Available online at www.sciencedirect.com
SCIENCE DIRECT®
Graphical Models 68 (2006) 212–234
Graphical Models
www.elsevier.com/locate/gmod

Animal gaits from video: Comparative studies

Laurent Favreau *, Lionel Reveret, Christine Depraz, Marie-Paule Cani

Laboratoire GRAVIR, INRIA Rhône-Alpes, 655 avenue de l'Europe, 38334 St. Ismier CEDEX, France

Received 14 January 2005; received in revised form 26 March 2005; accepted 1 April 2005
Available online 1 June 2005

Abstract

We present a method for animating 3D models of animals from existing live video sequences such as wild life documentaries. Videos are first segmented into binary images on which principal component analysis (PCA) is applied. The time-varying coordinates of the images in the PCA space are then used to generate 3D animation. This is done through interpolation with radial basis functions of 3D pose examples associated with a small set of key-images extracted from the video. In addition to this processing pipeline, our main contributions are: an automatic method for selecting the best set of key-images for which the designer will need to provide 3D pose examples, and a simple algorithm based on PCA images to resolve 3D pose prediction ambiguities. These ambiguities are inherent to many animal gaits when only monocular view is available.

Keywords: Animation from motion/video data; Interpolation/keyframing; Intuitive interfaces for animation

1. Introduction

Traditional motion capture methods—either optical or magnetic—require some cooperation from the subject. The subject must wear markers, move in a reduced

* Corresponding author. Fax: +33 4 76 61 54 66.
E-mail address: laurent.favreau@imag.fr (L. Favreau).

1524-0703/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2005.04.002

space, and sometimes has to stay on a treadmill. The range of possible captured motions is thus very limited: capturing the high speed run of a wild animal, such as a cheetah running after his pray is totally untractable using this method. This is unfortunate since this kind of motion data would be of great interest for 3D feature films and special effects, for which fantastic animals must be animated while no source of motion is available.

The new method we propose allows the extraction of 3D cyclic motion of animals from arbitrary video sequences (we are currently using live sequences from wild life animal documentaries). State of the art techniques in computer vision for markersless 3D motion tracking are still hard to use in an animation production framework. As an alternative, we propose to use robust existing techniques in a novel pipeline: we combine principal component analysis (PCA) of images and animation by interpolation of examples to reliably generate 3D animation of animal gaits from video data. PCA of images is well suited for animal gaits since this motion is naturally cyclic and PCA will factorize similar patterns and isolate main variation in images. Our experiments show what constraints and additional processing can be used to help PCA to focus on coding variation due to motion only. Our goal is to isolate and characterize, using PCA images, minimal sets of cyclic motion and to subsequently generate the associated 3D animation. More complex 3D animation with non-uniformly cyclic motion could later be generated using recent methods in motion synthesis. We improve existing techniques with two main contributions: an automatic criterion to select key-images from video and an algorithm to resolve ambiguities in the prediction of 3D poses from 2D video.

The resulting method greatly saves effort for the animator. Traditionally for the animation of quadrupeds, the artist must make several trails to set the key-frames and 3D poses. Our method, based on PCA images, allows us to provide directly the visually salient key-images with which to associate a 3D pose. The interpolation method automatically generates a long sequence of 3D animation mimicking the rhythm of the original video.

1.1. Previous work

One of the first attempts to reconstruct animal motion from videos is Wilhelms's work [11]. Deformable contours (snakes) are used to extract the 2D motion of each limb's contour from a video sequence of a running horse. This motion is then transformed to 3D motion by matching 2D contours of limbs in the image with contours of limbs of a 3D model aligned on the image sequence. It is well known now that active contours methods are very sensitive to noise and have parameters which are difficult to tune. Wilhelms et al. [11] mention this problem and allow the user to reinitialize the active contours. This makes the method difficult to use in the general case, especially when limbs occlude each others. More generally, Gleicher et al. [10] show that current computer vision techniques for the automatic processing of videos fail to provide reliable 3D information such as stable joint angles over time. They conclude that using this kind of approach for the direct control of 3D animation at the joint level is currently not feasible.

Example-based approaches have recently been recognized as a good alternative to traditional shape modeling and animation methods. The basic idea is to interpolate between a given set of examples, 3D poses or motions, mapped from an input parameter space to 3D data. Rose et al. [4] parameterize the synthesis of new motion from motion capture data labeled with abstract parameters characterizing the style of motion. Lewis et al. [3] interpolate shapes of a bending arm from joint angle values using radial basis functions (RBF). They show that pose space mapping avoids well-known artifacts of traditional skinning methods. Sloan et al. [2] extend this formulation by combining RBF and linear regression. All these approaches interpolate between well-defined data, i.e., examples of 3D shapes or motion, labeled with user-defined abstract parameters. Pyun et al. [5] show that a similar framework can be used to animate new faces from captured facial animation data. In this case, the abstract parameters are replaced by the 3D animation data that control the way the examples are interpolated over time. Visual features extracted from 2D images can also be used as input parameters to control pose space mapping. Bregler et al. [6] capture information from existing footage of 2D cartoon animation to control the blend shapes animation of 3D characters.

1.2. Overview

Our method is an example-based approach. We test video data as possible input parameters to control animation. Live video footage is challenging to process: because it lacks contrast and resolution, automatic feature extraction is not robust, and would require heavy user intervention. We rather convert the original images into normalized, binary images, on which PCA is applied. The images' coordinates in the principal component (PC) space provides an adequate set of parameters to control the 3D motion.

When input parameters are derived from a large set of data, all example-based methods require that the user explicitly designates the examples. We propose a new and automatic criterion for selecting these examples. RBF are used to interpolate between these pose examples over time, from the sequence of parameter values extracted from the video.

Section 2 presents our general pipeline for generating 3D animation from video: it details the chain of operations that we apply to the video sequences in order to extract adequate control parameters, and the way we interpolate given 3D pose examples to generate the animation. In particular, the conversion to binary images can either be fully automatic or use simple user input such as rough strokes sketched over the images: We show that both methods provide similarly good data for applying PCA.

Section 3 presents two extra contributions. First, we present a criterion for automatically selecting the best, minimal set of key-images from the video-data. Providing such a criterion prevents the user from spending hours carefully analyzing the input motion in order to find out which images he should associate with 3D pose examples. Second, we propose a simple algorithm to resolve ambiguities in the prediction of 3D poses from 2D silhouettes.

We validate our method in Section 4, by testing our approach on synthetic data: as our results show, we achieve a precise reconstruction of existing 3D animations of animal motion from video sequences of their 2D rendering, given that the right 3D shapes were associated with the automatically selected poses.

Section 5 presents several features of our method that concern live video processing, such as the option of filtering the coordinates in PC space before applying the interpolation.

Section 6 presents our final results: wild animal motion is extracted from real life documentaries. We conclude and give directions for future work in Section 7.

2. Predicting 3D animation using PCA on images

2.1. Overview of the method

Our approach combines statistical analysis of binary images by PCA and animation by pose space mapping. The binary images can be generated by automatic segmentation. When automatic segmentation fails, we propose a sketching tool to label the video. In this case, white strokes on a black background create the binary image. PCA is then applied on the binary images, taking each image as a single observation vector. The projection coefficients of input images onto the PCs are analyzed to extract optimal examples of 3D poses to interpolate. These projection coefficients serve as input parameters to the pose space mapping interpolation and control the temporal evolution of the animation (Fig. 1).

