Generation of Action Recognition Training Data Through Rotoscoping and Augmentation of Synthetic Animations

Nicola Covre[1(✉)], Fabrizio Nunnari[2], Alberto Fornaser[1], and Mariolino De Cecco[1]

[1] Department of Industrial Engineering, University of Trento, Trento, Italy
{nicola.covre,alberto.fornaser,mariolino.dececco}@unitn.it
[2] German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
fabrizio.nunnari@dfki.de

Abstract. In this paper, we present a method to synthetically generate the training material needed by machine learning algorithms to perform human action recognition from 2D videos. As a baseline pipeline, we consider a 2D video stream passing through a skeleton extractor (OpenPose), whose 2D joint coordinates are analyzed by a random forest. Such a pipeline is trained and tested using real live videos. As an alternative approach, we propose to train the random forest using automatically generated 3D synthetic videos. For each action, given a single reference live video, we edit a 3D animation (in Blender) using the rotoscoping technique. This prior animation is then used to produce a full training set of synthetic videos via perturbation of the original animation curves. Our tests, performed on live videos, show that our alternative pipeline leads to comparable accuracy, with the advantage of drastically reducing both the human effort and the computing power needed to produce the live training material.

Keywords: Action recognition · Random forest · Augmentation animation · Virtual Environment

1 Introduction

The mutual understanding and the collaboration between men and machines lead to a new concept of the relation between users and tools in several fields, such as automotive [1], medical [2,3], industry 4.0 [4], home assistance [5] or even gaming [6]. In these contexts, a more suitable way of managing the relation between man and machines relies on action recognition. The constraints introduced by joysticks, joypads, mice or any other physical controller will disappear. It is therefore important to study action recognition to fulfill the increasing need for teaching robots and make them 'understand' human movements. Augmented reality devices available in the market, like the Microsoft's Hololens[1] and

[1] https://www.microsoft.com/en-us/hololens – Feb 8th, 2019.

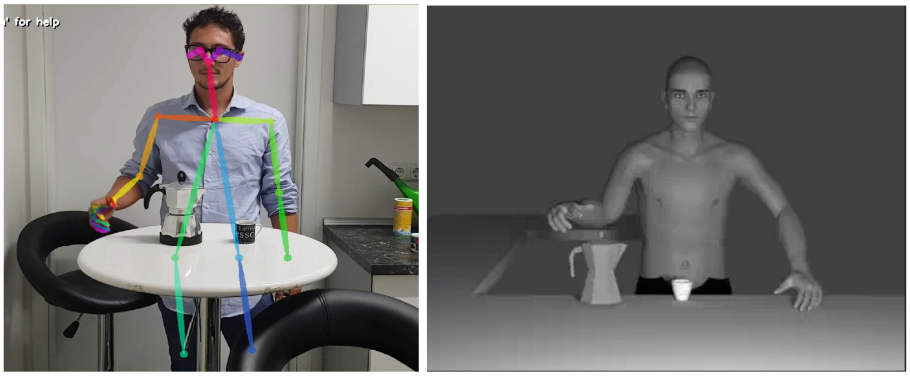


Fig. 1. Left: OpenPose applied on a live recorded video. Right: The same action recreated on a Virtual Human in the Virtual Environment.

the Meta 2[2], already allow for an interaction based on free hand movements, like pointing and grabbing[3].

Teaching to a machine how to recognize an action is performed by Machine Learning algorithms, which needs to be trained in order to produce a classifier. The classifier is then able to recognize recurring patterns. The training process is generally accomplished providing datasets. In the case of action recognition, a dataset consists of a number of videos of real persons performing actions, each video associated to information about the action performed. A person can perform several actions, or the same action several times, or a combination of the two. The datasets have to be properly organized and include a large amount of data to allow the artificial intelligence to discern and classify successfully different gestures and attitudes. However, this approach suffers of two main limitations. First, it is slow and expensive, due to the (usually) manual collection of data. Second, the data acquisition can not be automatized, and it is difficult to control parameters' evolution as long as the gestures are usually performed by actors.

We propose to analyze a new trend based on Virtual Environment (VE) data generation. Given an action, instead of collecting live videos of many real subjects performing the same action, we collect a single video of one subject performing a prototype of an action (See Fig. 1, left). Then, the prototype action is converted into a 3D animation of a virtual character (See Fig. 1, right) through Rotoscoping. Rotoscoping is a widely used technique in digital computer animation that consists of tracing over real videos images, frame by frame, to recreate a realistic replica.[4] Finally, the animation data of the prototype 3D animation are copied and altered, augmented, in order to generate a desired number of

[2] https://www.metavision.com/ – Feb 8th, 2019.
[3] https://www.ted.com/talks/meron_gribetz_a_glimpse_of_the_future_through_an_augmented_reality_headset?language=it.
[4] https://en.wikipedia.org/wiki/Rotoscoping.

variations. The resulting animation videos are used for training in place of the real ones.

The advantage of this approach is that the VE allows to control the parameters and automatize the data acquisition. In this way, it is possible to noticeably reduce the time and the effort required to collect the data.

This paper is structured as follows. Section 2 describes the approaches pursued in literature. Section 3 returns a general overview of our approach. Section 4 illustrates the procedures for VE and animation creation, data acquisition, and dataset generation. Sections 5 and 6 present the testing phase and discuss the results achieved. Finally, Sect. 7 concludes the paper by commenting on the limitations that affect the pursued methods and introduces future direction.

2 Related Work

Traditional approaches on gesture recognition are based on data collection from the real environment. Dataset generation has, usually, been pursued manually recording actors during the gesture performance [7–9]. However, machine learning requires often a large amount of data, but datasets are difficult to provide because of the need, together with actors, of a proper equipment, a post-production, and an accurate annotation phase. Eventually, it results to be an expensive and slow process.

