Simplified training for gesture recognition

Romain Faugeroux*†‡
*LIX, École Polytechnique.
Paris, France.

Thales Vieira†, Dimas Martinez†
†Institute of Mathematics, UFAL.
Maceió, Brazil.

Thomas Lewiner‡
‡Department of Mathematics, PUC-Rio.
Rio de Janeiro, Brazil.

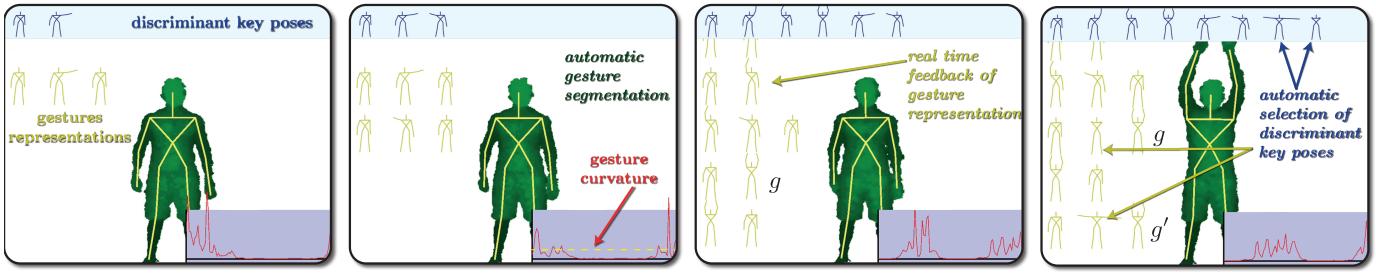

Fig. 1. Interface providing real-time feedback to the user during the training phase (left). Gestures are automatically segmented using the gesture curvature (middle left), avoiding the requirement to start and end at a key pose, or to interrupt the recording. A minimal set of key poses is automatically extracted to represent the gestures unambiguously (middle right), eventually adding discriminant key poses to a previous gesture g to distinguish it from the newly recorded one g' (right). This avoids training the key poses prior to training the gestures.

***Abstract*—Since gesture is a fundamental form of human communication, its recognition by a computer generates a strong interest for many applications such as natural user interface and gaming. The popularization of real-time depth sensors brings such applications to the public at large. However, familiar gestures are culture-specific, and their automatic recognition must therefore result from a machine learning process. In particular this requires either teaching the user how to communicate with the machine, such as for popular mobile devices or gaming consoles, or tailoring the application to a specific public. The latter option serves a large number of applications such as sport monitoring, virtual reality or surveillance — although it requires a usually tedious training.**

This work intends to simplify the training required by gesture recognition methods. While the traditional procedure uses a set of key poses, which must be defined and trained, prior to a set of gestures that must also be defined and trained, we propose to automatically deduce the set of key poses from the gesture training. We represent a record of gestures as a curve in high dimension and robustly segment it according to the curvature of that curve. Then we use an asymmetric Hausdorff distance between gestures to define a discriminant key pose as the most distant pose between gestures. This further allows to dynamically group gestures by similarity. The training only requires the user to perform the gestures and eventually refine the gesture labeling. The generated set of key poses and gestures then fits in previous human action recognition algorithms. Furthermore, this semi-supervised learning allows re-using a previous training to extend the set of gestures the computer should be able to recognize. Experimental results show that the automatically generated discriminant key poses lead to similar recognition accuracy as previous work.

***Keywords*-Gesture recognition ; Machine learning ; Training ; Pose identification ; Depth sensors ; 3d motion ; Natural user interface ;**

I. Introduction

Gestures constitute the earliest form of human communication, and their interpretation by mathematical algorithms puts gesture recognition at the heart of many applications, ranging from natural user interface to immersive games, emotion identification to surveillance, assistive robotics to sports monitoring, and more! The large availability of sensors motivates the visual computing community to push those applications to a very wide public: cheap depth sensors such as the Kinect platform are already capable of capturing depth maps and estimating a human skeleton at 30 frames per second [1].

However, most familiar gestures (the *emblems*) have a very culture-specific interpretation [2], which means that a gesture recognition system requires a learning approach [3]: either teaching the user how to use the system, or letting the computer learn the gestures from a specific group of users. The first approach is widely used for a restricted set of gestures in touch screen devices or game consoles. The second approach allows for a larger set of gestures, but requires a machine learning system, which entails recording gesture performances as (possibly dynamic) references for automatic action interpretation. This recording is part of the so-called *training* phase, which is often tedious and specific to a class of application. This work aims at simplifying the training until the user almost only needs to perform the reference gestures, in a way that the result can be used and re-used for gesture recognition.

Related Work: Gesture learning methods can be classified according to their representation [4]. Local representations use features extracted from the gestures, such as *key poses* [5] or *spatiotemporal interest points* [6,7], and build gestures as

a sequence of such features. On the one hand, this requires an additional training phase for the machine to learn the features, prior to training the gesture itself. On the other hand, the final recognition algorithm only need to compare a limited set of features, such as a few positions [8–10], angles [11–14] or a general bag of features [15–19], leading to extremely fast recognition which is suitable for interfacing with games or other resource-expensive processes.

Global representations, such as space-time accumulations [5,20–22] or templates [23,24], only need to train the gestures, but generally involves complex algorithm and requires more resource for the final recognition algorithm [25]. We aim at the best of both worlds: using a local representation to gain on speed for the final recognition but skipping the training of features. In this work, we use skeletons for the local representation since they are easily available from the Kinect platform, although the method proposed here easily adapts to other local representations.

