Real-time gesture recognition from depth data through key poses learning and decision forests

Leandro Miranda Thales Vieira Dimas Martinez Thomas Lewiner Antonio W. Vieira Mario F. M. Campos
Mathematics, UFAL Mathematics, PUC-Rio Computer Science, UFMG

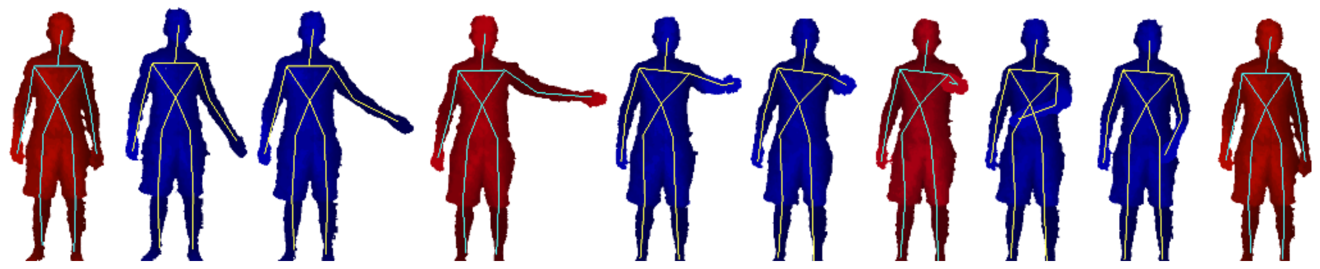

Fig. 1. Gesture representation through key poses: our method represents a body gesture as a sequence of a few extreme body poses, referred as key poses. In the example above, a gesture simulating a page turn is represented by the key poses in red.

***Abstract*—Human gesture recognition is a challenging task with many applications. The popularization of real time depth sensors even diversifies potential applications to end-user natural user interface (NUI). The quality of such NUI highly depends on the robustness and execution speed of the gesture recognition. This work introduces a method for real-time gesture recognition from a noisy skeleton stream, such as the ones extracted from Kinect depth sensors. Each pose is described using a tailored angular representation of the skeleton joints. Those descriptors serve to identify key poses through a multi-class classifier derived from Support Vector learning machines. The gesture is labeled on-the-fly from the key pose sequence through a decision forest, that naturally performs the gesture time warping and avoids the requirement for an initial or neutral pose. The proposed method runs in real time and shows robustness in several experiments.**

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

I. Introduction

Human gesture recognition has been a topic of active research for many years and covers a wide range of applications including, among others, monitoring, control and analysis. Monitoring applications covers classical problems of surveillance and automatic detection of suspicious gestures in environments with large flows of people, as well as monitoring in the home environment to detect accidents or monitoring children and elders. In the area of control, applications are usually interested in automatically recognizing human gestures to be used as a control for entertainment devices such as game consoles, virtual reality, motion capture to graphics model animation or automatic control of domestic utilities. The potential applications of gesture recognition in the area of analysis may help in diagnosis of orthopedic patients, in the study of athletes' performances and sports.

The variety of potential applications and growing availability of real time sensors has intensified efforts of the scientific vision community for capturing and automatic recognition of human gestures. The evolution and popularization of depth sensors, which currently generates depth maps in real time, as well as skeletons together with joint identification, is paving the way for the development of high quality natural user interface (NUI) for much more than usual game consoles. Improving the quality of a NUI essentially means increasing the robustness and execution speed of the gesture identification, and this is the objective of the present work.

The gesture recognition task can be stated as the process of labeling gestures performed by a person based on sensory observation captured by a device. This is a challenging task particularly when users perform gestures with different speeds or sequence of poses. In this work we propose a gesture recognition method from captured skeletons in real time. In particular, all our experiments are performed using the popular Kinect platform, a real-time depth sensing system that parses a depth-map stream at 30 frames per second, from which positions of skeleton nodes for each frame can be estimated in real time [1].

Contributions: A human gesture is essentially composed of body poses that continuously evolve over time. Interestingly, we verbally describe gestures by sequentially identifying a few extreme poses, referred as key poses [2], as illustrated in Fig. 1. Following this observation, we focus on improving and tailoring the three main ingredients of key-pose gesture recognition: the pose descriptor, the pose identification and the labeling of pose sequences as gesture.

Our pose descriptor relies on spherical angular representations of joints, similarly to the recent work of Raptis *et al.*[3]. However, it is more robust for usual gestures, and allows for real-time pose classification. In particular, it improves the representation of secondary joints (arms, hands, legs and feet) to suit for NUI applications.

The pose identification process combines several Support Vector Machine (SVM) classifiers [4], one per reference key pose, using the kernel distance in feature space as a confidence measure. The small pose descriptor size allows for online training in that context.

Finally, we propose a scheme for gesture recognition based on decision forests. Each forest node is a key pose, eventually including time constraints. This decision forest is learned during a training phase. The roots of the trees are the possible final key poses of gestures, where leaves represent the gestures. Each path from a leaf parent to the root gives a sequence of key poses for a gesture. This greatly simplifies the search for key pose sequences, allowing for real time gesture recognition. Moreover, the decision state machine provides a natural and robust temporal alignment. The whole process is robust even with noisy depth-based skeletons [1] as shown in the Results section.

