HMM

We've used a very simple left-right structure associating one state with a segment of the video.

The graph can be represented by the following transition matrix:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & & & \\ & a_{22} & a_{23} & & \\ & & \ddots & \ddots & \\ & & & a_{N-1N-1} & a_{N-1N} \\ & & & & a_{NN} \end{pmatrix}$$

With a_{ij} being the probability of going from state i to state j and N the number of states.

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i) \qquad 1 \leq i, j \leq N \tag{2.11}$$

It implicitly models the probability of staying a time t in a given state as an exponential distribution.

$$p_i(d) = (a_{ii})^{d-1}(1 - a_{ii}) \tag{2.12}$$

with d being the duration in state i. But we could extend it as in [18],[13], [14] and [15] to multiple states giving more degrees of freedom to model the duration or/and refine the segmentation in subsegments.

2.5.2 Observations

In our model, each state of the model emits a vector of continuous observations per video frame.

$$\mathbf{O} = \{o_1, o_2 \ldots, o_M\} \tag{2.13}$$

with M being the number of observations per state. The state is characterized by its probability distribution.

$$b_j(\mathbf{O}) = \varphi[\mathbf{O}, \mu_j, \mathbf{U}_j] \qquad 1 \leq j \leq N \tag{2.14}$$

with mean vector μ_j and the covariance matrix $\mathbf{U}_j$ in state j and φ being log–concave to ensure the convergence of the re-estimation procedures.

We then have a model $\lambda = [\mathbf{A}, \mathbf{B}, \pi]$ of our dance with $\mathbf{B} = \{b_j\}$ and π being the initial state probability distribution.

These observations can be any set taken among the motion features exposed before and we will compare their respective advantages and drawbacks in the next chapter.

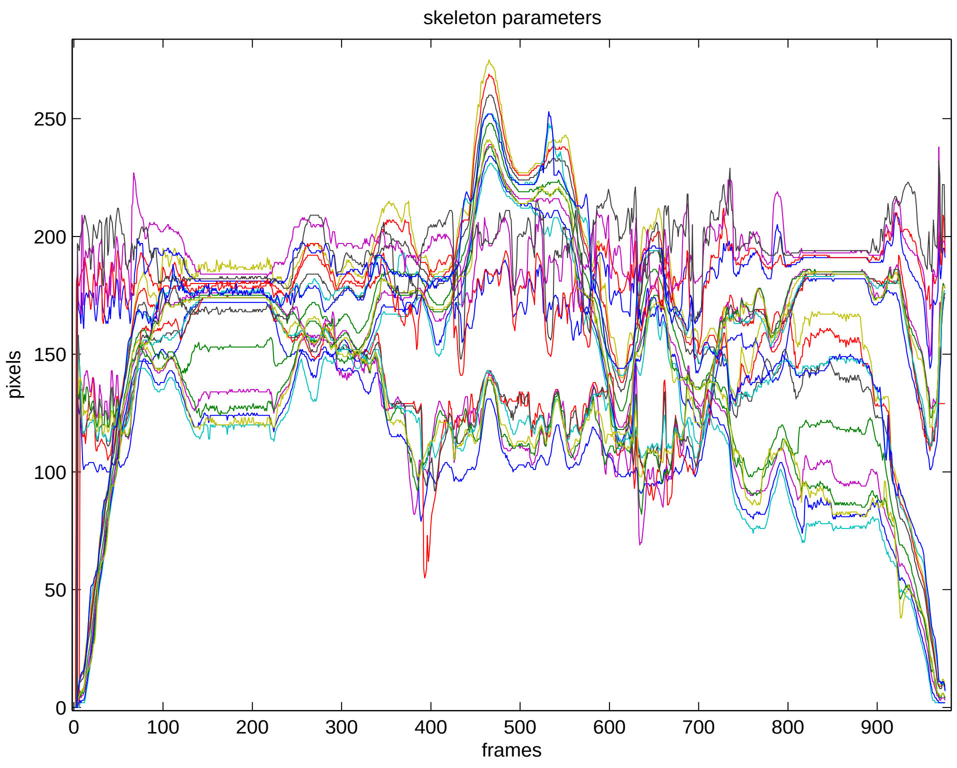

Figure 2.6: skeleton parameters for a dance

2.5.3 Temporal segmentation

Once we have one or several executions of a dance, in order to build a HMM, we have to divide it into several segments that can be associated to states of the HMM. Since we've used a simple left–right model, there is a one-to-one mapping between a segment and a state.

Manual

Most of the temporal segmentation of the dances in this work has been done manually either by looking at some key gestures–postures in the dances, or by looking at the parameters extracted from the video in order to create states well localized in the feature-space.

In the case of the synthetic dances, as we had the information relative to the footsteps for all the variations, we have used them as the markers for the segmentation. Therefore we were able to compare the same segments across the different animations with no error introduced by a bad segmentation.