Contributions: We propose here to automatically segment a record of gestures and extract a suitable set of key poses from a set of gestures (Fig. 1). A gesture is a sequence of poses and we first represent each pose by the position in $\mathbb{R}^3$ of its 9 joints, so that a record of gestures becomes a curve in $\mathbb{R}^{27}$. We derive a robust transition / neutral pose descriptor based on the curvature of that high-dimensional curve, automatically segmenting the record in gestures. We then use an invariant metric in the pose space, and look for *discriminant key poses*, *i.e.* a pose in one gesture that is the most distant to other gestures and previous discriminant key poses. For example, if two gestures have the same initial and final poses, two key poses, each discriminant for a gesture in relation to the other, are sufficient to distinguish the two gestures. When two recorded gestures are close in our metric, we label them as the same gesture. The gesture recognition can then be performed efficiently from an existing key pose based method using only the discriminant key poses.

This approach only requires that the user performs the gestures in a single record, and eventually refine the gesture identification in a semiautomatic manner. This minimal interaction leads to a simple training phase, and the efficiency of the algorithm allows for a real time feedback during the training. Moreover, as opposed to usual recognition methods relying on a pre-defined set of key poses, this allows to extend or re-use already trained gestures by simply concatenating the set of gestures, and our algorithm generates a minimal set of discriminant key poses from the extended set.

II. Technical Overview

Our method consists of three steps: an automatic gesture segmentation technique to extract individual gestures from a recording of gestures (Sec. III); a greedy algorithm to compute a compact, unambiguous key pose representation for each gesture (Sec. IV); and strategies to filter and refine the gesture representations, eventually incorporating corrections from the user, for the gesture recognition (Sec. V).

Gesture Segmentation: The user performs the gestures that the machine should recognize without the need to interrupt the recording between gestures or to go back to a neutral pose. At each frame of this recording, we extract a *pose*, *i.e.* the 9 relevant joints of the body skeleton [1] (Fig. 2). We segment the gestures from this stream of poses by analyzing the *gesture curvature*, *i.e.* the curvature of a curve in $\mathbb{R}^{27}$ whose points are the 3d coordinates of the position of the 9 joints of the poses, in sequence.

Discriminant Key Pose Selection: From the set of segmented gestures, we incrementally compute a few poses, called *key poses*, which compactly and unambiguously represent the gesture, in a greedy adaptive sampling manner. We start by adding the first and last pose of each gesture to the key poses. If a gesture starts and ends at the same pose, we also add the pose of the gesture that is the most distinct from the initial pose. Then, we check for gestures with identical representations, and add the discriminant key poses (the ones that better differentiate the gestures) to the set of key poses, until all representations become unique.

Gesture Identification: This simplification of the training phase may lead to over-segmentation or spurious gestures (like transition between gestures), which can be filtered using a semiautomatic approach. The user can also effortlessly confirm or correct the identification of two different performances of the same gestures. We finally feed the labeled gestures into a learning machine for recognition, and propose a simple (although slightly less efficient) classifier based on our discriminant measures to check for its coherence. The whole process needs very few resources to run in real time, which allows the user to fine-tune the gesture identification during the acquisition, and to re-use or increment the initial gesture recording to get a larger set of recognizable gestures.

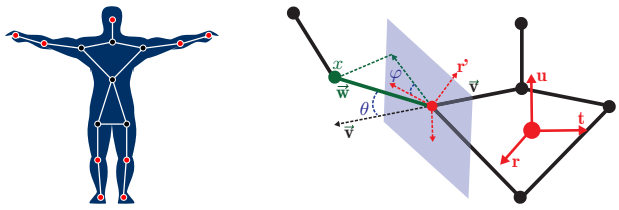

Fig. 2. Skeleton's graph: only the 9 joints beyond the torso, in red, are used (left). Joint-angle representation: each joint is represented by its spherical angles θ and ϕ in the torso referential (right). Figures adapted from [14].

III. Gesture Segmentation

The first step of our simplified training process consists in segmenting a recording of gestures into individual gestures. The main observation is that in-between gestures, the user naturally inserts a short pause where he stands almost still. Due to the sensor's use of random patterns, this pause is captured as a rapid oscillation between relatively distinct skeletons, which we detect through the curvature of the gesture curve. The user may filter this segmentation at the end of the process, although the method here turns out to be robust in practice.

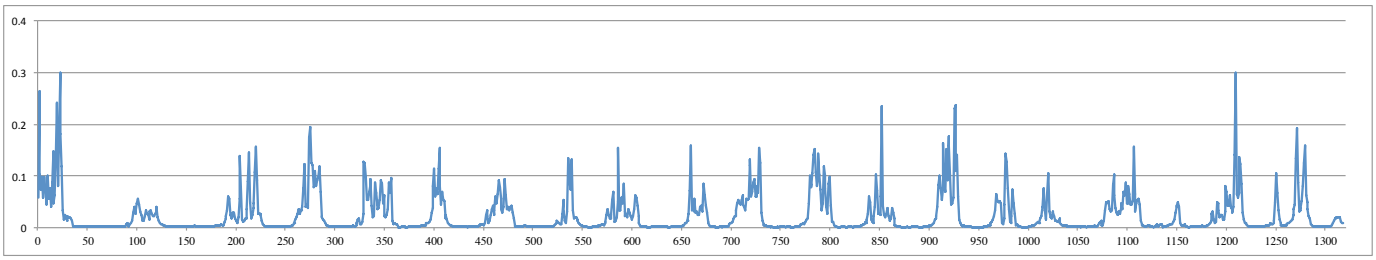

Fig. 3. Gesture curvature during a recording of one individual performing 11 gestures: short pauses can be easily identified by intervals of high curvature, while gesture executions are characterized by low curvature intervals.

A. Pose representations

In this work, we use two local representations derived from the skeleton produced by the OpenNI[1] from the Kinect sensor. This skeleton contains several labeled joints, out of which 9 are considered relevant for body gestures [8–10,13,14]. We refer to a configuration of those 9 joints as a *pose*, and use two distinct representation for it (Fig. 2).