It is also possible to rely on already available datasets, such as the KTH human motion dataset [10], the Weizmann human action dataset [11], INRIA XMAS multi-view dataset [12], UCF101 sport dataset [13]. These datasets are often used as reference for comparing different action recognition approaches. They are quite useful, because they allow to save a lot of time that usually data acquisition unavoidably takes. Unfortunately, they are frequently limited to specific cases [14]. This leads to use them mainly for different machine learning methods comparison and performance estimation.

Generally, both methods do not allow to control parameters as long as even the best actor can not introduce controlled variability thousand of times. Therefore, variability of the gesture and pattern distribution is populated repeating the action over and over, hoping to cover all the possible cases.

Training on synthetic data allows to manage these limitations. First of all, relying on a VE, data collection can be easily automatized. This relevant aspect permits to save time, funds, and effort, as long as it does not require neither actors, real cameras or equipment. Second, the procedure can control accurately all parameters, which leads to a more effective data generation: several versions of the same action can be achieved without repeating redundant information.

Similar research, which exploits a VE for Machine Learning, has been pursued on different classifications. For example, related to the autonomous driving, VE has been used to simulate a urban driving conditions. Data acquired from the synthetic world were used to generate a Virtual Training dataset for pedestrian detection [15].

Previous research gave us a reference guideline to follow as well as a benchmark for the results comparison. In addition, often they have been taken in to

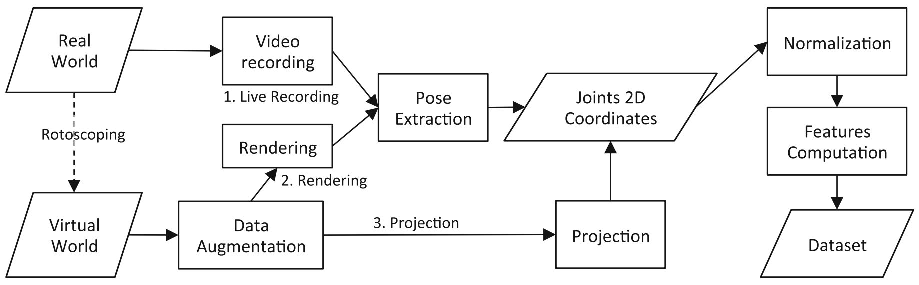


Fig. 2. Three different pipelines to produce the dataset for an action recognition system.

account during features selection. In our case, we focused the attention in establishing the performances of a VE dataset generation, obtained from synthetic data. In many research projects, the acquisition of the skeletal pose of a human can be obtained via affordable acquisition devices that can be found on the consumer market, like the Microsoft Kinect. Even if 3D skeleton acquisition provide much more spatial information [16], the Kinect is still an expensive solution with respect to a simple 640×480 2D camera [17]. For this reason, a 2D video-stream analysis has been pursued with Open Pose [18]. In this way, real and synthetic videos were processed in a similar way and in a reasonable amount of time.

Concerning the choice of a classification algorithm, a Random Forest [19] have been preferred with respect to Deep Neural Networks to limit the time needed for training and focus on the study of real vs. synthetic training material.

From the application point of view, gestures have been selected considering previous research projects in the field of rehabilitation [2,5,20]. In this context, the main goal of the application is to assist a mild cognitive patient while he or she is cooking, reconizing a few common kitchen actions such as: grabbing, pouring, mixing, and drinking.

3 Method Overview

Figure 2 outlines our approach for generating the dataset for gesture recognition. It shows the plots of three generation strategies, organized in three *paths* sharing the same starting and ending points.

Starting from top-left, *path 1* shows the most traditional approach, widely used in previous works [7,9]. It involves data collection from the Real Environment followed by a Pose Extraction step performed with Open Pose. The result of the pose extraction is a dataset (Joints 3D coordinates) containing the evolution in time of the 2D coordinates of the joints of the human skeleton. Obviously, the skeleton is not accurate, but only the approximation inferred by Open Pose through the analysis of 2D color video frames (See Fig. 1, left, for an example).

The two other paths, two alternative Virtual Training approaches, share a first step where a single live animation is transferred to a 3D editor (in our case

Blender) through rotoscoping (left side of the figure). Rotoscoping is a common animation technique consisting of drawing over a (semi-)transparent layer positioned over some reference animated material, which is visible in the background. This is a manual operation which leads to the Virtual Environment: a simplified representation of the real world where a virtual character performs a prototype version of some actions (See Fig. 1, right). The Virtual world provide the synthetic environment for the Data Augmentation process, which procedurally alters the animation data in order to produce a multitude of variations of the prototype animations. The augmented data is then used by paths 2 and 3.

Following *path 2*, the augmented animations are rendered as 2D videos, and the analysis proceed as for path 1, using OpenPose for pose detection. Differently, *path 3* skips the rendering and the joint coordinates are directly, and precisely, transformed into 2D coordinate via a computationally inexpensive world-to-screen projection.

Regardeless of the generation path, the time-evolving projected 2D coordinates proceed into a normalization process, followed by a feature extraction step, to become the reference training datasets.

4 Experiment: Action Recognition in a Kitchen

We applied our proposed approach in a real-world scenario set in a kitchen. The scenario accounts for a monitoring device which has to check if a user is correctly performing four actions: *grabing*, *pouring*, *mixing* or *drinking*. The goal of the assistant is to classify which of the four actions is being currently performed. The following subsections describe each of the phases needed to configure the classifier.

4.1 Virtual Avatar and Virtual Environment

In order to recreate an action which is feasible by a human, we edited a 3D environment composed of a virtual human, a kitchen, and three interactive objects: a coffe-pot, a coffe-cup, and a spoon (See Fig. 3). All the editing was performed using the Blender[5] 3D editor. The virtual human was generated using a freely available open-source add-on for Blender called MB-Lab[6]. This choice allowed us to save a conspicuous amount of time and rely on a well-made character. The VE creation takes the cue from a real environment. The reference objects have been crafted from scratch by one of the authors. Both the virtual human and the objects were disposed in the environment with the goal to realize animations for the four above-mentioned actions.