Automatic

When drawn in the feature-space, the point distributions generated by the dances seemed hard to cluster using an algorithm such as *K-means*. They would be better viewed as trajectories with local change of direction.

Therefore we have used a criterion δ based on the directness of the trajectory as defined by Volpe [26]. i.e. the ratio between the direct path between the 2 extreme points and the trajectory path.

$$\delta_t = \frac{\|\mathbf{P}_{t+N} - \mathbf{P}_{t-N}\|_2}{\sum_{i=t-N}^{t+N-1} \|\mathbf{P}_{i+1} - \mathbf{P}_i\|_2} \tag{2.15}$$

and used the local minima of the directness as markers.

2.5.4 Initialization

Once we have a segmentation for all the dances, we are able to compute the mean vector and the covariance matrix for each state and then model the probability density function per state.

We have used gaussian distributions, but it might be improved since the experimental distribution we have are far from being gaussian, but rather look like trajectories segments in the feature space.

We are also able to compute the mean duration for each state and then create the transition matrix using the following equation.

$$\bar{d_i} = \sum_{d=1}^{\infty} d p_i(d) = \sum_{d=1}^{\infty} d(a_{ii})^{d-1}(1 - a_{ii}) = \frac{1}{1 - a_{ii}} \tag{2.16}$$

2.5.5 "Following"

As for the score–following, we wanted our system to be implemented in real-time thus we have only used the *forward procedure* which has the advantage of being causal and can be implemented iteratively.

$$\alpha_t(i) = P(\mathbf{O}_1\mathbf{O}_2 \ldots \mathbf{O}_t, q_t = S_i|\lambda) \tag{2.17}$$

i.e. the probability of the partial observation sequence $\mathbf{O}_1\mathbf{O}_2 \ldots \mathbf{O}_t$ *and* state S_i at time t given the model λ

It can be computed using the following equations:

$$\alpha_1(i) = \pi_i b_i(\mathbf{O}_1) \qquad 1 \leq i \leq N \tag{2.18}$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(j) a_{ij}\right] b_j(\mathbf{O}_{t+1}), \qquad 1 \leq t \leq T-1, \quad 1 \leq j \leq N \tag{2.19}$$

If we follow this procedure we can then compute the probability of the observation sequence until time t given the model:

$$P(\mathbf{O}_1\mathbf{O}_2 \ldots \mathbf{O}_t|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \tag{2.20}$$

And if we scale the α_i such that

$$\hat{\alpha}_t(i) = \frac{\alpha_t(i)}{P(\mathbf{O}_1\mathbf{O}_2 \ldots \mathbf{O}_t|\lambda)} \tag{2.21}$$

we then have for each t the state distribution probability.

$$\hat{\alpha}_t(i) = P(q_t = S_i|\mathbf{O}_1\mathbf{O}_2 \ldots \mathbf{O}_t, \lambda) \tag{2.22}$$

Now that we are able to have the state distribution for each time instant t, we have to take a decision about the state that we consider the most likely for each t, that is called *decoding*. We have used two strategies that are described hereunder and will be compared in the next chapter.

Max decoding

In the max decoding, we simply select the most probable state $\tilde{q}_t$ so that:

$$\tilde{q}_t = \underset{i}{\arg\max}\{\hat{\alpha}_t(i)\} \tag{2.23}$$

and we can use $\max\{\hat{\alpha}_t(i)\}$ as the quality of the estimation.

Barycenter decoding

It is also possible to use the barycenter of $\{\hat{\alpha}_t(i)\}$:

$$\tilde{q}_t = \frac{1}{N}\sum_{i=1}^{N} i\hat{\alpha}_t(i) \tag{2.24}$$

which has a smoother behaviour when there is uncertainty among different states.

It would be possible to use other strategies using prior knowledge and restrictions on the possible paths but we've restricted ourselves to these two methods. In artistic applications, the decoding step could be left to the final application to better adapt to the context.

It's important to note that using the *forward procedure*, the decoded path can be non-optimal with respect to the path that would be decoded offline by the *viterbi* algorithm. It can even not be allowed by the HMMs topology.

Chapter 3

Results

In order to clarify the results that are going to be exposed in this chapter, I will define the different measures of error we have used and explain their advantages and drawbacks.

First we have used the mean rms error : mre

$$mre = \frac{1}{TN}\sqrt{\sum_{t=1}^{T}(\tilde{q}_t - q_t)^2} \tag{3.1}$$

and the mean absolute error : mae

$$mae = \frac{1}{TN}\sum_{t=1}^{T}|\tilde{q}_t - q_t| \tag{3.2}$$

They both give an average measure of the how much the recognized path differs from the original one.