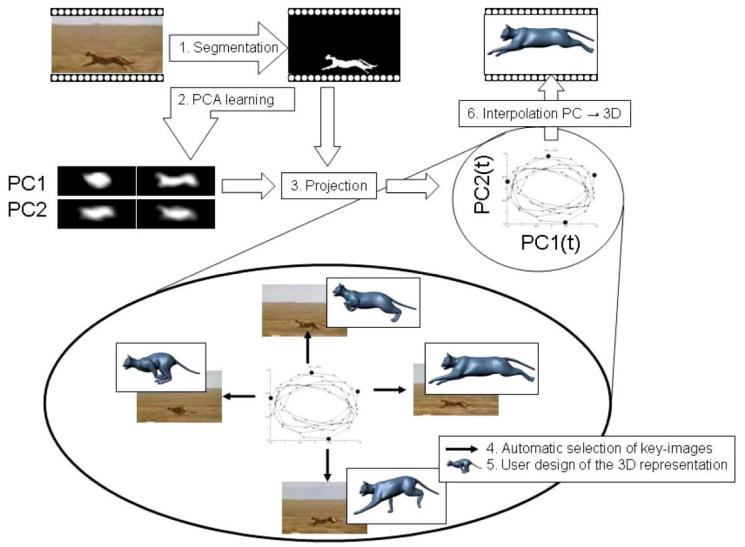

Fig. 1. Overview of the method.

2.2. Reducing variability into binary images

Using PCA directly on images would encode any variation in appearance. In addition to variation due to motion, changes in illumination, camera motion, and occlusion would be coded as well by PCA. Thus, before applying PCA, video images are segmented into binary images in order to filter such variation and isolate the foreground subject from the background. Assuming the user can provide some initial guess on the subject and background location on the first image by selecting two rectangular areas for each one on the first image of the sequence, a simple segmentation based on mixture of Gaussians can still provide accurate results (Fig. 1 and top of Fig. 2). This method is easy to implement and was sufficient for the purpose of our work on gaits generation. More elaborated techniques could be used and provide even more accurate input data to our approach [7].

When automatic segmentation fails, we propose to the user a sketching interface to label the video footage. The sketches does not need to be accurate as in Davis et al. [9], where the drawing needs to be precise enough so that the joints can be automatically recognized. In our case, the huge change in illumination and high occurrence of occlusions make impossible to claim for a careful joint to joint labeling. Instead, we rely on a raw labeling with strokes of the main features such as the spine, legs, and head. It is not required to label every joint individually if they do not appear in the image. The idea is to similarly apply a PCA on images, either generated from segmentation or resulting from sketching.

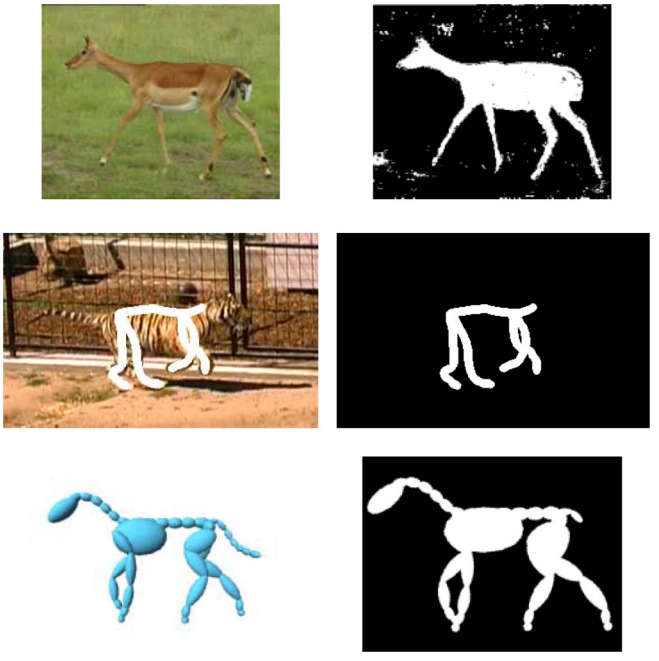

Fig. 2. Results of segmentation for our three sources of data: live video, sketching, and synthetic.

Table 1
Number of frames and image size for some tested sequences

Sequence	Number of frames	Original image	Binary image
Horse walk	100	320×240	320×240
Horse canter	100	320×240	320×240
Horse gallop	100	320×240	320×240
Cheetah run	137	192×144	90×34
Tiger run	60	720×480	448×216
Antelope walk	122	352×288	232×173
Giraffe walk	73	352×288	195×253

Once the subject is isolated from the background, a region of interest is automatically extracted around the silhouette by standard morphological analysis and detection of connected components. This process is applied to all the images in the video sequence. We keep track of the center of mass of the binary silhouette evaluated at the previous image so that the region of interest is still focussed on the correct connected component.

This step allows us to get rid of variance due to camera motion which is not relevant to the true motion of the tracked subject. Unfortunately, it also filters out the translation of the animal, which is relevant to motion. Nevertheless, we are not trying to extract the translation as an independent parameter. Instead, our aim is to capture the overall timing and predict animation by interpolation of 3D pose examples. Consequently, if a vertical translation is set in the pose examples, assuming such a motion is correlated with the rest of the visible motion in the images sequence, it will appear in the final animation. Typically, a full body extension is correlated with a flight part in the gait scenario, which should be enough to recover the vertical translation of the animal. The horizontal translation could be recovered using optical flow analysis techniques.

From this pre-process, we end up with a sequence of binary images. Table 1 shows the data specifications for our 7 test sequences in terms of number of frames, size in pixels of the original image, and size in pixel of the tracked window of the silhouette.

2.3. Principal components as input visual features

Principal components analysis is a standard method for data reduction. It consists in finding a set of principal axes, linearly spanning the variance of multidimensional data. Turk and Pentland [8] introduced one of the first implementations of PCA on images to perform face recognition (eigen-faces). In this case, each image is considered as an independent observation where all the pixel values are stacked in a single vector. Eigen-images have been widely used to reduce, classify, and recognize regular patterns from images. As a new contribution we show that PCA on images can encode variation due to motion only and can be used not only to classify shapes but also to continuously predict change in motion. We will take benefit of this property in the interpolation scheme.

Table 2
Percentage of the variance covered by each PC with respect to the total variance of the data for our 7 test sequences

Sequence	PC1	PC2	PC3	PC4
Horse walk	33.7	23.7	11.4	8.56
Horse canter	32.5	14.5	9.17	8.78
Horse gallop	31.1	19.9	11.0	8.33
Cheetah run	44.7	11.6	9.93	7.79
Tiger run	15.2	10.5	6.14	4.69
Antelope walk	21.5	12.2	8.40	6.91
Giraffe walk	42.8	15.8	11.1	5.63

PCA consists in calculating the eigenvectors and eigenvalues of the covariance matrix of the collected data. In our case, each rectangular image of the sequence is viewed as a row vector $\mathbf{i}(t)$ of all the pixels values stacked together. We gather all the n images over a sequence in a matrix $\mathbf{I}$, after having subtracted the mean image $\bar{\mathbf{i}}$

$$\bar{\mathbf{i}} = \frac{1}{n}\sum_{t=1}^{n}\mathbf{i}(t), \tag{1}$$

$$\mathbf{I} = [(\mathbf{i}(t_1) - \bar{\mathbf{i}})^t, \ldots, (\mathbf{i}(t_n) - \bar{\mathbf{i}})^t]^t. \tag{2}$$

The PCA is then formulated as:

$$\frac{1}{n}\mathbf{I}^t\mathbf{I}\mathbf{E} = \mathbf{E}\mathbf{D}, \tag{3}$$

$$\mathbf{E}^t\mathbf{E} = \mathbf{1}. \tag{4}$$

Finally, we take as input vector of the animation the projection coefficients onto the PCs stacked as column vectors in matrix $\mathbf{E}$ and normalized by the square roots of the eigenvalues stacked in the diagonal matrix $\mathbf{D}$

$$\mathbf{p}(t) = (\mathbf{i}(t) - \bar{\mathbf{i}})\mathbf{E}\sqrt{\mathbf{D}^{-1}}. \tag{5}$$

Table 2 recapitulates the results of PCA in terms of part of the variance covered by each PC with respect to the total variance of the data for our 7 test sequences.