[5] https://www.blender.org/ – Apr 30, 2019.
[6] https://github.com/animate1978/MB-Lab – Feb 9th, 2019.

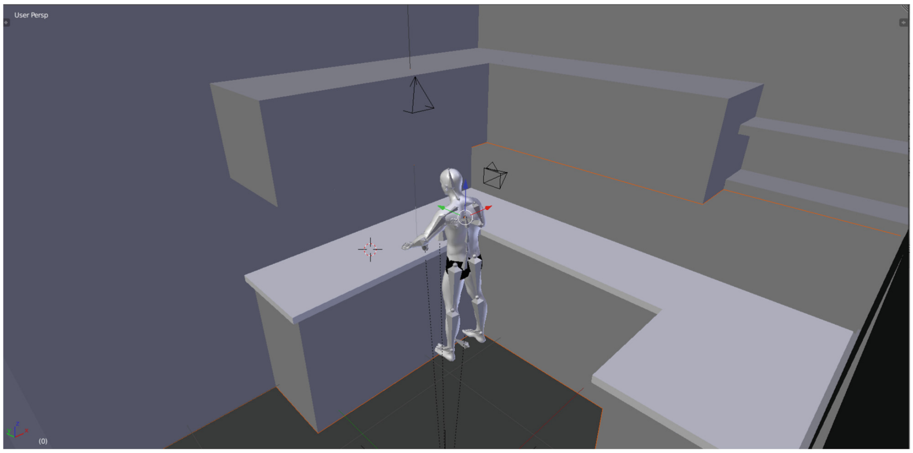


Fig. 3. Virtual Environment recreated in Blender 3D

4.2 Animation Generation Through Rotoscoping

As already anticipated in Sect. 3, a limited amount of reference animations where recorded live and meticulously replicated in the 3D environment through rotoscoping. We call these the *prototype* animations.

Two classes of animation have been generated. The first class of animations was created for *calibration* purposes. We identified five key locations in front of the body, with the right arm extended centrally, up, down, left, right, with respect to the related shoulder. For each position, we edited an animation where the hand is waving around its reference location. In this way, a simple check on features and armature comparison was easily pursued between real world and virtual one.

The second class of animations consists of four kitchen-related actions: grabbing, pouring, mixing, and drinking. These actions involve the interaction with the three objects in the environment: a coffee pot, a spoon, and a coffee cup. While the first class of animations was edited completely manually, the second class of animations went through a data augmentation process.

To ease the rotoscoping process, inverse kinematic (IK) controls were configured on the hands of the virtual character. The author of the rotoscoping procedure must only move the bone representing the palm of the hand. The IK routine will automatically compute the rotation values of all the joins of the arm, up to the spine. This is a standard time-saving technique widely used in the digital animation industry. Consequently, the animation curves defining an action affects only the IK controller, while the rotation of the intermediate bones are computed in real-time during the playback of the animation.

An *animation curve* is a function mapping the time domain to the value of an element of the VE. In our context, an animated element can be one of the six degrees of freedom (3 for position and 3 for rotation) of an object or skeletal joint. Hence, for gesture or posture recognition, animation curves are useful to track the evolution over time of the position and rotation of IK controllers in the 3D space, as well as the evolution of the rotation of the joints. The Animation curves are strictly related with the animation generation. During the animation authoring (in our case, through rotoscoping) the author stores a set of reference body poses as *key frames*. A key frame maps a specific time point to a value for each of the character's animation curves. Afterward, during animation playback, all the animation curves are generated by interpolating between the key-points saved along the animation timeline. In Blender, animation curves can be modified programmatically. This allows us to generate and modify automatically several animations with a routine.

4.3 Data Augmentation: Procedural Variation of the Prototype Animations

It is important to remember that the introduction of a significant variability that populate the feature distribution returns often better training outcome. For this reason, we implemented four data augmentation methods using the Blender's internal scripting system, which is based on Python.

Randomization of Objects' Position. This first method is designed to change, randomly, within certain intervals, the disposition of the interactive objects (coffee pot and cup) in front of the avatar. In this way, the bone structure is re-arranged by the inverse kinematic in order to reach, grab, and move these object in different locations the 3D space during the action development.

Camera Rotation. This method takes care of moving the camera in different positions and rotations around the vertical axis of the avatar. The function requires as input the total number of variations – always an odd number, in order to keep the central framing – and the angular step between two consecutive views. Eventually, the procedure takes as reference the frontal perspective, which is located at 90°, and moves the camera along a circumference of fixed radius. The animation videos were augmented with a combination of five different points of view and an angular step of 18°, leading to positions of the camera respectively at 54°, 72°, 90°, 108°, 126° (Fig. 4).

Time Scaling. The time scaling method controls the duration of the animation sequence, shrinking or expanding the time interval between two consecutive key-frames. This augmentation technique takes into consideration that different actors perform the same action along different time intervals. The transformation function takes as input the animation curve and the scaling factor. The process does not modify the first frame, which is kept as the reference starting point.

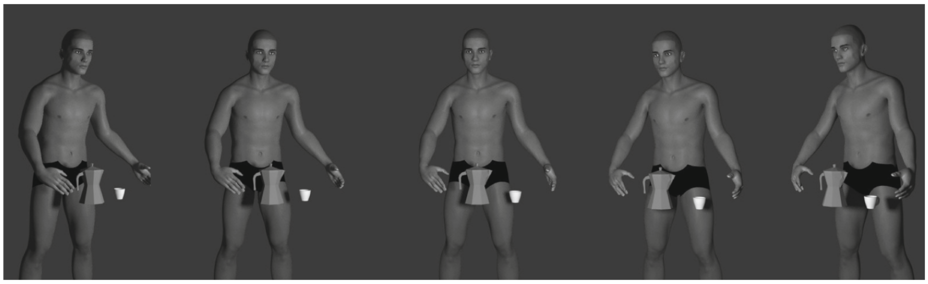


Fig. 4. Rendering of the same action frame using different camera viewing angles.