II. Related work

Human gesture recognition has been extensively studied and a large body of literature has been produced with applications in areas such as surveillance, home monitoring and entertainment. The methods for gesture recognition can be categorized according to the spatial representation used: local, global or parametric.
The methods in the local category use point-wise descriptors evaluated on each point of interest, and then use a bag of features (Bof) strategy to represent actions. This approach has attracted a lot of attention in the past few years [5], [6], [7], [8] and an example of largely used local feature is the Space-Time Interest Point (STIP) presented by Laptev and Lindeberg [9]. A bag of 3D points was proposed by Li *et al.*[10] as local features for action recognition from depth map sequences. A drawback of local features approaches is that they lose spatial context information between interest points.
The methods in the second category use global features such as silhouettes [2], [11], [12] and template based approaches [13], [14], where spatial context information are preserved. Vieira *et al.*[15] proposed a global feature called Space-Time Occupancy Patterns (STOP) for action recognition from depth map sequences where space and time axes are used to define a 4D grid. A drawback of global features approaches is the lack of body joints identification.

Finally, parametric methods try to reconstruct a model of the human body with identification of joints to obtain an skeleton. One way to obtain skeletons is by using a marker based strategy as in the Motion Capture (MoCap) models [16], [17], where a set of markers is attached to the human body at specific points that are tracked during motion. Using this representation, spatial features are constructed using some geometric measures and relations, e.g. angles, [18], [19] and body joints [20], [21]. However, MoCap skeletons strongly depend on the joint markers and the capture system.

Skeletons can also be estimated from 2d maps. In particular, Shotton *et al.* [1] obtain skeletons without markers by computing joint coordinates in real time from depth maps. Such skeletons can be obtained from the popular *XBOX Kinect* sensor. Compared to Mocap data, skeletons from Kinect are easier to obtain, which have driven the popularity of this sensor, but they are still noisy, turning gesture recognition from depth data a challenge, that is the focus of the present work.

Beyond spatial representation, another issue to be addressed for human gesture recognition is the temporal alignment among different sequences that may vary significantly. To address time alignment, Hidden Markov Models is highly used as in the work from Davis and Tyagi [22] and from Zhang *et al.*[23]. Li *et al.*[12] propose a graphical modeling of human action where key poses define nodes on the graph and each action describes a path in the Action Graph. In the works from by Müller *et al.*[20], [21], Dynamic Time Warping (DTW) is used to address time alignment in their Motion Templates using MoCap data. In this work, we propose a simpler, yet robust and efficient method to perform alignment inside a decision forest scheme.

Gesture recognition using skeletons from Kinect is even more recent. Reyes *et al.*[24] presented an action recognition system using DTW for time alignment, where weights are assigned to features based on inter-intra class action variability. They obtain 3D coordinates of skeletal models using a reference coordinate system to make description invariant to view point and tolerant to subject corporal differences. Results are reported for only five different categories. Raptis *et al.*[3] presented a method for classifying dance gestures using Kinect skeletons where a large number of gestures classes are used. Their features are constructed using angular representation on a coordinate system obtained by Principal Component Analysis on a subset of the body points that define the torso. Despite high recognition rate reported, their classification method is limited to dance gestures and is conceived based on the strong assumption that the input motion adheres to a known music beat pattern.

III. Technical Overview

Our method relies on three main ingredients, as illustrated in Figure 2: a pose descriptor that concisely represents a human body pose, described in Section IV-A; a multi-class SVM learning machine that robustly identifies key poses in real-time, detailed in Section IV-B; and a decision forest built to recognize human gestures, optionally considering time/speed constraints (Section V).

Pose Descriptor: We convert the set of points representing the nodes of the skeleton to a more appropriate representation to robustly solve the gesture recognition problem. Essentially, we developed a joint-angles representation that provides invariance to sensor orientation; reduces redundancy and space dimensionality; while preserving relevant information to the classification of key poses. Our representation is an improvement of the work of Raptis *et al.*[3].

Key Pose Learning: A multi-class SVM approach recognizes key poses from skeletons in real time. More precisely, the training set is used to build several SVM binary

