Simulating actions for learning

Philippe Saade, Philippe Joly, Ali Awada

Simulating actions for learning

Philippe SAADE, Philippe JOLY
philippe.saade@irit.fr, philippe.joly@irit.fr
SAMOVA
UPS IRIT
118 route de Narbonne F-31062 Toulouse Cedex 9, France

Ali AWADA
Lebanese University
Hadath, Lebanon
al_awada@ul.edu.lb

***Abstract*—This paper presents a novel approach for generating new actions to learn supervised algorithms such as the Adaboost in the context of human action recognition. Indeed, the learning process requires a large amount and variety of data. Our motivation in this work is to reduce the dependency on public databases and allow learning with small sets of actions. We overcome the problem of non-discriminatory action datasets for action recognition by enlarging a set of actions performed by different persons in different ways and captured by a Kinect. We present a way to enlarge the originally captured dataset from a Kinect device or from simply annotated data. This is done by combining the extrema of the action sequences into intervals, creating random points within them, and adding certain variables to discriminate the samples. These actions are learned and tested with a late fusion Adaboost using simple features and a strong classifier for each joint. Finally, a confidence coefficient is calculated and used as input of a higher level Adaboost classifier.**

Keywords—Action Recognition; Simulating actions; Late fusion Adaboost

I. INTRODUCTION

The recent emergence of the Kinect, a RGB and depth infrared-based camera capturing technology, helped facilitate and improve the study of human action recognition. In addition to its RGB-D capability, it provides a framework for finding 20 landmarks on the human skeleton and an Active Appearance Model (AAM) face recognition, as well as speech recognition.

Its usage rapidly spread across many fields: in health where it was used as a monitoring system in senior homes [1], security, health [2] [3] and of course gaming [4]. Furthermore, researchers benefitted of this technology to solve the problem of action recognition; they used the captured data as features by running them through algorithms such as Support Vector Model (SVM) [5], bag-of-features and Adaboost.

Of course, all of these require a large and discriminative action dataset for the training phase. Therefore, the same action should be performed by multiple persons in very different ways. Yet, the available online databases do not contain enough samples for each action and there is also the problem of asking people to perform a gesture or giving them instructions to perform actions they are not familiar with (e.g. not everyone will know how to perform a tennis smash for instance).

In this paper, we present a training model from simulated actions generated from a limited number of initially captured ones. These simulated actions are expected to be relevant to train a two-level Adaboost algorithm with Mid-Level features extracted from the 20 joints of the human body.

II. RELATED WORK

Using video data, algorithms such as the SVM with the bag-of-features, are very common for recognizing actions; as an example in [6], both algorithms were used on a YouTube dataset with the Actlets features... Also, we state the use of the 3D Joint Angles [7] of a human body as features for training a Hidden Markov Model or even simpler algorithms as the Multidimensional Dynamic Time Wrapping MD-DTW for aligning extracted features from each time instance [8]. Other studies have considered the depth data captured from a Kinect and results have been compared in the framework of the HARL international campaign [9].

The usage of the Adaboost algorithm in this contribution was inspired by previous work on face and action recognition. It showed good results with the well-known Viola Jones algorithm [10] and has proved to be a promising algorithm for its application on images with Mid-Level features for action recognition [11].

The Kinect has improved the accuracy and speed of tracking a person's location. Consequently, it is very easy to work with skeleton data; a C# code was posted online as Open Source for a DTW algorithm that aligns 3D coordinates from the Kinect's Joints and labels simple gestures [12]. This method is very simple and its learning phase does not require a lot of training data as opposed to the algorithms stated above which require much larger action datasets.