Perturbation of the Hand Trajectory. This method modifies the animation curves of the hand IK controller. The animation curves keep the track of the position and rotation changing along the animation.

This method requires as input the three animation curves covering the translation of the hand IK controller and introduces, in between the main key-frames, an extra key-frame which "perturbs" the interpolation connection, imposing a passage by a different point. This method aims to simulate the human feedback control while an object reaching is pursued. The introduced perturbation is randomized in a range with the same order of magnitude of the other key points.

Combining these methods as nesting dolls a large number of variations of the same gesture took place. These methods produced already a good amount of variability to appreciate the potentiality of the augmentation approach.

4.4 Joint Coordinate Collection

This phase consists of storing in a data frame the evolution of the movement of the body joints. Each line of the data frame will be associated to a time stamp. Each column of the data frame is one cartesian component (x or y) of the join 3D position after a projection on the 2D image space. Table 1 shows an example.

Table 1. Example of the data frame resulting from the projection of the joint coordinates in time.

Time (ms)	hand.x	hand.y	wrist.x	wrist.y	elbow.x	elbow.y	...
0.00	0.123	0.234	0.345	0.345	0.762	0.324	...
16.66	0.124	0.232	0.346	0.344	0.760	0.328	...
33.33	0.125	0.230	0.348	0.342	0.759	0.329	...
...	...	...	...	...	...	...	

Given an input animation, the joint coordinates collection was performed using two different methods: (i) pose estimation using OpenPose, and (ii) direct

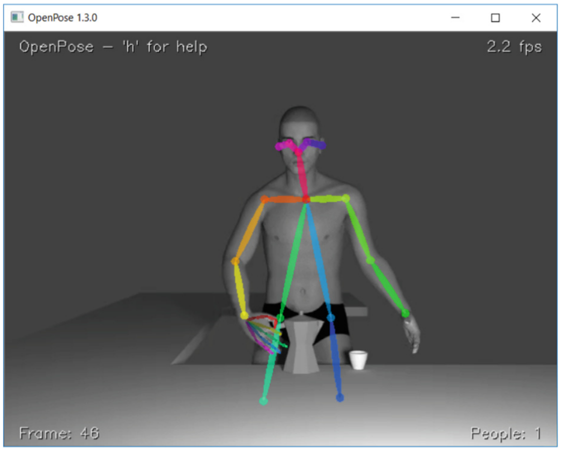


Fig. 5. OpenPose applied on a frame of a rendered video

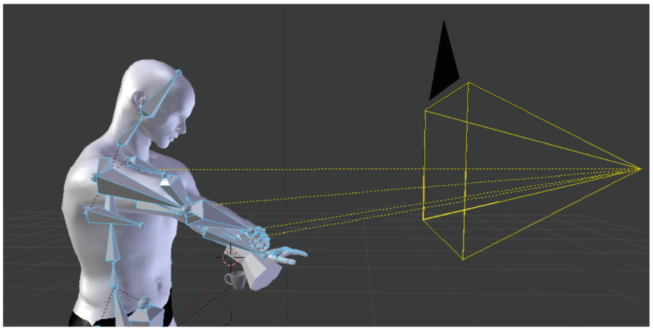


Fig. 6. Joint coordinates projection on the camera view plane

projection (in Blender) of the 3D joint coordinates into the 2D camera plane. With reference to Fig. 2, the first method is used by paths 1 and 2, while the second method is used in path 3.

As already mentioned, the first method exploits Open Pose and reflects the more traditional approach, as it is possible to point out from Fig. 5. Open Pose extrapolates the skeleton animation from 2D videos and returns the coordinates on a pixel-based metric. The result of the processing on synthetic videos is very similar to the one on real videos. With this method, the processing is highly computational expensive because the augmented animations must be rendered.

Differently, the second method skips the rendering step and uses the virtual camera embedded into the virtual environment to collect the 3D joint coordinates and projects them on the 2D virtual screen (See Fig. 6). These coordinates are expressed in a meter-base metric.

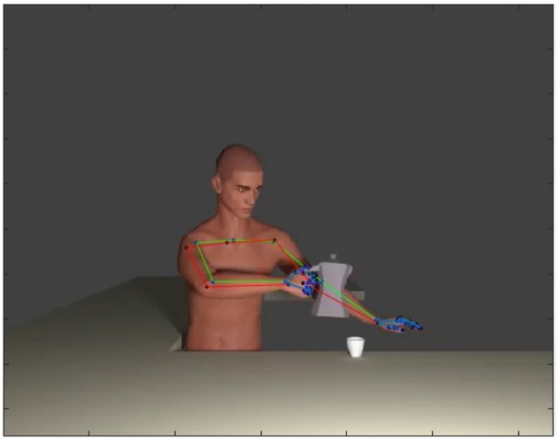


Fig. 7. Comparison between Open Pose (red) and projected coordinates (green) skeletal structures on the same frame. (Color figure online)

This strategy emulates the Open Pose video analysis with the difference of avoiding any error in the estimation of the joint positions. The absence of error in the projection creates a discrepancy between the perfectionism of the data set with respect to the estimation errors introduced by Open Pose (See Fig. 7). In this case, perfectionism is not an advantage, because Open Pose will be anyway used to analyse real world video during the action recognition on human subjects. This implies that training data and test data will come from different distributions.

Even if the second method results to be faster with respect to the first one, it returns a skeleton that appears to be slightly different from the Open Pose standard structure. The main difference being the presence, in the Open Pose structure, of a central upper-spine bone always positioned as mid-point between the clavicles. Hence, we aligned the two structures by computing, in the Blender skeleton, the coordinates of a fictional spine bone by averaging the coordinates of the clavicles.

Regardless the used method (pose extraction or projection), the coordinates obtained as output are then collected, normalized, and used for computing Classification Features.

4.5 Coordinates Normalization

The projected joint coordinates have to be pre-processed before features computation takes place. As anticipated, this occurs because the two methods use different coordinate systems: the open pose output is in a pixel-based metric, while the projected coordinates are expressed in meters (unit of reference used in the VE). See Fig. 8, left.