For the gesture recognition, we encode a pose as the list of spherical angles of each of the 9 joints [13,14], so that a pose is tuple of 9 points on a sphere: $p \in \left(\mathbb{S}^2\right)^9$. We use the geodesic distance on the sphere $\delta(p_l, p'_l)$ to compare the same joints l in two distinct poses p and p' using their spherical angles (θ_l, ϕ_l) and (θ'_l, ϕ'_l) using the usual formula:

$$\delta(p_l, p'_l) = \arccos\left(\sin\theta_l \sin\theta'_l + \cos\theta_l \cos\theta'_l \cos|\varphi_l - \varphi'_l|\right) \quad .$$

We can then measure the distance between two poses as suggested in [14]:

$$\Delta(p, p') = \sum_{l=1}^{9} \left[\delta\left(p_l, p'_l\right)\right]^2 \quad .$$

This distance measure is used for identifying poses and distinguish discriminant key poses in the Sec. IV.

For the gesture segmentation, we encode a pose by the coordinates of each of the 9 joints in $\mathbb{R}^3$ [8–10], in which case a pose is a point in $\mathbb{R}^{27}$ and a gesture is a curve in $\mathbb{R}^{27}$. This representation is straightforward, and the use of curvature of the gestures ensures the segmentation is invariant to change in the relative position of the user to the sensor.

B. Curvature estimation

Although the gesture curve is embedded in $\mathbb{R}^{27}$, we only use its first curvature, and the estimator can be adapted from usual curvature estimators. Since the curve has a high co-dimension, parametric estimators gain performance.

Denoting the gesture curve by function $p(t) \in \mathbb{R}^{27}$, its curvature can be computed from the associated Frenet frame [26], *i.e.* for each time parameter t an orthonormal frame whose origin is at $p(t)$. Its first vector $\mathbf{e}_1(t)$ points in the direction of the tangent to the curve: $\mathbf{e}_1(t) = \frac{p'(t)}{\|p'(t)\|}$. The second vector $\mathbf{e}_2(t)$ points in the direction of the first normal: the plane $Span(\mathbf{e}_1(t), \mathbf{e}_2(t))$ matches the plane $Span(p'(t), p''(t))$ and $\langle \mathbf{e}_2(t), p''(t) \rangle > 0$. The following vectors of the frame point in the direction of the higher-order normals.

The curvature $\kappa(t)$ of the curve at $p(t)$ follows from the time derivative of $\mathbf{e}_1(t)$ [26]:

$$\kappa(t) = \frac{\langle \mathbf{e}'_1(t), \mathbf{e}_2(t) \rangle}{\|p'(t)\|} = \frac{\langle p''(t), \mathbf{e}_2(t) \rangle}{\|p'(t)\|^2} \quad .$$

The second form is deduced using the definition of the frame, and avoids estimating $\mathbf{e}'_1(t)$.

We estimate the derivatives of $p(t)$ using an extension of parametric curve fitting [27]: we fit a portion of the gesture curve around $p(t)$ to a parabola of the form

$$\tilde{p}(s) = p(t) + \tilde{p}' \cdot s + \tfrac{1}{2} \cdot \tilde{p}'' \cdot s^2 \quad ,$$

where the unknown vector coefficients $\tilde{p}'$ and $\tilde{p}''$ are the estimates for the derivatives $p'(t)$ and $p''(t)$. The fitting is obtained by a weighted least-squares minimization of the distance of poses $p(t')$ of the gesture curve to the fitted curve $\tilde{p}(s)$. The arc-length parameter s is estimated as $\|p(t') - p(t)\|$ and the weights are given by a Gaussian function of s.

C. Gesture segmentation

Usual protocols for gesture recognition training require the user to separate the gestures by a neutral pose, which can be trained and detected [22]. We propose here to remove this requirement and let the user record the sequence of gestures with the only restriction to perform a short pause between two gestures. Since the Kinect sensor projects random patterns to extract the depth image, the skeleton extraction [1] receives slightly different inputs even for a static body and its example-oriented nature returns distinct, unrelated skeletons for each input. This results in a rapidly oscillating sequence of poses, which can be directly detected from the gesture curvature, as shown in Fig. 3.

We thus detect the pauses in-between gestures with a simple threshold on a window inside the gesture curvature graph. In our experiments, we use a threshold of 0.01 and a window of 5 poses. This may eventually result in over-segmented gestures (actually it occurred only once in our experiments, as recorded in Fig. 10), and the user can simply indicate at the end of the process that two half-gestures should be concatenated.

[1] http://www.openni.org

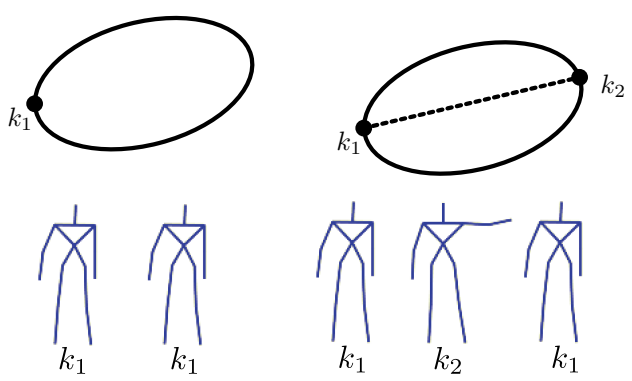

Fig. 4. A gesture that starts and finishes at a neutral pose k_1 (left) does not have a valid representation and must be refined by inserting the most discriminative pose k_2 (right).

IV. Discriminant Key Pose Selection

Once the recording is segmented into a set $\mathcal{G}$ of gestures, we aim at representing each gesture as a minimal sequence of poses, called *key poses*, such that distinct gestures correspond to different sequences of key poses. We greedily select key poses that are the most discriminant between pairs of gestures.