Many databases have been made available for public use for actions that have been captured by the Microsoft Kinect like the MSRC-12 database [13], which contains 12 gestures performed by 30 people. So far, the actions are simple: crouch, lift both arms, move hand, wear goggles, kick... The MSRC-12 database has been analyzed in [14] using a Hidden Markov Model and it has been stated that it is the largest one

that can be found. There are other databases available like the MoCap BVH [15] [16] which contains captured actions from 40 infrared sensors that follow white spots located on a person wearing black. The recorded data is converted into BVH files with flawless joint angles data. One of the other databases available is the MSR [16] which has a very large set of actions where some of them are performed numerous times. These actions are not very different since they have been performed by a maximum of 12 persons for the largest database. We also state the UMD-Telluride Kinect Dataset [17], the G3D gaming action dataset [18] and the Cornell Activity Dataset 60 [19]; however, the actions are captured by at most 4 subjects.

III. METHOD

Our approach consists of capturing the data from the Kinect, calculating the joint angles for simulation and finally testing the resulting simulated actions with a late fusion Adaboost.

A. Microsoft Kinect

This infrared-based camera uses RGB-D data to recognize the positions of the following 20 joints: Hip Center, Spine, Shoulder Center, Head, Shoulder Left, Elbow Left, Hand Left, Wrist Left, Shoulder Right, Elbow Right, Hand Right, Wrist Right, Hip Left, Knee Left, Ankle Left, Foot Left, Hip Right, Knee Right, Ankle Right and Foot Right. All this data is collected using the Microsoft Kinect SDK which is an API available online for public usage.

B. Capture and analysis

By recording few actions from the Microsoft Kinect, we simulate and add different variables to the initially captured actions. To achieve this, we first test the Kinect to find if the initial data generates a quantifiable error when tracking the joints. This will be taken into account in the simulation model.

To this end, we capture data from a mannequin for approximately 10 minutes from which, 100 frames are displayed in Fig. 1 and Fig. 2.

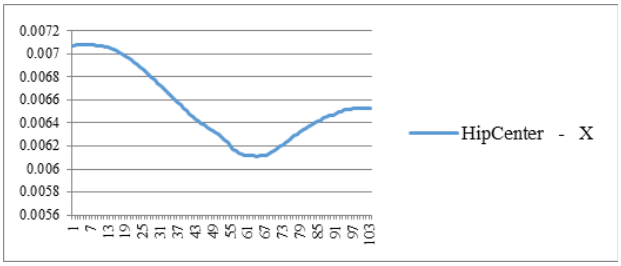

Fig. 1. 100 frames samples for captured Kinect Hip Center joint X position from a manequin. This graph plots variations in the detected hip position along the time. The scale of the vertical axis is the meter

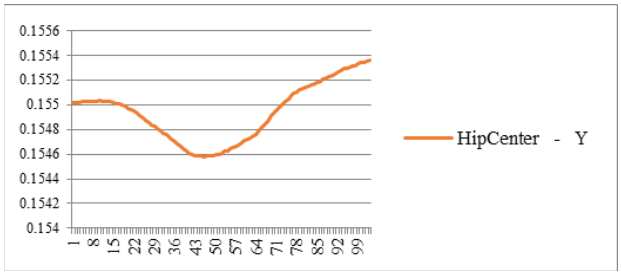

Fig. 2. 100 frames samples for captured Kinect Hip Center joint Y position from a manequin. This graph plots variations in the detected hip position along the time. The scale of the vertical axis is the meter

We conclude that the error is quantified in millimeters and sometimes less, and is therefore negligible. Yet, we should take into consideration the joints which are not tracked because they might be concealed or are located out of the camera's viewpoint.

We will therefore add a small random error to the generated action, later in the simulator.