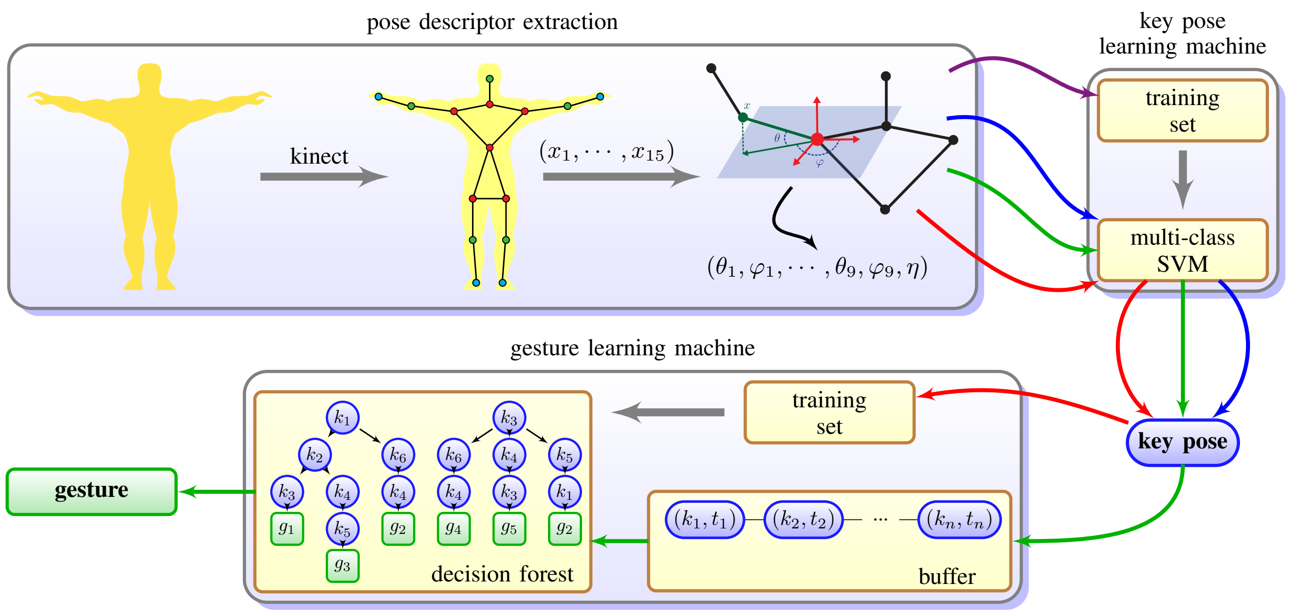

Fig. 2. Our method learns and recognizes in real time key poses and gestures from a compact joint-angles skeleton representation. There are four main use cases: training the key pose learning machine (purple arrow); recognizing user key poses in real time (blue arrows); training the gesture learning machine (red arrows); recognizing user gestures in real time (green arrows).

classifiers that robustly recognize key poses, in a *one-versus-all* approach. The efficiency of this method permits real-time training, allowing the user to improve the training set by correcting occasional misclassified key poses.

Gesture Training and Recognition: After training the key pose learning machine, the user is allowed to execute and label examples of gestures, defined as finite sequences of key poses. For each gesture performance, the classified key poses are accumulated into a circular buffer.

During training, a decision forest is built such that all the sequences of the same gesture correspond to connected paths in the forest, and those paths are labeled with the gesture. The decision forest is optimized to efficiently perform the search for key pose sequences, as described in Section V. It can optionally consider time/speed constraints. Even when different users perform the same gesture with different duration of key poses, the decision forest provides an effective and robust solution to that temporal alignment problem.

IV. Key Pose Statistical Learning

Key-based gesture recognition methods are highly dependent on the robustness of pose classification, and the pose identification requires efficiency to perform in real time. To solve this multi-class classification problem, we propose a supervised learning approach, where the machine learns key poses from user examples. We further aim to deliver robust pose identifications even with small training sets, as the ones provided by a single short training session. Finally, we would like the user to be able to, at any moment, provide labeled training data to correct and improve the classifier robustness, while keeping it efficient.

We build such a classifier using a multi-class variation of the *Support Vector Machines* (SVM) binary classifier, whose formulation is well suited to meet our requirements. SVM received a lot of attention in the machine learning community since it is optimal in the sense of the *VC* statistical learning theory [4]. We refer the reader to the book of Smola and Schölkopf [25] for a complete introduction to SVM techniques. This section briefly describes some basics of the multi-class SVM approach we adopted, and our joint-angles skeleton representation.

A. Joint-angles Representation

The skeleton representation must be invariant to sensor orientation and global translation of the body, minimize issues with skeleton variations from different individuals and still concisely capture all the relevant body pose information.

For each frame, the skeleton stream is a sequence of graphs with 15 nodes, where each node has its geometric position represented as a 3d point in a global Cartesian coordinate system (Fig. 3). The joints adjacent to the torso are usually called first-degree joints, while joints adjacent to first-degree joints are classified as second-degree joints. First-degree joints include the elbows, the knees and the head, while second-degree joints are the extremities: the hands and feet.

Different body poses are essentially obtained by rotating first and second-degree joints. Note that each joint movement has 2 degrees of freedom: a zenith angle θ and an azimuth angle φ; while the distance between adjacent joints (i.e., the radial distance) is always constant (Fig. 4).

In the work of Raptis *et al.*[3], a straightforward joint-angles representation is developed by converting each joint position $x_l \subset \mathbb{R}^3$ to local spherical coordinates. First, a torso basis is estimated by applying a PCA to a 7-3 torso matrix filled with the torso node positions. Then, the spherical coordinates

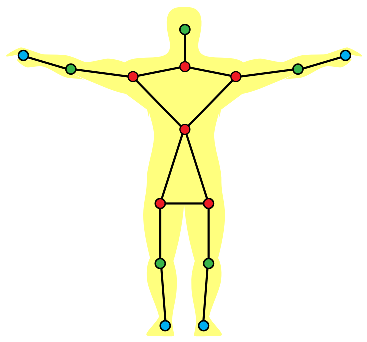
Fig. 3. Skeleton's graph: torso joints are represented in red; first-degree joints in green; and second-degree joints in blue.

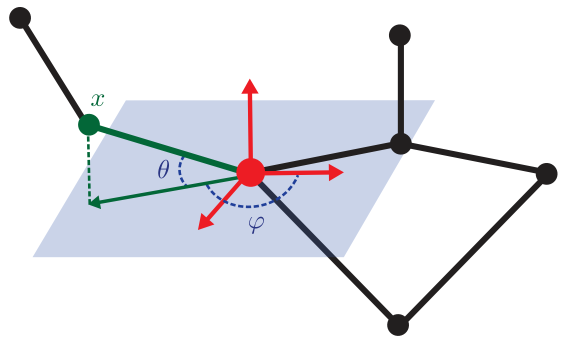

Fig. 4. Joint-angles representation: each body pose is a product of joints movements, which have 2 degrees of freedom in local spherical coordinate systems: the zenith angles θ and the azimuth angles φ.

of each first-degree joint are computed as a translation of this torso basis to the joint.