A. Gesture representation

For a given set $\mathcal{K} = \{k_1, \dots, k_n\}$ of key poses, we compute a representation $\hat{g} = (k_i, k_j, \dots)$ for each gesture $g = (p_1, p_2, \dots) \in \mathcal{G}$ as a sequence of key poses as follows. For each pose p of g, we look for its closest key pose $k_p \in \mathcal{K}$, using the pose distance Δ introduced in Sec. III-A: $k_p = \operatorname{argmin}_{k \in \mathcal{K}} \Delta(p, k)$. Then, we traverse the poses of gesture p, and insert k_p in the sequence $\hat{g}$ representing g if $\Delta(p, k_p) < \epsilon$ for a given threshold ϵ. To improve robustness in noisy examples, we only consider inserting a key pose k if it corresponds to two successive poses p_t, p_{t+1} of g, *i.e.* if $k = k_{p_t} = k_{p_{t+1}}$. Additionally, consecutive repetitions of key poses in $\hat{g}$ are ignored, *e.g.*, gesture representation $\hat{g} = (k_1, k_2, k_2, k_3)$ simplifies to $\hat{g} = (k_1, k_2, k_3)$.

B. Discriminant key pose

We look for discriminant key poses in two situations: either to refine a single gesture or to distinguish two gestures.

Initially, each gestures is represented by their initial and final poses, both inserted to the key pose set. In that context, a gesture g may reduce to the representation (k_1, k_1), in particular when k_1 is a kind of neutral pose. To avoid this degenerated representation, we look for an intermediate key pose as the pose of g that is the farthest from k_1:

$$k_2 = \operatorname*{argmax}_{p \in g} \Delta(p, k_1) .$$

Eventually, k_2 is still similar to k_1: $\Delta(k_2, k_1) < \epsilon$, in which case gesture g reduces to a static pose and we discard it. Otherwise, k_2 is inserted into $\mathcal{K}$ and, using the process described in Sec. IV-A, gesture g will be represented by $\hat{g} = (k_1, k_2, k_1)$ (Fig. 4).

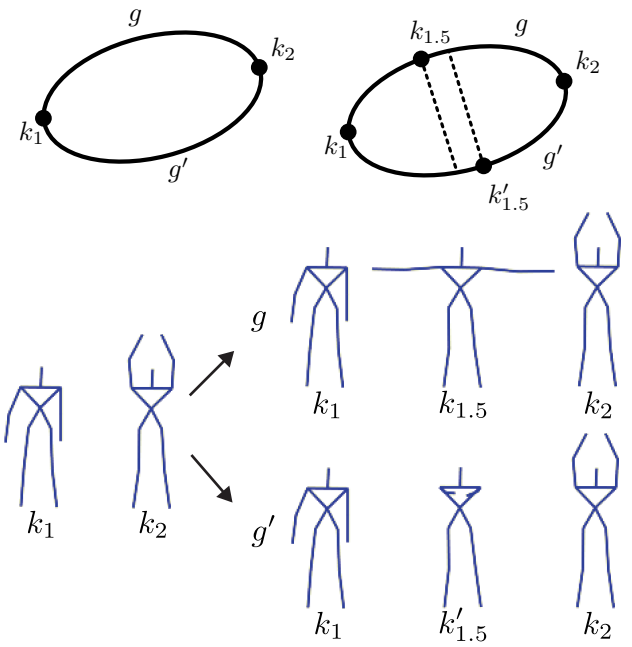

Fig. 5. An ambiguous gesture representation (k_1, k_2) representing different gestures g and g' (left) are refined by finding and inserting discriminant poses $k_{1.5}$ and $k'_{1.5}$ that better distinguishes gestures g and g' (right).

In order to distinguish two gestures g and g' that have the same representation $\hat{g} = \hat{g}' = (k_1, k_2, \dots, k_n)$, we look for two key poses to insert in the sequences $\hat{g}$ and $\hat{g}'$ in order to differentiate them. Let's look first at the case $n = 2$: $\hat{g} = \hat{g}' = (k_1, k_2)$ (Fig. 5). We can define the most discriminant pose of g in relation to g' as the pose $k_{1+1/2}$ where the asymmetric Haussdorff distance is achieved:

$$k_{1+1/2} = \operatorname*{argmax}_{p \in g} \min_{p' \in g'} \Delta(p, p') \quad .$$

Similarly, we compute the most discriminant pose of g' in relation to g as: $k'_{1+1/2} = \operatorname{argmax}_{p' \in g'} \min_{p \in g} \Delta(p', p)$.

In the general case, we repeat the process above for every sub-sequence between successive key poses of g and g' and select the most distinctive pair (Fig. 6). More precisely for $\hat{g} = \hat{g}' = (k_1, k_2, \dots, k_n)$, let g_i and g'_i be the sub-gestures of g and g' respectively whose initial and final key poses are k_i and k_{i+1}. We compute the most discriminant pairs of key poses $k_{i+1/2}$ and $k'_{i+1/2}$ between g_i and g'_i for every $i \in \{1, \dots, n-1\}$, and select for $\mathcal{K}$ only the pair $k_{i+1/2}$ and $k'_{i+1/2}$ with the highest sum of asymmetric Haussdorff distances:

$$j = \operatorname*{argmax}_i \left\{ \min_{p' \in g'_i} \Delta\left(k_{i+1/2}, p'\right) + \min_{p \in g_i} \Delta\left(k'_{i+1/2}, p\right) \right\} .$$

If $\Delta\left(k_{j+1/2}, k'_{j+1/2}\right) < \epsilon$, both gestures receive identical labels. Otherwise, $k_{j+1/2}$ and $k'_{j+1/2}$ are inserted in $\mathcal{K}$, and we re-compute the representations of all gestures as described in Sec. IV-A.