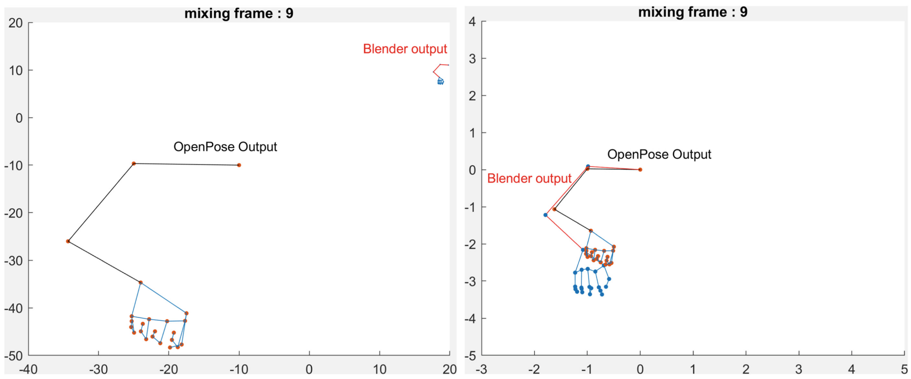


Fig. 8. Comparison between the coordinates projected from Blender skeleton structures and Open Pose skeleton structure. Left: before normalization. Right: after normalization.

Eventually, the midpoint between the two shoulder sets the common origin of the reference system, which represents, from now on, every other couple of joints coordinates. Therefore, every joint coordinate was translated to have the midpoint as origin.

After that, the two representations needed to be scaled. The clavicle extension appeared to be the most suitable length to choose as reference. In fact, it remains constant enough during the entire gesture performance. This is due to the fact that the analysis has been conducted mainly with respect to a frontal camera point of view, which did not change remarkably over time. Another advantage of this strategy is that farther and closer subjects are normalized and analyzed as they were at the same distance.

Hence, every distance is divided by the reference length in order to represent every skeleton structure with the same dimensions. In this way it has been possible to overlap accurately the Open Pose skeleton and the Blender one (See Fig. 8, right).

4.6 Features Selection

The feature selection was the most critical step of this work, as it can deeply affect the final result and jeopardize the output performances. Literature and academic paper of reference mainly suggest two approaches. The first one, based on a Kinect data acquisition or 2D skeleton extraction, compute the features from geometrical relations between joints, such as distances, angles or relative velocities [7,9,21]. The second approach, based on 2D images analysis, suggest a Deep Learning approach where pixels are provided to the algorithm and the best features combinations are selected by the network itself [1,17].

For this work, we used random forests as classifiers, hence features were selected manually. Different choices have been pursued and performances were compared with respect to each other to find out the best combination.

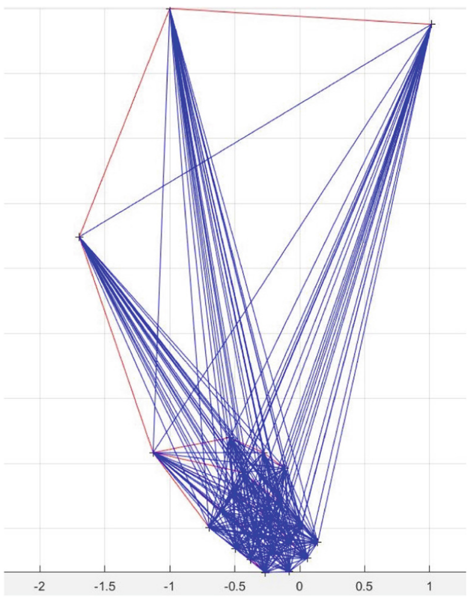


Fig. 9. Feature set one: all the possible combination of relative distances between two non-consecutive joints.

The computation of the features dataframe uses a temporal sliding window of 5 frames. The first 5 rows of the coordinates dataset produces the first row of the features dataframe. The procedure iterates by shifting the time window of 2 frames at each step, until the end of the coordinates dataset is reached.

The time windows size (5) and sliding step (2) were empirically chosen considering the high frame rate (compared with the gesture duration), and with the goal of reducing the similarities with respect to the previous five frames and the consecutive ones. Therefore reducing the chance of computating two identical samples. This approach has also the advantage of acting as low-pass filter against the high frequency jittering of the coordinates returned by Open Pose.

The computed features (columns of the features dataframe) belong to four categories: raw normalized coordinates, distances between joints, joint movement speed (computed over the 5 frames of the time window), and angles (of the armpit and elbow).

Data analysis and features computation have been accomplished focusing on the right arm, which involves 23 bones, from the clavicle and down to the fingers. The results reported in the remainder of this paper come from the two feature sets which returned the best results. The best choice is a list of features computed as all the possible combinations of distances between two non-consecutive joints (Fig. 9), while the second selection, simpler and more intuitive, collects the normalized $<x, y>$ joints coordinates.

The two choices of features returned different results on different groups of actions. In fact, for the first group, composed by the five different hand position – mainly accomplished as a calibration experiment – the x, y joint coordinates returned the best result. Differently, the list of relative distances between all the non consecutive joints returned a better outcome in classifying the grab, pour, mixing, and drinking actions.

Moreover, further comparison and focused analysis led us to introduce also angles measurements of the shoulder and the elbow joint and to exclude the relative distance combination between fingers – due to Open Pose low accuracy in locating them – in order to increase the performances.

5 Training and Testing

Once the features are computed and filled up the dataset, the Machine Learning method receive them as input in order to proceed with the training. When training is done, the performance of the algorithm are evaluated using a test dataset.

It is worth now reminding that the purpose of this study is mainly to assess the advantages (or disadvantages) of using a virtual environment, rather than a real one, to train a classifier. Hence, the classifiers were trained with two different type of video sources: the first made of live-recorded videos of humans (path 1), the second with the video material generated synthetically (paths 2 and 3). However, in order to assess the behaviour on real scenarios, the test set was always entirely composed of live videos.

