Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc

An Innovative System for Supporting Acquisition and Reproduction of Gestures in Storytelling Humanoid Robots

Agnese Augello[a], Angelo Ciulla[b], Alfredo Cuzzocrea[c], Salvatore Gaglio[a,b], Giovanni Pilato[a] and Filippo Vella[a]

[a]*Institute for High Performance Computing and Networking National Research Council of Italy (ICAR-CNR), Palermo, Italy*
[b]*Dipartimento di Ingegneria, Università degli Studi di Palermo, Palermo, Italy*
[c]*iDEA Lab, University of Calabria, Rende, Italy*

ARTICLE INFO

Article History:
Submitted 4.26.2020
Revised 7.14.2020
Accepted 7.27.2020

Keywords:
Storyteller
humanoid robot
gesture
automatic mimic

ABSTRACT

The work describes a module that has been implemented for being included in a social humanoid robot architecture, in particular a storyteller robot, named NarRob. This module gives a humanoid robot the capability of mimicking and acquiring the motion of a human user in real-time. This allows the robot to increase the population of his dataset of gestures. The module relies on a Kinect based acquisition setup. The gestures are acquired by observing the typical gesture displayed by humans. The movements are then annotated by several evaluators according to their particular meaning, and they are organized considering a specific typology in the knowledge base of the robot. The properly annotated gestures are then used to enrich the narration of the stories. During the narration, the robot semantically analyses the textual content of the story in order to detect meaningful terms in the sentences and emotions that can be expressed. This analysis drives the choice of the gesture that accompanies the sentences when the story is read.

1. Introduction

Nowadays, robots collaborate more and more with human beings, helping them to achieve different goals. In this context, the recognition and subsequent reproduction of gestures becomes extremely important.
Considering that communication between human beings is based not only on verbal interaction but also through nonverbal cues, a humanoid robot capable of interacting with people combining both speech and gestures would improve the effectiveness of social robots. On the other hand, other studies like [28, 22, 10] consider *knowledge management techniques* (e.g., [10]) to improve this phase. Sometimes, it is possible to enrich or even replace one's narrative exposure with gestures. This ability can be particularly relevant in a storytelling context, and, in general, in assistive robotics. In this context, a gestural expressiveness is indeed desirable to strengthen the meaning conveyed by the words spoken by a robot.
Processing the semantic content of a text and adding the execution of proper gestures while the robot is telling something, or generally, while it is interacting with human beings, is essential for improving the effectiveness of communication, avoiding trivial and boring situations. The capability to exhibit emotions and to show expressiveness during storytelling is fundamental to obtain an effective engagement [32] [13] [3]. As reported in [35], the use of bodily expressions in robots facilitates a mood induction process of the story and improves the storytelling experience.

In our Lab we are working on *NarRob*, a social robot

*Giovanni Pilato
**Alfredo Cuzzocrea
agnese.augello@icar.cnr.it (A. Augello); angelo.ciulla@community.unipa.it (A. Ciulla); alfredo.cuzzocrea@unical.it (A. Cuzzocrea); salvatore.gaglio@unipa.it (S. Gaglio); giovanni.pilato@cnr.it (G. Pilato); f.vella@pa.icar.cnr.it (F. Vella)
https://www.icar.cnr.it/persone/augello (A. Augello); http://si.deis.unical.it/cuzzocrea (A. Cuzzocrea); https://www.unipa.it/persone/docenti/g/salvatore.gaglio (S. Gaglio); https://www.icar.cnr.it/persone/pilato (G. Pilato); https://www.icar.cnr.it/persone/vella (F. Vella)
ORCID(s):

DOI reference number: 10-18293/JVLC20 20-N1-017

which plays the role of a storyteller [4] [3]. It is embodied in the Softbank robotic platforms Pepper and NAO, and it can show a body expressiveness to emphasize the semantic and the emotions arising from the text. In an early version, NarRob performs an analysis of the text in a story, and it then annotates each sentence by *a)* the emotion expressed in the text and *b)* the terms that can be associated with gestures, which are then performed while the robot is telling the story. In the specific, NarRob exploits: 1) a gesture module to acquire gestures and store them in a knowledge base 2) a chatbot module integrating an OWL ontology, used by the robot to have a conversation with users, before and after the narration, to deepen specific concepts involved in the narration; 3) an annotation module that makes it possible for the robot to detect meaningful terms as well as an emotion expressed in the text and to annotate the story with specific tags 4) an expressivity module that is used by the robot to convert the tags in gestures and expressions.

NarRob was equipped with a repository of manually annotated gestures. The gestures mainly came from the Pepper gesture dataset, and some other samples have been added using the timeline utility from the SoftBank Choregraphe suite. In this case, the gestures are acquired by using the timeline of Choregraphe, recording the sequence of postures obtained by using the robot like a puppet. Another modality consists of using an acquisition device that allows for tracking of the skeleton of a human performing the posture. This paper reports the evolution of NarRob with the implementation of a module designed to acquire and reproduce the gestures performed by a human during an interactive session.

In particular, we illustrate the development of a module of NarRob, named *Gestures Module*, aimed at acquiring in real-time gestures from human users, recording them, and executing them both as they are being recorded, and in a mirrored version. The module allows the robot to reproduce also simple poses instead of complete gestures. We have defined a mapping algorithm to allow a *SoftBank Pepper* robot to reproduce the tracked gestures as close as possible to the original ones.