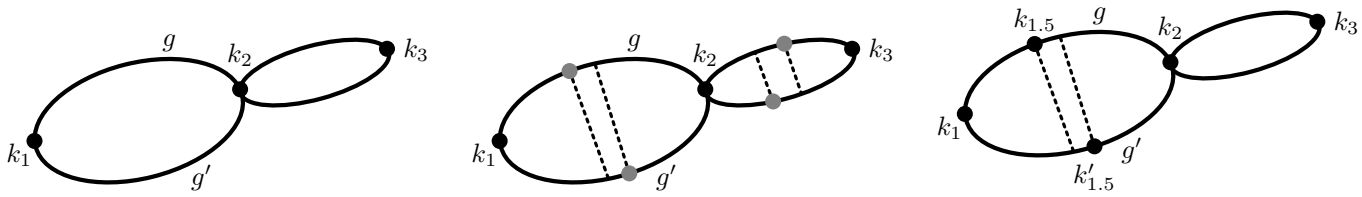

Fig. 6. When an ambiguous gesture representation is composed of three or more key poses, as the example (k_1, k_2, k_3) (left), each pair of consecutive key poses is analyzed. Here, discriminant key poses are found in pairs (k_1, k_2) and (k_2, k_3) (center), and only the most discriminative key poses $k_{1.5}$ from $k'_{1.5}$ are inserted (right).

C. Greedy selection of discriminant key poses

At the beginning of the algorithm, or each time a gesture is added, the initial and final poses of each gesture $g \in \mathcal{G}$ are inserted in $\mathcal{K}$, and similar poses (within Δ-distance inferior to ϵ) are identified. We then compute the gesture representations with this initial set $\mathcal{K}$ as described in Sec. IV-A. If a gesture g has a degenerate representation $\hat{g} = (k_1, k_1)$, we look for a discriminant key pose to refine $\hat{g}$ as described in Sec. IV-B, either discarding g or inserting a new key pose k_2 to $\mathcal{K}$, in which case we re-compute the gesture representations with the new set $\mathcal{K}$.

Now that all gestures have a valid representation, we proceed iteratively to refine gestures that have identical representations. If two gestures g and g' have the same representation $\hat{g} = \hat{g}' = (k_1, k_2, \ldots, k_n)$, we look for a pair of discriminant key poses as described in Sec. IV-B, either identifying g or inserting both key poses to $\mathcal{K}$, in which case we re-compute the gesture representations.

This process is indeed an adaptive sampling of the gesture curves, refined until all distinct gestures have different representations.

V. Gesture identification

During a training session, the user sequentially performs several different gestures with short pauses and no labeling. The automatic process described so far copes with a very simple training protocol, but it may generate spurious gestures or it may fail to detect performances of the same gestures. Those issues are handled by semiautomatic filters described in this section.

A. Spurious gesture elimination

The segmentation may extract spurious gestures, like transitions between gestures, long pauses or static gestures. We opt for a semiautomatic filter, where some of the spurious gestures are automatically removed and others are submitted to the user for validation.

We detect static gestures during the refinement of a single gesture (Sec. IV-B). Then, we suggest to the user very short gestures sorted according to their duration, but presenting first gestures that start at a non-neutral pose and end at the neutral pose. The first gestures of that list can safely be automatically classified as spurious, while the following are left to the user's decision.

B. Semiautomatic labeling

A gesture recognition method requires several exemplars of the same gesture, eventually performed by different users. This may lead to different key poses sequence for the same gesture, and several gesture recognition method incorporate this fact using templates [8], decision forests [14] or action graphs [19,21,22], to cite a few.

During the search for discriminant key poses (Sec. IV-B), we already detect that some gestures are identical, and generate a single label for both. This identification heavily depends on the pose similarity threshold ϵ, so we opt to use a safe value for ϵ and ask the user to complete the labeling. In our interface, the user confirms whether two performances correspond to the same gesture. If not, we look for discriminant key poses as described in Sec. IV-B, but threshold ϵ is exceptionally ignored to allow the insertion of similar key poses in $\mathcal{K}$, and effectively distinguish the gestures. Note that, for natural user's interface applications, this interface can be coupled to the assignment of an action the computer should perform in response to a user's gesture.

C. Gesture recognition

The result of our simplified training is a set $\mathcal{G}$ of gestures together with a set of key poses that can represent the gestures unambiguously. This is indeed the common format for gesture recognition databases such as the MSR[2], UTKinect[3], CAD[4] or KGD[5] datasets, and can thus be used by most gesture recognition algorithms.

In order to validate our approach, we embedded the result of our simplified training in a modification of the key pose based gesture recognition system of Miranda *et al.*[14]. The key pose learning machine uses the set $\mathcal{K}$ of key poses, and a classifier $\tilde{f}$ that is capable of recognizing poses acquired from the Kinect sensor in real time. Instead of the original SVM definition of $\tilde{f}$, we use a nearest-neighbor classifier combined with the pose similarity threshold to classify a pose p:

$$\tilde{f}(p) = \begin{cases} k_p = \underset{k \in \mathcal{K}}{\operatorname{argmin}}\ \Delta(k, p) & \text{if } \Delta(k_p, p) < \epsilon, \\ \text{-1} & \text{otherwise.} \end{cases}$$

Note that if the current pose p is not similar to any key pose in $\mathcal{K}$, the classifier returns -1.

[2]http://research.microsoft.com/en-us/um/people/zliu/ActionRecoRsrc/
[3]http://cvrc.ece.utexas.edu/KinectDatasets/HOJ3D.html
[4]http://pr.cs.cornell.edu/humanactivities/data.php
[5]http://www.im.ufal.br/professor/thales/gesturedb.html