C. Simulation algorithm

1) Calculating the angles

The 3D joint angles are expected to be more invariant and more discriminant for an action then the coordinates. We calculate them to be used as features to represent bone orientations for the connected joints. In this phase, we use the BVH angles from the MoCap database and also compute them in our custom way by considering every bone rotation from an initial body T position. Consequently, we obtain two angles for each bone as in the following algorithm:

a) Move the joints to form a T position for the body while conserving the length of each bone.

b) Calculate the angles (θ_x, θ_y, θ_z) *for each joint:*

For each coordinate as i:

Calculate next vector

$$V_{ni} = \left|(\text{nextjoint})_i - (\text{currentjoint})_i\right| \quad (1)$$

Calculate previous vector

$$V_{pi} = \left|(\text{previousjoint})_i - (\text{currentjoint})_i\right| \quad (2)$$

At T position (Unit Vector)

According to each orientation of the bone in T position,

If bone is vertical:

$$\theta_x = SignedAngle(V_{ny}, V_{pz}) \quad (3)$$

Calculate rotation Matrix R_x

$$V'_{nx} = V_{nx} \times R_x \quad (4)$$

$$\theta_z = SignedAngle(V'_{nx}, V_{py}) \quad (5)$$

If bone is horizontal:

$$\theta_y = SignedAngle(V_{nx}, V_{pz}) \quad (6)$$

Calculate rotation Matrix R_y

$$V'_{ny} = V_{ny} \times R_y \quad (7)$$

$$\theta_z = SignedAngle(V'_{ny}, V_{py}) \quad (8)$$

2) Aligning the local minima and local maxima of the action

a) Joint Angles with DTW

Let $k=\{\theta_x, \theta_y, \theta_z\}$ the joint angles as calculated from (1)

For every action coordinate sequence A_{ik}, we find the set of local minima and maxima and form another sequence S_{ik}. This step is supposed to reduce dependencies to the gesture dynamicity.

We choose a reference action R randomly.

We align all the sequences S_{ik} with R by applying the DTW algorithm. As a result, we obtain a sequence of intervals.

We join all of these intervals according to the frame number of the reference action's extrema, and obtain a final sequence of intervals I_{ik}

3) Choosing the points

We triple the size of every I_{ik} to increase the diversity of the actions' angles as following:

$$I_{ik} = [2 * LB(I_{ik}) - UB(I_{ik}), 2 * UB(I_{ik}) - LB(I_{ik})] \quad (9)$$

Where LB(I) and UB(I) are respectively the lower and the upper bounds of the interval I.

This step is a parameter that we choose arbitrarily to add the diversity. The more the I_{ik} is big, the more the simulated action changes from the initially captured ones.

Then, the points are chosen arbitrary by one of the following three methods:

i. For the first frame,

We choose an angle K_r as reference.

We get a random point inside an I_{ik} for K_r.

We calculate the rest of the K angles proportionally.

We find the rest of the frames' angles the same way.

ii. We choose randomly the first point P_0 (as in (i)) in I_i, then, to improve the action's smoothness, we calculate the rest of the points proportionally to the previous interval I_{i-1}.

iii. For each interval we calulate P from (i), P' from (ii) and average (i) and (ii) to smooth the action. We calculate all the intervals the same way.

4) Adding the variables

We multiply the length of the sequence that we've obtained from the previous step by a random number between 1 and the length of the longest initial action divided by the length of the shortest one.

We also add a small random error that changes the position of the joints and that may result from an unknown joint position caused by a miscalculation, an error in captured data as shown in III.B, or a hidden joint that does not appear in the Kinect's field.

5) Generating the action

After obtaining the sequence of points, the action is generated by simple proportionality between the frames and the points. Finally, the 3D coordinates are calculated by using rotation matrices.

D. Recognizing the actions with the Adaboost

Our aim in this paper is to develop the simulator, therefore, we choose the Adaboost as a training algorithm. It is a simple algorithm to implement and observe and has proven to have given significantly positive results for face recognition. Moreover, we choose the Adaboost for its boosting capabilities which selects the most discriminate features out of its input, considering that an action can be described with features that are specific to it.

Also, every action can be translated by the movement of specific joints. Therefore, we choose to run a separate Adaboost algorithm on each joint. This method was used previously in [11] and described as early and late fusion Adaboost.

Thus, we compute our features from each joint, then calculate a confidence coefficient and use it as an input to a high level Adaboost for final classification. The advantage of our method is that we consider a very developed set of features.