The development of this module makes it possible for the robot to learn more and more gestures in real-time by using a low-cost RGBD camera. This allows enriching in a faster manner the gestures dataset of the robot.

In our approach, we exploited a *Microsoft Kinect* sensor to capture the motion data from a user. The Microsoft Kinect is a popular choice for any research that involves body motion capture. It is an affordable and low-cost device that can be used for noninvasive, marker-less tracking of body gestures. In particular, we used the *OpenNi* driver for the *Kinect*, the NiTE 2.2 libraries for detecting the user skeleton, and the Kinetic version of ROS with the module pepper_dcm to provide package exchange and bridging between the computer and the robot and Ubuntu 16.04. We focused on the movements of the arms and the head, laying the basis for the extension of the same approach to the remaining parts.

In what follows, after a section where related works are reported, we describe the NarRob components, focusing on the module aimed at the acquisition and reproduction of gestures.

2. Related Works

Using motion capture to control or teach a robot is a concept that have gained more and more attention in recent years, due to the many applications that it can have, from industrial work to social interactions. A device often used for this purpose is Microsoft Kinect, due to the cost and complexity of a standard motion capture setup. For example, Baron et al. [6] controlled a Mindstorm NXT artificial arm with sensor Kinect, employing gesture recognition to regulate arm movement. Chang et al. [8] developed a Kinect-based gesture command control method for driving a humanoid robot to learn human actions, using a Kinect sensor and three different recognition mechanisms: dynamic time wrapping (DTW), Hidden Markov model (HMM) and principal component analysis (PCA).

Sylvain Filiatrault and Ana-Maria Cretu [18] used sensor Kinect to mimic the motion of a human arm to a NAO humanoid robot. In their case, the software architecture is based on three modules: Kinect Manager, Interaction Manager, and NAO manager. The Kinect Manager deals with the events and data captured by the Kinect. The class Kinect Transformer is used to get the Euler angles of the desired joints. The Interaction Manager is the intermediary between the Kinect and the robot and contains the repository for the joints used by the other two modules. The use of a joint repository of all articulations allows reducing the data to be processed as some joints are not needed. Finally, the NAO manager contains the static and dynamic constraints to apply to each one of the articulations, as well as the methods that allow the control of the robot movements. To be sure that the robot has enough time to execute the gesture, a delay of 200 ms between one cycle and the next has been introduced.

Augello et al. [1] modelled a computational creativity behavior in a dancing robot using deep learning . The dataset used as a base was collected using a Kinect and together with a Variational Autoencoder that allows mapping input patterns in a latent space it allowed the robot to create new dancing moves in real time as it listened to music. Itauma et al. [19] used a Kinect to teach an NAO robot some basic Sign Language gestures. The aim was teaching Sign Language to impaired children by employing different machine learning techniques in the process. Shohin et al. [24] used three different methods to make a robot NAO imitate human motion: direct angle mapping, inverse kinematics using fuzzy logic and iterative Jacobian.

Miguel et al. [25] used a Kinect sensor and a Convolutional Neural Network (CNN) trained with the MSRC-12 dataset [33] to capture and classify gestures of a user and send related commands to a mobile robot. The used dataset was created by Microsft and had 6244 gesture instances of 12 actions. To have gestures of the same length, without losing relevant information, the system used a Fast Dynamic Time Warping algorithm (FastDTW) to find the optimal match between sequences by non linearly warping them along the time axis. This resulted in all gestures normalized to sequences of 667 frames, with each frame having 80 variables, corresponding to the x,y,z values for each of the 20 joints, plus a separation value for each joint. The resulting 667x80 matrix is used as the input of the CNN, which classifies it in one of the 12 possible gestures. The CNN was trained using two strategies, combined training consisting of a single CNN to recognize all 12 gestures and individual training with 12 different CNN, each capable of recognizing only one gesture. The accuracy rates were 72.08% for combined training and 81.25% for the individual training.

Moreover, Unai et al. [36] developed a natural talking gesture generation behavior for *Pepper* by feeding a Generative Adversarial Network (GAN) with human talking gestures recorded by a Kinect. Their approach in mapping the movements detected by Kinect on the robot is very similar to what we used, but while they feed the resulting values to a neural network (a GAN), we use the (filtered) values directly.

3. The Humanoid Storyteller Modules

In what follows we mainly describe the new *Gestures Module* that has been implemented in NarRob and its interaction with the other storytelling modules of NarRob, i.e. the *Annotation Module*, the *Chatbot Module*, and the *Expressivity Module*.

3.1. Gestures Module

The Gesture module deals with collecting acquired gestures, saving them in a knowledge base. In a previous implementation of this module [4] the dataset was mainly based on the animations available in the NAO robot animations package, and enriched with a limited set created by recording and annotating a sequence of postures of the robot through Softbank Choregraphe Suite.

The gesture module has been improved to enrich the dataset with new gestures acquired by capturing the movements of human people with a low cost RGB-D camera.

The module is structured in a set of components, to increase its versatility for future projects and to simplify extensions of the current project.

The first module is named *Viewer*, and it extracts data frames from the Kinect camera (consisting of nine float values: three values for a joint position in 3D space, four values for quaternion from the origin and reliability values for both) and sends them in a pipe. The module also provides the feed of the Kinect camera with the overlay of the tracked user's skeleton. The frame data is long 8640 bytes. It is composed of 15 joints, with 9 values each and a representation of 64 bytes for each value. The data is sent to the *Gesture_Brain* module.

