Pattern Recognition Letters xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

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

A real-time system for motion retrieval and interpretation

Mathieu Barnachon [a,*], Saïda Bouakaz [a], Boubakeur Boufama [b], Erwan Guillou [a]

[a] Université de Lyon, CNRS, Université Lyon 1, LIRIS, UMR5205, 8 bd Niels Bohr, F-69622 Villeurbanne, France
[b] School of Computer Science, University of Windsor, Canada N9B 3P4

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Motion retrieval
Motion interpretation
Human action recognition
Motion Capture
Automaton
Examplar-based

ABSTRACT

This paper proposes a new examplar-based method for real-time human motion recognition using Motion Capture (MoCap) data. We have formalized streamed recognizable actions, coming from an online MoCap engine, into a motion graph that is similar to an animation motion graph. This graph is used as an automaton to recognize known actions as well as to add new ones. We have defined and used a spatio-temporal metric for similarity measurements to achieve more accurate feedbacks on classification. The proposed method has the advantage of being linear and incremental, making the recognition process very fast and the addition of a new action straightforward. Furthermore, actions can be recognized with a score even before they are fully completed. Thanks to the use of a skeleton-centric coordinate system, our recognition method has become view-invariant. We have successfully tested our action recognition method on both synthetic and real data. We have also compared our results with four state-of-the-art methods using three well known datasets for human action recognition. In particular, the comparisons have clearly shown the advantage of our method through better recognition rates.

1. Introduction

Computer vision aims at being used in our every day life by providing solutions in areas such as entertainment (Freeman et al., 1996), medical applications (Perini et al., 2006) or intelligent video-surveillance (Kilambi et al., 2006), just to name a few. One particular and futuristic application of computer vision is to achieve a Human–Computer Interaction (HCI) (Okwechime et al., 2011; Raptis et al., 2011) of the same level as the human–human one. Ideally, humans will be able to interact with computers as if they were interacting with other humans.

In this work, we are proposing a twofold contribution. First, we significantly reduce the MoCap raw data with linear regressions, both incrementally and online, so that we can provide a comparison space for human motion. Second, we classify and enrich a graph-based automaton of actions where, recurrent motifs and critical transitions are highlighted. As a consequence, we have proposed a new framework to use MoCap data for interaction with a computer in a much richer way. Note that the MoCap data we are referring to could come from any source as long as it consists of a set of spatio-temporal 3D points representing some important features on the human body. These MoCap points usually represent the extracted skeleton joints.

This paper is organized as follows. The next section presents a selection of previous works. Section 3 presents the MoCap data reduction method used before any learning or recognition stage. Section 4 introduces our spatio-temporal metric for comparing and matching actions. Section 5 describes the graph-based automaton for our online recognition and learning processes. Experiments on synthetic and real data are presented in Section 6 and a conclusion is provided in Section 7.

2. Previous works

We have divide this section into three parts. The first deals with Human Computer Interaction, the second concerns the use of human body to identify activities, and the third relates to some works coming from computer animation.

2.1. Human Computer Interaction

Although many researchers have proposed new solutions to improve Human–Computer Interaction, most of them are still simplistic as they were usually limited to a small number of basic interactions. In early solutions, markers were used to ease the tracking/matching problem. The work by Zhang et al. (2001) was among the first to propose a computer vision-based solution to interact with a computer. They have used an ordinary piece of paper as a marker and they were able to point, click, draw and turn. Their main drawback was the use of a simple marker that has limited the number of recognizable actions. Van den Bergh et al. (2005) proposed a generic approach to remotely interact with a computer by detecting the fingers of one hand and localizing the eyes. In theory, they can use all fingers of one hand together with

* Corresponding author. Tel.: +33 (0) 4 26 23 44 45.
E-mail addresses: mathieu.barnachon@liris.cnrs.fr (M. Barnachon), saida.bouakaz@liris.cnrs.fr (S. Bouakaz), boufama@uwindsor.ca (B. Boufama), erwan.guillou@liris.cnrs.fr (E. Guillou).

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2012.12.020

a single finger from the other hand to control a pointing device with several options. This solution can only be used in front of large projected screen and it is not user-friendly for standard interactions. Argyros and Lourakis (2006) detect and use the fingers configuration to control the mouse move and click actions. This markerless solution is a step forward in HCI as it is based on the bare human hand. However, it is limited to the mouse move and click actions and does not exploit the much richer gesture possibilities. Using two cameras, Li et al. (2007) have proposed a vision-based system to simulate touch screen devices. They suffer from similar drawbacks of other previous methods as they only propose touch screen interactions, which is limited in term of interpretation. Going beyond simplistic HCI, Okada and Stenger (2008) have presented a framework, using silhouettes, to build a shape hierarchy tree. They capture motion at the same time as they use it in a virtual environment where, the user can interact through his/her avatar. Because of the inter-silhouette dependency, they had to use 30 human models to produce a set for all recognizable configurations. They build their tree in a batch process and a Cell Broadband Engine™ was required to evaluate the tree, which is far from real time on a typical computer. Although their work is among the first ones to use the potential of full body MoCap to interact with computers, the proposed method is cpu-time consuming and not flexible enough as it requires the presence of very similar actions in the tree. Okwechime et al. (2011) have proposed a framework to synthesize motion from a user input. Their works is closed to action recognition as they have to segment and recognize motion to find transitions. In particular, they are more synthesis oriented in their interaction scheme.

2.2. Human body analysis

Fujiyoshi and Lipton (1998) have performed skeletonization of the body contour to identify walking, running and even gait analysis. Their work is among the first ones to perform human body skeletonization to identify human action. The solution has the advantage of being easy to use but it is mainly useful for simple interpretations. Bobick and Davis (2001) have proposed a method using spatio-temporal templates for human activity recognition. Their method runs in real-time and uses a database of actions they have previously extracted. Although their recognition process is real-time, adding new actions to their database is time consuming and not very flexible. Working with silhouettes for action recognition, Elgammal et al. (2003) used the exemplar paradigm with Hidden Markov Model. This solution has the advantage of being flexible and time efficient. However, it is limited by the 2D primitives, extracted from the video stream. Parameswaran and Chellappa (2003) use MoCap with markers in an invariant motion space. As mentioned by the authors, there is no 3D invariance in motion space, and therefore all actions have to be described independently. To achieve view independence, it is necessary, in most cases, to work in the 3D space and to use an animation skeleton. Huang and Trivedi (2005) have presented the concept of a cylindrical histogram, in a voxel-based representation, to perform recognition using Hidden Markov Chains. In the proposed method by Weinland et al. (2007) exemplars are used to learn view-independent actions of human with hidden Markov chains. The reconstruction process was not required as they performed recognition in 2D space, without prior knowledge of the cameras' positions. Xiong and Liu (2007) have also used Hidden Markov Model on extracted silhouettes, but their target was limited to simple human behaviors. In the work by Lv and Nevatia (2007), actions were modeled as sets of virtual key-poses to be used in the matching process. This method is however limited by the high computation cost and by the number of available virtual key poses. In 2008, Cuntoor et al. (2008) have suggested that trajectories provide the most discriminative solution to understand human behavior. Hence, more and more recent research works are using trajectories for action recognition. Li and Fukui (2008) have proposed a trajectory-based solution where, variations and 2D projections were generated through camera motion only. The experiments presented in their paper were not extensive enough to draw a strong conclusion based on their results. The work by Yilmaz and Shah (2008) is another silhouette-based method where, a 2D spatio-temporal silhouette curve is transformed into a three-dimensional spatio-temporal volume. They detect feature (important) points in these curves then, with a bipartite graph, these features are used to categorize actions. Unfortunately, temporal correspondence, used for feature points, together with extensive calculations produce a lengthy silhouette volume that is prone to errors. Baak et al. (2009) create motion priors base to ensure that certain constraint dependencies are in agreement with the behaviors. For example, they have used the constraint, foot on the floor, during walking. However, achieving a costly high accuracy for motion tracking is not crucial for motion interpretation in general. Han et al. (2010) exploit the skeleton hierarchy to achieve human decomposition and compute trajectories in manifold space on huge animation databases. As they are using a subset of the whole MoCap data, their solution requires a large database of similar actions for correct interpretation. Moreover, a computationally complex and time-consuming preprocessing stage is needed to cluster similar actions. Ahmad and Lee (2010) have extended the concept of motion history to the case of silhouettes. Their method, based on SVM algorithm, depends on too many parameters making the obtained results sensitive to the extracted silhouette accuracy and camera viewpoint. Note that most of the silhouette-based methods suffer from the simple "star" skeletonization model and are mostly surveillance oriented rather than action recognition oriented. Furthermore, they are usually vulnerable to ambiguities in the recognition as they are using two-dimensional information only. Shotton et al. (2011) are capable to obtain MoCap data with good accuracy, largely enough for most HCI applications, as it has been proved by Raptis et al. (2011). Although the efficiency for dance actions is excellent, the whole process was limited by a known number of actions to prevent overall error rate increase.