Training Material. For each action, each prototype animation has been augmented into nine videos of roughly one second each; for a total of 36 videos. At the render rate of 60 frame per second, we collected around 28 feature samples per video. The same amount of videos have been collected from the real environment in order to generate a balanced training dataset using the traditional approach.

Similar criteria applied for testing material collection from the Real Environment. Four different actors performed nine times each action, for a total of 144 videos. The four actors accomplished each gesture in different amount of time. Consequently, even if the same number of videos have been exploited, for each action there is a slightly different amount of samples.

Table 2 reports on the time needed to create a dataset for each of the three generation paths.

Environment setup refers to the time needed to prepare the live scene, or the 3D editing. Data acquisition refers to the time needed for live shooting, rendering, or coordinates projection. Skeleton Extraction is the time taken by OpenPose. Data Conversion refers to the time needed to transform the skeleton extracted coordinates in matrix form, which is more suitable for Matlab operations. Converting Open Pose data is slower because the coordinates of each frame are saved in a separate JSON file. Finally, Feature Computation refers to

Table 2. Measured time for the creation of the training material

Training method	Environment setup	Data acquisition per single action	Skeleton extraction per single action	Data conversion per single action	Feature computation per single action
Traditional (path 1)	30 s (per single action)	1 min	3 min	6 s	4 min
VE rendering (path 2)	30 min	10 s	3 min	6 s	4 min
VE projection (path 3)	30 min	3 s	n/a	2 s	4 min

the conversion from coordinates to features dataset, as described above in this section.

With this timings, the creation of the 36 videos of the training set took a total of 292.1 min for path 1, 291.6 min for path 2, and 177 min for path 3. The latter, represents a 40% time saving compared to path 1. It is possible to appreciate, that the virtual training approach results to be much faster if the training require a large dataset composed by many actions. In different conditions of video length, computational procedures or feature selection, timings would proportionally scale. The traditional approach would eventually remain the slowest, while the coordinates projection based would always result to be the fastest.

The Training and Testing processes exploit two different methods. The first one is the Classification Learner of Matlab, available as toolbox already implemented and accessible from the toolbar of the main window. The second one is an experimental Random Forest developed by the University of Trento [19].

MATLAB Classification Learner. The training procedure of the Matlab Classification Learner asks the user, at first, to select a validation strategy. Due to the large amount of samples – rows of the dataset – the "Cross out Classification" and the "No Classification" options have been discarded, while the "Hold Out Selection"[7] appeared to be the most suitable one. This choice allows selecting the percentage of samples that the trained classifier uses for the validation. This subgroup of samples does not take part to the training, but provides the entries for the validation procedure.

Following, the user is further asked to select the list of features to use for the training. This is useful to experiment with different combination of features or to simply discard a subset if is already known that some features are less relevant with respect to the others, and therefore can be neglected.

Once the Classification Learner has loaded the dataset, it is possible to choose the Classification Algorithm, selecting it from a list of available ones. We chose to use the Fine Tree algorithm, because we wanted to compare a basic decision Tree

[7] https://www.mathworks.com/help/stats/select-data-and-validation-for-classification-problem.html – Apr 30, 2019.

based classifier with the sigma-z Random Forest developed by the University of Trento [19].

After selecting the classifier algorithm, the training process begins, followed by validation and testing. The training uses the features measurements to learn how to classify the action. The validation set is used during the training to help adjusting the hyper parameters and improve the performance of the classifier. Finally, the real performance of the classifier is measured on the test set.

Using Matlab Classification Learner, Validation and Testing process are quite different to each other. During our tests, the validation percentage returned often very high accuracy levels, between 98% and 100%. This means that the classification algorithms works extremely well on video of the same type. However, even when training on synthetic videos, test is performed on live videos. For this reason, testing process results to be far more significant because it represent how good the VE data emulate the Real Environment ones.

Random Forests. The Sigma-z Random Forest, proposed by the University of Trento, works differently. First of all, it is important to introduce that this Random Forest does not process data in the same way pursued by traditional Random Forests [19]. In fact, it relies on a probability computation analyzing two main aspects. The first one is the population uncertainty, computed considering the variability over the whole dataset, the second one is the sampling uncertainty, which reflects the uncertainty related to the new entry measurement (through the variance computed over the five frames of the feature computation). These two aspects, combined together, determine the probability for a certain new entry to belong to a specific class. In this way, it is possible to provide also a confidence level for each new entry classification in addition to the accuracy level, which is usually reported after every testing procedure. This remarkable aspect allows knowing the percentage of confidence the classifier is providing while every new entry is classified, in other words "how much it is sure". The Random Forest does not accomplish the validation on a subgroup of samples of the training dataset, but it directly pursues the validation using the testing dataset. Hence, in the Random Forest case, Validation and Testing are actually the same process.

Results. We decided to report the results of the Random Forest proposed by the University of Trento, because it returned an outcome which is slightly better with respect to the Classification Learner of Matlab. Each one of the following tables is an average result of multiple testing sessions conducted separately with different actors. Tables 3, 4 and 5 report the confusion matrix and the overall accuracy achieved during the testing procedure over the four cooking actions. Table 3 presents the results obtained from classifier trained with real environment videos (Traditional approach, path 1). Tables 4 and 5 report the results obtained training the classifier with a virtual environment through video rendering (path 2) and avatar joint coordinates projection (path 3), respectively.

Table 3. Confusion matrix of the classifier trained with the **traditional approach**, using real environment videos

True\Predicted	Drinking	Mixing	Grabbing	Pouring
Drinking	94.27%	1.67%	1.32%	2.74%
Mixing	13.38%	68.79%	16.36%	1.47%
Grabbing	0.96%	23.45%	71.47%	4.12%
Pouring	10.16%	1.87%	2.71%	85.26%
Average accuracy	79.94%			

Table 4. Confusion matrix of the classifier trained with the **proposed approach, path 2**, using video rendering.