The *Gesture_Brain* module works both as a gateway for the ROS system [34] and as the module where actual data processing takes place. The gathered data cannot be used directly: a mapping is required to correctly associate each joint user position to the equivalent one in the Pepper robot. For this reason, the data is parsed and structured in a 15×9 float matrix, which is then split into three matrices: one for coordinates, one for quaternions, and one for reliability values. In our approach, we exploit only the first matrix considering always high the reliability. We, therefore, assume at this stage that the captured joint data is accurate enough for our purpose, as the Kinect already discards joints whose reliability values are too low.

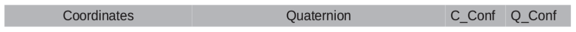


Figure 1: Structure of a single row of the array sent by the Viewer module

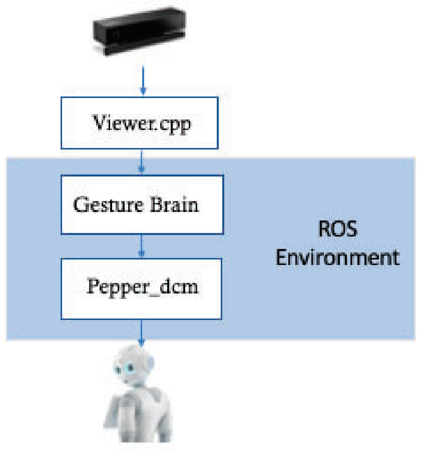


Figure 2: Structure of the gesture acquisition module

The joint position data is then used to estimate Pepper joint angles, specifically shoulder pitch, shoulder roll, elbow roll and elbow yaw for both arms and head yaw for the head.

The additional left and right wrist yaw and head pitch joint angles have been set to 0, since the Kinect camera is too imprecise to give a good estimate of that points.

When all the required values have been collected, the joint angles are sent to the robot using the ROS threads provided by the *pepper_dcm* bridge. These threads consist of multiple joint angles divided into groups, each group representing a body part. As we are interested only in the movement of arms and head, we use three of them: head, left arm, and right arm.

The bridge reads the sent values and the time stamp between each capture to dynamically compute in real-time the gesture trajectory. The spatio-temporal information allow the system to be as accurate as possible so that the gesture can be properly executed. The bridge itself is endowed with the in-built Self Collision Avoidance (part of the NaoQi library) while the *wait* and *breathe* animations have been deactivated, since they interfere with the commands sent by the pepper_dcm.

3.1.1. Mapping user Gestures into Pepper Movements

The *Pepper* robot has five degrees of freedom for each arm (each one associated with a joint), unlike human beings who have seven. A mapping is thus required between the detected user gestures and their reproduction by the robot.

From the *Kinect* camera the Cartesian coordinates for each joint, the quaternion for each segment (both referenced globally), and a reliability value for both are extracted. The bridge *pepper_dcm* uses the Euler angles to communicate to the robot the new position of its joint angles. We preferred to use the 3D space coordinates since the extracted quaternions do not represent the rotation from the previous position, but rather the rotation from a reference quaternion. This could lead to excessive inaccuracies when the values are converted in Euler angles.
Let $\overline{x}$, $\overline{y}$ and $\overline{z}$ be the unit vectors for each axis, that is:

$$\begin{aligned} \underline{x} &= (1,0,0) \\ \underline{y} &= (0,1,0) \\ z &= (0,0,1) \end{aligned}$$

Let S_L , E_L and W_L be the coordinates of the shoulder, the elbow and the wrist of the left arm respectively, $\overline{S_L E_L}$ and $\overline{E_L W_L}$ are defined as:

$$\begin{aligned} \overline{\underline{S_L E_L}} &= E_L - S_L \\ E_L W_L &= W_L - E_L \end{aligned}$$

$SR_L$is the supplementary to the angle between $\overline{S_L E_L}$ and $-x$ axis:

$$SR_L = \frac{\pi}{2} - arcos(\overline{S_L E_L} \cdot -x) \tag{1}$$

$SP_L$is the angle between the projection of $\overline{S_L E_L}$ on zy plane and z axis, shifted in range to avoid the jump discontinuity at π and $-\pi$:

$$SP_L = \pi - mod_{2\pi}(\frac{3}{2}\pi + arctan(\overline{S_L E_{Lz}}, \overline{S_L E_{Ly}}) \tag{2}$$

For values of SR_L close to $\frac{\pi}{2}$, SP_L become unstable. As such, in the algorithm is assigned a value of 0 for $SR_L > 1.3$.
$ER_L$is the angle between $\overline{E_L W_L}$ and $\overline{S_L E_L}$, shifted by $\frac{\pi}{2}$:

$$ER_L = \frac{\pi}{2} + arcos(\overline{E_L W_L} \cdot \overline{S_L E_L}) \tag{3}$$

$EY_L$is the angle between the projection of $\overline{E_L W_L}$ on zy plane and z axis, shifted in range for stability reasons, plus $-SP_L$:

$$\begin{aligned} EY_L = \pi - mod_{2\pi}(\frac{3}{2}\pi + \\ + arctan(E_L W_{Lz}, E_L W_{Ly}) - SP_L \end{aligned} \tag{4}$$

The right arm is almost the same as the left arm, the only difference is that some angles have the opposite sign.

