POLITECNICO DI TORINO
Repository ISTITUZIONALE

Virtual character animation based on affordable motion capture and reconfigurable tangible interfaces

(Article begins on next page)

20 May 2024

Virtual Character Animation Based on Affordable Motion Capture and Reconfigurable Tangible Interfaces

Fabrizio Lamberti, *Senior Member, IEEE,* Gianluca Paravati, *Senior Member, IEEE,*
Valentina Gatteschi, Alberto Cannavò, and Paolo Montuschi, *Fellow, IEEE*

Abstract—Software for computer animation is generally characterized by a steep learning curve, due to the entanglement of both sophisticated techniques and interaction methods required to control 3D geometries. This paper proposes a tool designed to support computer animation production processes by leveraging the affordances offered by articulated tangible user interfaces and motion capture retargeting solutions. To this aim, orientations of an instrumented prop are recorded together with animator's motion in the 3D space and used to quickly pose characters in the virtual environment. High-level functionalities of the animation software are made accessible via a speech interface, thus letting the user control the animation pipeline via voice commands while focusing on his or her hands and body motion. The proposed solution exploits both off-the-shelf hardware components (like the Lego Mindstorms EV3 bricks and the Microsoft Kinect, used for building the tangible device and tracking animator's skeleton) and free open-source software (like the Blender animation tool), thus representing an interesting solution also for beginners approaching the world of digital animation for the first time. Experimental results in different usage scenarios show the benefits offered by the designed interaction strategy with respect to a mouse & keyboard-based interface both for expert and non-expert users.

Index Terms—Tangible User Interfaces, Natural User Interfaces, Motion Capture, Human-Machine Interaction, Computer Animation.

✦

1 INTRODUCTION

ANIMATION of virtual characters is essential for a wide range of applications, from the production of movies and video games to the creation of virtual environments used in education, cultural heritage, product design, and social networking scenarios, to name a few [1], [2].

Generating 3D animated characters is usually a very labor-intensive task that requires a lot of time and significant expertise, also because of the sophisticated interfaces used [3]. This fact may be particularly critical for unskilled users, who might want to use them in rapid prototyping tasks that are typical of the early production stages of both digital and non-digital contents [4], [5]. Hence, approaches capable to trade-off the tight requirements of existing animation systems with a higher productivity (possibly paid in terms of control accuracy) could be of great interest [6].

According to [7], in common animation systems based on *keyframing*, much of the complexity associated with the creation of an animated character through traditional mouse & keyboard interfaces lays in the *posing* step. In fact, animators need to select and manipulate in sequence, through an interface that is natively 2D, a potentially large number of on-screen handles associated with a character's virtual skeleton, often referred to as "armature" or "rig" (a tree of rigid segments called "bones"). Handles let animators adjust, directly or indirectly, the degrees of freedom (DOFs) of all the individual character's joints [8].

The research community tried to address these issues

by proposing different interaction paradigms. One of these paradigms is *performance-driven animation*, where an actor's physical performance is captured and interactively transferred to the virtual character to be animated. Character's motion can be controlled in real time, thus giving the animators an immediate feedback about the generated animation and letting them focus on improvisation [10]. Hence, this technique is often referred to as *virtual puppetry* [11].

A form of virtual puppetry that has become very common in many movie and video game productions is *motion capture* [12]. However, requirements in terms of equipment and skills make this technique suitable especially for professional animators [13]. Moreover, defining the mapping between performer's and character's motion is a complex task that requires sophisticated configuration steps [14], and automatic retargeting solutions proposed so far do not allow the animators to keep a fine control on resulting poses [15].

Another interaction paradigm that is gaining increasing attention is represented by the so-called *tangible user interfaces* (TUIs) [16]. TUIs have proved already to be particularly suited, especially for novice users, for controlling "rigged" characters by offering a number of advantages compared to alter techniques, such as direct tactile feedback and intuitive 3D perspectives [5], [17]. TUIs have been used both for keyframing and performance-driven animation, though not together in a single animation pipeline. Moreover, they are rarely combined with other interaction paradigms, except for traditional ones. Lastly, they are often based on specially shaped hardware, which is meant to animate only a particular character. When designed for re-configurability, they may get particularly costly and hard to assemble, and mapping physical modifications onto the character's motion

• *The authors are with the Dipartimento di Automatica e Informatica of Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129, Torino, Italy. E-mail: see http://grains.polito.it/*

may require complex setup mechanisms.