We have also used the max error : me

$$me = \frac{\max|\tilde{q}_t - q_t|}{N} \tag{3.3}$$

It gives an idea about what the worse case error can be, which is of extreme importance in live performance.

Since these three errors are normalized by the number of states N they should be treated with care when comparing segmentations with different number of states.

And finally, we have used the mean binary error : mbe

$$mbe = \frac{1}{T}\sum_{t=1}^{T}\lfloor \tilde{q}_t - q_t \rfloor \tag{3.4}$$

with

$$\begin{array}{rclccc} \lfloor x \rfloor & = & 1 & if & |x| & \geq & 0.5 \\ & & 0 & otherwise & & & \end{array}$$

We could have called it the coincidence rate instead, it gives an idea of how much the recognized path coincides with the original one. It is very sensitive because of the binary threshold and is only useful when the two paths are very close to each other. Otherwise the it quickly becomes very high. For example a perfect recognition associated with a constant delay would lead to a very bad results.

3.1 Features

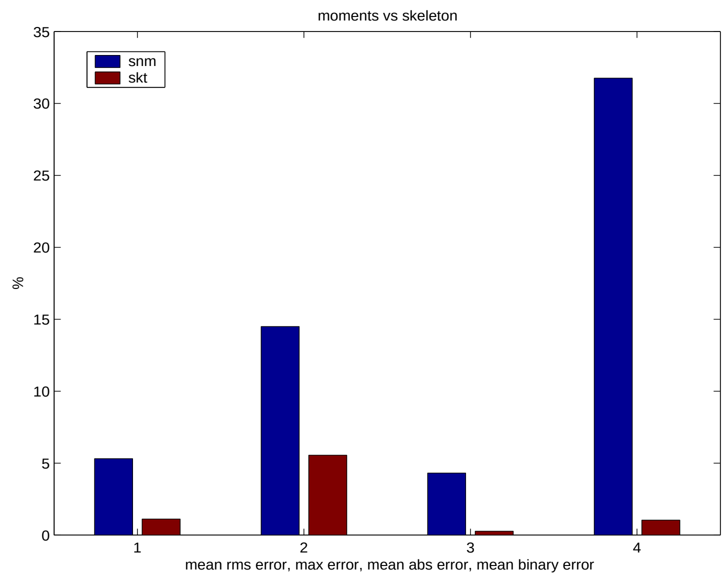

Figure 3.1: moments vs skeleton

Though we have tried many kinds of parameters during this work, we have mainly used the skeleton parameters and image moments. We will compare here the skeleton parameters with the Normalized Moments taken on the silhouette image that we will shorten to *SNM*. We could use the Hu moments or the cartesian moments but the results are quite similar and we won't expose them here because of space. We have used HMM trained on the four examples to follow each example individually and have averaged the results to give a global mark to the feature.

We can see on figure 3.1 that the skeleton gives significantly better results than the SNM for the different kind of errors. That can be explained by the fact that:

- The SNM only goes until order 3, which only conveys low spatial frequency information whereas the skeleton is a kind of multi–resolution description of the silhouette and thus catches more of its spatial frequency contents.
- We have 22 parameters for the skeleton and only 9 for the SNM. If we consider that each parameter brings the same amount of information – which is not the case since they are correlated – the skeleton has more chance to give us a discriminant description.

In the rest of the results exposed in this paragraph, we have used either the SNM, the skeleton or both depending on what we wanted to highlight.