Let $\overline{HN}$ be the difference between the coordinates of the joints H (head) and N:

$$\overline{HN} = H - N$$

The head yaw HY is the angle between the projection of $\overline{HN}$ on the xz plane and the z axis:

$$HY = -arctan(\overline{HN}_z, \overline{HN}_y) - \frac{\pi}{2} \tag{5}$$

Smoothing movements through a Line of Best Fit: The *Kinect* joint detection is based on the shape of the user, which is redrawn at every frame. While calibrating the sensor helps to reduce the resulting jerkiness but a significant amount of noise can still remain. This noise can be approximately classified in two categories:

- a constant gaussian noise caused by small alteration on the detected shape and
- large "spikes" when the $Kinect$ fail to guess the position of one or more joints (especially common when part of the limb is outside the frame or when two or more joints overlap).

A simple way to compensate part of this noise is to use a *line of best fit*.

In particular, given k points in (x, y) coordinates system, we must find the values c_0 and c_1 in the equation

$$p(x) = c_0 x + c_1$$

that define the straight line minimizing the squared error

$$E = \sum_{j=0}^{k} |p(x_j) - y_j|^2$$

in the equations

$$\begin{aligned} x_0 c_0 + c_1 &= y_0 \\ x_1 c_0 + c_1 &= y_1 \\ &\dots \\ x_k c_0 + c_1 &= y_k \end{aligned}$$

The result is a smoother movement, especially when the Kinect camera is not able to detect the exact coordinates of a given joint. This is because, given a disturbing signal, the line of best fit can be seen as an approximation of the tangent line that the signal would have at that point if all the noise were removed. This is not always true, especially when the signal changes rapidly, but it's close enough in most cases to give a generally cleaner movement.

3.1.2. Modes of Operation

Besides acquiring the user movements in order to increase the gestures knowledge base of the robot, the Gesture module has also some additional features implemented to increase the breadth of experiments that can be performed with the system or to help with future projects. In particular, the module can act into three modalities: 1) Mimic mode, 2) pose mode, and 3) Playback mode.

The behavior of the module is managed by a set of input parameters, named *mode, mirror_flag, json_file_name, LAjpos, RAjpos, Hjpos*. The first parameter determines which one of the three different modes of operation will be used (default 0), the second one determines if the mirror mode is activated or not (default false), the third defines the name of the text file used to record (in mode 0 and 1) or read (mode 2) the gestures (the default value is NULL, that is no recording) and set a flag (record_flag) to 1, the fourth, fifth and sixth ones are used to determine the pose to use in mode

1 (as default, the robot will spread its arms parallel to the ground, in a pose that in animation is known as "T-pose"). More in details, the modes of operation of the main program are:

- Mode 0, or "Mimic Mode", is the default mode and it allows the robot mimicking the movements of the user. The record flag is used to store the data, so the output is not just sent to the ROS publishers, but recorded in a JSON(JavaScript Object Notation) file. This file is then linked to a dataset of gestures that can be stored and annotated in order to be reproduced later. If the *mirror* flag is active, each movement is mirrored during the execution. In case both the record and mirror flags are active, the mirrored movement will be recorded and saved in a specific file to be linked to the gesture dataset.

- Mode 1, or "Pose Mode", make the robot execute a pose (defined at the beginning by the value of the given parameters) already stored in the gestures ontology. We considered a scenario where the Pepper robot showed a specific pose and the user had to replicate it as closely as possible. The scenario involves the use of both the normal mode and the mirror mode. A proper distance algorithm calculates how close is the pose of the user to that one performed by the robot. The distance is separately evaluated for the head, the right upper arm, the left upper arm, the right forearm, and the left forearm. If the user pose keeps all body parts below their respective distance thresholds (defined separately for each body part), the pose is assumed to be correctly replicated. If the record flag is activated the returned distance values are saved. When the mirror flag is set to "on" the user should try to replicate the mirrored version of the shown pose.

- Mode 2, or "Playback Mode", consists of reproducing a previously recorded gesture. The mirror flag, even if selected, doesn't have any effect on this procedure. The name, necessary to activate the record flag, is used as the name of the gesture linked to the gesture ontology. In this case the goal is twofold: just emulating a gesture or asking a user to emulate the gesture itself. This could be done either for learning activities (e.g. teach pupils some gestures) or for therapeutic ones (e.g. physical exercises for elderly).

3.1.3. Auxiliary Algorithms

In the following we describe a set of auxiliary algorithms that we have included in the Gesture Module to implement its tasks.

Distance measurement To measure the distance between the robot pose and the detected pose, we used a distance measurement algorithm. For each arm, the related joint angle group (consisting of Elbow roll, Elbow yaw, Shoulder pitch,

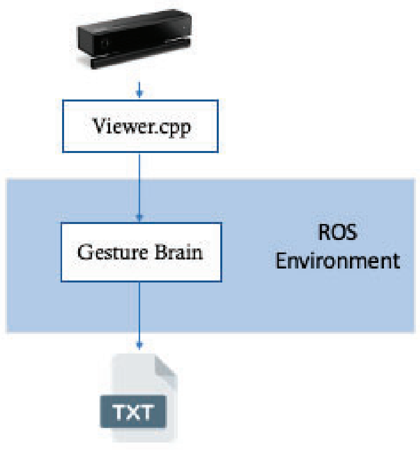


Figure 3: Structure of the algorithm when recording (the text file can be either the recorded gesture in mode 0 or the record of distance values in mode 1)