1) Features:

Since we do not perform any profound study on the features, we use a great number of simple features as an input of the Adaboost calculated from both coordinates and joint angles. For obvious reasons, we normalize the distance between the Shoulder Center and Hip Center.

All the features are stated in Table I.

2) Joints as Mid-level features

Since the Kinect gives us each of the joint position's coordinates separately, we calculate a set of features for each

joint. We then input each joint's low-level features in a separate Adaboost (Lower Level Classifier LLC).

We calculate a confidence coefficient (cc) similar to the one stated in [11] by first calculating the distance between the feature value and the threshold which separates the positive and negative feature values during the Adaboost decision for each action.

$$d_i = | \, featureVal\,ue - threashold \, | \qquad (10)$$

With *d'* the distance of the currently tested action

Finally, we obtain the cc as follows:

If $\overline{d_i} < d' < UpBound\ (d_i)$

$$cc = (d' - \overline{d_i}) / (UpBound\ (d_i) - \overline{d_i}) \qquad (11)$$

Else If $\overline{d_i} > d' > LowBound\ (d_i)$

$$cc = (d' - LowBound\ (d_i)) / (\overline{d_i} - LowBound\ (d_i)) \qquad (12)$$

Else $cc=0$

The confidence coefficient multiplied by the binary result of LLC testing is considered as a Mid-level feature and is used as an input for the Higher Level Classifier (HLC).

Consequently, the HLC will choose which joint defines the action the best. This procedure is shown in Fig. 3

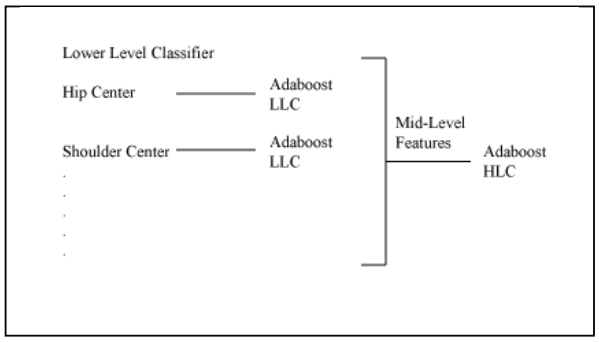

Fig. 3. Adaboost late fusion

The usage of the late fusion method permits the Adaboost to test each joint by itself and improve its classification instead of working only on the features.

TABLE I. FEATURES USED AS INPUT OF THE LOW LEVEL ADABOOST CLASSIFIER

Feature name	Variations and comments
Velocity	Mean max min
Acceleration	Mean max min
Signed velocity	Mean max min
Signed acceleration	Mean max min
Min	
Max	
Mean	
Distance for each joint according to previous joint	Mean max min, standard deviation
Local maxima	Min mean
Local minima	Max mean
Extrema	Mean max min Deviation Standard deviation
FFT coefficients	We consider only the 1/3 of the smallest Action

a. Calculated features for training and as an input of the Adaboost LLC

3) Training data

Since the Adaboost is a binary classification algorithm, we train all actions, one against another, and then display the results of false positive and false negative that we obtain in Table III.

We input 200 negative and 200 positive samples to train the Adaboost. Compared to the original dataset which is composed of some actions that have been captured and others that we took from MSRC-12 dataset, we count 39 MSRC-12 Start (flap hands in air), 39 MSRC-12 crouch, 11 tennis backhand drive, 10 tennis forehand, 4 raise hand in air and 8 hand wave. The results from the captured dataset are displayed in Table III.

4) Performances

We must also note that during the process of our algorithm, we are not working in real-time for calculating the features for each action; it takes on average 10 seconds on a 2.0 GHZ CPU to convert an action of approximately 100 frames to our set of features except for the MSRC-12 action dataset, which contains sequences of 1200 frames and more, requiring 20 minutes for processing each. After generating the features, finding the action type has a negligible time. It would be interesting to point out that during the learning of the LLC the algorithm can be launched in parallel since we are working on each joint by itself. Therefore, the learning process is a lot faster than a simple sequential Adaboost

5) Results

We test our dataset using 38 MSRC-12 start and crouch, 10 hand wave, 15 right shoulder up, 11 Tennis forehand and 11 Tennis backhand. We state the False Positive (FP) and False Negative (FN) results in Table III.