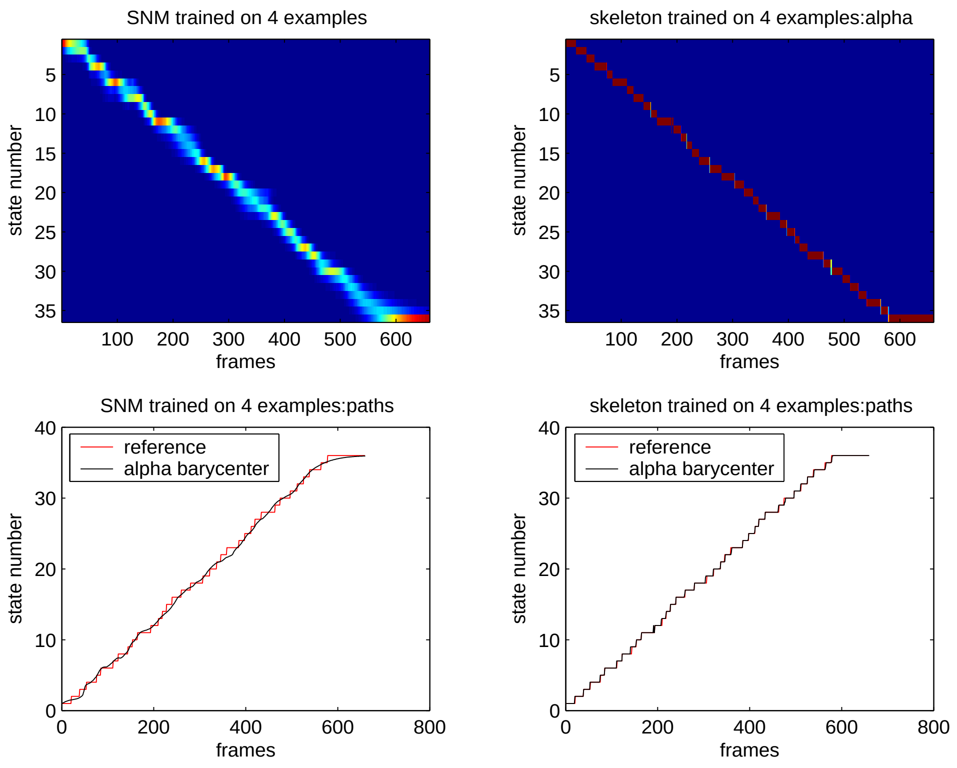

Figure 3.2: learning

3.2 Learning

In this chapter, we have tried to evaluate the effect of learning on the "following" performances independently of similarities and dissimilarities between the different examples, we have computed all the possible couples (L, R) where L is the set of examples used for the learning step and R is the set of examples being recognized. For example when learning on 3 examples we have the following possibilities: $(\{1,2,3\},\{1,2,3,4\})$, $(\{1,2,4\},\{1,2,3,4\})$ and $(\{2,3,4\},\{1,2,3,4\})$. Then all the results have been averaged in order to a have a unique error for each learning situation[1]. For figure 3.2 we have used the skeleton parameters on the set of real dances videos.

On the figure 3.2 we can see that the different errors are very high (about 76% for the *me*!) when the learning is only performed on one example. In fact the model is under-trained, and is totally unable to recognize the other examples[2]. Then the errors quickly fall very low (5.55% for the *me* and 1.04% for the *mbe*) giving very

[1]1 example, 2 examples, 3 examples and 4 examples.

[2]In some cases, the $\alpha_t(i)$ are so low that they exceed the machine's precision range.

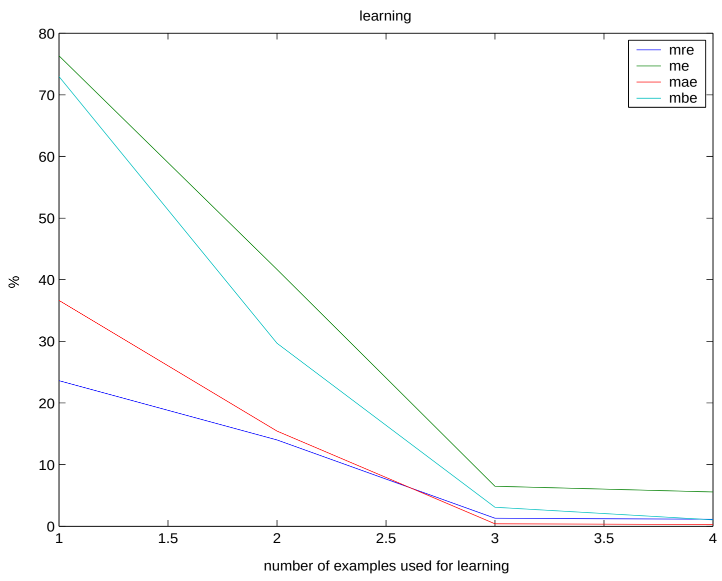

Figure 3.3: learning

good results even with three examples and improving slighly with four. Because of the limited number of examples we are not able to know if the error would rise with more examples or if it would converge to a stable point. In comparison, the *snm*, which are not put in this chapter because of space, gives almost constant results across the different possible training but with greater errors, as can be seen on figure 3.1.

Note that when learning on three examples and recognizing the remaining one which is the situation we would meet in live performances but that we will not show due to space restriction, the results are slightly lower (between 5% and 15% for the *mbe*) than when the example belongs to the training set but it is still very usable. However, when the learning is performed on only 2 examples and following an example that is not in the training set, the results are very bad (60% for the *mbe*) and the error is maximum when the dancers are different.