However, this same torso basis is used as reference to convert the second-degree joints, leading to a non-local description of the angles. Also, as mentioned by the authors, some combinations of joint positions can result in collapsed projections and consequently inconsistent angles, as in the open arms position [3].

We use the same torso basis for first-degree joints, but improve the representation of second-degree joints by considering rotations of the orthonormal torso basis $\{\mathbf{u}, \mathbf{r}, \mathbf{t}\}$.

Let $\mathbf{v}$ be the vector defined by the right arm and the right elbow and $\mathbf{w}$ the vector between the right elbow and the right hand. To define a local basis for the right hand, we rotate the torso basis $\{\mathbf{u}, \mathbf{r}, \mathbf{t}\}$ by the angle $\beta = \arccos(\mathbf{v} \cdot \mathbf{r})$ around the axis $\mathbf{b} = \mathbf{v} \times \mathbf{r}$. Note that if $\mathbf{v} \cdot \mathbf{r} = 1$, no rotation is applied. Also $\mathbf{v} \cdot \mathbf{r} \neq -1$ since the right arm can never rotate completely left due to body constraints. The rotated basis is translated to the right elbow and the spherical coordinates of the right hand are computed as

- θ - the angle between $\mathbf{v}$ and $\mathbf{w}$
- φ - the angle between the rotated $\mathbf{t}$ and the projection of $\mathbf{w}$ in the plane whose normal is $\mathbf{v}$

If $\mathbf{v}$ and $\mathbf{w}$ are collinear, we just set $\phi = 0$, as the azimuth is not defined, and this will not be an issue to our SVM pose classifier. The other second-degree joints are similarly constructed using variants of the torso basis, such that collapsing issues are avoided by other body constraints.

Finally, each joint position x_l is represented using a pair of spherical angles (θ_l, φ_l) that specifies it in a locally defined spherical coordinate system. We also compute the angle η between the directional vector $\mathbf{z}$ from the Kinect sensor and the inverted vector $-\mathbf{t}$ from the torso basis, to detect torso inclinations. Thus, a body pose joint-angles representation is a pose descriptor vector $\mathbf{v} = (\theta_1, \varphi_1, \ldots, \theta_9, \varphi_9, \eta) \in \mathbb{R}^{19}$.

B. Multi-class SVM formulation

The classification step identifies key poses from a pre-defined, verbally described set $\mathcal{K} = \left\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{|\mathcal{K}|}\right\}$ of key pose classes used to build gesture representations later on. During key pose training, the user creates a training set by providing several examples of each key pose. In our experiments, 30 examples per key pose were usually enough. The multi-class SVM learning machine is supplied with the training set $\mathcal{T} = \{(\mathbf{v}_1, \mathbf{c}_1), (\mathbf{v}_2, \mathbf{c}_2), \ldots\}$, where each pair corresponds to an example of a key pose trained by the user. Each vector $\mathbf{v}_i \in \mathbb{R}^{19}$ is the pose descriptor of the trained key pose, while $\mathbf{c}_i \in \{1, \ldots, |\mathcal{K}|\}$ is an integer identifying the key pose class.

For each key pose $\mathbf{p} \in \{1, \ldots, |\mathcal{K}|\}$, we build a classifying function $\hat{f}_p$ as the kernel distance to some of the trained pose:

$$\hat{f}_{\mathbf{p}}(\mathbf{v}) = \textstyle\sum_{j \in SV} \alpha_j \ \psi_p(\mathbf{c}_j) \ \phi(\mathbf{v}_j, \mathbf{v}) + b,$$

where

$$\psi_p(\mathbf{c}) = \begin{cases} 1 & \text{if } \mathbf{c} = \mathbf{p}, \\ -1 & \text{otherwise.} \end{cases}$$

The function $\phi \colon \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ maps a pair of pose descriptors to their scalar product in a feature, possibly infinite-dimensional space turning $\hat{f}_{\mathbf{p}}$ into a non-linear classifier. In this work, we used a Gaussian kernel, i.e.

$$\phi(\mathbf{v_1}, \mathbf{v_2}) = \exp\left(-\frac{\|\mathbf{v_2} - \mathbf{v_1}\|^2}{2\sigma^2}\right).$$

We chose $\sigma = 5$ since it lead to fine results in our experiments. Given a queried pose represented by its descriptor $\mathbf{v}$, each classifying function $\hat{f}_p$ essentially returns positive values if $\mathbf{v}$ is likely to be of key pose class $\mathbf{p}$, and negative values otherwise. The higher the value, the higher the confidence. Such use of the SVM-distance for classification confidence has been successful in other contexts, as for intelligent galleries design [26].

Finally, the key pose classification is decided through a voting process, where the higher confidence through all key pose classifiers identifies the key pose class of the queried pose. The key pose class from descriptor $\mathbf{v}$ is then

$$\hat{f}(\mathbf{v}) = \begin{cases} \mathbf{q} = \underset{\mathbf{p}}{\operatorname{argmax}} \hat{f}_{\mathbf{p}}(\mathbf{v}) & \text{if } \hat{f}_{\mathbf{q}}(\mathbf{v}) > 0, \\ \text{-1} & \text{otherwise.} \end{cases} \tag{1}$$

Note that if all classifiers return non-positive values, then the queried pose does not belong to any trained key pose class, causing the key pose classifier to return -1.