We note an improvement in the results as shown in Table III, especially for complicated gestures like tennis forehand drive and tennis backhand drive. Nevertheless, the False Negative classification increases when evaluating the MSRC-12 Crouch gesture, this is due to the limits of the simulator as explained in III.C.5

6) Limits of the Simulator

When simulating actions from very long ones, we note that the resulting actions are not very "Humanlike". So, we

ask 3 persons to identify 3 samples, chosen randomly from each simulated set after miming the action and stating its name. We obtain the results in Table II where we note, for each person, the number of actions that are identified correctly over the total number of presented actions

Problems occur when simulating from the MSRC-12 dataset because the gestures are repeated multiple times in the same action and at unknown frame positions. Hence, the DTW will not be able to perform a proper alignment of the local maxima and minima. The DTW fails to align very different sequences in size and content.

Nevertheless, after running the simulator on the MSRC-12 database which contains very large sets of data, the Adaboost identifies the initially captured actions.

We also note that the *tennis backhand drive* was confused 5 times with the *push object to right* which explains the low average from the tennis captured data.

IV. CONCLUSION

In this paper, we introduced a method to generate simulated actions since there are not enough available, in particular, labeled skeleton joint data captured from a Kinect device. We have applied a late fusion Adaboost algorithm on joint data using very simple features. After running some tests, the results are satisfying; however, we aim to develop the late fusion algorithm to be able to recognize more complicated action sequences with overlaying gestures. To this end, we will have to segment the sequences and focus on the classification by developing the Adaboost, and work in depth on the discriminatory features. In addition, we will compare the results from the Adaboost to those from commonly used algorithms such as the SVM while enlarging our datasets.

TABLE II. VISUAL RESULTS FOR THE SIMULATED SAMPLES

Action	Original dataset	Person 1	Person 2	Person 3	Avg. (%)
Raise both arms to the sides	MSRC-12	1/3	1/3	1/3	33
Crouch	MSRC-12	3/3	3/3	3/3	100
Push object with right hand to the right	MSRC-12	2/3	1/3	0/3	33
Wear Goggles	MSRC-12	1/3	1/3	1/3	33
Wave hands in air	MSRC-12	2/3	3/3	3/3	89
Walk	MoCap BVH	3/3	3/3	3/3	100
Kick	MoCap BVH	3/3	1/3	2/3	67
Shoulders up	Captured	3/3	3/3	3/3	100
Tennis backhand drive	Captured	2/3	2/3	2/3	67
Tennis forehand drive	Captured	3/3	2/3	2/3	78
Surrender	Captured	3/3	3/3	2/3	89
Hand Wave	Captured	3/3	3/3	3/3	100

b. 1/3: 1 action identified correctly from 3 actions.

c. Avg: the average of the positive result from Person 1, 2 and 3

TABLE III. FALSE POSITIVE (FP) AND FALSE NEGATIVE (FN) RESULTS

	MSRC-12 start (flap hands in air)		MSRC-12 crouch		Tennis forehand drive		Tennis backhand		Right shoulder up		Right hand wave	
	FP	***FN***	***FP***	***FN***	***FP***	***FN***	***FP***	***FN***	***FP***	***FN***	***FP***	***FN***
Captured	8.9%	12.6%	4.4%	3%	11.3%	23%	20.5%	1%	7.6%	26%	2%	4.3%
Simulated	5.1%	2.1%	3.5%	8.6%	2.6%	1%	8%	2%	2.8%	6%	3%	4.5%

REFERENCES

[1] S. Winbaum, "Microsoft's Kinect Technology Enters World of Seniors," 12 12 2011. [Online]. Available: http://www.retirementhomes.com/library/microsofts-kinect-technology-enters-world-of-seniors-2/.