3.3 Decoding

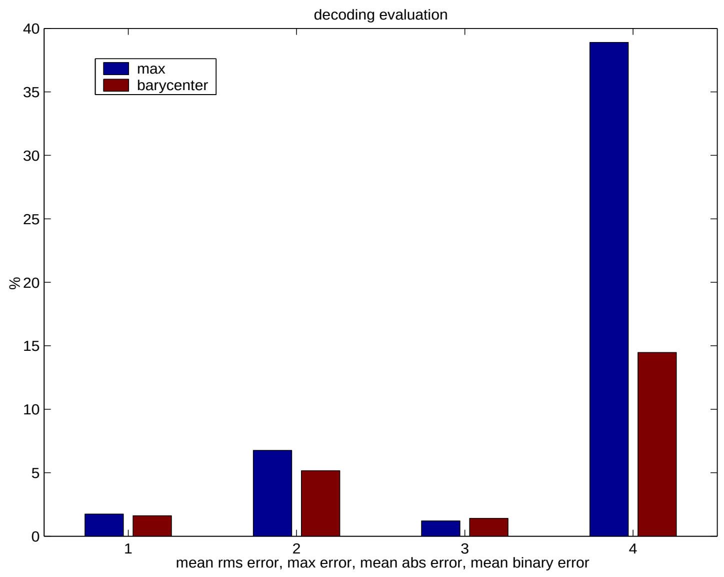

Figure 3.4: decoding evaluation

We wanted to evaluate how the decoding strategy affects the recognition performance. Optimizing the decoding strategy improves the overall recognition rate. This becomes particularly apparent when using a suboptimal set of features such as the SNM taken on the silhouette, as shown in figure 3.3. We used both the real dances and the animations and only used tests where HMMs where trained with the four examples.

We can see in table 3.3 that the barycenter method has lower *mre*, *me* and *mbe* than the max method. The difference is particulary striking with the *mbe*. However the *mae* is in favour of the max method. Since the *mre* penalizes the distance to the original path more than the *mae* it means that the barycenter method stays closer to the original path.

In the rest of this document, if not mentioned explicitely, the decoding method will always be the barycenter.

	max	barycenter
mean rms error (%)	1.74	1.61
max error (%)	6.75	5.16
mean abs error (%)	1.20	1.41
mean binary error (%)	38.89	14.47

Table 3.1: decoding evaluation

3.4 Principal Components Analysis

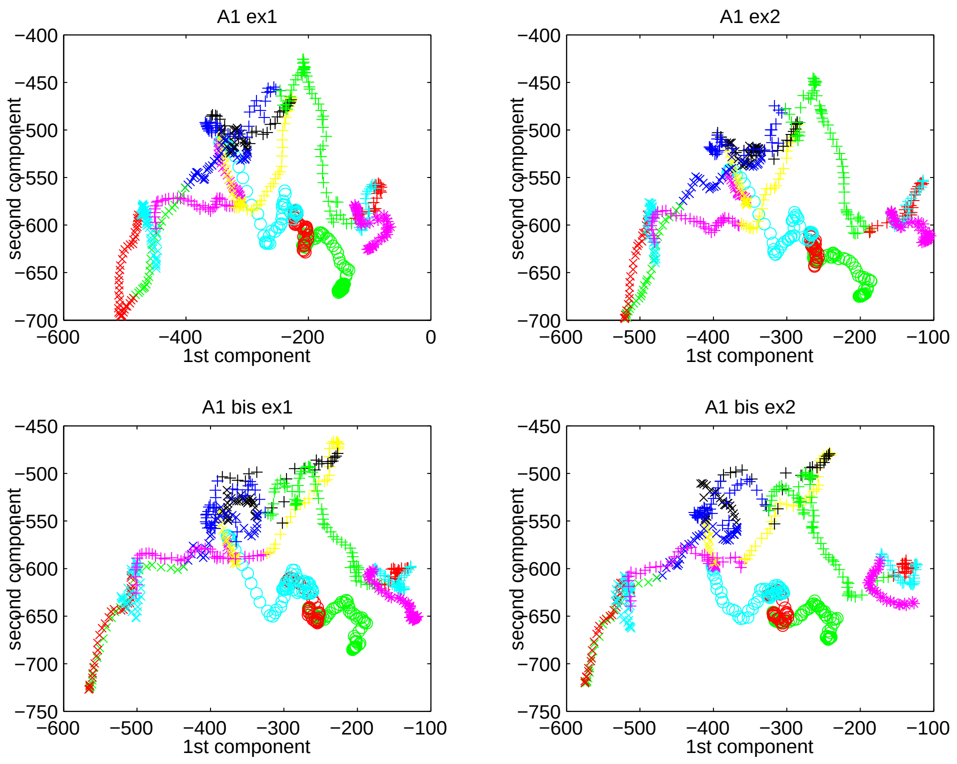

Figure 3.5: trajectories of each video dance projected on the 2 first principal components of the 4 dances data set

We can see on figure 3.4 that the four different examples have very similar trajectories when projected on the first two principal components. The different segments are well localized in the feature space and the manual segmentation is consistent across the examples. It is even possible to view that A1 ex1 and A1 ex2 have more in common than A1 bis ex1 and A1 bis ex2. It is an interresting result since they correspond to two different dancers.