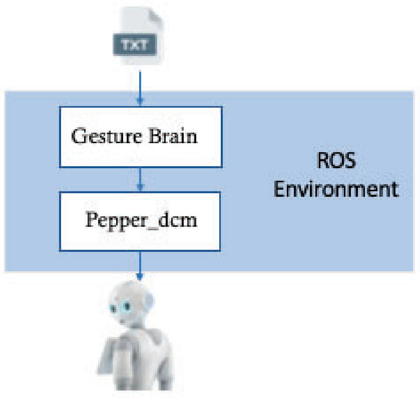


Figure 4: Structure of the algorithm in mode 2 (the txt file in this case is the coding of a previously recorded gesture)

Shoulder roll, and Wrist yaw) is split into two vectors, representing the upper arm (Shoulder pitch and Shoulder roll) and the forearm (Elbow roll, Elbow yaw, and Wrist yaw). Including the joint angle group for the head (Head yaw and Head pitch). The position is represented with five vectors. To avoid that joint angles with a larger range of values could excessively affect the final result, each angle is normalized dividing the value by its maximum value. This process is applied both to the robot pose and the detected pose of the user, producing five pairs of vectors. Finally, a Mean Squared Error algorithm is applied at each pair of vectors, resulting in five distances defined in the interval [0,1]. With this separated evaluation we found that the final measure is more precise and reliable allowing to define separate thresholds if a pass/fail system is implemented (like in our case).

Mirroring: The activation of the mirror flag activates the mirror algorithm for any detected movement. This means that in mode 0, the robot will mirror the user (and if the program is recording, it will record the mirrored movement), while in mode 1, the user will have to mirror the shown pose. The mirroring mode consists in switching the detected angles between the left and the right arm, changing the sign of every angle except shoulder pitch. For the head yaw angle, a simple change in the sign is enough.

Recording and Playback: If the recording flag is active, the program will write a text file in *JSON* syntax with the joint angles sent to the robot (mode 0) or the distance from the given pose (mode 1) for each frame.
In mode 2, the records created in mode 0 can be reproduced by the robot.
In mode 0, the fields for each frame are:

- AbsTime: Date and time when the frame was captured, to the microsecond.
- Left_Arm: Vector containing the joint angles of the left arm joint group.
- Right_Arm: Vector containing the joint angles of the right arm joint group.
- Head: Vector containing the joint angles of the joint head group.

In mode 1, the fields for each frame are:

- AbsTime: Date and time when the frame was captured, to the microsecond.
- Error_Left_Arm_Shoulder: the measured distance between the given position and the detected position of the left upper arm.
- Error_Left_Arm_Elbow: the measured distance between the given position and the detected position of the left forearm.
- Error_Right_Arm_Shoulder: the measured distance between the given position and the detected position of the right upper arm.
- Error_Right_Arm_Elbow: the measured distance between the given position and the detected position of the right forearm.
- Error_Head: the measured distance between the given position and the detected position of the head.

4. From Words to Gestures and Expressions

The gestures acquired by the kinect are used by the robot in order to improving its communication abilities by adding expressiveness to the narration, by including gesticulation to emphasize what is being said by the robot.

The operation of the module that draws the gestures from the dataset has two main phase; during the first phase, it parses the sentence that has to be said, looking for the terms that directly correspond to a given gesture; if a match is found, the term is annotated for being associated with a gesture; otherwise, the focus of the parser in oriented at finding the verbs in a sentence. A POS is then executed and the verbs present in the sentence are analyzed by using Wordnet: in particular, a lemmatization task is run in order to find the lemma of the verb under exam; once the lemma has been found, a score $s(lemma\_verb1, lemma\_kb) \in [0, 1]$ is computed between the lemma of the verb in a sentence and the lemmas that are associated, in the knowledge base of the annotated gestures. The score is determined by calculating the shortest path linking two senses in WordNet by considering the "is-a" relationship [23]. In this first version of the system, the score computation takes into account only the most frequent sense for each lemma, speeding up the analysis. A similarity threshold $T_{sim} \in [0, 1]$ is experimentally fixed. If $s(lemma_{verb1}, lemma_{kb})$ g T_{sim}), the action corresponding to the $lemma_{kb}$ that gives the highest value of $s(lemma_{verb1}, lemma_{kb})$ between the lemma associated to the verb detected in the sentence and the lemma associated to the action stored in the list of gestures that can be performed by the robot. If the value of $s(lemma\_verb1, lemma\_kb)$ is below the value of T_{sim}), the verb is ignored and no specific gesture is performed by the robot.

Furthermore, an emotion detection module, presented in [26], has been exploited to find if there is a basic emotion that could be associated to that particular sentence that will be said. The emotion detection module acts on the communicative channels of the robot that could be associated by a human observer to some emotions, e.g., the color of its LEDs, the speed of its speech, and the inclination of the head [17][20] [5].

The emotion that can be associated to a given sentence is one of the well known Ekman basic emotions: *anger*, *disgust*, *fear*, *joy*, *sadness* and *surprise*. If no emotion is detected, the *neutral* label is used. The module exploits a literature lexicon derived from the Word-Net Affect Lexicon [30] [31] and the methodology, based on the Latent Semantic Analysis (LSA)[21] paradigm, that has been presented in [26].