2.4. Interpolation

Our goal is to generate animation parameters (position and joint angles) $\mathbf{x}(t)$ from the values of projection coefficients $\mathbf{p}(t)$ computed from PCA. We use interpolation of m 3D pose examples $[\mathbf{x}(t_i)]_{i=1\ldots m}$, corresponding to m images in the video sequence for which we know the projection coefficients $[\mathbf{p}(t_i)]_{i=1\ldots m}$ at time t_i in the video sequence. For clarity, we note $\mathbf{x_i}$ and $\mathbf{p_i}$ for, respectively, $\mathbf{x}(t_i)$ and $\mathbf{p}(t_i)$.

Three main methods for scattered data interpolation are used in example-based method approaches: linear interpolation [6], radial basis function [3] or a combination of both [2]. In the latter case, linear interpolation allows us to cope with cases where input data could be sparse and require a stable behavior for extrapolation. In our case, input data are the results of PCA and as such are already linearly compact. For this reason, RBF were enough to deal with our case. This general interpolation scheme is formulated as linear combination of distance functions $h(r)$ (the RBF) from m interpolation points in the input space

$$\mathbf{x}(\mathbf{p}) = \sum_{k=1}^{m} h(\|\mathbf{p} - \mathbf{p_k}\|)\mathbf{a_k}, \tag{6}$$

where $\mathbf{p}$ is the input vector and $\mathbf{x}$ the predicted vector. $h(r)$ are the RBF. $\mathbf{a_k}$ are unknown vectors to be determined. If the RBF are stacked into a single vector $\mathbf{h}(\mathbf{p})$ and the unknown coefficients $\mathbf{a_k}$ as row vectors into a matrix $\mathbf{A}$, we have the formulation:

$$\mathbf{x}(\mathbf{p}) = \mathbf{h}(\mathbf{p})\mathbf{A}, \tag{7}$$

$$\mathbf{h}(\mathbf{p}) = [h(\|\mathbf{p} - \mathbf{p_1}\|) \dots h(\|\mathbf{p} - \mathbf{p_m}\|)]. \tag{8}$$

$$\mathbf{A} = [\mathbf{a_1^t}, \dots, \mathbf{a_m^t}]^t. \tag{9}$$

As interpolation points, we use m 3D pose examples $\mathbf{x_i}$ and the the values of the m associated input parameters $\mathbf{p_i}$ of the corresponding key-image. $\mathbf{A}$ has to be solved so that $\|\mathbf{x}(\mathbf{p_i}) - \mathbf{x_i}\|$ is minimal. This minimization in a least square sense leads to the standard pseudo-inverse solution

$$\mathbf{A} = (\mathbf{H^t H})^{-1}\mathbf{H^t X}, \tag{10}$$

where,

$$\mathbf{X} = [\mathbf{x_1^t}, \dots, \mathbf{x_n^t}]^t, \tag{11}$$

$$\mathbf{H} = [\mathbf{h}(\mathbf{p_1})^t, \dots, \mathbf{h}(\mathbf{p_m})^t]^t. \tag{12}$$

The final formulation is then

$$\mathbf{x}(\mathbf{p}) = \mathbf{h}(\mathbf{p})(\mathbf{H^t H})^{-1}\mathbf{H^t X}. \tag{13}$$

Note that this can be re-formulated exactly as an interpolation of the $\mathbf{x_i}$

$$\mathbf{x}(\mathbf{p}) = \sum_{i=1}^{m} w_i(\mathbf{p})\mathbf{x_i} \tag{14}$$

by extracting the matrix $\mathbf{h}(\mathbf{p})\ (\mathbf{H^t H})^{-1}\mathbf{H^t}$. Alexa et al. [1] compress and animate 3D sequences from PCs learnt on a fully available sequence of 3D data. In our case, PCs are learnt from image space and animation of 3D data is controlled by interpolation.

The value of $\sum_{i=1}^{m} w_i(\mathbf{p})$ should stay close to 1 to guarantee that a point $\mathbf{p}$ in the input space is close enough to interpolation points and any $w_i(\mathbf{p})$ should be close to $[0,1]$ so that $\mathbf{x}(\mathbf{p})$ stays close to the convex hull of the 3D pose examples.

For the choice of $h(r)$, a common practice is to use a gaussian function for its $\mathscr{C}^{\infty}$ continuity properties:

$$h(r) = \mathrm{e}^{-\alpha r^2}. \tag{15}$$

The parameter α in Eq. (15) needs to be determined. Statistically, projections on PC are homogeneous with standard deviation. This means data will be spread approximatively in every projected direction over the same interval $[-1;+1]$—varying according to the nature of distribution. Assuming interpolation points are well spread, we take a value of 2 as a raw estimate of the distance between interpolation points. At midpoint between two interpolation points, we expect an equal influence. This can be translated into the fact that we want $h(r)$ to be equal to 0.5 when $r = 1$. This leads to an estimate of $\alpha = \ln 2$.

All previous works on example-based animation rely on the user to decide where 3D pose examples need to be provided [3,2,5]. In our case, this would mean selecting key-images among thousands of a video sequence. Given the number of key-images to provide, we present an automatic criterion to select these ones within the video sequence.

3. Key-images selection

3.1. Criterion for automatic selection

We want smooth mapping between the image space and the animation space as we based all our timing control on images. A small change in the image space must produce a small change in the animation space. We notice that the interpolation scheme on RBF involves the inversion of a matrix $\mathbf{H}^t\mathbf{H}$, build from the interpolation points, as it has been shown in the previous section. Consequently, to ensure a stable interpolation, and thus a smooth animation, we select key-images over the sequence which minimize the condition number of the matrix $\mathbf{H}^t\mathbf{H}$ to invert. The condition number is evaluated as the ratio of the largest singular value of the matrix to the smallest. Large condition numbers indicate a nearly singular matrix.

This criterion is generally applicable to any example-based method. It can be used to select any number of input examples, key-images in our case, when they have to be chosen within a large set of data. The singular values of $\mathbf{H}^t\mathbf{H}$ are the squared singular values of $\mathbf{H}$. This matrix measures the respective distances between the interpolation points. Intuitively, the criterion on condition number thus selects input examples which are equally spread within the data set. Having all the singular values closed to each other means they equally sample every direction of the input space.