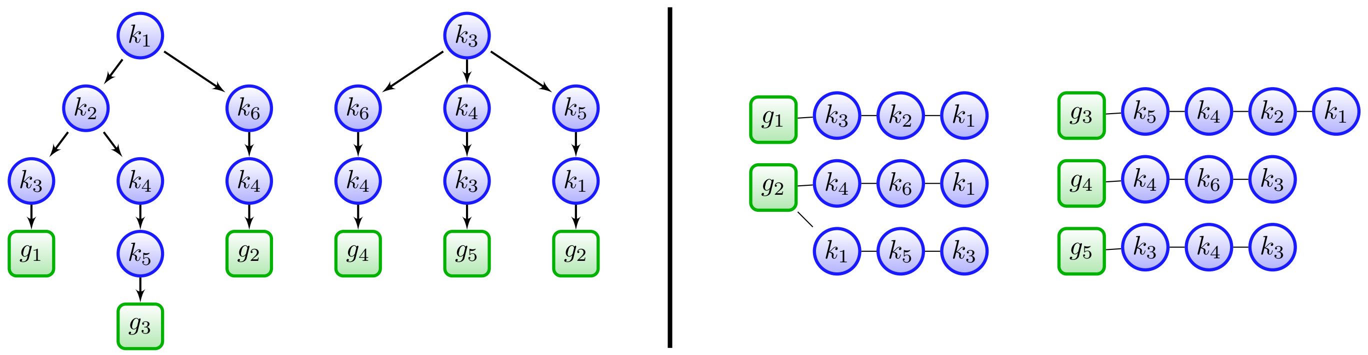

Fig. 5. Gesture learning machine example: a forest containing 6 key poses (left); and the 5 gestures represented by the forest (right).

V. Gesture Recognition through Decision Forest

We represent each gesture as finite sequences in the set $\mathcal{K} \subset \mathcal{P}$ of key poses. This representation is used to build a gesture set during training sessions. Afterwards, the training set is structured in a decision forest that efficiently recognizes gestures, parsing the key poses sequence of the user's performance.

A. Defining a gesture

In our setting, a gesture is represented as a sequence $g = \{k_1, k_2, \cdots, k_{n_g}\}$ of n_g key poses $k_i \in \mathcal{K}$. Usually, a gesture can be identified through a small sequence of two to five key poses. For example, a gesture simulating a page turn may need as few as 4 key poses, as shown in Fig. 1.

A straightforward solution to train a gesture example is to manually insert into the training set a sequence $\{k_1, k_2, \cdots, k_{n_g}\}$ of key poses composing the gesture g. Although effective, this approach can miss important key poses or ignore an alternative way of performing the same gesture.

In our approach, the user executes and labels examples of gestures during a gesture training session. The key pose classifiers searches in real time for key poses, which are then stored in the training set as a sequence of key pose identifiers representing the gesture. Often, slightly different executions of the same gesture class can lead to different sequences of key poses. These sequences are recorded separately as different representations of the same gesture class.

B. Recognizing gestures

Our decision forest scheme efficiently identifies gestures in real time. Given a gesture training set composed of key pose sequences, we build a forest whose nodes represent key poses. Each path in a tree of the forest, from a leaf parent to the root, represents a recognizable gesture. Thus, each tree represents all the gestures whose final key pose is given by its root key pose, while each leaf stores a gesture identifier. Note that there are at most as many trees as key poses.

During the recognition phase, the key pose classifiers try to recognize key poses performed by the user, and accumulate them into a circular buffer $\mathcal{B}$ of the last identified key poses. Note that the buffer must be bigger enough to accumulate a key pose sequence representing any trained gesture. We avoid doubly inserting a key pose in $\mathcal{B}$ by checking if the last added key pose is identical to a newly detected key pose.

Each time a key pose k is recognized, it is inserted in $\mathcal{B}$. Then, starts the search at the root of the tree representing gestures ending in k. We search down the tree by reversely iterating the buffer starting at k. If the preceding element in $\mathcal{B}$ (i.e., the previously detected key pose) is a child of the current node, the search continues. Otherwise, the search fails, as there is no trained key pose sequence which is a suffix of the last performed key pose sequence. If we manage to reach a leaf, the gesture stored in that leaf is recognized and reported.

The choice of storing the gestures back-to-front in the forest trees simplifies the recognition work. In this way, the successful search for the performed gesture begins only when its last key pose is detected, while in previously queries (i.e., when a previous key pose is recognized), the search will rapidly fail. Also, as we never know which key pose of $\mathcal{B}$ is the first key pose of the next gesture the user can be executing, we would need to perform searches for gestures initializing in every detected key pose.

We emphasize that two different key pose sequences can be safely tagged as the same gesture. Fig. 5 shows an example of a simple forest with six key poses and five recognizable gestures. Note how two different sequences of key poses are assigned to the same gesture g_2. This is convenient, for example, in applications where a gesture done with the right hand is considered to be the same as the one made with the left hand. Also, it is possible that, to perform the same gesture, one pass through slightly different sequences of key poses.

Finally, our formulation does not allow sub-gestures of gestures to be in the same training set, avoiding ambiguity issues. However, if this behavior is required, one can easily adapt our method by making possible to represent a sub-gesture in an interior node of the tree, along the path of its complete gesture.

C. Time constraints

For some applications, the execution speed of the gesture matters. In our gesture representation, we optionally include the duration between consecutive key poses as a time vector $[t_1, t_2, \cdots, t_{n-1}]$. Thus, the same key pose sequence performed at different speeds can be considered as executions of gestures belonging to different classes, or even not considered a gesture at all.

We store in each leaf of the forest one or more time vectors of the gestures sharing the same corresponding key pose sequence. These time vectors are captured during the training phase, representing the intervals between each pair of consecutive key poses.