True\Predicted	Drinking	Mixing	Grabbing	Pouring
Drinking	96.02%	1.3%	0.25%	2.43%
Mixing	1.03%	59.02%	37.15%	2.80%
Grabbing	0.91%	17.28%	81.16%	0.65%
Pouring	1.1%	1.7%	3.08%	94.12%
Average accuracy	82.58%			

It is possible to state that the reached accuracy is averagely over 80%. During the multiple training and testing trials the average accuracy mainly fluctuated between 71% and 85%.

Both the approaches based on VE (Tables 4 and 5) returned results performances comparable with the traditional approach. Moreover, there is not a conspicuous difference between the two VE based approaches.

6 Discussion

Even if average performances of the proposed method results to be definitely promising, sometimes the classifier has difficulties in classifying a specific class, lowering the session performances around 70%, while usually state of art accuracy values results to be higher than 80% [2,6]. Higher accuracy resulted difficult to pursue mainly due to several limitations which occurred during data acquisition and feature selection. Even if Open Pose skeleton mismatching has been compensated through data filtering – selection based on the joint acquisition confidence value, provided by Open Pose – the 2D spatial characterization between actions has been hardly achieved.

The cooking group of actions characterizes each gesture with geometrical features, such as the rotation of the wrist or the hand opening/closure tracking through the relative position between fingers. However, these intuitive features are quite difficult to monitor properly from a 2D flat screen. This problem is further increased due to the prospective point of view. Hence, while the five hand

Table 5. Confusion matrix of the classifier trained with the **proposed approach, path 3**, using projected coordinates.

True\Predicted	Drinking	Mixing	Grabbing	Pouring
Drinking	92.14%	0.74%	1.15%	5.97%
Mixing	1.79%	57.98%	38.28%	1.95%
Grabbing	1.0%	16.19%	82.16%	0.65%
Pouring	1.5%	1.2%	2.97%	94.33%
Average accuracy	81.65%			

position group of gesture – just used for calibration purposes and, therefore, not widely reported – has been better classified by x, y features, the second group (of cooking actions) returned better results comparing a much larger number of features. These features, as mentioned before, are the relative distances between non consecutive joints, angles between joints, and their average difference between two consecutive frames (within the interval of five frames).

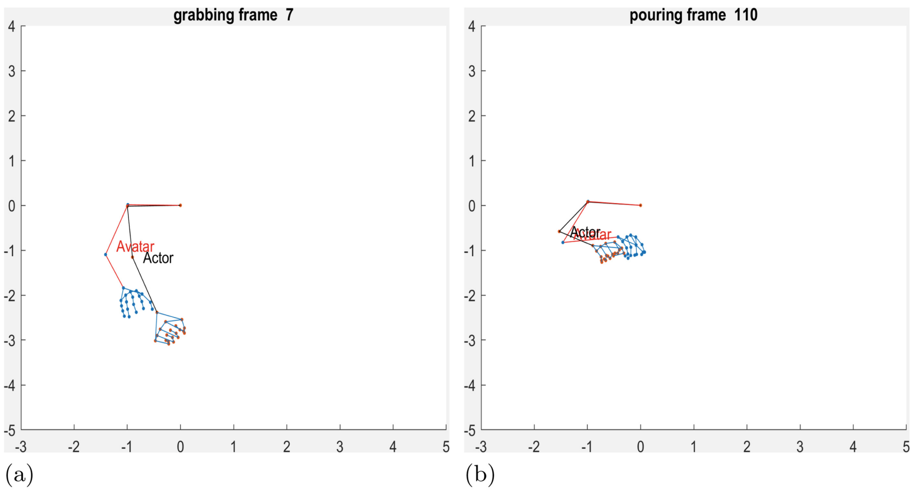
In fact, the lack of a depth component affected considerably this second group testing procedure, as soon as cooking actions discernment relied on features less distinct and less trackable in a 2D context.

It is intuitive to understand–and possible to observe–that the accuracy level is strictly related to the feasibility of the Avatar gestures features with respect to the real ones that the actor performs. For this reason, a meticulous iterative analysis has been performed over the gesture evolution in order to understand, once for all, how to realize a natural movement. The group of the five hand position helped remarkably in this process, because of the simplicity of the action allowed to focus more deeply on the joint representation aspects. Furthermore, translation and scaling augmentation procedures played an important role in coordinate feasibility, forcing the similarity between the skeleton structures of the real and the virtual environments. As it is possible to state from the images of Fig. 10, all along the four cooking actions performance, the sequence of the avatar joints render a skeleton structure indistinguishable from the actors' ones.

7 Conclusions

This paper presented a novel approach to train machine learning systems for the classification of actions performed by humans in front of video cameras. Instead of training on a multitude of live-recorded real-world videos, we proposed to realize a limited amount of live videos and manually translate them in to 3D animations of a virtual character. The few hand-crafted 3D animations are then augmented via procedural manipulation of their animation curves.

Tests on real, live videos showed comparable results between the two training approaches. Hence, despite the action recognition accuracy (81%–84% with the classifiers used in this study) being far from human-level, this research allowed us to show the versatility and the performances of the virtual training.



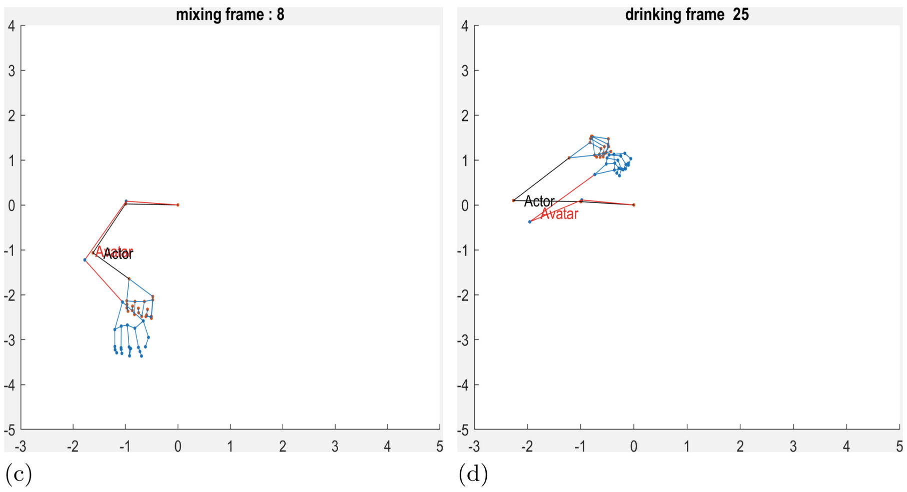


Fig. 10. Comparison between the Avatar (Synthetic) skeleton and the Actors' one.