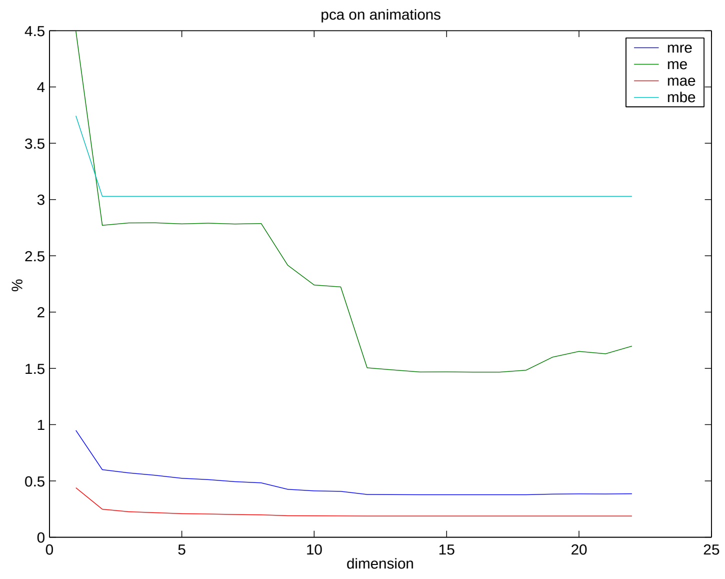

Figure 3.6: pca dimension reduction effect on error, animation data

We wanted to know how the use of PCA as a preprocessing step would help improve following results and reduce the number of parameters required to perform the computation.

We have used both tests with HMMs being trained with an example and following the same example and tests with HMMs being trained with the four different examples and following these four examples successively. Then the respective errors have been averaged for each dimension in order to give a global result.

We can see in figures 3.4 and 3.4 – as we would have expected – that the errors decrease almost monotonously with the number of dimensions in both animations and real dances. We can also note that there is no real benefit in using more than about eight parameters for the real dances and twelve for the animations, when the original number of parameter for the skeleton is 22. It is easily explained by the fact that most of the skeleton parameters are correlated as it can be seen in figure 2.5.2.

We can also note that in the case of the real dances, which were subject to different kinds of perturbations because of the lighting condition, the DV compression and the silhouette segmentation step, the PCA does not help improve the results. There is no minimum in the error curves for which the perturbations would have been removed while keeping the useful signal.

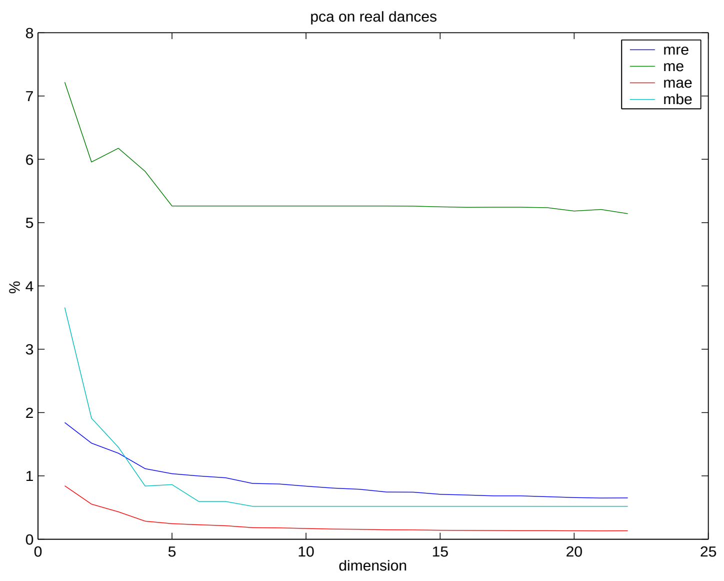

Figure 3.7: pca dimension reduction effect on error, real data

3.5 Segmentation

We wanted to evaluate the influence of the segmentation method on “following”. Because we couldn’t have an automatic segmentation having the same meaning across different dances, we have only used HMMs trained on one example and recognizing the same example. This way we should have the best possible results (since the model is fitted to this example) and only evaluate the influence of the segmentation method.

We have also taken care of the number of states in each segmentation so that they either match or are similar in order to be able to compare the errors.

Animations

For the animations, we already had a kind of manual segmentation given by the footsteps of the character, that had been used to generate the animation. Furthermore, their use can be justified by the fact that they are associated to well defined instants of the movement and can be considered as a “natural” segmentation.