That approach assumes that a sentence d can be encoded as a point in a Data Driven "conceptual" space, by calculating a vector $\mathbf{d}$ whose i-th component is the number of times the i-th word of the vocabulary is present in d. The vector $\mathbf{d}$ is subsequently mapped into a reduced-dimensionality "conceptual" space induced by LSA.
At the same time, a set of vectors acting as emotional "beacons" have been used to map a text from the conceptual space to an emotional space. In particular, six sets E_{anger} , $E_{disgust}$, $\cdots$, $E_{surprise}$ of vectors constituting the sub-symbolic coding of each "beacon" identifying a basic emotion have been used. The generic vector belonging to one of the sets is encoded in the same "conceptual" space together with the sentence s of the story that has to be said. Once the sentence s is mapped into the "conceptual" space, it is possible to compute its emotional fingerprint following the methodology illustrated in [26], which consists in exploiting seman-

tic similarity with the vectors that are associated to each one of the six E_i sets. The emotional space is build as a six-dimensional hypersphere where all the sentences are encodes. Each region of this hypersphere is associated to a set of emotional expression of the robot. The element associated to the highest value of emotion determines the main emotion expressed by the sentence. A minimum value of threshold $Th\_e \in [0, 1]$ has been experimentally established in order to label sentences that do not carry any emotion as "neutral".

Thanks to this encoding, the system is provided with a module that makes it possible to generate an expressive behavior, according to the story annotations. In particular a gesture tag triggers a proper movement according to what is is stored in the gestures dataset, and to produce an emotional expression.

For the emotions, we have encoded six possible emotional expressions, corresponding to the Ekman categories (i.e., anger, disgust, fear, joy, sadness, surprise), that has been defined by setting some of the robot communicative channels that can be correlated by a human observer to some emotions, such as the color of its LEDs, the pitch and speech rate, and the head inclination. [17] [20] [5] [29].

4.1. Annotation Module

This module of NarRob is aimed at the semantic annotation of possible actions and the emotional labelling of sentences.

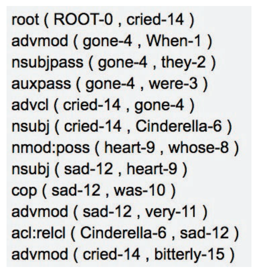
The gestures acquired with the Gestures Module are annotated in an interactive manner according to their specific meaning and labelled into the gesture ontology.

The robot, before reading a story, analyzes its composing sentences, extracts the actions to perform and the emotions to manifest as well.

Each story is then offline analysed through a dependency-parsing by using the Stanford CoreNLP tool. This analysis is performed to obtain the dependencies graphs of the sentences. Each graph is then analyzed starting from the root, where each node is compared with the annotations of the gestures in the KB. Each time a match is found, the tag $^{s}tart(gesturename)$ is added to the text to give an indication of the gesture that robot should perform in that specific point of the story. Figure 5 shows an example for the sentence "When they were gone, Cinderella, whose heart was very sad, cried bitterly", extracted from the well known story of Cinderella [1]. In the specific case there are two words corresponding to the annotations in the gestures dataset. However, the priority is given according to the order of the dependencies, in this case one of the words (the verb *cry*), is the root of the sentence and will be chosen to add the gestures tag.

The annotations related to the emotional content of the story are inserted according the six Ekman categories. These categories have been widely used also in other contexts [27] [16] [14]. Here we exploit the approach described in section 4 When a specific emotion is detected, the tag *mood("EmotionName", EmotionWeight)* is added in the text, where the weight specify how intense is the emotion according to the emotion detection algorithm.

[1] The text is extracted from https://www.storyberries.com/fairy-tales-cinderella-or-the-little-glass-slipper/



```
root ( ROOT-0 , cried-14 )
advmod ( gone-4 , When-1 )
nsubjpass ( gone-4 , they-2 )
auxpass ( gone-4 , were-3 )
advcl ( cried-14 , gone-4 )
nsubj ( cried-14 , Cinderella-6 )
nmod:poss ( heart-9 , whose-8 )
nsubj ( sad-12 , heart-9 )
cop ( sad-12 , was-10 )
advmod ( sad-12 , very-11 )
acl:relcl ( Cinderella-6 , sad-12 )
advmod ( cried-14 , bitterly-15 )
```

Figure 5: The dependencies of the analyzed sentence

This text passage is an example of annotated story, taken from " The Real Princess"[2] by Hans Christian Andersen:

> *mood("SADNESS") gesture(ThunderGesture)*
> *One evening a fearful tempest arose, it thundered and lightened, and the rain poured down from the sky in torrents:*
> *mood("SADNESS")*
> *besides, it was as dark as pitch.*
> *mood("ANGER") gesture("HearGesture")*
> *All at once there was heard a violent knocking at the door, and the old King, the Prince's father, went out himself to open it.*

4.2. The Chatbot Module

This module relies on a chat-bot engine based on the SALVE architecture [2], and a conversational Knowledge Base described through a set of question-answers modules, named categories, and written in an enhanced AIML markup language, named S-AIML. This rule engine manages the interaction according to typical practices of interaction.

The chatbot exploits also an OWL[3] ontology, formalizing the main concepts related to the specific domain for which the robot has been suited[4].