2.3. Related work from computer animation

The segmentation of MoCap data has been recently addressed for motion recognition in computer animation. Barbič et al. (2004) have used Principal Component Analysis (PCA) to search for peaks in probabilistic distribution, using Mahalanobis distance. In their experiments, only synthetic data have been used for a known number of behaviors. Their major drawback is the level of details they have used in the method. In particular, all body parts are of equal importance in the action which means that irrelevant motions are also included. They do not consider global orientation for segmentation as well. Beaudoin et al. (2008) extract motifs in MoCap databases. According to their own definition, a motif is a subsequence of animations that are very similar to each other in the motion space. They build transitions and obtain a motion-pattern graph by hand labeling. This structure is like a recognition structure as well as a smoothing transition tool. The lengthy preprocessing of clustering and motif extraction makes this solution not a real-time one and therefore, less suitable for HCI. Another similar work by Müller et al. (2009) proposes a solution to automatically annotate large set of MoCap data. It suffers from the same drawbacks and do not propose a structure useful to recognize new actions.

Please cite this article in press as: Barnachon, M., et al. A real-time system for motion retrieval and interpretation. Pattern Recognition Lett. (2013), http://dx.doi.org/10.1016/j.patrec.2012.12.020

3. MoCap data reduction

Unlike Dynamic Time Warping methods of Sakoe and Chiba (1978), where action recognition is tightly related to time, our solution is based on the matching of action elements, obtained by segmenting actions into small temporal poses. These action elements arise from our local regressions based on the exhibition of sudden local changes. In particular, we have expressed trajectories along each of the three spatial axis in a spatio-temporal space. Then, these trajectories are reduced into line segments via linear regressions. The classical view dependence problem has been overcome by using a skeleton-centric coordinate system.

3.1. Skeleton-centric coordinate system

In the context of Human–Computer Interactions, we have to place the user in the center of the recognition process. Instead of using a pseudo-invariant action space or looking for a reference system which is the most discriminant for each action, like PCA approaches, we have used a view-invariant coordinate system, based on the user itself. Depending on the target application, different body-centric coordinate systems have been used in the context of MoCap. In particular, the skeleton root is often used as the origin and articulations are described hierarchically from this root. These coordinate systems are straightforward to compute as they are mostly implicit in the incoming raw MoCap data. Given that Motion Capture of humans have some invariant articulations, such as the skeleton root, we propose in this paper to use this invariance to create a view-invariant coordinate system. Note that any efficient MoCap solution provides an axis for the torso (at least one), a distinction between front and back and, the hips. Building on this MoCap information, we can use the directions of relevant bones to create an orthogonal skeleton-centric coordinate system. In this context, human actions are independent of the viewpoint and trajectories are expressed in a way that is independent from the choice of the coordinate system.

3.2. Trajectory modelisation

Given human body 3D MoCap points in the scene's coordinate system, trajectories are generally often smoothed and approximated by polynomials. Although higher degree curves would give us better approximation of the curvatures, a linear approximation is good enough to identify the significant changes at a much lower cost (see Fig. 1). As one can see from Fig. 1, important changes are typically located at local extrema. In several previous works, especially for machine learning methods, these extrema points are clustered and used instead of raw data. These points indicate geometrical changes in motion but they lack information on the intrinsic velocity and acceleration. Our linear regressions quickly incorporate some indicators of the geometrical changes, such as directions and velocities. These regressions can be computed incrementally and, the obtained line segments provide an estimation of the MoCap point trajectory as well as an error criterion.

3.2.1. Linear regression

We make a piece-wise linear trajectory model with the well-known linear regression and its least-square estimator. Linear regression aims at minimizing the least-square estimator $\hat{\beta}^n$, where

$$\hat{\beta}^n = \arg\min_{\beta_0,\beta_1} \sum_{i=1}^{n} (y_i - \beta_1 x_i - \beta_0)^2 \quad (1)$$

where, in our case, x_i is the time coordinate, and y_i is one of the spatial coordinates (X, Y or Z).

It can be easily shown that the above estimator $\hat{\beta}^n$ can be evaluated incrementally.

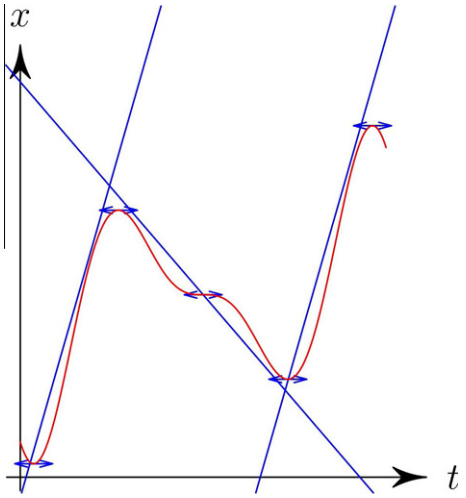


Fig. 1. This curve represents a trajectory along the X-axis of one MoCap point projected onto a 2D spatio-temporal space. The critical changes are marked with double side arrows and the lines represent the approximations of the trajectory using linear regressions.

Rule 1. A point P is added to the linear regression $RegLin_n = [P_0 \quad P_1 \quad \cdots \quad P_n]$ if and only if $\hat{\beta}^{n+1} \leqslant \varepsilon$, otherwise, P will be the starting point of a new linear incremental regression.

Using rule 1, each point trajectory will be transformed into linear segments. The ε parameter describes the granularity (or the precision) of the action segmentation. It can be adjusted to deal with noise from MoCap points extraction.

3.2.2. Trajectory decomposition

For efficiency reasons, we have merged information from all the three axes into one, motion-reduced, state vector. The latter includes information on extrema for each axis, as well as speed and velocity for each MoCap point. Indeed, speed is indicated by the time axis, in-between two sharp changes, and acceleration is encoded in the line slope.

To make the explanation of the next sections easier, we introduce some definitions below.

Definition 1. An *action cut* t_i is a time where a C^1 discontinuity (singularity) of the trajectory occurs.

Such transition happens when each of the three trajectories (along the three axes) of a MoCap data has had at least one C^1 discontinuity. In other words, an *action cut* is located at the latest C^1-discontinuity of the three projections. During the learning stage at the beginning (resp. end) of a MoCap data stream, a *top-start* (resp. *top-end*) is manually set. We assume that *top-start* (resp. *top-end*) is the first (resp. the last) *action cut*.