gesture	id	segmentation accuracy	recognized gestures per user										ours (%)	[14] (%)
			u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}		
Turn Next Page	$\hat{g}_A$	10	10	8	10	10	9	9	10	9	9	9	**93**	**95**
Turn Previous Page	$\hat{g}_B$	10	10	9	10	10	9	6	9	9	9	10	**91**	**95**
Raise Right Arm	$\hat{g}_C$	10	9	8	9	8	7	10	9	10	8	10	**88**	**94**
Raise Left Arm	$\hat{g}_D$	10	10	10	9	10	9	9	10	9	10	9	**95**	**94**
Open Clap	$\hat{g}_E$	8	10	10	10	9	9	8	10	9	8	10	**93**	**99**
Open Arms	$\hat{g}_F$	9	9	9	10	8	9	10	10	9	8	9	**91**	**97**
Put Hands Up Lat.	$\hat{g}_G$	9	10	10	10	10	10	9	10	10	10	10	**99**	**100**
Put Hands Up Front	$\hat{g}_H$	10	10	9	7	9	10	9	10	10	10	9	**93**	**96**
Lower Right Arm	$\hat{g}_I$	8	8	7	6	8	7	8	8	8	8	7	**75**	**82**
Bow	$\hat{g}_J$	6	10	10	10	9	10	10	10	9	10	10	**98**	**100**
Goodbye	$\hat{g}_K$	7	9	9	10	7	9	10	9	10	8	7	**88**	**92**
average (%)		**88**	**90**	**92**	**89**	**89**	**91**	**89**	**93**	**89**	**91**	**92**		

Tab. 7. Trained gestures, segmentation accuracy and recognition rate for 10 individuals. Individuals were asked to train gestures respecting the sequence given in this table. Segmentation accuracy per gesture counts how many users had the gesture successfully segmented during the first execution (spurious gestures are not considered). The recognized gestures per user column shows the number of successfully recognized gestures after 10 executions per gesture.

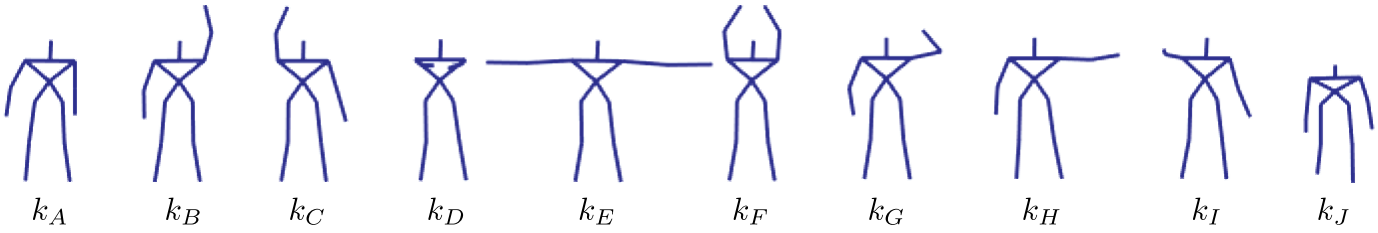

Fig. 8. Discriminant key pose set $\mathcal{K}$ computed online during a training session from a single user. During the session, 11 gestures were performed sequentially, and our method successfully found a small set of 10 key poses capable of representing all gestures as key pose sequences, avoiding ambiguity and allowing high recognition rates.

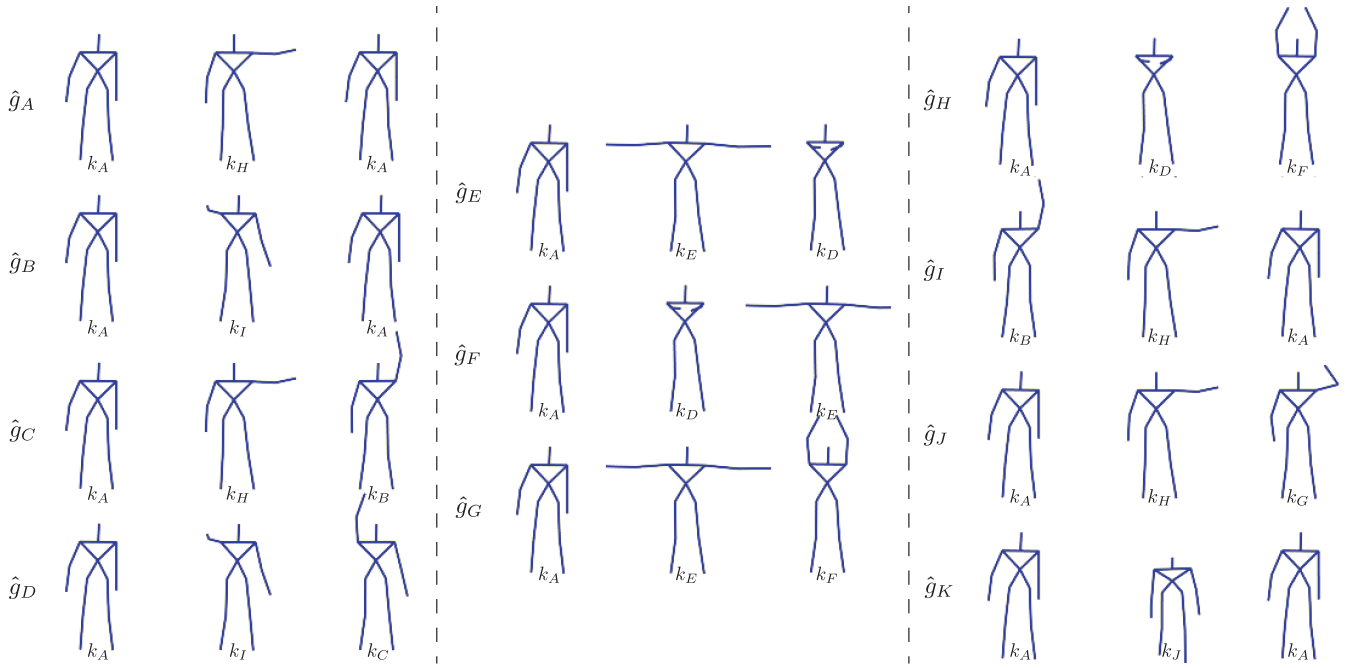

Fig. 9. Gesture representations computed for 11 gestures captured from individual u_1. 10 discriminant key poses shown Fig. 8 were automatically computed, and coincidentally all gestures were described by a sequence of exactly 3 key poses.