Applied on a large scale set of different actions, the proposed approach can contribute in significantly cutting the time needed for the collection of training material and, consequently, lower production costs, yet with no loss in performances.

As a matter of improving the classification accuracy, it is important to remember that the entire work has been pursued based on a 2D analysis. It is thus reasonable to believe that performances could further increase if an extension to the 3D analysis – for example using a Microsoft Kinect – is accomplished. This aspect would lead to a more accurate similarity between the real and the virtual skeletons and to more precise information regarding the joint coordinates position and rotation.

It is also important to remark that the basic animation relies on the rotoscoping of one single person, while the testing procedure has been conducted over a sample of four people. A more general approach – which should involve several different people, each one with different personal movement style – may encounter more difficulties in discriminating gestures with similar posture patterns, such as mixing and pouring, and would lead to a lower accuracy estimation, independently from the followed path (live or synthetic).

As a possible improvement, it would be also possible to skip skeleton recognition and rely on deep convolutional neural networks for an end-to-end mapping between frame pixels and action class. This would include, for example, an object recognition algorithm. The object recognition would support the gesture recognition one, helping the selection of the candidate action in relation to the detected object.

References

1. Nilsson, M.: Action and intention recognition in human interaction with autonomous vehicles (2015)
2. D'Agostini, J., et al.: An augmented reality virtual assistant to help mild cognitive impaired users in cooking a system able to recognize the user status and personalize the support. In: 2018 Workshop on Metrology for Industry 4.0 and IoT, pp. 12–17. IEEE (2018)
3. Dariush, B., Fujimura, K., Sakagami, Y.: Vision based human activity recognition and monitoring system for guided virtual rehabilitation. US Patent App. 12/873,498, 3 March 2011
4. Gorecky, D., Schmitt, M., Loskyll, M., Zühlke, D.: Human-machine-interaction in the industry 4.0 era. In: 2014 12th IEEE International Conference on Industrial Informatics (INDIN), pp. 289–294. IEEE (2014)
5. Mizumoto, T., Fornaser, A., Suwa, H., Yasumoto, K., De Cecco, M.: Kinect-based micro-behavior sensing system for learning the smart assistance with human subjects inside their homes. In: 2018 Workshop on Metrology for Industry 4.0 and IoT, pp. 1–6. IEEE (2018)
6. Bloom, V., Makris, D., Argyriou, V.: G3D: a gaming action dataset and real time action recognition evaluation framework. In: 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 7–12. IEEE (2012)
7. Papadopoulos, G.T., Axenopoulos, A., Daras, P.: Real-time skeleton-tracking-based human action recognition using kinect data. In: Gurrin, C., Hopfgartner, F., Hurst, W., Johansen, H., Lee, H., O'Connor, N. (eds.) MMM 2014. LNCS, vol. 8325, pp. 473–483. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04114-8_40
8. Wang, C., Wang, Y., Yuille, A.L.: An approach to pose-based action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 915–922 (2013)
9. Li, W., Zhang, Z., Liu, Z.: Action recognition based on a bag of 3D points. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 9–14. IEEE (2010)

10. Blank, M., Gorelick, L., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes, pp. 1395–1402. IEEE (2005)
11. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local SVM approach. In: Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, vol. 3, pp. 32–36. IEEE (2004)
12. Weinland, D., Ronfard, R., Boyer, E.: Free viewpoint action recognition using motion history volumes. Comput. Vis. Image Underst. **104**(2–3), 249–257 (2006)
13. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: a dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
14. Poppe, R.: A survey on vision-based human action recognition. Image Vis. Comput. **28**(6), 976–990 (2010)
15. Marin, J., Vázquez, D., Gerónimo, D., López, A.M.: Learning appearance in virtual scenarios for pedestrian detection. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 137–144. IEEE (2010)
16. Abbondanza, P., Giancola, S., Sala, R., Tarabini, M.: Accuracy of the microsoft kinect system in the identification of the body posture. In: Perego, P., Andreoni, G., Rizzo, G. (eds.) MobiHealth 2016. LNICST, vol. 192, pp. 289–296. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58877-3_37
17. Lie, W.-N., Le, A.T., Lin, G.-H.: Human fall-down event detection based on 2D skeletons and deep learning approach. In: 2018 International Workshop on Advanced Image Technology (IWAIT), pp. 1–4. IEEE (2018)
18. Simon, T., Wei, S.-E., Joo, H., Sheikh, Y., Hidalgo, G., Cao, Z.: OpenPose (2018)
19. Fornaser, A., De Cecco, M., Bosetti, P., Mizumoto, T., Yasumoto, K.: Sigma-z random forest, classification and confidence. Meas. Sci. Technol. **30**(2), 025002 (2018)
20. Fornaser, A., Mizumoto, T., Suwa, H., Yasumoto, K., De Cecco, M.: The influence of measurements and feature types in automatic micro-behavior recognition in meal preparation. IEEE Instrum. Meas. Mag. **21**(6), 10–14 (2018)
21. Yoo, J.-H., Hwang, D., Nixon, M.S.: Gender classification in human gait using support vector machine. In: Blanc-Talon, J., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2005. LNCS, vol. 3708, pp. 138–145. Springer, Heidelberg (2005). https://doi.org/10.1007/11558484_18