When searching for gestures in the decision forest, more than a time vector can be found in a leaf. To choose or discard a gesture based on the time vectors, we use two criteria. Let $\mathbf{t}_i$ be a time vector stored on a leaf representing the gesture g_i and $\mathbf{t}$ the current user time vector. If $\|t_i - t\|_\infty > T$, where T is a small threshold, then g_i is discarded. Among all non-discarded gestures on the leaf, the gesture g_i that minimizes $\|t_i - t\|_1$ is chosen as the recognized gesture.

VI. Results

We present in this section the experiments we have performed to validate the robustness and evaluate the performance of the proposed method. We also compare our method to two state of the art methods, and discuss limitations.

A. Experiment setup

To evaluate the robustness of our key pose learning machine, we designed a key pose set $\mathcal{K}$ composed of 18 key poses to be used in all tests. One example of each key pose class is shown in Fig. 6. Note that we focused mainly on superior limbs poses, which are more suitable for natural user interfaces. To create the key pose training set $\mathcal{T}$, a single trainer performed around 30 examples of each key pose, resulting in approximately 600 examples of key poses.

Then, we designed a set $\mathcal{G}$ of 10 gestures, as shown in Table I. We asked the single trainer to perform 10 times each gesture from this set, and captured the sequences of key poses. We also considered time constraints in gesture g_7 to validate our formulation. Here, the last pose k_{11} must be kept for 1 second to characterize that gesture.

Note that $\mathcal{K}$ restricts the set of recognizable gestures $\mathcal{G}$ to all finite combinations of key poses from $\mathcal{K}$. Thus, the design of $\mathcal{K}$ must take into account the desired recognizable gesture set $\mathcal{G}$, exactly as compiling a step-by-step tutorial.

B. Key pose recognition

Robustness: We asked the trainer and 10 inexperienced individuals to perform all trained key poses to evaluate the recognition rate of our classifiers. Each individual performed each key pose 10 times. Table II shows the results. The key pose learning machine was able to recognize the users key poses in most cases, achieving an average recognition rate of 94.84%. Even in similar poses, like k_{13} and k_{17}, the machine succeeded in classifying the right pose in most examples. We noted that most failures were in challenging poses to the skeleton tracker, such as the pose k_{18}. Also, one female individual with long frizzy hair was troublesome for the skeleton tracker, and consequently to our method. However, when she tied her hair, better results were achieved.

gesture	**id**	**key pose seq.**	**rec. rate**
Open-Clap	g_1	$k_1,\ k_4,\ k_7$	**99%**
Open Arms	g_2	$k_1,\ k_7,\ k_4$	**96%**
Turn Next Page	g_3	$k_1,\ k_2,\ k_5,\ k_1$ $k_1,\ k_6,\ k_3,\ k_1$	**83%**
Turn Previous Page	g_4	$k_1,\ k_5,\ k_2,\ k_1$ $k_1,\ k_3,\ k_6,\ k_1$	**91%**
Raise Right Arm Laterally	g_5	$k_1,\ k_2,\ k_8$	**80%**
Lower Right Arm Laterally	g_6	$k_8,\ k_2,\ k_1$	**78%**
Good Bye (k_{11} time constraint:1sec.)	g_7	$k_1,\ k_{11}$	**92%**
Japanese Greeting	g_8	$k_1,\ k_{14},\ k_1$	**100%**
Put Hands Up Front	g_9	$k_1,\ k_5,\ k_{18}$ $k_1,\ k_5,\ k_8$ $k_1,\ k_5,\ k_{11},\ k_8$ $k_1,\ k_8$	**96%**
Put Hands Up Laterally	g_{10}	$k_1,\ k_4,\ k_{10}$	**100%**

TABLE I
Trained gestures and average recognition rate from experiments with 10 individuals. Key poses are described in Table II. Note that some gestures have multiple definitions.

Stability: To verify the stability of the key pose learning machine, we performed out-of-sample tests. In this experiment, we removed at random 20% of the data from our training set, computed the SVM classifiers and tried to classify the removed poses. After executing this experiment 10 times, the average number of false classifications was only 4.16%, while 3.45% could not be classified.

C. Gesture recognition

To check the robustness of the gesture learning machine, we described the trained gestures to 10 individuals. Then, we measured the recognition rate of the machine when the individuals executed each trained gestures 10 times. Excellent results were obtained in the majority of the gestures, while more tricky gestures achieved satisfactory results, as shown in Table I. Also, very good results were obtained in the time constrained gesture g_7. Further validation of time constrained gestures can be the object of a future research.

D. Performance

During the preprocessing phase, the only small bottleneck is computing the SVM binary classifiers functions. For a large training set with around 2.000 key pose examples of 18 classes, the 18 functions were computed in 3.9 secs, with an average of 105 support vectors per binary classifier. Note that these functions only need to be computed once, as long as the training set remains unchanged.

During training and recognition phases of our experiments, performance was negligible. The key pose learning machine, composed of several SVM binary classifiers, was easily capable of recognizing key poses at 30fps (the maximum Kinect sensor frame rate) in an average desktop computer.

Also, most gestures are composed of just a few key poses, generating decision trees with very low depths. In the other side, the trees breadths depends on the number of trained gestures, which is also a low number in most cases. Thus, the decision forest search complexity is irrelevant in the total complexity of our gesture recognition approach.