Definition 2. An action element (AE_P) of a single MoCap point P is the motion of P expressed between two *action cuts*.

Therefore an action element AE_P can be expressed as $AE_P = [P_B \quad P_E \quad T_B \quad T_E]$, where $P_B, P_E \in \mathbb{R}^3$, are the spatial locations of P at times T_B(beginning) and T_E(end), respectively (see Fig. 2).

Let $\mathbb{A}$ be the set of all the skeleton's articulations. According to Definition 2, a path of each MoCap point for the time interval $[T_0, T_1]$, is represented by a sequence of contiguous segments. In particular, for each point $P \in \{\mathbb{A}\}$, we define the *Motion Segment* of P, with respect to $[T_0, T_1]$, to be the concatenation over time of all the *AE*s.

Definition 3. Considering all points of $\{\mathbb{A}\}$ within the time interval $[T_0, T_1]$, an *Action Segment* (*AS*) is defined by

Please cite this article in press as: Barnachon, M., et al. A real-time system for motion retrieval and interpretation. Pattern Recognition Lett. (2013), http://dx.doi.org/10.1016/j.patrec.2012.12.020

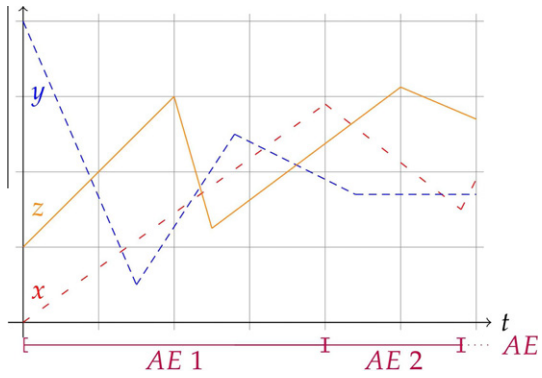


Fig. 2. Example of an action element (*AE*) defined from the linear regressions of a MoCap point trajectory along the three axes.

$$AS = \bigcup_{P\in\{\mathbb{A}\}} \langle\langle AE_P \rangle\rangle_{T_0}^{T_1} = \langle\langle \bigcup_{P\in\{\mathbb{A}\}} (AE_P) \rangle\rangle_{T_0}^{T_1} \tag{2}$$

where $\langle\langle \cdots \rangle\rangle_{T_0}^{T_1}$ denotes the concatenation operator for the interval $[T_0, T_1]$.

As the length of the time interval has to be defined, we have set $|T_0, T_1|$ to be equal to c/fps, where c is a constant defining the *AS* resolution and *fps* is the frame rate of the input video. In practice c could be assigned a value between 6 and 10 for a good compromise between speed and resolution. In other words, an *AS* can be seen as the union of the *AE*s from all the MoCap points during a time interval. Note that at the beginning of the MoCap data input, the starting time of the first *AS* is given by a *Top-start*.

4. Spatio-temporal metric

Because *AE*s are not necessary time invariant, their comparison must involve both spatial and temporal information. The criterion given in Eq. 1 provides an error evaluation for the current acquisition pose that gives too much weight to the temporal error versus the spatial one. Therefore, this simple criterion was not used in our recognition process. Instead, we have defined a spatio-temporal metric for measuring the distance between two *AE*s. When comparing two action elements $AE_P = [P_B^P \;\; P_E^P \;\; T_B^P \;\; T_E^P]$ and $AE_Q = [P_B^Q \;\; P_E^Q \;\; T_B^Q \;\; T_E^Q]$, the distance is given by the following relation:

$$d(AE_P, AE_Q) = \left|\left(\left|P_B^P - P_B^Q\right| - \left|P_E^P - P_E^Q\right|\right)\right| \times \left(1 + \frac{\left|T_E^P - T_E^Q\right|}{\max\{T_E^P, T_E^Q\}}\right) \tag{3}$$

(see Fig. 3 for an illustration)

To keep the notations simple and without loss of generality, let us assume that $T_0 = 0$, as shown on Fig. 3. This is a consequence of our action recognition strategy being based on the independent matching of *AE*s. In addition, the temporal dissimilarity has less weight than the spatial one in our distance. Due to the natural variation of body proportion and movement styles, instances of the same action can be different, depending on the performer.

Given that each *AS* consists of several *AE*s that correspond to all MoCap points, we need to sum up the distances from Eq. 3 for all *AE*s in order to compare two *AS*s. Furthermore, this sum has to be done for each spatial axis as we have three dimensions. Hence, the distance between two *Action Segments* AS_i and AS_j is given by:

$$D(AS_i, AS_j) = \sum_{t=0}^{T} \left[\sum_{P\in\{\mathbb{A}\}} d\left(AE_P^i, AE_P^j\right)\right] \tag{4}$$

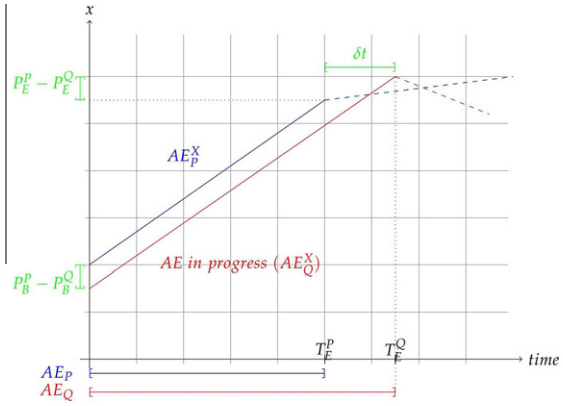
where AE_P^i and AE_P^j are the action elements of AS_i and AS_j, respectively, and $T = \max\{T_1^{AS_i}, T_1^{AS_j}\}$ (see Fig. 3).

The distance defined so far can be used for comparing two *AS*s only. However, as an action is usually made of a number of *AS*s, comparing an action in progress to a known candidate action can be done incrementally. In particular, we have used a voting-like strategy to move from one *AS* to the next one. If the distance *D* between the two current *AS*s is below a threshold, then the match is accepted and the recognition process moves to the next *AS*. Otherwise, the candidate action is dropped and another one is explored. The similarity score between two actions *A*, the one to be recognized, and *B*, a candidate match, at any given time is given by the following relation:

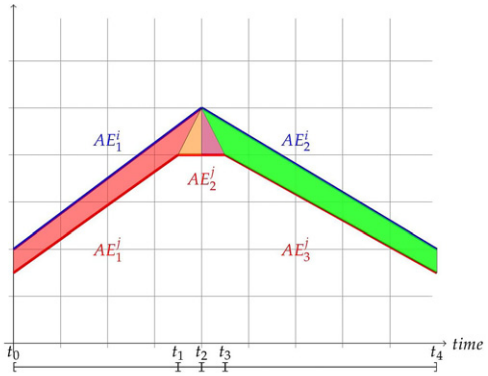
$$C(A, B) = \sum_{i\in[0,\ldots,N]} (1 - M) \tag{5}$$

$$M = \frac{D(AS_i^A, AS_i^B)}{\max\limits_{j\in[0,\ldots,N]} \left\{D(AS_j^A, AS_j^B)\right\}} \tag{6}$$

where $N = |A|$, i.e. the number of *AS*s.



(a) AE_P being compared to an *Action Element* in progress, AE_Q Only the *x*-projection is shown. (Better seen in colors).