We then proceed as in [14] to detect gestures executions: key poses recognized by the classifier are accumulated into a circular buffer that continuously queries a decision forest representing all gestures. The modification proposed does not intend to improve the performance of the classifier, but to validate the key poses extracted during our simplified training and the pose similarity distance Δ.

VI. Experiments

To evaluate each step of our method and fairly compare with previous work, we use the same set of 11 gestures as Miranda *et al.*[14][5], reported in Tab. 7. We provided a brief description of each gesture to 10 inexperienced individuals, and asked them to sequentially perform each gesture once, with minor pauses, in front of a Kinect sensor. Each user took an average of 46 seconds and 1,400 frames.

Both segmentation and discriminant key pose selection were executed online, while the few spurious gestures were manually eliminated by each user in a post-processing phase. Unsuccessfully segmented gestures were exceptionally retrained to allow gesture recognition experiments. Finally, we asked each user to perform again each gesture 10 times to check the gesture recognition accuracy.

A. Real-time segmentation robustness

To evaluate the robustness of the gesture segmentation, we count, for each of the 10 users' recordings, how many gestures were correctly separated by the curvature criterion.

The algorithm was able to accurately segment most gestures in at least 8 out of 10 users, as reported in Tab. 7. Some over-segmentation occurred, mainly for the Bow and the Goodbye gestures. The curvature graphs of these executions (Fig. 10) show that some performances of those gestures contain a small pause (around pose k_J) that is interpreted by our algorithm as a transition between gestures.

B. Discriminant key pose selection

Computing discriminant key poses for each user separately, our algorithm succeeded in computing small sets of 10 to 12 discriminant key poses (Fig. 8). Most of our automatically selected key poses were similar among all individuals, and also similar to the manually designed key poses from the work of Miranda *et al.*[14], revealing good stability.

In most individuals training sets, small key pose sequences were obtained for all gestures. Fig. 9 shows gesture representations for user u_1. Coincidentally, all gestures were represented by a sequence of exactly 3 key poses, which seems visually adequate to both describe and discriminate all trained gestures.

We empirically found that better key pose representations were found by setting $\epsilon = 0.2\pi$. Larger values of ϵ result in fewer key poses and less accurate representations. On the one hand, setting $\epsilon = 0.4\pi$, poses similar to k_J would be recognized as a key pose similar to k_A, turning the Bow gesture invalid. On the other hand, setting smaller values of ϵ, the number of key poses increases, resulting in visually similar key poses in $\mathcal{K}$. Furthermore, the recognition rate decreases the classifier rejects more poses.

C. Gesture recognition

After computing for each user u_i its key pose set $\mathcal{K}_i$ and its set of gesture representations $\hat{\mathcal{G}}_i$, we asked them to execute each gesture 10 times to check the recognition accuracy. For most gestures, the recognition rate is similar to manually selected and trained key poses [14], as reported in Tab. 7. Note that this previous work relies on 18 key poses, as opposed to 11 automatically generated, and uses several executions of the same gestures with an SVM classifier, which is more robust than our validation-purposed nearest neighbor classifier.

The major cause of recognition inaccuracy was the confusion between key poses k_B and k_G (Fig. 8), in particular for the Raise Right Arm or Lower Right Arm gestures. Another issue for the recognition occurred in the repetitive Goodbye gesture, where some users trained sequences like $\{k_A, k_H, k_G\}$ but later performed symmetrically $\{k_A, k_G, k_H\}$.

D. Performance

As mentioned before, our method is capable of segmenting gestures, computing key pose representations and recognizing gestures in real time with minor CPU usage, at 30 fps (maximum Kinect sensor frame rate), on a Core i7 laptop at 2.4 GHz.

During a training session, the method computes curvatures for each frame, which induces a lag of 3 frames to achieve the window where the time derivatives are computed. The main bottleneck of the discriminant key pose selection is the nearest neighbor queries for existing key poses in $\mathcal{K}$: since the amount of key poses is small, no spatial data structures were used. Finally, during recognition phase, only one nearest neighbour query is executed to detect key poses, while the decision forests keeps gesture search complexity very low.

We also performed experiments offline by segmenting and computing a key pose representation for a training session composed of 1329 frames (44.3 seconds) and 20 gestures (including spurious ones, see Fig. 3). The discriminant key poses extraction took 0.33 seconds, reinforcing that the performance of our method allow for CPU-intensive process to run concurrently.

VII. Limitations and future work

The method proposed here allows extracting a small set of key poses that represent unambiguously a given set of gestures, as long as the speed of execution of a gesture or part of it is not relevant. In some applications, like dance gestures [13], periodicity and time are intrinsic to the gesture validation, and our gesture representation (Sec. IV-A) must be modified accordingly.

We use the same threshold ϵ for all our comparisons, and even in the key pose classifier adapted for our validation experiments. Tailored thresholds for the refinement of a single gesture, the discriminant key poses between two gestures, the identification of key poses and in particular for the classifier would certainly improve the key pose extraction and recognition rate.