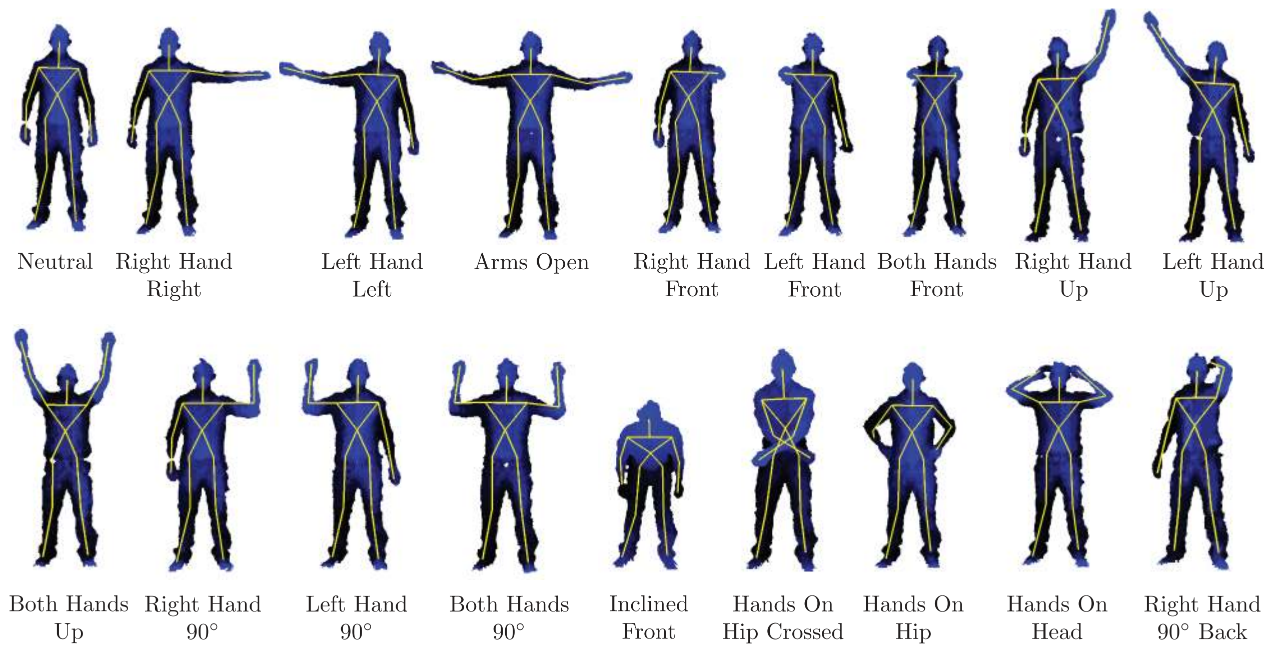

Fig. 6. Example of key poses from the training set.

key pose	id	recognized key poses per user u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}^1	u_{10}^2	total (%)
Neutral	k_1	10	10	10	10	10	10	10	10	10	10	10	**100.00**
Right Hand Right	k_2	10	10	10	10	10	10	10	10	10	10	8	**98.18**
Left Hand Left	k_3	10	10	10	9	10	10	9	10	10	10	10	**98.18**
Arms Open	k_4	10	10	10	7	10	10	10	9	10	7	10	**93.63**
Right Hand Front	k_5	10	10	10	10	10	10	10	10	10	8	7	**95.45**
Left Hand Front	k_6	10	10	9	10	10	10	10	10	10	10	10	**99.09**
Both Hands Front	k_7	10	10	10	10	10	10	10	10	10	10	10	**100.00**
Right Hand Up	k_8	10	10	10	10	10	10	10	10	10	10	10	**100.00**
Left Hand Up	k_9	10	10	10	10	10	9	10	10	10	9	10	**98.18**
Both Hands Up	k_{10}	10	10	10	10	10	10	10	10	10	10	10	**100.00**
Right Hand 90°	k_{11}	10	8	9	10	10	10	10	10	8	10	10	**95.45**
Left Hand 90°	k_{12}	10	10	10	10	10	6	10	10	10	5	10	**91.81**
Both Hands 90°	k_{13}	10	10	10	10	10	10	10	10	10	10	10	**100.00**
Inclined Front	k_{14}	8	10	10	10	10	8	10	10	10	5	7	**89.09**
Hands-on-Hip Crossed	k_{15}	7	8	6	8	8	10	10	10	8	10	8	**84.54**
Hand-On-Hip	k_{16}	10	10	10	10	10	10	10	9	10	10	10	**99.09**
Hands on Head	k_{17}	9	10	10	8	10	10	9	7	10	10	6	**90.00**
Right Hand 90° Back	k_{18}	8	10	9	6	7	7	7	10	10	3	8	**77.27**
total (%)		**95.5**	**97.7**	**96.1**	**93.3**	**97.2**	**94.4**	**97.2**	**97.2**	**97.7**	**87.2**	**91.11**	

TABLE II
RESULTS OF THE EXPERIMENTS WITH 10 INDIVIDUALS, PERFORMING VARIATIONS OF EACH KEY POSE 10 TIMES. u_1 IS THE TRAINER USER, u_{10}^1 IS THE FEMALE WITH LONG FRIZZY HAIR, SHOWING THE WORST RESULTS, AND u_{10}^2 IS THE SAME FEMALE WITH TIED HAIR, ACHIEVING BETTER RESULTS. ALL KEY POSES WERE VERY WELL RECOGNIZED BY THE CLASSIFIERS, EXCEPT FOR THE LAST ONE, DUE TO SKELETON TRACKING ISSUES.