(b) Comparison $AS_i = \langle\langle AE_1^i, AE_2^i \rangle\rangle_{t_0}^{t_4}$ and $AS_j = \langle\langle AE_1^j, AE_2^j, AE_3^j \rangle\rangle_{t_0}^{t_4}$.

Fig. 3. Comparison of *AE*s themselves and inside *AS*s.

Please cite this article in press as: Barnachon, M., et al. A real-time system for motion retrieval and interpretation. Pattern Recognition Lett. (2013), http://dx.doi.org/10.1016/j.patrec.2012.12.020

Hence, this similarity score is always positive and the higher its value the better for the recognition. This score can be translated into a percentage value when divided by the number of *AS*s used. Note that this score is calculated online while the corresponding action is progressing.

5. Recognition automaton

This section describes the framework we have used to learn and recognize actions. Our main goal here is to perform online action recognition using MoCap data as input.

As described in the previous sections, our trajectories are expressed in a very efficient way as they have been significantly reduced then expressed as compact state vectors. We have also coupled our trajectory reduction with an efficient structure, the proposed recognition automaton illustrated on Fig. 4. As this structure has to be easy to create and flexible enough to include new actions in a straightforward way, we have found that our best choice for this task is the graph structure, representing an automaton. Note that trees would not be appropriate in our case as we have decided to avoid data hierarchy and the use of databases. Although graphs have been used in the animation community Kovar et al. (2002), it was with a different goal as the main target was the creation of new smooth animations from existing ones. In our case, we are aiming at highlighting shared parts of actions as well as reaching a quick decision on the recognition process. The state nodes in our automaton graph represent the *AS*s of the learned actions and each transition represents the sequence of *AE*s leading to the next *AS*. As an action typically consists of a few *AS*s, it is represented by a set of state nodes that are linked in the graph over time. An *AS* can have more than one transition when it is part of more than one action. However, each state transition in the graph has a unique label representing the sequence of *AE*s leading to the next *AS*. The automaton, see Fig. 4, has different kinds of state nodes. It has *Starting AS*, which are the initial *AS*s of each learned action. Each *Starting AS* represents a starting point for the recognition process of a given action. We have also *Accepting AS* to mark the recognition or accepting states. In some cases, a *Starting AS* can be also an *Accepting AS*, which translates into a loop state transition in the graph. Such state transitions describe repetitions inside an action. Moreover, these loops can be used for noise detection, e.g. repetitions of successive small irrelevant actions.

5.1. Adding (learning) new actions

The efficiency of our reduction process and the use of *AS*s in the automaton have made the process of adding new actions straightforward. An incoming action, coming as MoCap data, is first reduced using our linear regression method then, its *AS*s are constructed in sequence. If the first *AS* is matched to an existing state node in the automaton, this node becomes a *Starting AS* (if it was not one already). Otherwise, a new *Starting AS* node is created. The same process is repeated for the remaining *AS* of the action with the difference that newly created state nodes are regular *AS*s or an *Accepting AS* for the last one. The appropriate transitions, made up of *AE*s, are also created between *AS*s of the same action. The pseudo-code given in Algorithm 1 summarizes the learning process of a new action.

Algorithm 1. Adding a new action (learning) into the automaton.

Input: Automaton Graph $\mathscr{G}$ and a video stream $\mathscr{V}$
repeat
  if *Top-Start* **then**
    while *not Top-End* **do**
      $AS_i \leftarrow$ GetAS (from MoCap ($\mathscr{V}$));
      forall $AS_i \in \mathscr{G}$ **do**
        if $AS_i == AS$ **then**
          MergeSuccessors (AS_i);
          MergePredecessors (AS_i);
        else
          Add (AS_i, to $\mathscr{G}$);
        end
      end
    end
  end
until *EndLearningProcess*;

Although our recognition method is an examplar-based one, actually more than one action instance is used in the learning stage. First, using one examplar, an action is learned and added to the automaton graph as described above. Then, the same action, now known, is repeated t times and for each instance, the distances

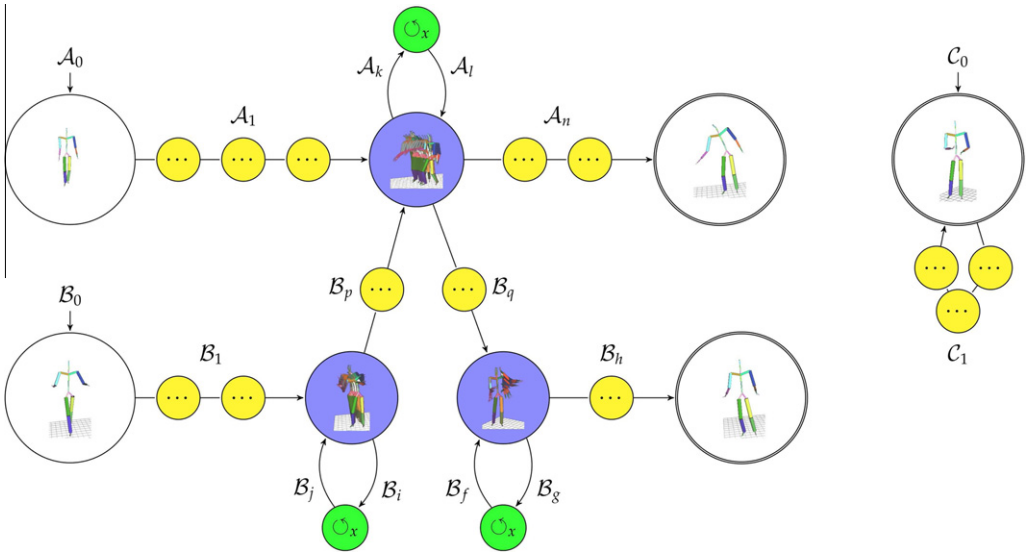


Fig. 4. Part of the automaton for three learned actions $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$. Starting *AS*s are shown with an arrow on them while accepting *AS*s are shown with double circles. Regular *AS*s are represented as yellow states. For example, action $\mathscr{A}$ is recognized through the path $\mathscr{A}_0 \rightarrow \mathscr{A}_1 \rightarrow \{\mathscr{A}_k \rightarrow \mathscr{A}_l\}^* \rightarrow \mathscr{A}_n$, while action B could be recognized by $\mathscr{B}_0 \rightarrow \mathscr{B}_1 \rightarrow \{\mathscr{B}_i \rightarrow \mathscr{B}_j\}^* \rightarrow \mathscr{B}_p \rightarrow \{\mathscr{A}_k \rightarrow \mathscr{A}_l\}^* \rightarrow \mathscr{B}_q \rightarrow \{\mathscr{B}_f \rightarrow \mathscr{B}_g\}^* \rightarrow \mathscr{B}_h$. On the other hand, action $\mathscr{C}$ does not share any node with other actions.

Please cite this article in press as: Barnachon, M., et al. A real-time system for motion retrieval and interpretation. Pattern Recognition Lett. (2013), http://dx.doi.org/10.1016/j.patrec.2012.12.020

D, defined in Eq. 3, between the *AS*s, are calculated. These distances are used to estimate the standard deviations σ_i for the different *AS*s, and the mean of *AS*s is used inside the automaton. Note that there will be no overhead for the recognition process as only one examplar per action is added to the automaton.

5.2. Action recognition

Similar to the learning process, an incoming action, coming as MoCap data, is reduced using our linear regression method then, *AS*s are constructed and matched to the automaton nodes in sequence. Each state transition, made of *AE*s, is used to compare the current action with a learned one, with respect to the current *AS*. Hence, recognizing a whole action turns into finding a path in our automaton graph, see the pseudo-code given in Algorithm 2.