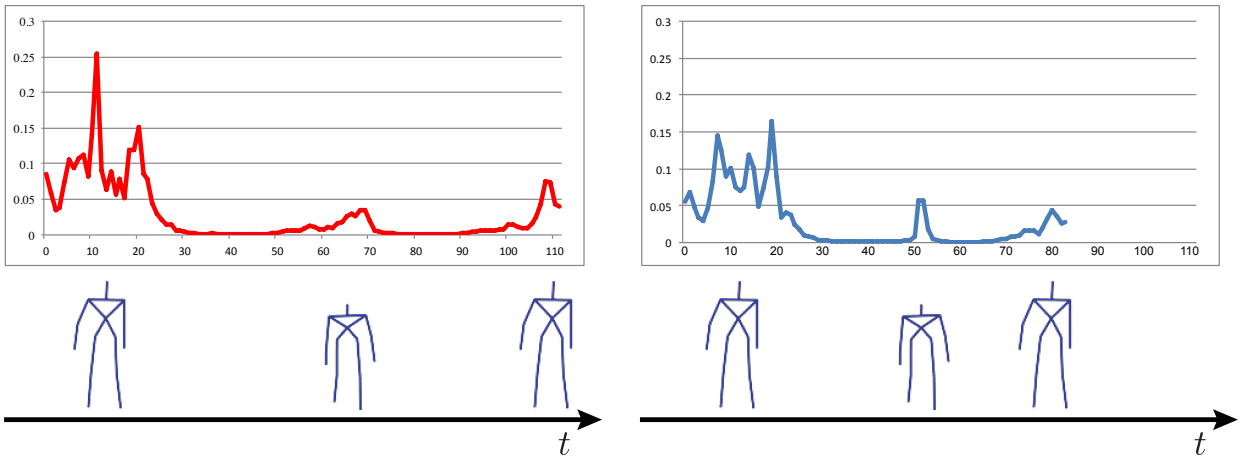

Fig. 10. Over-segmentation due to a long pause in the Bow gesture before turning the head up, as shown in the left curvature graph between frames 63 and 71. In this case, the method incorrectly segmented the gesture at frame 70. On the right graph, even though high curvatures were detected around frames 51 and 53, they were not enough to segment the gesture.

Finally, regarding the segmentation, we only use the first of the 26 gesture curvatures. Although higher-order curvatures are more sensitive to noise, they may help in reducing the over-segmentation of some gestures (Fig. 10). Better heuristics for the spurious gesture detection may also further simplify the user's interaction during the training.

ACKNOWLEDGMENT

The authors would like to thank CNPq, FAPEAL, FAPERJ and École Polytechnique for partially financing this research.

REFERENCES

[1] J. Shotton, A. W. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR*, 2011, pp. 1297–1304.

[2] M. Desmond, P. Collet, P. Marsh, and M. O'Shaughnessy, *Gestures: Their origins and distribution*. Cape, 1979.

[3] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[4] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 28, no. 6, pp. 976–990, 2010.

[5] F. Lv and R. Nevatia, "Single view human action recognition using key pose matching and Viterbi path searching," in *CVPR*, 2007, pp. 1–8.

[6] I. Laptev and T. Lindeberg, "Space-time interest points," in *ICCV*, vol. 1, 2003, pp. 432–439.

[7] Y. Zhu, W. Chen, and G. Guo, "Evaluating spatiotemporal interest point features for depth-based action recognition," *Image and Vision Computing*, 2014.

[8] M. Müller and T. Röder, "Motion templates for automatic classification and retrieval of motion capture data," in *SCA*, 2006, pp. 137–146.

[9] M. Müller, A. Baak, and H.-P. Seidel, "Efficient and robust annotation of motion capture data," in *SCA*, 2009, pp. 17–26.

[10] A. W. Vieira, T. Lewiner, W. Schwartz, and M. F. M. Campos, "Distance matrices as invariant features for classifying mocap data," in *ICPR*. IEEE, 2012, pp. 2934–2937.

[11] L. Kovar, "Automated extraction and parameterization of motions in large data sets," in *Siggraph*, vol. 23, 2004, pp. 559–568.

[12] K. Forbes and E. Fiu, "An efficient search algorithm for motion data using weighted PCA," in *SCA*, 2005, pp. 67–76.

[13] M. Raptis, D. Kirovski, and H. Hoppe, "Real-time classification of dance gestures from skeleton animation," in *SCA*, 2011, pp. 147–156.

[14] L. Miranda, T. Vieira, D. Martinez, T. Lewiner, A. W. Vieira, and M. F. M. Campos, "Online gesture recognition from pose kernel learning and decision forests," *Pattern Recognition Letters*, vol. 39, pp. 65–73, 2014.

[15] J. Sun, X. Wu, S. Yan, L. Cheong, T. Chua, and J. Li, "Hierarchical spatio-temporal context modeling for action recognition," in *CVPR*, 2009, pp. 2004–2011.

[16] L. Cao, Z. Liu, and T. Huang, "Cross-dataset action detection," in *CVPR*, 2010, pp. 1998–2005.

[17] A. Kovashka and K. Grauman, "Learning a hierarchy of discriminative space-time neighborhood features for human action recognition," in *CVPR*, 2010, pp. 2046–2053.

[18] J. C. Niebles, C. W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *ECCV*, 2010, pp. 392–405.

[19] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3D points," in *CVPR Workshops*, 2010, pp. 9–14.

[20] D. Weinland and E. Boyer, "Action recognition using exemplar-based embedding," in *CVPR*, 2005, pp. 1–7.

[21] W. Li, Z. Zhang, and Z. Liu, "Expandable data-driven graphical modeling of human actions based on salient postures," *Circuits and Systems for Video Technology*, vol. 18, no. 11, 2008.

[22] A. W. Vieira, E. R. Nascimento, G. L. Oliveira, Z. Liu, and M. F. Campos, "On the improvement of human action recognition from depth map sequences using Space–Time Occupancy Patterns," *Pattern Recognition Letters*, vol. 36, no. 15, pp. 221–227, 2014.

[23] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *TPAMI*, vol. 23, 2001.

[24] D.-Y. Chen, H.-Y. M. Liao, and S.-W. Shih, "Human action recognition using 2-D spatio-temporal templates," in *Multimedia and Expo*, 2007, pp. 667–670.

[25] C. Ellis, S. Z. Masood, M. F. Tappen, J. La Viola, Joseph J., and R. Sukthankar, "Exploring the trade-off between accuracy and observational latency in action recognition," *International Journal of Computer Vision*, vol. 101, no. 3, pp. 420–436, 2013.

[26] M. do Carmo, *Differential geometry of curves and surfaces*. Prentice Hall, 1976.

[27] T. Lewiner, J. Gomes, H. Lopes, and M. Craizer, "Curvature and torsion estimators based on parametric curve fitting," *Computers & Graphics*, vol. 29, no. 5, pp. 641–655, 2005.