[2] Kinect for Windows Team, "Unique Cancer Treatment Center alex's place Uses Kinect for Windows to Help Put Kids at Ease," Microsoft Corporation, 15 11 2012. [Online]. Available: http://blogs.msdn.com/b/kinectforwindows/archive/2012/11/15/unique-cancer-treatment-center-uses-kinect-for-windows-to-help-put-kids-at-ease.aspx.

[3] C. C. Martin, D. C. Burkert, K. R. Choi, N. B. Wieczorek, P. M. McGregor, R. A. Herrmann, and P. A. Beling, "Member, IEEE," in *A Real-time Ergonomic Monitoring System using the Microsoft Kinect*, Charlottesville, 2012.

[4] Kinect for Windows Team, "Build-A-Bear Selects Kinect for Windows for "Store of the Future"," Microsoft Corporation, 13 12 2012. [Online]. Available: http://blogs.msdn.com/b/kinectforwindows/archive/2012/12/13/build-a-bear-selects-kinect-for-windows-for-quot-store-of-the-future-quot.asp.

[5] C. Schuldt, I. Laptev, B. Caputo, *Recognizing Human Actions: A Local SVM Approach,* Stockholm: Computational Vision and Active Perception Laboratory, 2004.

[6] M. M. Ullah and I. Laptev, *Actlets: A novel local representation for human action recognition in video,* France: INRIA - Willow Project, Laboratoire d'Informatique, Ecole Normale Supérieure, 2012.

[7] Md. Z. Uddin, N. Duc Thang, J. T. Kim, and T-S. Kim, "Human Activity Recognition Using Body Joint-Angle," *ETRI,* vol. 33, no. 4, pp. 569-579, 2012.

[8] G.A. ten Holt, M.J.T. Reinders, E.A. Hendriks, *Multi-Dimensional Dynamic Time Warping for Gesture Recognition,* Landbergstraat: Delft University of Technology,, 2007.

[9] "ICPR - HARL 2012," LIRIS, INSA, CNRS, ICPR, 2012. [Online]. Available: http://liris.cnrs.fr/harl2012/.

[10] P. Viola, M. Jones, "Robust Real-time Object Detection," in *Second International workshop on statistical and computational theories of vision – modeling, learning, computing, and sampling, Vancouver, 2001*.

[11] A. Fathi and G. Mori, *Action Recognition by Learning Mid-level Motion Features,* Burnaby: School of Computing Science Simon Fraser University, 2008.

[12] G. Duncan, "Open source Kinect gesture recognition project, Kinect DTW," 24 8 2011. [Online]. Available: http://channel9.msdn.com/coding4fun/kinect/Open-source-Kinect-gesture-recognition-project-Kinect-DTW.

[13] S. Fothergill and H. M. Mentis and P. Kohli and S. Nowozin, "MSRC-12 Kinect gesture data set Microsoft Research Cambridge," ACM, 2012. [Online]. Available: http://research.microsoft.com/en-us/um/cambridge/projects/msrc12/.

[14] P. Gomes, S-M. Morgens, S-R. Smith, *Gesture Classification from Kinect Data,* Santa Cruz: CMPS242: Machine Learning, 2012.

[15] Carnegie Mellon University, "CMU Graphics Lab Motion Capture Database," [Online]. Available: http://mocap.cs.cmu.edu/.

[16] Yuan, Z. Liu, Y. Wu, "MSR Action Recognition Datasets and Codes," Microsoft Corporation, 6 2010. [Online]. Available: http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/default.htm.

[17] "UMD-Telluride Kinect Dataset," [Online]. Available: http://www.umiacs.umd.edu/research/POETICON/telluride_dataset/.

[18] "G3D: A Gaming Action Dataset," 28 5 2012. [Online]. Available: http://dipersec.king.ac.uk/G3D/.

[19] "Personal Robotics," Cornell University, 2009. [Online]. Available: http://pr.cs.cornell.edu/humanactivities/data.php#format.