During the recognition process of an action, we have used a score, typically 0.9 (or 90% of similarity, like most other proposed recognition methods), to determine whether or not an action in progress matches a learned one. However, this score is computed incrementally, making it possible to decide on the outcome, at any step, of the recognition process. That is, whether the current action (path) is a learned action or not. Furthermore, at any step, the automaton is able to tell us whether there are multiple solutions. In particular, a strong divergence of the current action at its end, would not significantly affect the score. For instance, the recognition process of a human fall action should not be sensitive to the motion occurring right after the person hits the ground.

Algorithm 2. Action recognition using the automaton.

Input: Automaton Graph $\mathscr{G}$ and a video stream $\mathscr{V}$
repeat
  $AS_i \leftarrow$ GetAS (from MoCap ($\mathscr{V}$));
  CurrentNode $\leftarrow$ FindStartingAS (from $\mathscr{G}$, to match AS_i);
  if *no matching AS found* **then**
    // no action found for this AS_i;
  else
    while *CurrentNode is not an accepting AS* **do**
      $AS_i \leftarrow$ GetAS (from MoCap ($\mathscr{V}$));
      CurrentNode $\leftarrow$ FindAS (from *Succesors*(*CurrentNode*), to match AS_i);
      if *no matching AS found* **then**
        break
      end
    end
  end
until *EndOfVideoStream*;

5.3. Complexity

Our proposed method allows to recognize learned actions as well as to add new ones to the current automaton. Yet, the process remains real-time as its computational complexity has been kept very low, thanks to our compact trajectory linear regression and to our automaton.

Let's start with the recognition part and estimate its complexity in term of number of operations (comparisons). If we assume that we have already n actions coded in our automaton, then we have at most n staring *AS* (starting points). In the worst case, we will need to make n comparisons to find a starting point in the automaton. Once this starting point has been found, we navigate from one node (*AS*) to another, by making a few comparisons on the *AE*s (transitions) at each step, until we reach an accepting state. Although the number of transitions from one *AS* can be more than one when that *AS* is shared by two or more actions, it remains a very small number that is often equal to one. The number of *AS*s per action can also affect the recognition time. The worse-case recognition complexity will be $\mathcal{O}(n \times m)$, where m is the number of *AS*s per action. However, both n and m are typically two small constants. The number of learned actions n cannot be very large, a few dozen learned actions is a very realistic number. On the other hand, m depends on the action itself and it could range from one to a few hundreds for very long actions. In all situations, the product $n \times m$ remains a constant with an upper bound and therefore, the complexity of the recognition can be considered to be $\mathcal{O}(1)$. Another important observation about very long actions is that the recognition process is done during the MoCap data acquisition. In other words, when a long action with a lot of *AS*s is to be recognized, there will be more time available for this recognition because of the lengthy data acquisition. Hence, the recognition process remains real time regardless of the length of the action.

On the other hand, there is an additional overhead for adding new actions to the automaton graph during the learning phase. Let's assume again that we have already n actions coded in our automaton. This translates into having at most n starting *AS*s. Let's also assume that our automaton has a total number of $n \times m$ nodes, where m is the average number of *AS*s per action or the average number of nodes between a starting node and an accepting one. When a new action coming from the MoCap stream is being processed, its first *AS* will be compared to all the $n \times m$ nodes of the automaton, in a bid to find an existing *AS* for a match. If a match is found, then there is no need to create a new node, otherwise, a starting *AS* is created and added to the automaton. This process is repeated again for each of the remaining *AS* of the new action. Assuming that the new action has also m *AS*s, the learning process requires a total of $n \times m^2$ operations in the worse case. Hence, the complexity of this task is $\mathcal{O}(n \times m \times m)$. As discussed earlier, n and m are small constants, for example, if we have already 30 known actions and our m is around 50, the total number of needed comparisons for the worse case will be is 75,000. As for the recognition task, although it takes more time to learn a very long action, there is also more time available because of the lengthy data acquisition. Note also that the vast majority of the automaton work is to recognize actions, while adding new ones is done occasionally. As a conclusion, even if the addition of a new action to the automaton comes with a small overhead, it has a negligible effect on the whole process given that this is a relatively rare event, when considering a long period of time.

6. Results

This section presents the experiments we have carried out on both synthetic and real data. For the latter, in addition to comparing our results with other methods, we have also developed an online demonstrator that uses video-extracted MoCap data to control a "software" by actions.

6.1. Experiments with synthetic data

The synthetic data we have used here were obtained from the CMU Graphics Lab Motion Capture Database (MoCap CMU, 2003). We have performed action recognition after adding different levels of noise to our data. To achieve this, we have simply changed the actions' timing, and consequently velocity and gravity, to simulate real actions done by human actors. In order to perform physical plausible variations, we have used the method proposed by McCann et al. (2006). The deformation is based on a cubic spline expressed in the $\{sourcetime\} \times \{playbacktime\}$ space. In this system, variations are expressed by a set of constraints with friction, gravity, ground contact, etc. The obtained result is an optimal

Please cite this article in press as: Barnachon, M., et al. A real-time system for motion retrieval and interpretation. Pattern Recognition Lett. (2013), http://dx.doi.org/10.1016/j.patrec.2012.12.020

Table 1
Obtained results on 15 actions from MoCap CMU (2003), where A and B are two actors. The "# *AS*" column shows the number of action segments obtained from our reduction process while the next column gives the percentage ratio between these *AS*s and the original number of frames.

	Action name	# *AS*	Reduction Rate (%)	Learning time	Video length (s)
1	Walk, shake hands (1)	87	71.10	0.002	2.50
2	Walk, shake hands (2)	87	71.10	0.101	2.50
3	A pulls B (1)	87	85.30	0.168	4.93
4	A pulls B (2)	89	78.02	0.946	3.38
5	A pulls B by the elbow (1)	87	80.13	0.994	3.65
6	A pulls B by the elbow (2)	87	78.62	0.517	3.39
7	Navigate busy sidewalk	69	76.28	0.586	2.42
8	Conversation (1)	87	95.83	0.647	17.38
9	Conversation (2)	70	62.50	0.948	1.93
10	Quarrel (1)	87	95.18	1.122	15.04
11	Quarrel (2)	87	93.06	1.404	10.44
12	Friends meet, hang out	87	95.05	1.471	10.44
13	Run, scramble for last seat (1)	87	81.33	1.934	3.88
14	Run, scramble for last seat (2)	87	73.80	2.013	2.77
15	Chicken dance	87	92.96	2.113	12.8

increasing time

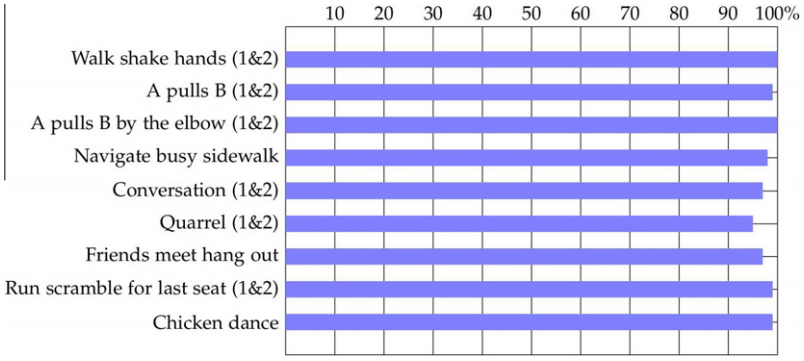


Fig. 5. Recognition results for 15 synthetic actions made on 100 random variations for each action (variations were obtained using McCann et al. (2006)).

motion related to the spline made by the user. As a consequence, this type of simulation is plausible, i.e. a real actor will likely do the same motion given these constraints.