In table 3.5 and figure 3.5 we can see that the error rates are significantly lower when using the footsteps to segment the animation, than when using an automatic segmentation. This shows that the footsteps allows for an effective segmentation

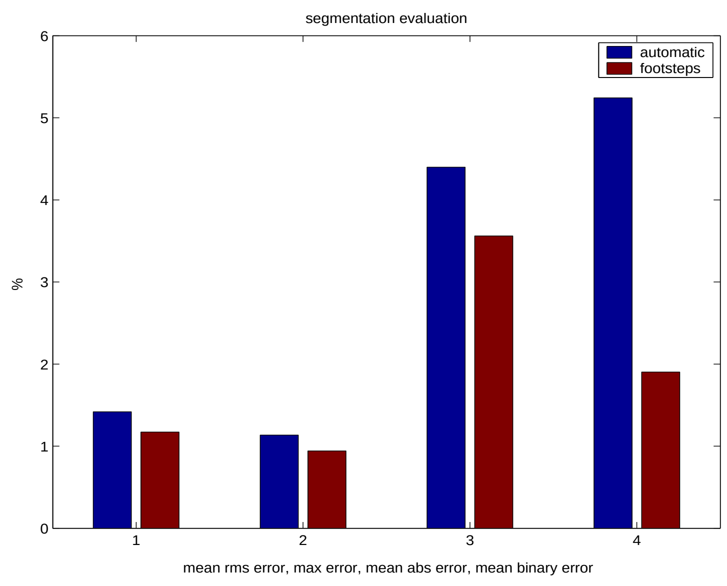

Figure 3.8: segmentation evaluation on animation data

	automatic	footsteps
mean rms error (%)	1.42	1.17
max error (%)	1.13	0.94
mean abs error (%)	4.40	3.56
mean binary error (%)	5.24	1.90

Table 3.2: segmentation evaluation on synthesis data

of the movement.

Videos

For the videos, the manual segmentation was made by searching some key instants and postures simultaneously in the different examples, and the automatic segmentation was still made using the the curvature of the trajectory in the feature space.

In table 3.5 and figure 3.5 we can see that in this case the automatic segmentation has better results. It is easily explained by the fact that automatic segmentation is designed to have each state associated to – as much as possible– non overlapping regions of the feature space.

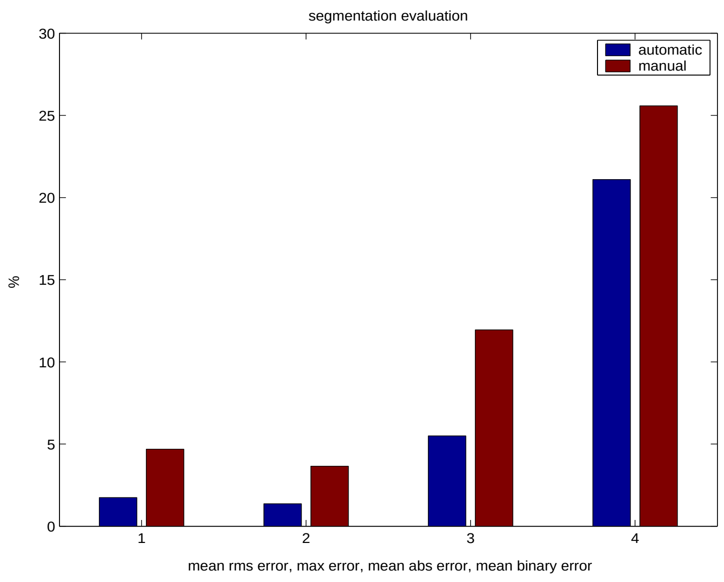

Figure 3.9: segmentation evaluation on real data

	automatic	manual
mean rms error (%)	1.42	1.17
max error (%)	1.13	0.94
mean abs error (%)	4.40	3.56
mean binary error (%)	5.24	1.90

Table 3.3: segmentation evaluation on real data

Furthermore, in this experiment we have forced the automatic segmentation to use the same number of states as the manual one in order to compare them equally. But the natural number of states given by the automatic segmentation was originally about twice the number of states given by the manual one. It means that the manual segmentation is under-specified with respect to the data set.

It would be interesting to investigate a mixed multilevel segmentation with the top layer being defined manually by the choreographer with respect to some key instants in the dance – the ones to be retrieved with precision – and let the system define automatically a finer grained structure inside those states in order to ease the following task.

Chapter 4

Summary and conclusion

We have presented a general “following” system applied to dance with which we had rather good experimental results on both animations and videos of real dances, using the skeleton features. Our methodology has thus being validated which can let us suppose that it might be transposed successfully in other fields.

We have shown that the system can be exposed to under-learning when there are too few examples, but also that it can converges quickly on good results, even with only 4 examples. This suggests that when there is no possibility of collecting several examples we should find a way to give the system more tolerance. This could be realized by using a variance estimator with its interval of confidence instead of a direct computation.

We have also shown that altough Principal Component Analysis is not required to have good results, it could help reduce the number of required dimensions. As the eigenvectors of the covariance matrix can be pre-computed before launching the “following”, it could help reduce the CPU cost of the decoding at the expense of a preprocessing computation. Again, this would need to be investigated in detail.