In practice, as will be shown in Sections 4 and 5, only few PCs and few 3D pose examples are needed. This allows us to implement a simple combinatory approach for the condition number criterion: for each sequence of n frames, given a number

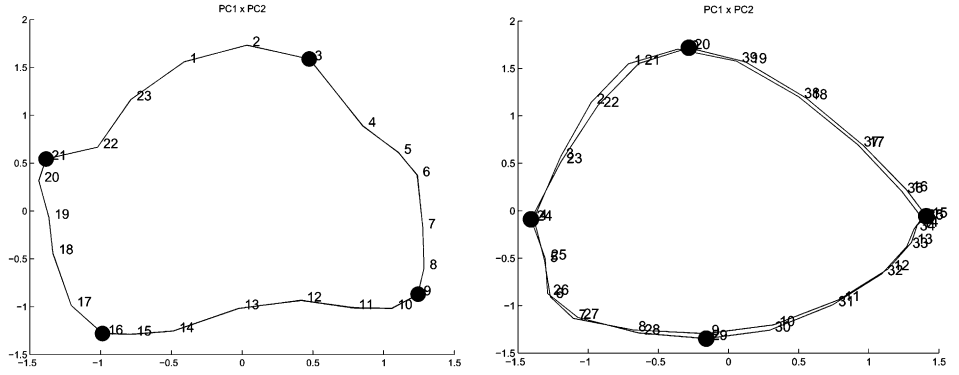

Fig. 3. PC1(t) × PC2(t): PC projections across time for two synthetic sequences of horse: canter and walk. Frames are numbered for one cycle. Circles are selected examples by the condition number criterion.

of c PCs to consider and a number m of key-images to select, we evaluate the condition number of all the $\binom{n}{m}$ matrices $\mathbf{H}^t\mathbf{H}$. The $\mathbf{H}^t\mathbf{H}$ matrix is square and its dimension is m. We keep the set of m key-images within the whole sequence providing the $\mathbf{H}^t\mathbf{H}$ matrix having the smallest condition number. Keeping only a few PCs makes the computation fast. We tested with up to 5 PCs, but experiments showed that 2 were enough as will be detailed in following sections.

As an example, for the prediction of three sequences of animation from synthetic images, we plot the projections on the two first components as a 2D graph and search for the best four examples based on the condition number criterion (next section will show that 2 PCs and 4 key-images is the best configuration for the prediction of this specific gait). In this case, the condition number criterion has the particularity to select examples at approximatively the extreme variation of the two first PCA projections (Fig. 3).

Intuitively, the more key-images are given, the better the interpolation will be. As any key-image will require the user to provide a 3D pose example, a compromise must be found. The question of the number of key-images needs to be examined on a case-to-case basis. From our experiments on animal gaits, we observed good results with 4 pose examples for the running cases and 8 pose examples for the walking cases.

3.2. Resolving 2D ambiguities with switching models

At this point, our method predicts 3D motion from silhouette images. It results in a unique 3D pose for each distinct input image. In some cases however, two different 3D poses can lead to very similar silhouettes when viewed from the side (Fig. 4). This is very common in motions that consist in a succession of two symmetric phases, such as quadrupeds walking. The motion predicted by RBF still provides good results but only on one half of the period of the original gait.

To avoid this problem, it is first necessary to provide two different 3D pose examples for each of the ambiguous silhouettes of the key-images and second to build a

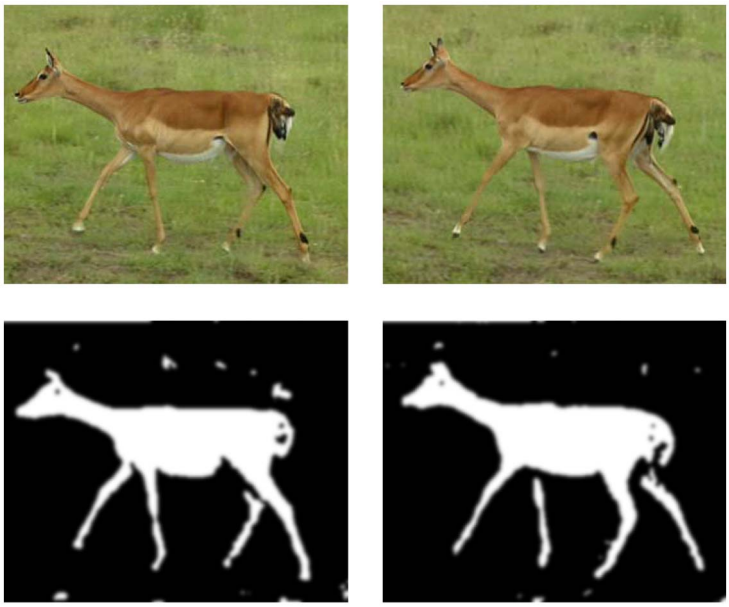

Fig. 4. Two different poses can produce similar silhouettes.

method to correctly choose between these two poses during the generation of the 3D animation.

We solve for the first problem with a simple algorithm:

(1) We select m initial key-images with the standard method and build the animation by associating 3D poses to key-image and using RBF prediction. If the user acknowledges issues about pose ambiguities, we go to step 2.
(2) For each key-image, we automatically search for its closest image in the PCA space and propose it to the user as the alternative pose for this silhouette. We constrain this image to be at least 3 frames further than the initial key-image to guarantee that we are in another half-cycle.
(3) When the user validates the proposed image as the key-image corresponding to the same silhouette but at a different pose, we ask the designer to provide the appropriate 3D pose example.
(4) We iterate until each of the m initial key-images of step 1 has its associated key-image corresponding to the opposite pose.

At the end of this process, we have doubled the number of m initial key-images and corresponding 3D pose examples. Fig. 5 provides an example for this algorithm with $m = 4$ initial key-images. We are now able to generate a full cycle of motion. To generate animation, the same method of prediction from images is kept, but instead of keeping the same m 3D pose examples, we switch between q sets of m 3D pose examples as time evolves, taken from the $2m$ 3D pose examples selected by the

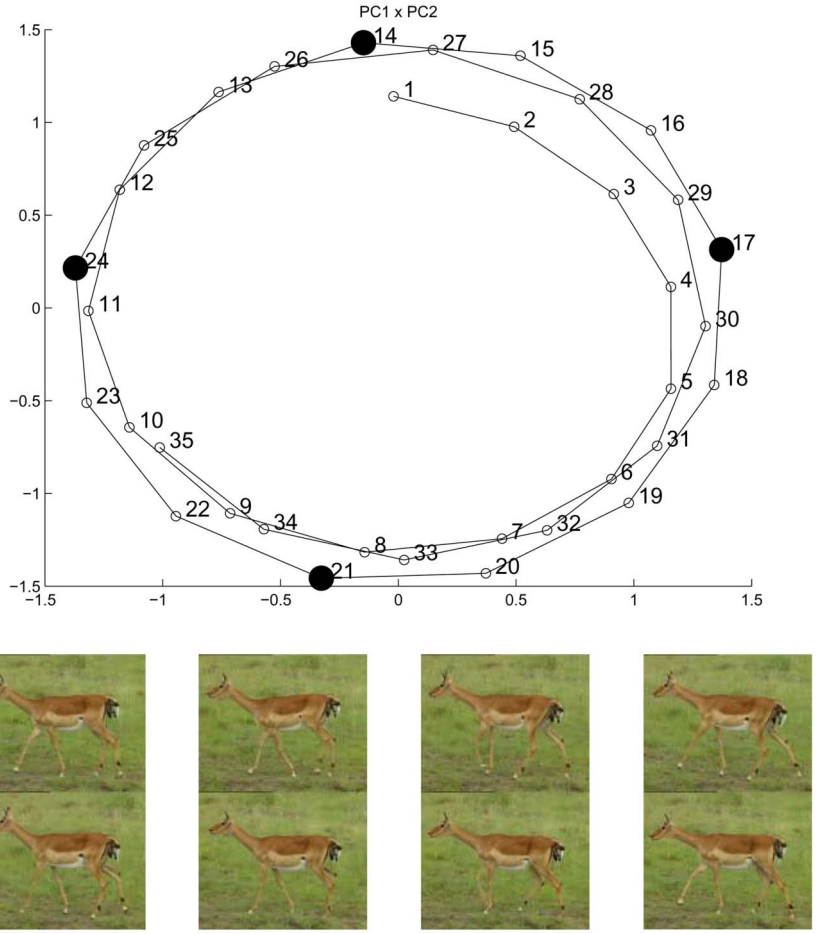

Fig. 5. Selecting more key-images to resolve ambiguities in the PCA space. Images 14, 17, 2,1 and 24 are selected as initial key-images. In a second step, 27, 4, 8, and 11 are selected as candidates for ambiguous pose, based on their coordinates in the PCA space. First row: frames 14, 17, 21 24; second row: frames 27, 4, 8, 11.