Table 1 summarizes our results using 15 actions from MoCap CMU (2003). As shown in the "Reduction Rate" column, the number of *AS*s, resulting from our data reduction process, is much smaller than the MoCap raw data. We have computed, as a percentage, the ratio between the obtained number of *AS*s and the original number of frames per action. Note that each frame represents a set of joints (articulations of the tracked skeleton) within an action. When noise is added, as described above, the number of *AS*s increases slightly by 1–5%. Although the learning time increases with the size of the graph, it remains very small and less than the video length for all the 15 actions. On the other hand, the recognition process remains real time regardless of the video length and the graph size. Using the method by McCann et al. (2006), we have generated 100 variations for each of these 15 actions and used them for our recognition tests. Fig. 5 summarizes the obtained results with an excellent recognition rate close to 100% for all actions.

We have also carried out experiments to illustrate qualitative results when noise is added. Fig. 6 shows some kind of variations applied to our actions. We can see that the proposed solution is not that sensitive to time variation in action recognition. Even in presence of high time variations, we are still able to recognize the original actions performed, and most of the time, common parts to other actions.

6.2. Experiments with real data

We have carried out three sets of experiments with real data, described below in parts I, II and III. In the first set, we have tested the proposed method for robustness and accuracy, while in the second set of experiments we have compared it to several state-of-the-art action recognition methods. Finally, we have developed an online demonstrator that uses video-extracted MoCap data, to control a "software" by actions.

In all our real experiments, we have used live Motion Capture data obtained with the method proposed by Shotton et al. (2011). The latter performs real-time Motion Capture from the kinect sensor system (see Fig. 7).

6.2.1. Part I: first set of experiments

We have created our own database of 23 different actions to demonstrate the recognition ability and effectiveness of our system (see Fig. 8). In particular, each action has been performed 5 times by one actor to set the means for the *AS*s, as described in Paragraph 5.1. Then our recognition system was used to recognize the different actions performed by 5 different actors, where each actor repeated the same action 5 times. As it can be seen from Fig. 8, the success rate is always above 80% and mostly above 90%. This is a very good success rate given the variety of the actions and actors.

6.2.2. Part II: comparison

We have compared our method to Yao et al. (2011); Müller et al. (2009); Tenorth et al. (2009) and Krausz and Bauckhage (2010). These solutions involve three different datasets, the HDM dataset and a restriction of the CMU dataset, both proposed by Müller et al. (2009); and the TUM dataset, proposed by Tenorth et al. (2009). The HDM and CMU datasets are made by markers Motion Capture systems while the TUM dataset is an acquisition of different *scenarii* in a kitchen. For the latter, the actor is asked to dress a

Please cite this article in press as: Barnachon, M., et al. A real-time system for motion retrieval and interpretation. Pattern Recognition Lett. (2013), http://dx.doi.org/10.1016/j.patrec.2012.12.020

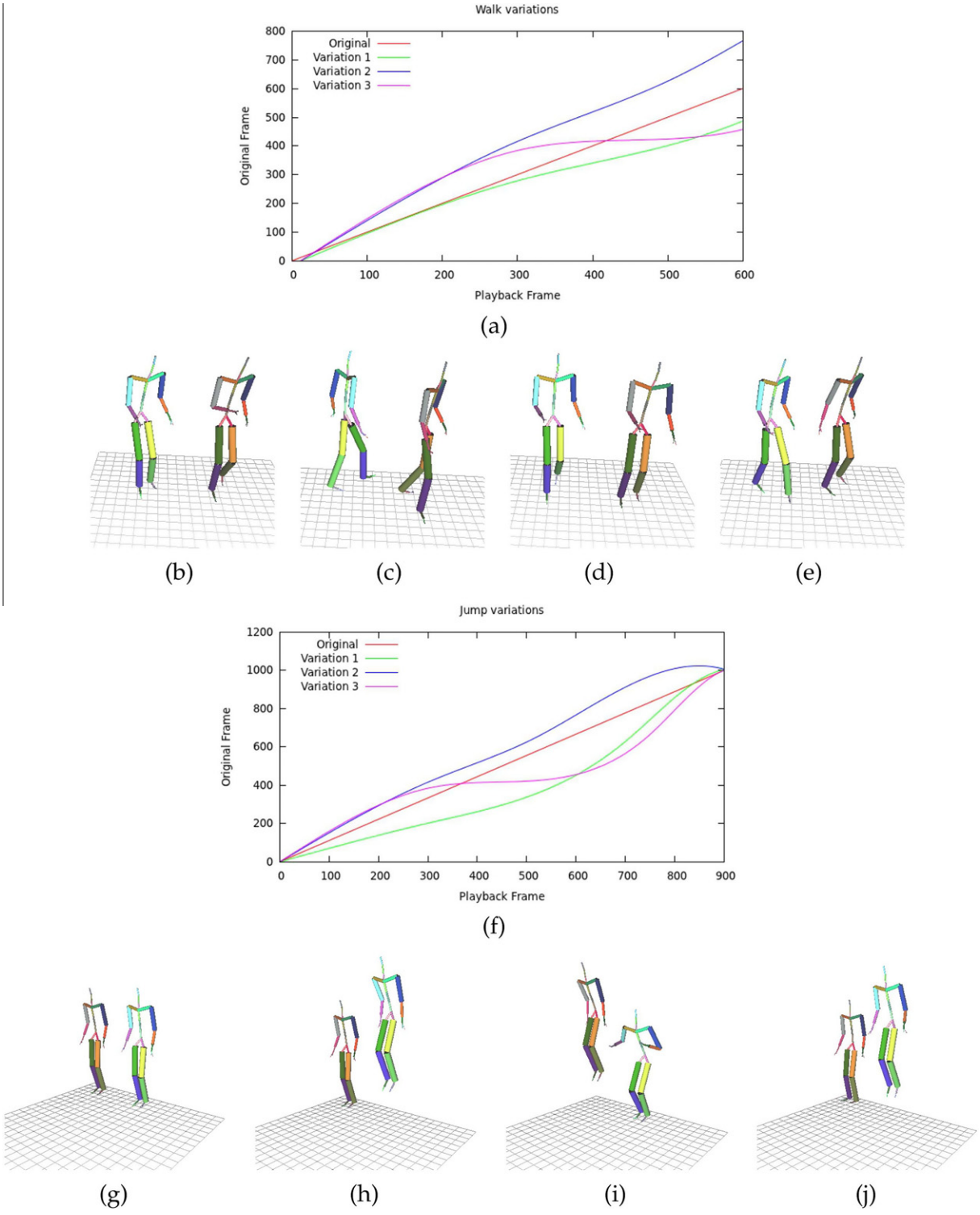
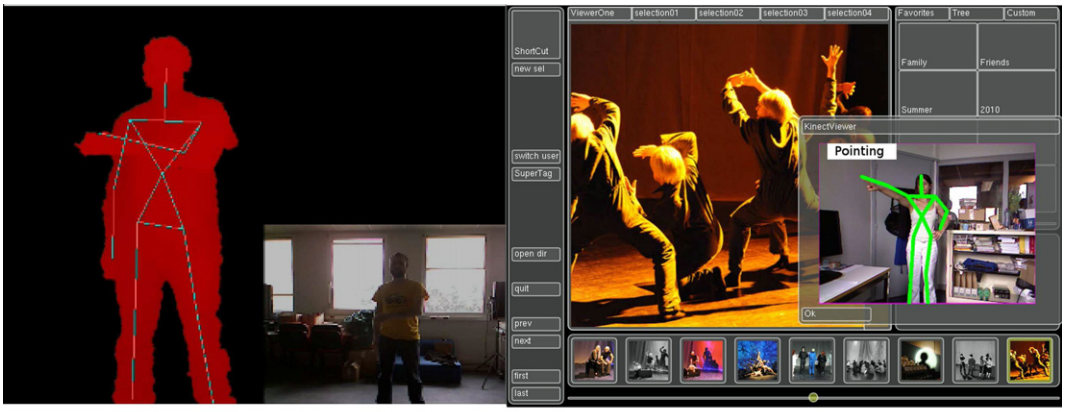


Fig. 6. Variations of the walk action 18_01 (figures (b)–(e)) and jump action 16_03 (figures (g)–(j)) where the original actions are represented with the red line on figures (a) and (f) while the other splines represent their physics-based motion re-timing. Figures (b)–(e) and figures (g)–(j) illustrate the differences between original and perturbed poses for a number of motion re-timings.

table, carrying one or several objects at the same time. Previous proposed solutions are machine learning oriented, i.e. they involve multiple training samples and need a statistical validity of action (intra-class variations).