E. Comparison

We compared our approach to two state of the art methods. In the work of Li *et al.*[10], three subsets of 8 gestures each are selected from a dataset of 20 gestures, as shown in Table III. For each subset, three different training sets were considered: training one third of the samples, two third of the samples, and samples from half of the subjects (cross-subject test). Then, the proposed method tried to recognize the non-trained samples.

The same dataset and subsets are used to compare with [10], [15] and our method, although we extract skeletons from the original depth maps. We analyzed the gestures of each of the three subsets to design sets of distinctive key poses for learning. For each subset AS1, AS2 and AS3, we trained key poses and gestures using the skeletons of half of the subjects from the dataset. We manually labeled key poses for each performance, feeding the gestures learning machine from the resulting pose sequence. Then, we performed cross subject tests, trying to recognize non-trained gesture samples using the corresponding training set.

The obtained results are shown in Table IV. Note that excellent results were obtained in AS1 and AS3, outperforming [10] and [15], while AS2 performed badly. The low performance of AS2 recognition was mainly due to 3 gestures composed of not very distinctive key poses: *draw x*, *draw circle* and *draw tick*. While these gestures required very accurate movements, many subjects executed them through slightly different poses.

AS1	**AS2**	**AS3**
Horizontal arm wave	High arm wave	High throw
Hammer	Hand catch	Forward kick
Forward punch	Draw x	Side kick
High throw	Draw tick	Jogging
Hand clap	Draw circle	Tennis swing
Bend	Two hand wave	Tennis serve
Pickup & throw	Side boxing	Pickup & throw

TABLE III
GESTURE DATASET OF LI *et al.*[10]

Gesture subset	**Li [10]**	**Vieira [15]**	**our method**
AS1	72.9%	84.7%	**93.5%**
AS2	71.9%	81.3%	**52.0%**
AS3	79.2%	88.4%	**95.4%**
Average	74.7%	84.8%	**80.3%**

TABLE IV
COMPARISON OF RECOGNITION RATE THROUGH A CROSS-SUBJECT TEST.

F. Limitations

Most robustness issues were mainly due to two reasons: skeleton tracking and accurate gestures, like drawing an x. Using a single Kinect, the user must be in front of the sensor, since side positions can occlude joints, degrading the skeleton. In the other side, the skeleton tracker can generate different skeletons for different individuals performing the same pose. These differences can degrade the invariance of pose descriptors, requiring some kind of skeleton normalization in extreme cases. Also, our method is limited to gestures composed of distinctive key poses. Gestures that requires very accurate movements can be troublesome for our learning machines. Finally, relying on key pose design and training may not be the friendliest solution for a casual user.

VII. FUTURE WORK

As the skeleton extraction and tracking algorithms still cope with large amount of noise from the sensor, robustness is a main issue for future work. A common problem for these algorithms is the lack of 3d information in some joint positions, requiring the user to face the camera. Working with two or more kinect sensors could be of help in robustly capturing and processing the skeleton stream.

The algorithm introduced here may be improved in three different directions. First, the use of time in recognizing gestures may be improved to distinguish complex gestures, using complementary SVM machines, maybe incorporating velocity of the joints. Second, automatic key pose generation may greatly ease the usability of the interface. In this setting, the computer should be able to decide the best set of key poses to train, in order to get good results in gesture recognition. Finally, SVM classification can be improved by taking into account the periodicity of the descriptor.

Acknowledgment: The authors would like to thank CNPq, FAPEAL, FAPERJ and FAPEMIG 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] F. Lv and R. Nevatia, "Single view human action recognition using key pose matching and viterbi path searching," in *CVPR*, 2007, pp. 1–8.

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

[4] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 2000.

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

[6] L. Cao, Z. Liu, and T. Huang, "Cross-dataset action detection," in *CVPR*, 2010.

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

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

[9] I. Laptev and T. Lindeberg, "Space-time interest points," in *ICCV*, 2003.

[10] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *CVPR Workshop for Human Communicative Behavior Analysis*, 2010.

[11] D. Weinland and E. Boyer, "Action recognition using exemplar-based embedding," in *CVPR*, 2005.

[12] 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.

[13] 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.

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

[15] A. W. Vieira, E. R. Nascimento, G. L. Oliveira, Z. Liu, and M. M. Campos, "Stop: Space-time occupancy patterns for 3d action recognition from depth map sequences," in *CIARP*, 2012.

[16] "Carnegie mellon university motion capture database," http://mocap.cs.cmu.edu.

[17] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber, "Documentation mocap database hdm05," Universität Bonn, Tech. Rep. CG-2007-2, 2007.

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

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

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

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

[22] J. W. Davis and A. Tyagi, "Minimal-latency human action recognition using reliable-inference," *Image and Vision Computing*, vol. 24, 2006.

[23] J. Zhang and S. Gong, "Action categorization with modified hidden conditional random field," *Pattern Recognition*, vol. 43, no. 1, pp. 197–203, 2010.

[24] M. Reyes, G. Domínguez, and S. Escalera, "Feature weighting in dynamic time warping for gesture recognition in depth data," in *ICCV Workshop on Consomer Depth Cameras for Computer Vision*, 2011.

[25] B. Schölkopf and A. J. Smola, *Learning with Kernels*. MIT, 2002.

[26] T. Vieira, A. L. Bordignon, A. Peixoto, G. Tavares, H. Lopes, L. Velho, and T. Lewiner, "Learning good views through intelligent galleries," *Computer Graphics Forum*, vol. 28, no. 2, pp. 717–726, 2009.