Moreover, the Chatbot is not only aimed at interacting with the user during the storytelling activity. It is exploited also to label the recently acquired gestures with proper descriptions through an interactive session. An example of dialogue is reported below:
U: NarRob, please pay attention to this new gesture
R: Ok. I am waiting and recording
The user makes a gesture that is acquired by the Kinect
U:End of gesture
R: I have acquired a new gesture. I can repeat it
Narrob executes the PlayBack mode of the recently acquired gesture
R: Has the gesture been correctly reproduced?
U: Yes, it has
R: What label should I assign to the gesture?
U: The gesture can be labelled with "come here"

[2] http://www.gutenberg.org/cache/epub/1597/pg1597.txt
[3] W3C Web Ontology Language (OWL)

R: Ok. Done. Gesture acquired, labelled and saved for future storytelling.

5. Conclusions and Developments

We have presented in detail the realisation of a specific module for our NarRob Architecture aimed at acquiring gestures from an user by using a low-cost RGB-D Camera, such as the Kinect.

NarRob exploits an annotated corpus of text and an ontology of gestures to associate body signs and emotional expressions to the narration, by selecting the best appropriate gestures fitting the meaning of a sentence or part of a sentence. The goal is to obtain a not trivial, not pre-defined, and a more involving interaction with a user. Gestures have been acquired by a Gesture Module, which is capable of detecting, storing and reproducing the user poses with a Kinect camera with sufficient accuracy.

The first experiments show that the reproduced movements are quite accurate and smooth; the recording and execution of the gestures are very close to the real-time movements. However, sometimes, certain positions cannot be reliably detected, due to imprecise behavior of the Kinect output when joints overlap each other, and to excessive reliance on the silhouette to detect the human body and the lack of joints in key points of the detected skeleton (like the hands). There is also an environmental factor, like lightning and positioning, that can make accurate user detection problem.

The robot can acquire gestures form an human user and reproduce them in an adequate manner by using also a proper mapping procedure that allows to approximate the gestures of human beings. Furthermore, the robot, thanks to the Gesture Module, is potentially autonomously capable of acting both as an instructor and a learner by exploiting the gesture mirroring feature.

In future works, a neural network will be also trained to recognize and classify the gestures to give a proper answer, creating a more realistic verbal communication between humans and robots. Possible improvements should include a more effective detection algorithm for the Kinect, more efficient ways to execute the mapping, the use of all points detected (not just limbs and head), the use of the official Microsoft SDK to have even more points detected (provided it's possible to retrieve all the necessary libraries) and a more general user-friendly experience (like the ability to set a timer for recording). Moreover, we plan to improve our framework as to deal with novel and emerging *big data trends* including performance (e.g., [7, 12, 9]), and privacy and security (e.g., [11, 15]).

References

[1] Augello, A., Cipolla, E., Infantino, I., Manfre, A., Pilato, G., Vella, F., 2017. Creative robot dance with variational encoder. arXiv preprint arXiv:1707.01489 .

[2] Augello, A., Gentile, M., Weideveld, L., Dignum, F., 2016. Dialogues as social practices for serious games, in: Proceedings of the Twenty-second European Conference on Artificial Intelligence, pp. 1732–1733.

[3] Augello, A., Infantino, I., Maniscalco, U., Pilato, G., Vella, F., 2018a. Introducing narrob, a robotic storyteller, in: International Conference on Games and Learning Alliance, Springer. pp. 387–396.

[4] Augello, A., Infantino, I., Maniscalco, U., Pilato, G., Vella, F., 2018b. Narrob: A humanoid social storyteller with emotional expression capabilities, in: Biologically Inspired Cognitive Architectures Meeting, Springer. pp. 9–15.

[5] Bänziger, T., Scherer, K.R., 2005. The role of intonation in emotional expressions. Speech communication 46, 252–267.

[6] Baron, G., Czekalski, P., Malicki, D., Tokarz, K., 2013. Remote control of the artificial arm model using 3d hand tracking, in: 2013 International Symposium on Electrodynamic and Mechatronic Systems (SELM), IEEE. pp. 9–10.

[7] Bonifati, A., Cuzzocrea, A., 2006. Storing and retrieving xpath fragments in structured P2P networks. Data Knowl. Eng. 59, 247–269.

[8] Chang, C.w., He, C.j., et al., 2014. A kinect-based gesture command control method for human action imitations of humanoid robots, in: 2014 International Conference on Fuzzy Theory and Its Applications (iFUZZY2014), IEEE. pp. 208–211.

[9] Chatzimilioudis, G., Cuzzocrea, A., Gunopulos, D., Mamoulis, N., 2013. A novel distributed framework for optimizing query routing trees in wireless sensor networks via optimal operator placement. J. Comput. Syst. Sci. 79, 349–368.

[10] Cuzzocrea, A., 2006. Combining multidimensional user models and knowledge representation and management techniques for making web services knowledge-aware. Web Intelligence and Agent Systems 4, 289–312.

[11] Cuzzocrea, A., Bertino, E., 2011. Privacy preserving OLAP over distributed XML data: A theoretically-sound secure-multiparty-computation approach. J. Comput. Syst. Sci. 77, 965–987.

[12] Cuzzocrea, A., Moussa, R., Xu, G., 2013. Olap*: Effectively and efficiently supporting parallel OLAP over big data, in: Model and Data Engineering - Third International Conference, MEDI 2013, Amantea, Italy, September 25-27, 2013. Proceedings, pp. 38–49.