As actions are simpler in the HDM and CMU datasets, our solution outperformed the method by Müller et al. (2009) on these two datasets. Note that their goal was to create a tool for animators while ours is to recognize human action. They have usually made several classes for single actions. For instance they have different classes for the action "walk": starting left foot, starting right foot, one step, two steps, etc. In our case however, walking has to be in the same class, regardless of the starting foot. Hence, we ended up with less classes than Müller et al. (2009) as we have combined several of their classes into single ones.

Please cite this article in press as: Barnachon, M., et al. A real-time system for motion retrieval and interpretation. Pattern Recognition Lett. (2013), http://dx.doi.org/10.1016/j.patrec.2012.12.020



(a) Sample of action used in our real dataset. (b) A snapshot of our slide viewer demonstrator.

Fig. 7. Example of real actions used in our experiments.

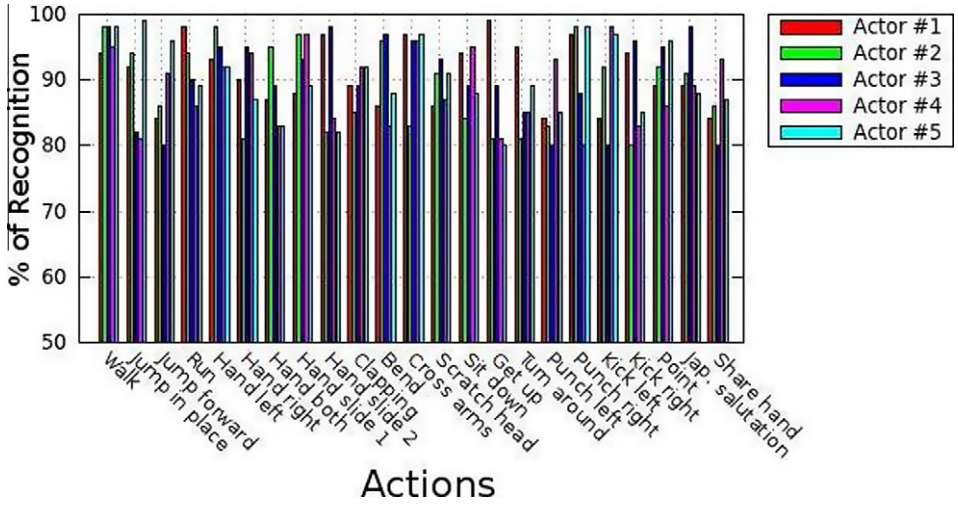


Fig. 8. Results on 23 real human actions where the recognition percentage is scaled to be between 50% and 100% for clarity.

Table 2
Comparison of our solution with state-of-the-art datasets.

Method	Dataset		
	HDM (%)	CMU (%)	TUM (%)
Müller et al. (2009)	80	75	–
Yao et al. (2011)	–	–	81.50
Krausz and Bauckhage (2010)	–	–	67
Tenorth et al. (2009)	–	–	62.77
Our	91.34	90.78	85.30

The TUM dataset is more complex as it is made of everyday human activities. Actions are shorter, close to each other and sometimes difficult to identify. Indeed, Tenorth et al. (2009) have proposed three different labels for each frame: left hand, right hand and torso. To be consistent with others (Tenorth et al., 2009; Krausz and Bauckhage, 2010; Yao et al., 2011), we have used the same learning procedure, i.e. seven sequences for testing and the others for training. We have used 10 classes of actions and used the right hand labels as ground truth.

The obtained comparison results are summarized in Table 2. Our solution outperformed the best result of Yao et al. (2011) for this dataset with a 3.8% margin, without using 2D information from images as they do. The other methods were significantly outperformed even that Tenorth et al. (2009) were using RFID sensors for doors.

We can also mention and discuss two other methods that use 3D information as we do, without being able to perform experimental comparisons because of lack of the datasets used in these publications. The method by Weinland et al. (2007) uses 3D information to avoid view ambiguity but it is aimed at distinguishing between very different actions. The other method, by Okada and Stenger (2008), has the advantage of using a simple single camera for Motion Capture. However, their 3D information is a model-based approximation of the actual 3D structure. In particular, they have to generate thousands of models to fully and precisely approximate the human body. Furthermore, they did not propose action identification and/or recognition. Our examplar-based solution is not based on complex model nor does it need an expensive batch process to compute a recognition-ready space.

Our proposed method is able to overcome the previous limitations as it is using the full 3D skeleton, extracted using MoCap techniques. We can easily distinguish between very similar actions, like sitting and bending actions (see the recognition rate in Fig. 8, which is around 90%). For example, using a hand, we can replace mouse action (even without recognizing actions). We can "record/learn" many computer shortcuts and replace them with simple human body actions. We can also navigate through "noisy actions" as we did not use generic models to perform Motion Capture.

Please cite this article in press as: Barnachon, M., et al. A real-time system for motion retrieval and interpretation. Pattern Recognition Lett. (2013), http://dx.doi.org/10.1016/j.patrec.2012.12.020

6.2.3. Part III: a demonstrator

We have used a demonstrator to illustrate HCI possibilities. The demonstrator is a simple slide viewer where, actions are mapped to common commands such as next/previous slide, begin/end, etc. The user can choose its own way of making these commands, with a "top start" and a "top end" from the MoCap stream (this part is done with the help of a WiiRemote). The video, given in supplemental material, shows its efficiency in a presentation context (see Fig. 7 for an snapshot).

In order to minimize noise sensitivity for this demonstrator, our reduction process required at least 10 frames before creating a single *AS*. In addition to overcoming the noise problem, this minimum number of frames has allowed us to prevent over segmentation, as 10 frames for this MoCap solution represents $\frac{1}{3}$ second of video input. Small range motions for the case of this proposed demonstrator were also filtered out thanks to this requirement.

7. Conclusion

We have proposed in this paper a new framework for online action recognition, also known as online Motion Capture interpretation, using 3D markerless MoCap data as input. The contributions of this paper are twofold. (1) A new method to reduce the raw MoCap data into linear regressions, expressed in compact state vectors, was proposed and, (2) a real-time method, using a graph-based automaton, to learn and recognize actions. We have also overcome the view-dependency problem by defining and using a skeleton-centric coordinate system. The proposed method has the advantage of being linear and incremental and therefore, making the recognition task very fast. Furthermore, the addition or learning of a new action is also straightforward.