Though the automatic segmentation of the data which has been used may be better than the manual one, the manual segmentation has nevertheless proven to be effective both for animations and for real dances and to allow easy and accurate training on several examples. Different kinds of automatic segmentation should also be investigated. We could imagine using the HMM as in [19] to perform the segmentation.

We have also proved that the barycenter decoding resulted in smoother paths than the max decoding in the presence of ambiguities among different states. This is a better choice in the case of live performance because we did not wanted our system to move back and forth between different states. We should recommend that the decoding strategy should be choosen or designed on a case by case basis depending on the application, in order to achieve optimal results.

Bibliography

[1] *Gesture-based Communication in Human-Computer Interaction.* Springer Verlag, 2004, ch. Analysis of Expressive Gesture: The EyesWeb Expressive Gesture Processing Library.

[2] Bobick, A. F. Movement, activity, and action: The role of knowledge in the perception of motion (1997).

[3] Bowden, R. Learning statistical models of human motion (2000).

[4] Bradski, G., and Davis., J. Motion segmentation and pose recognition with motion history gradients. *WACV* (2000), 238–244.

[5] Bradski, G. R., and Davis, J. W. Motion segmentation and pose recognition with motion history gradients. In *Machine Vision and Applications* (2002), vol. 13, Springer – Verlag, pp. 174 – 184.

[6] Brand, M., and Hertzmann, A. Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 183–192.

[7] Chenevière, F., Boukir, S., and Vachon, B. Compression and recognition of spatio-temporal sequences from contemporary ballet. *International Journal of Pattern Recognition and Artificial Intelligence* (2003).

[8] F. Bevilacqua, E. F. Captation et analyse du mouvement pour l'interaction entre danse et musique. In *actes des rencontres musicales pluridisciplinaires: le corps et la musique. grame* (2004).

[9] Gavrila, D. M. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding: CVIU 73*, 1 (1999), 82–98.

[10] Ge, X., and Smyth, P. Deformable markov model templates for time-series pattern matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (2000), ACM Press, pp. 81–90.

[11] Hyun-Ju Lee A1, Yong-Jae Lee A1, C.-W. L. Gesture classification and recognition using principal component analysis and hmm. In *IEEE Pacific Rim Conference on Multimedia* (2001), H.-Y. Shum, M. Liao, and S.-F. Chang, Eds., vol. 2195 of *Lecture Notes in Computer Science*, Springer, pp. 756–763.

[12] J. K. Aggarwal, Q. C. Human motion analysis: A review (1999). *Computer Vision and Image Understanding: CVIU 73*, 3 (1999), 428–440.

[13] Orio, N., and Déchelle, F. Score following using spectral analysis and hidden markov models. In *ICMC: International Computer Music Conference* (La Havane, Cuba, 2001).

[14] Orio, N., Lemouton, S., Schwarz, D., and Schnell, N. Score following: State of the art and new developments. In *New Interfaces for Musical Expression (NIME)* (Montreal, Canada, Mai 2003).

[15] Pellegrini, T., Duée, R., and Georges, L. Suivi de voix parlée grace aux modèles de markov cachés, 2003.

[16] Pengyu Hong, Matthew Turk, T. S. H. Gesture modeling and recognition using finite state machines (2000).

[17] Puckette, M. EXPLODE: A User Interface for Sequencing and Score Following. pp. 259–261.

[18] Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (1989), 257–286.

[19] Raphael, C. Automatic segmentation of acoustic music signals using hidden markov models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1999), vol. 21, pp. 360–370.

[20] Tian-Shu Wang, Heung-Yeung Shum, Y.-Q. X.-N.-N. Z. Unsupervised analysis of human gestures. In *IEEE Pacific Rim Conference on Multimedia* (2001), H.-Y. Shum, M. Liao, and S.-F. Chang, Eds., vol. 2195 of *Lecture Notes in Computer Science*, Springer, pp. 174–181.

[21] Troje, N. F. Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. In *Journal of Vision* (2002), vol. 2, pp. 371–387.

[22] Turk, M. Gesture recognition.

[23] Turk, M. A random walk trough eigenspace. In *IEICE Trans. Inf. Syst.* (2001), vol. E84-D, pp. 1586–1695.

[24] Vercoe, B. The Synthetic Performer in the Context of Live Performance. pp. 199–200.

[25] Vercoe, B., and Puckette, M. Synthetic Rehearsal: Training the Synthetic Performer. pp. 275–278.

[26] Volpe, G. *Computational models of expressive gesture in multimedia systems.* PhD thesis, InfoMus Lab, DIST – University of Genova, 2003.

[27] Ying Wu, T. S. H. Vision-based gesture recognition: A review.

[28] Yoshua Bengio, Pascal Vincent, J.-F. P.-O. D.-M. O., and Roux, N. L. Spectral clustering and kernel pca are learning eigenfunctions. Tech. rep., Centre de Recherches Mathematiques, Universite de Montreal, 2003.