previous algorithm. We call these q sets the switching models. The prediction of animation parameters is extended as follows:

$$\mathbf{x}(\mathbf{p}_t) = \sum_{k=1}^{q} w_{\sigma_k(s_t)}(\mathbf{p}_t)\mathbf{x}_{\sigma_k(s_t)}, \tag{16}$$

$$s_t = \text{switch}(\mathbf{p}_t, s_{t-1}), \tag{17}$$

where s_t represents a phase state index in term of switching model, $\mathbf{x}_i$ the $2m$ pose examples, and w_i the model weight given the input image and the current phase state. The function switch$(\mathbf{p}, s)$ indicates which set of m pose examples needs to be used in

the prediction algorithm. It is a discrete state variable, incremented each time we detect that we have reached the last 2D silhouette within a set of m key-images. The change of silhouette in key-images is easily detected by a distance function in the PCA space.

Switching between $q = 2$ models of m pose examples would allow to explore the whole animation space, as we have just doubled the number of initial m pose examples. However, the transition between two models turned out to be unstable. We solved this problem by introducing overlaps between intermediate models. The use of $q = 4$ switching models allows smooth transitions. In practice, we use $m = 4$ pose examples to describe half of a cycle motion. The function $\sigma_k(s)$ gives the 4 indices of pose examples at 4 state positions with 2 overlapping pose examples between two consecutive steps:

$$\sigma_i(j) = \sigma_{ij}, \tag{18}$$

$$(\sigma_{ij}) = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 5 & 6 \\ 5 & 6 & 7 & 8 \\ 7 & 8 & 1 & 2 \end{pmatrix}. \tag{19}$$

4. Validation on synthetic images

We have validated our method by taking as input images the rendering of a skeleton horse 3D model. Joints are represented as ellipsoids (Fig. 2). The choice of such a model was made to get rid of any bias that a skinning algorithm would introduce. We report results for three sequences: gallop, canter, and walk. By using synthetic images, we can still test the full pipeline as described in Section 2. In addition, we can compare with the original animation parameters. This evaluation gave us hints on the number of PCs and examples that should be used.

We have exhaustively evaluated the results using an increasing number of key-images (starting at two) and an increasing number of PCs (starting at one). Given the number of key-images to select from the video, the condition number criterion tells what key-images to select. The corresponding 3D pose examples are provided by the original animation sequence. We evaluate the results by computing the mean (and standard deviation) of the absolute difference over all the joint angles for the main rotation axes (perpendicular to the image plane, 36 angles in the case of our model) between original and predicted values.

From Fig. 6, we immediately observe that adding the third and following components introduce noise. This suggests that they are coding information not relevant to the gait motion. As for the number of keys, as expected, the more examples are provided, the smaller the error. A good compromise arises on 4. Adding a fifth key-image decreases the mean error less than a degree. Two or three pose examples, although optimally selected by the condition number criterion, are not enough. With

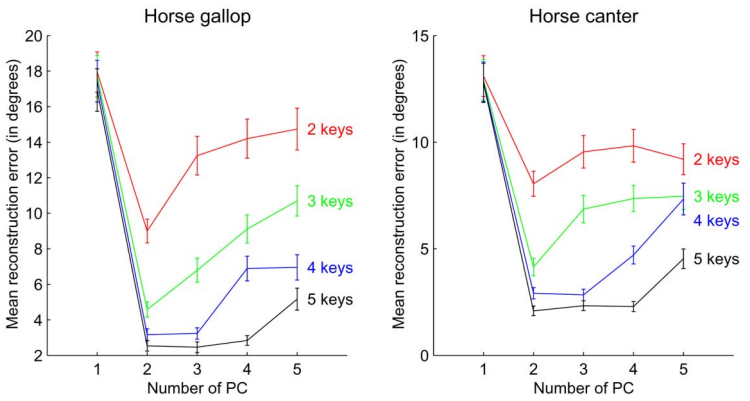

Fig. 6. Evaluation for the gallop and canter sequences. Each curve corresponds to the mean error with a fixed number of examples (2–5), with respect to the number of components used as input parameters.

four 3D pose examples and two PCs, we obtain a very good match between the original animation and the predicted animation from images.

5. Processing live video sequence

We discuss now how to apply our approach on live video images, sometimes emphasized by a rough sketch as mentioned in Section 2. As detailed below, strictly focussing on the first two PCs and applying a band-pass filter to the PC trajectories along time enables us to achieve as good visual results as with the synthetic data.

5.1. Restricting to the two first PCs

In the case of the synthetic examples, the first two components exhibit consistently interpretable behavior. For example, for the gallop of the horse, The first component (PC1) encodes a variation between a flight phase, when none of the feet touch the ground, and a grouped phase. The second component (PC2) corresponds to an opposition between a rising phase, when the horse jumps off the ground, and a descending phase, when the horse front feet hit the ground (Fig. 7).

Numerical evaluation on synthetic images have suggested that the two first PCs are optimal to achieve good prediction. Image segmentation and sketching by hand will naturally introduce more noise in PC curves, making PC unstable and poorly reliable to predict relevant motion. We decide from these observations that the only two first PCs should be kept for live video and sketched images.

We confirm this hypothesis on the cheetah sequence where similar interpretation as that for the horse gallop can be made on the two first PCs (Fig. 8).

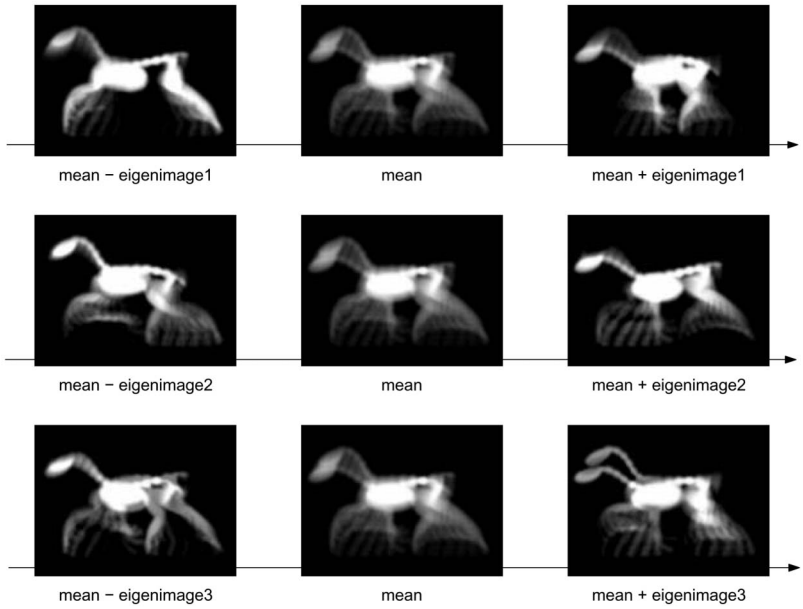

Fig. 7. Variation encoded by the first three eigenvectors for the horse gallop sequence. Middle column is the mean shape, each row corresponds to the variation along an eigenvector.