Our experiments with synthetic data, obtained from the CMU Graphics Lab Motion Capture Database, have yielded extremely high recognition rate even when noise was added to the simulations. Our experiments with real datasets, constructed by our group, have also shown the robustness and effectiveness of our method. In addition, we obtained better recognition rate when we compared our method to four others, using well known datasets. We have also successfully demonstrated our system on real videos for the case of a slide viewer, where a speaker uses his hands to command a slide show.

Note that the proposed recognition method has numerous applications as it can be applied to almost any Human Computer Interaction situation. In particular, it can be used to enhance and make 3D computer games more attractive with plenty of new possibilities.

Our future work will include the extension of this approach to cope with very different accelerations in actions and to make distinction between multi-resolution actions, i.e., to distinguish inner behaviors within an action, like exhibiting step during walk or dance. We also plan to investigate the possibility of including machine learning aspects to our method in order to make it more robust when MoCap fails to extract the correct motion information.

Acknowledgment

This work has been supported by ANR (french National Research Agency) and CNSA (french National Solidarity Fund for Autonomy).

References

Ahmad, M., Lee, S.W., 2010. Variable silhouette energy image representations for recognizing human actions. Image Vision Comput. 28 (5), 814–824.

Argyros, A., Lourakis, M., 2006. Vision-based interpretation of hand gestures for remote control of a computer mouse. In: Computer Vision in Human–Computer Interaction, pp. 40–51.

Baak, A., Rosenhahn, B., Müller, M., Seidel, H.P., 2009. Stabilizing motion tracking using retrieved motion priors. In: IEEE 12th Internat. Conf. on Computer Vision, pp. 1428–1435.

Barbič, J., Safonova, A., Pan, J.Y., Faloutsos, C., Hodgins, J.K., Pollard, N.S., 2004. Segmenting motion capture data into distinct behaviors. In: GI '04: Proc. Graphics Interface 2004. Canadian Human–Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, pp. 185–194.

Beaudoin, P., Coros, S., van de Panne, M., Poulin, P., 2008. Motion-motif graphs. In: SCA '08: Proc. 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Eurographics Association, Aire-la-Ville, Switzerland, pp. 117–126.

Bobick, A.F., Davis, J.W., 2001. The recognition of human movement using temporal templates. IEEE Trans. Pattern Anal. Machine Intell. 23 (3), 257–267.

Cuntoor, N., Yegnanarayana, B., Chellappa, R., 2008. Activity modeling using event probability sequences. IEEE Trans. Image Process. 17 (4), 594–607.

Elgammal, A., Shet, V., Yacoob, Y. Davis, L.S., 2003. Learning dynamics for exemplar-based gesture recognition. In: IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, vol. 1, p. 571.

Freeman, W.T, Tanaka, K., Ohta, J., Kyuma, K., 1996. Computer vision for computer games. In: IEEE Internat. Conf. on Automatic Face and Gesture Recognition, p. 100.

Fujiyoshi, H., Lipton , A.J., 1998. Real-time human motion analysis by image skeletonization. In: IEEE Workshop on Applications of Computer Vision, p. 15.

Han, L., Wu, X., Liang, W., Hou, G., Jia, Y., 2010. Discriminative human action recognition in the learned hierarchical manifold space. Image Vision Comput., 836–849.

Huang, K.S., Trivedi, M.M., 2005. 3D shape context based gesture analysis integrated with tracking using omni video array. In: CVPR '05: Proc. 2005 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR'05) – Workshops. IEEE Computer Society, Washington, DC, USA, p. p. 80.

Kilambi, P., Masoud, O., Papanikolopoulos, N., 2006. Crowd analysis at mass transit sites. In: IEEE Internat. Conf. on Intelligent Transportation Systems, pp. 753–758.

Kovar, L., Gleicher, M., Pighin, F., 2002. Motion graphs. ACM Trans. Graph. 21 (3), 473–482.

Krausz, B., Bauckhage, C., 2010. Action recognition in videos using nonnegative tensor factorization. In: Internat. Conf. on Patterns Recognition. pp. 1763–1766.

Li, X., Fukui, K., 2008. View invariant human action recognition based on factorization and hmms. IEICE Trans. Inf. Systems E91-D (7), 1848–1854.

Li, S., Lv, J., Xu, Y., Jia, Y., 2007. Eyescreen: A gesture interface for manipulating on-screen objects. In: Human–Computer Interaction. HCI Intelligent Multimodal Interaction Environments. pp. 710–717.

Lv, F., Nevatia, R., 2007. Single view human action recognition using key pose matching and viterbi path searching. In: CVPR, pp. 1–8.

McCann, J., Pollard, N.S., Srinivasa, S., 2006. Physics-based motion retiming. In: SCA '06: Proc. 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 205–214.

MoCap CMU, 2003. The data used in this project was obtained from <http://mocap.cs.cmu.edu/>. The database was created with funding from nsf eia-0196217. <http://mocap.cs.cmu.edu/>.

Müller, M., Baak, A., Seidel, H.P., 2009. Efficient and robust annotation of motion capture data. In: SCA '09: Proc. 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. ACM, New York, NY, USA, pp. 17–26.

Okada, R., Stenger, B., 2008. A single camera motion capture system for human–computer interaction. IEICE Trans. Inf. Systems, 1855–1862.

Okwechime, D., Ong, E.J., Bowden, R., 2011. MIMiC: Multimodal interactive motion controller. IEEE Trans. Multimedia 13 (2), 255–265.

Parameswaran, V., Chellappa, R., 2003. View invariants for human action recognition. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, vol. 2, p. 613.

Perini, E., Soria, S., Cucchiara, APR., 2006. Facemouse: A human-computer interface for tetraplegic people. In: Computer Vision in Human–Computer Interaction, pp. 99–108.

Raptis, M., Kirovski, D., Hoppe, H., 2011. Real-time classification of dance gestures from skeleton animation. In: Proc. 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation – SCA '11. ACM Press, p. 147.

Sakoe, H., Chiba, S., 1978. Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Signal Process. 26, 43–49.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A., 2011. Real-time human pose recognition in parts from a single depth image. In: CVPR, pp. 1297–1304.

Tenorth, M., Bandouch, J., Beetz, M., 2009. The TUM kitchen data set of everyday manipulation activities for motion tracking and action recognition. In: IEEE Internat. Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS), in conjunction with ICCV.

Van den Bergh, M., Servaes, W., Caenen, G., De Roeck, S., Van Gool, L., 2005. Perceptive user interface, a generic approach. In: Computer Vision in Human–Computer Interaction, pp. 60–69.

Weinland, D., Boyer, E., Ronfard, R., 2007. Action recognition from arbitrary views using 3d exemplars. In: Proc. Internat. Conf. on Computer Vision, Rio de Janeiro, Brazil. 2007. p. 1–7.

Xiong, J., Liu, Z., 2007. Human motion recognition based on hidden markov models. In: Advances in Computation and Intelligence, pp. 464–471.

Yao, A., Gall, J., Fanelli, G., Gool, L.V., 2011. Does human action recognition benefit from pose estimation? In: British Machine Vision Conference, pp. 67.1–67.11.

Yilmaz, A., Shah, M., 2008. A differential geometric approach to representing the human actions. Computer Image and Vision Understanding 109 (3), 335–351.

Zhang, Z., Wu, Y., Shan, Y., Shafer, S., 2001. Visual panel: Virtual mouse, keyboard and 3d controller with an ordinary piece of paper. In: PUI '01: Proc. 2001 Workshop on Perceptive User Interfaces. ACM, New York, NY, USA, pp. 1–8.

Please cite this article in press as: Barnachon, M., et al. A real-time system for motion retrieval and interpretation. Pattern Recognition Lett. (2013), http://dx.doi.org/10.1016/j.patrec.2012.12.020