[13] Cuzzocrea, A., Pilato, G., 2018. Taxonomy-based detection of user emotions for advanced artificial intelligent applications, in: International Conference on Hybrid Artificial Intelligence Systems, Springer. pp. 573–585.

[14] Cuzzocrea, A., Pilato, G., 2019. An innovative user-attentive framework for supporting real-time detection and mining of streaming microblog posts. Soft Computing , 1–20.

[15] Cuzzocrea, A., Russo, V., 2009. Privacy preserving OLAP and OLAP security, in: Encyclopedia of Data Warehousing and Mining, Second Edition (4 Volumes), pp. 1575–1581.

[16] D'Avanzo, E., Pilato, G., Lytras, M., 2017. Using twitter sentiment and emotions analysis of google trends for decisions making. Program: electronic library and information systems 51, 322–350.

[17] Feldmaier, J., Marmat, T., Kuhn, J., Diepold, K., 2016. Evaluation of a RGB-LED-based emotion display for affective agents. arXiv preprint arXiv:1612.07303 .

[18] Filiatrault, S., Cretu, A.M., 2014. Human arm motion imitation by a humanoid robot, in: 2014 IEEE International Symposium on Robotic and Sensors Environments (ROSE) Proceedings, IEEE. pp. 31–36.

[19] Itauma, I.I., Kivrak, H., Kose, H., 2012. Gesture imitation using machine learning techniques, in: 2012 20th Signal Processing and Communications Applications Conference (SIU), IEEE. pp. 1–4.

[20] Johnson, D.O., Cuijpers, R.H., van der Pol, D., 2013. Imitating human emotions with artificial facial expressions. International Journal of Social Robotics 5, 503–513.

[21] Landauer, T.K., Foltz, P.W., Laham, D., 1998. An introduction to latent semantic analysis. Discourse Processes 25, 259–284. doi:10.1080/01638539809545028, arXiv:https://doi.org/10.1080/01638539809545028.

[22] Lau, M.C., Anderson, J., Baltes, J., 2019. A sketch drawing humanoid robot using image-based visual servoing. Knowledge Eng. Review 34, e18.

[23] Lingling, M., Runqing, H., Junzhong, G., 2013. A review of semantic similarity measures in wordnet, in: International Journal of Hybrid Information Technology, p. Vol 6 No 1.

[24] Mukherjee, S., Paramkusam, D., Dwivedy, S.K., 2015. Inverse kinematics of a nao humanoid robot using kinect to track and imitate human motion, in: 2015 International Conference on Robotics, Automation, Control and Embedded Systems (RACE), IEEE. pp. 1–7.

[25] Pfitscher, M., Welfer, D., Cuadros, M.A.d.S.L., Gamarra, D.F.T., 2018. Activity gesture recognition on kinect sensor using convolutional neural networks and fastdtw for the msrc-12 dataset, in: International Conference on Intelligent Systems Design and Applications, Springer. pp. 230–239.

[26] Pilato, G., D'Avanzo, E., 2018a. Data-driven social mood analysis through the conceptualization of emotional fingerprints. Procedia Computer Science 123, 360 – 365. doi:https://doi.org/10.1016/j.procs.2018.01.056. 8th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2017 (Eighth Annual Meeting of the BICA Society), held August 1-6, 2017 in Moscow, Russia.

[27] Pilato, G., D'Avanzo, E., 2018b. Data-driven social mood analysis through the conceptualization of emotional fingerprints. Procedia computer science 123, 360–365.

[28] Regier, P., Milioto, A., Karkowski, P., Stachniss, C., Bennewitz, M., 2018. Classifying obstacles and exploiting knowledge about classes for efficient humanoid navigation, in: 18th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2018, Beijing, China, November 6-9, 2018, pp. 820–826.

[29] Rodriguez, I., Manfre, A., Vella, F., Infantino, I., Lazkano, E., 2018. Talking with sentiment: Adaptive expression generation behavior for social robots, in: The 19th International Workshop of Physical Agents (WAF).

[30] Strapparava, C., Mihalcea, R., 2007. Semeval-2007 task 14: Affective text, in: Proceedings of the 4th International Workshop on Semantic Evaluations, Association for Computational Linguistics, Stroudsburg, PA, USA. pp. 70–74.

[31] Strapparava, C., Mihalcea, R., 2008. Learning to identify emotions in text, in: Proceedings of the 2008 ACM Symposium on Applied Computing, ACM, New York, NY, USA. pp. 1556–1560. doi:10.1145/1363686.1364052.

[32] Striepe, H., Lugrin, B., 2017. There once was a robot storyteller: Measuring the effects of emotion and non-verbal behaviour, in: International Conference on Social Robotics, Springer. pp. 126–136.

[33] Url, a. Msrc-12 dataset; https://www.microsoft.com/en-us/ download/ details.aspx? id=52283.

[34] Url, b. Ros kinetic; http://wiki.ros.org/kinetic.

[35] Xu, J., Broekens, J., Hindriks, K., Neerincx, M.A., 2015. Effects of a robotic storyteller's moody gestures on storytelling perception, in: Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on, IEEE. pp. 449–455.

[36] Zabala, U., Rodriguez, I., Martínez-Otzeta, J.M., Lazkano, E., 2019. Learning to gesticulate by observation using a deep generative approach. arXiv preprint arXiv:1909.01768 .