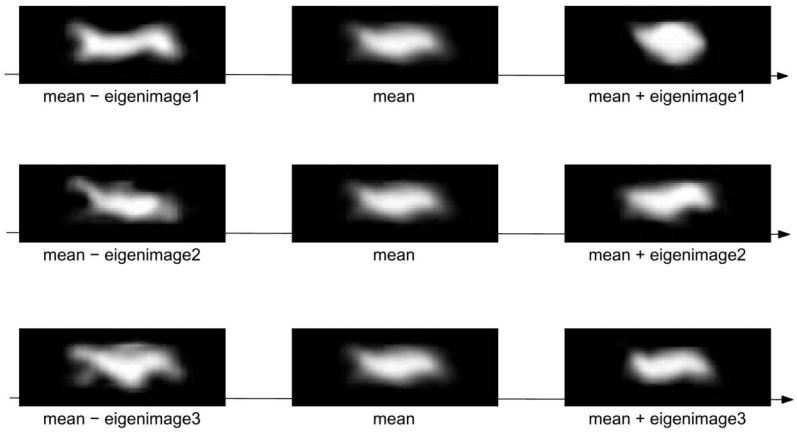

Fig. 8. Variation encoded by the first three eigenvectors for the cheetah sequence.

5.2. Spectrum regularization

On the synthetic sequence, the time variation of the projections on the first two components shows a shift in phase of one fourth of the cycle period, corresponding to an alternation of jump, flight, landing, and grouping legs. This produces the

circular pattern shown on Fig. 3. This configuration has been reproduced on every examples of synthetic images. Consequently, we adopt the configuration of projections on PC1 and PC2 in a circular pattern as a characterization of a video sequence to be usable with our method. In the Fourier domain, this configuration corresponds to peaks at the same location for projections on PC1 and PC2, and a phase difference of approximatively $\pi/2$.

Live video can thus be diagnosed as not usable by our method if it does not have projections on PC1 and PC2 staying within a certain bandwidth that we automatically estimate. The first component encodes most of the variance and is considered to be representative of the fundamental cyclic variation. Its spectrum will thus be centered around a frequency corresponding to the period of the cycle. All our experiments confirm this hypothesis. From a Fourier transform, we get the frequency of maximum amplitude. We fit a peak function centered on this frequency of the form

$$\text{peak}(f) = \frac{1}{1 + (\frac{f - f_0}{f_b - f_0})^2}, \tag{20}$$

f_0 is set at the frequency of maximum amplitude and f_b is set so that it corresponds to the closest frequency to f_0 having an amplitude of half of the maximum. We deduce a bandwidth of $[-3(f_b - f_0); +3(f_b - f_0)]$, corresponding to end points at 10% of the maximum amplitude (Fig. 9).

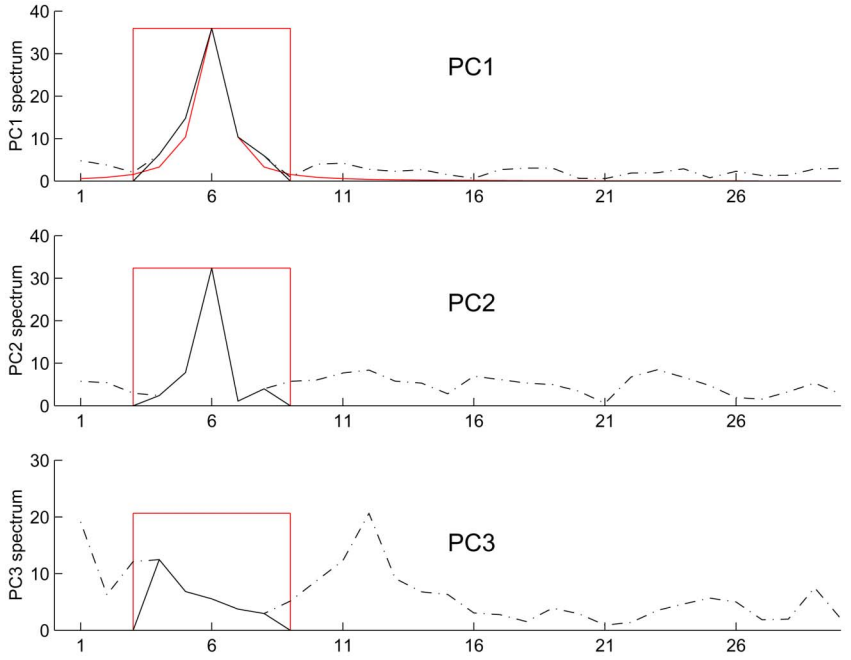

Fig. 9. Spectrum of the three first PCs of the cheetah spectrum. The peak function is fitted to PC1, and a rectangular window is deduced.

Table 3
Correlation coefficient between the filter and unfiltered signal for the five first components of some test sequences

PC:	PC1	PC2	PC3	PC4	PC5
Horse gallop	.90	.92	.01	<.01	<.01
Horse canter	.94	.89	.08	<.01	.01
Horse walk	.99	.94	<.01	.05	<.01
Cheetah run	.91	.61	.14	.21	.25
Antelope walk	.78	.91	.14	.10	.06
Tiger run	.87	.72	.26	.18	.23
Giraffe walk	.92	.82	.24	.19	.28

The second component is filtered by the same band-pass filter. What is expected is that the second component shows a similar peak, creating the circular pattern. We can evaluate how this hypothesis is respected by comparing how much the reconstructed signal after filtering, matches the original signal. For this, we compute the correlation coefficient between the original PC signal and the filtered PC signal. We have observed that for our test sequences, the two first components shared the same peak ($r \geqslant .6$), while the following components do not ($r \leqslant .3$) (Table 3). This provide a numerical criterion to evaluate if our method can be successfully applied to a video sequence as we have presented it.

Fig. 9 shows the results in the Fourier domain for the cheetah sequence which conforms to the criterion. When the live video sequence fails to conform to this criterion, we suggest using the sketch approach. If the sketch approach still fails to meet the criterion, we diagnose that our method cannot work on the analyzed video.

5.3. Non-cyclic motion factorization

In our early experiments [12], a sequence was considered as usable with our method if its two first PCs shared a single similar peak in the frequency domain. In some cases however, we report that the first component is not purely related to the animal locomotion. For example, one of our test sequences represents a rhinoceros walking. While walking, the rhinoceros slowly lowers its head throughout the sequence, to end picking up food on the ground. Fig. 10 shows the variation encoded by the first three eigenvectors of this sequence. The first component encodes an alternance of lifting and lowering the head. This information is spread all over the sequence and is not relevant with the animal locomotion. It does contain only a purely cyclic evolution (Fig. 11). Thus, this component cannot be used by our method. The second and third component, however, share the same peak in the frequency domain and exhibit a circular pattern. They encode an alternance of grouping and spreading the rear legs and front legs, respectively. Our method can be successfully applied using these components, which is consistent with our early findings. In this case, our method has been robust to image variation that does not correspond to locomotion. This unwanted information has been filtered out.

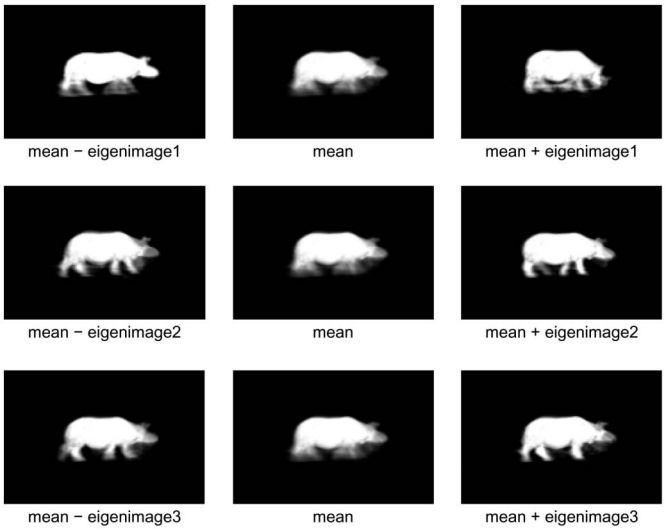

Fig. 10. Variation encoded by the first three eigenvectors of the rhinoceros walk sequence.

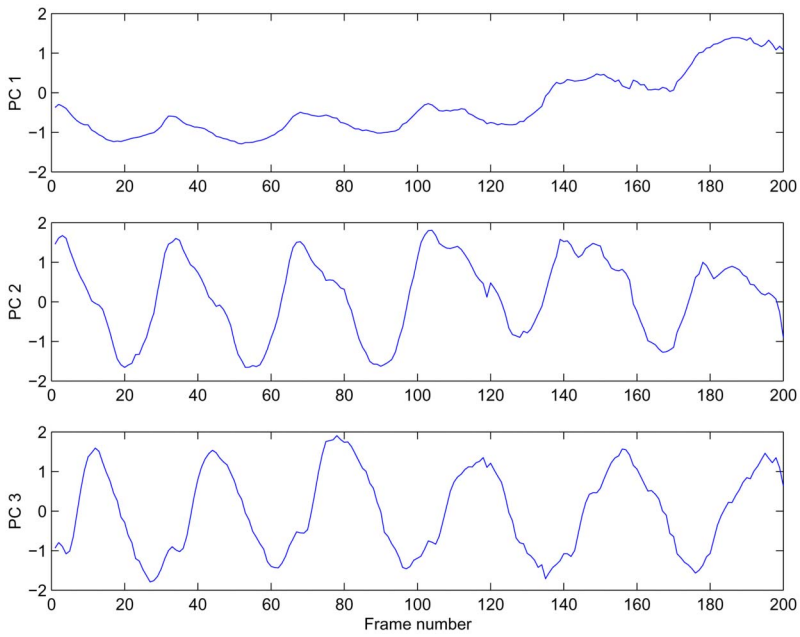

Fig. 11. Evolution of first three PCs of the rhinoceros walk sequence.

6. Results

6.1. Keys-images selection: comparison between different animals

Fig. 12 shows the evolution of the weights of the 3D pose examples for the cheetah sequence and for the rhinoceros sequence. We have an exact interpolation at these pose examples. For the rest of the sequence, we observe a correct generalization, the influence of each pose example appears at a right pace in a coherent order. Note that the sum of weights stays close to one, guaranteeing that the input parameters are always close to an interpolation point. The weights are sometimes outside of the range of $[0, 1]$ as they are not constrained in the RBF formulation. This lets the

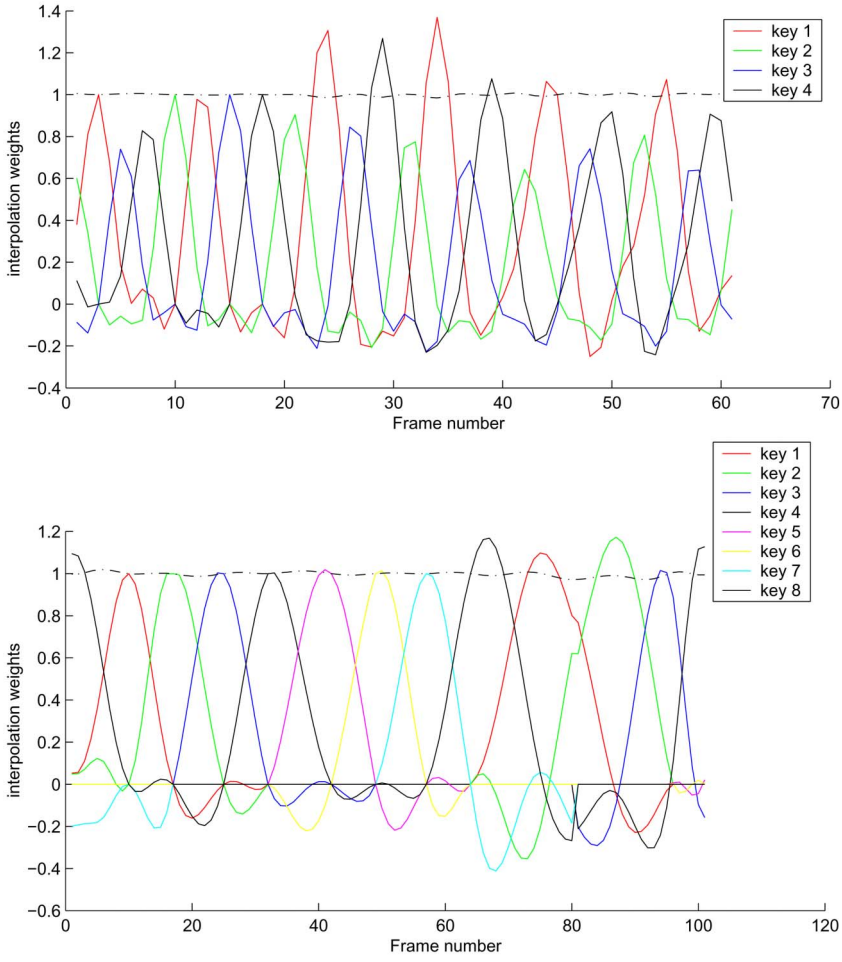

Fig. 12. Evolution of the weights of the four examples for the cheetah sequence (top), and of the eight examples for the rhinoceros sequence (bottom). The dashed line is the sum of weights.

resulting pose leave the convex hull of the pose examples. This flexibility allows some extrapolation in 3D space introduced by image variations along the sequence. A control could be easily added to maintain these weights within a safe range in order to avoid the generation of strange pose, too far outside of the convex hull of the pose examples. So far however, none of the tested sequences required this control.

We applied our method to various species of quadrupeds, ranging from light ones such as the cat to the heaviest ones such as the elephant. Heavy animals have slower gaits. This can be seen on Fig. 12, where the number of frames between two successive peaks in the interpolation weights are greater for the rhinoceros than for the cheetah. However, the selected key-images are very similar between light and heavy animals.

Fig. 13 gathers the final results about key-images selection. It shows the automatically selected key-images and the associated 3D poses provided by the artist.

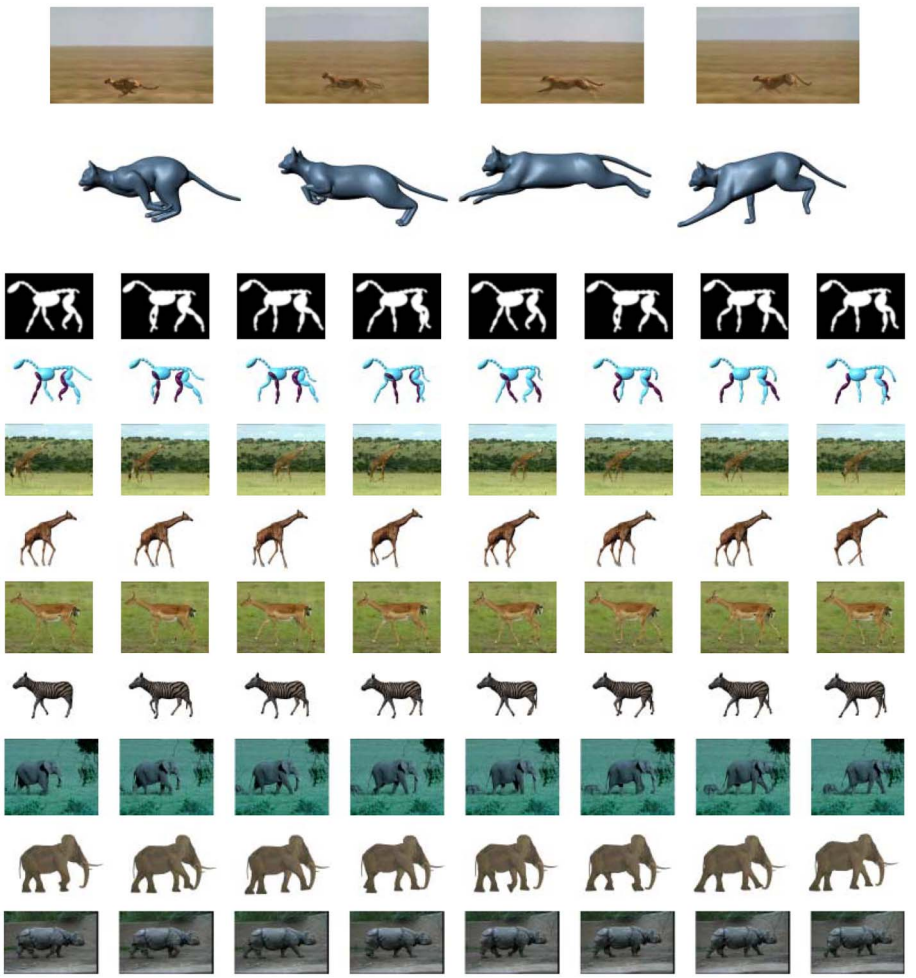

Fig. 13. Selection of key-images from video sequences.

6.2. Key-images selection: comparison between live images and sketched images

We tested our method on two horse walk sequences: a live video and a synthetic sequence, and compared the results. The key-images selected in both cases are very similar (Fig. 14). Note that while the appearance of the synthetic horse is not realistic, its motion is well captured. This is due to the fact that our method relies on the 2D dynamics of a video sequence to capture the 3D dynamics of an animal gait, but does not require realism at the image level.

6.3. Key-images selection: comparison between live images and synthetic images

To validate the use of sketching, we chose an elephant walk sequence where segmentation could be successfully applied. We sketched the sequence and compared the results obtained using the two silhouette extraction methods: image segmentation and sketching. The key-images selected in both cases are slightly different (Table 4), but correspond to very similar poses (Fig. 15). This shows that sketching is a good alternative to automatic segmentation. It does not capture the exact shape of the animal, but correctly encodes the animal gait.

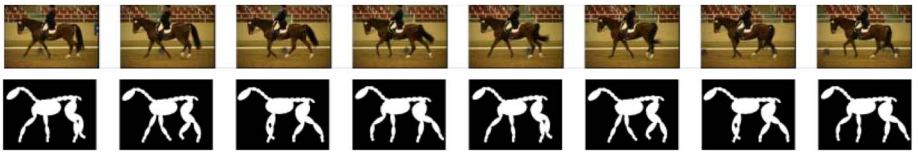

Fig. 14. Keys-images selected for the real horse walk sequence (top), and for the synthetic horse walk sequence (bottom).

Table 4
Indices of the key-images selected for elephant walk sequence, using sketching and image segmentation

Sketch	6	12	18	24	30	36	42	48
Segmentation	8	13	18	24	32	37	42	48

Fig. 15. Key-images selected for the elephant walk sequence, using sketching (top), and image segmentation (bottom).

7. Conclusion

When traditional motion capture of non-cooperative subjects such as wild animals is not feasible, live video footage can still provide significant information on motion. This paper has proposed novel and robust tools to analyze video data and make it applicable to the generation of 3D motion. We rely on PCA to extract parameters from binary version of the input images. As our result show, a small number of parameters is sufficient for cyclic animal gaits: using the two PCs already gives good results. We provide a criterion for selecting the best set of key-images from the video. In our application, the selected poses can easily be interpreted in terms of extremal images in the 2D PC space.

Our work shows that PCA can be applied onto a sequence of 2D images to control 3D motion. PCA on images helps to give a quantification of the significant changes in the appearance of the video. The RBF interpolation of pose examples aims at transposing the pace of video changes into the animation domain. The automatic selection of examples helps to focus the effort of the designer on the most important key-frames.

Our method could benefit from using more advanced video analysis techniques. For instance, it is very sensitive to camera rotation or out-of-plane translation. These are non-gait-related motions that may be encoded by the PCs. Thus, camera compensation methods could help widening the range of usable video.

As a future work, we are planing to explore non-uniformly cyclic motions such as transitions between gaits. The addition of physically based constraints could help animating non-cyclic parts of the motion. Non-linear dimension reduction methods could also used for a better visualization and parameterization of complex motions.

We are also studying how to re-use existing PCA basis and its 3D associated poses to automatically analyze a new video sequence thanks to morphological adaptation in the image space.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.gmod.2005.04.002.

References

[1] M. Alexa, W. Müller, Representing animation by principal components, in: Proc. EUROGRAPHICS'00, 2000.
[2] P.-P.J. Sloan, C.F. Rose III, M.F. Cohen, Shape by example, in: Proc. I3D'01, 2001.
[3] J.P. Lewis, M. Cordner, N. Fong, Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation, in: Proc. SIGGRAPH'00, 2000, pp. 165–172.
[4] C. Rose, B. Bodenheimer, M.F. Cohen, Verbs and adverbs: multidimensional motion interpolation using radial basis functions, IEEE Comput. Graph. Appl. 18 (5) (1998) 32–40.

[5] H. Pyun, Y. Kim, W. Chae, H.W. Kang, S.Y. Shin, An example-based approach for facial expression cloning, in: Proc. EG/SIGGRAPH Symposium on Computer Animation, SCA'03, 2003, pp. 167–176.
[6] C. Bregler, L. Loeb, E. Chuang, H. Deshpande, Turning to the masters: motion capturing cartoons, in: Proc. SIGGRAPH'02, 2002.
[7] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (1997).
[8] M. Turk, A. Pentland, Eigen faces for recognition, J. Cognit. Neurosci. 3 (1) (1991).
[9] J. Davis, M. Agrawala, E. Chuang, Z. Popović, D. Salesin, A sketching interface for articulated figure animation, in: Proc. EG/SIGGRAPH Symposium on Computer Animation, SCA'03, 2003.
[10] M. Gleicher, N. Ferrier, Evaluating video-based motion capture, in: Proc. of Computer Animation, CA'02, 2002.
[11] J. Wilhelms, A. Van Gelder, Combining vision and computer graphics for video motion capture, Vis. Comput. 19 (6) (2003).
[12] L. Favreau, L. Reveret, C. Depraz, M.-P. Cani, Animal gaits from video, in: Proc. EG/SIGGRAPH Symposium on Computer Animation, SCA'04, 2004.