By moving from the above considerations, this paper presents the design of an animation system where information about the pose of an instrumented tangible interface based on reusable components can be combined with data obtained by tracking animator's body to generate 3D character's motions with both keyframing and performance-driven animation techniques. This way, while articulating the character with the tangible interface, the animator can pose it in the 3D space by using his or her body. Currently, body tracking is based on the Microsoft Kinect[1], a device exploited to design many kinds of natural user interfaces (NUIs). The tangible interface is built using motors, sensors and bricks in Lego Mindstorms EV3[2] packs. An automatic configuration mechanism is implemented to assist the animator in the assembly and configuration process, with the goal of helping him or her to setup the interface that best fits the DOFs of the character to be animated. The above technologies have been integrated in the Blender[3] open source animation tool, making the overall system suited, also in terms of costs, to both professional and non-professional users. Common animation functionalities provided by the considered tool and configuration mechanisms required by the proposed interaction method are made available via customized voice commands, thus letting the animator focus on the handed prop and on him or her body performance.

Experimental results aimed to compare the devised solution with the traditional mouse & keyboard interface confirmed its suitability for both skilled and unskilled animators under different perspectives.

2 RELATED WORK

In this section, key research activities pertaining techniques and interfaces for computer animation are reviewed.

2.1 Body and Hand Motion Tracking

Motion tracking systems were developed to overcome intrinsic limitations of 2D interfaces (e.g., based on pen and touch/multitouch input [18]). Early motion tracking systems were characterized by a low-dimensional control of animated virtual characters, and let animators use their hands (with or without gloves) to define key poses or produce performance-driven animations [19], [20].

The control of virtual characters with a large number of DOFs is generally achieved more effectively by using high-dimensional full-body motion capture [21]. However, the quality of animations produced is often paid with the need for expensive equipment, large performance spaces, skilled actors, and laborious post-processing steps [8]. Technological advancements recently reduced, at least for some usage scenarios, the impact of these constraints, by letting also novice animators record full-body motion data using, e.g., consumer-level vision-based marker-less tracking systems or wearable sensors [22], [23].

Independently of the technology used, a point that needs to be considered both for keyframing and performance-driven animation is how to map the DOFs captured by the

tracking system onto the DOFs of the character to be animated [24]. Mapping can be either direct or indirect. Direct mapping is generally suited to the control of human-shaped characters with a number of target DOFs comparable to that of the capture technology [21], whereas indirect mapping is often used to animate non-antropomorphic characters or to retarget motion to a different number of DOFs [25].

Various solutions have been defined to enable the automatic retargeting of performer's motion onto arbitrarily-shaped characters' motion [26]. An approach tailored to rigged characters that considers the similarity between controlling and controlled DOFs is proposed in [27]. Alternative approaches not requiring character's mesh to be prepared for animation are also available [3], [15]. The drawback is that the animator is required to act as the digital character, by assuming poses that could be non-natural, which makes it difficult or even impossible to create the desired motions. Several techniques have been developed to address the above limitation, e.g., by simulating a realistic character's motion from set of predefined action samples that are blended together based on actual performance [28]. These techniques are generally not suitable for professional productions, where animators want to achieve a fine-grained control over the configuration of each character's joint.

Hence, manual solutions have been developed, which often require animators to work with sophisticated interfaces making mapping a task quite hard to accomplish [29], [30].

It is worth noting that both the retargeting approaches suffer from the fact that the control interface (e.g., the hand, or body) has a fixed topology, which can be quite different from the structure of the virtual character. Thus, mental effort required during animation could be non-negligible.

2.2 Generically and Specially Shaped Tangible Devices

TUIs were introduced to make general-purpose human-machine interaction more intuitive thanks to the affordances offered by physical objects [16]. Their use for computer animation is conventionally dated back to the late 1980's, when mechanical devices known as "waldos" started to be used for the control of computer-generated puppets appearing in TV shows and other interactive performances [31], [32].

Devices used to build such interfaces are rather heterogeneous. In some cases paper tags have been used [33], [34]. Sometimes, physical and haptic props recreating classic interfaces exploited, e.g., by puppeteers, to pull the strings of a real mannequin were proposed [35]. More frequently, special-purpose mechanical devices have been adopted, often designed to mimic the shape of the character to be animated under the assumption that intuitiveness can be improved by reducing the separation between the tangible and virtual worlds [6], [36]. For instance, in [37], an instrumented human-shaped doll is used to specify key poses and retrieve matching motion capture data from a repository. A similar solution is reported in [38], where motion is captured by tracking the joints of the physical device by using a stereo camera. In [5], a tangible interface with a robotic design where passive joints are replaced by servo motors was presented. According to the authors of [5], besides allowing animators to reconfigure the device to previous poses, active joints could be used to provide physical feedback, e.g., by compensating gravity or recreating natural behaviors.

---

1. https://developer.microsoft.com/en-us/windows/kinect
2. http://www.lego.com/en-us/mindstorms/
3. https://www.blender.org/

Comparable approaches have been adopted also for animating non-anthropomorphic characters. For instance, in [39] a chicken-shaped plush toy is used to control the behavior of a similar character in an interactive game. In [40], a teddy bear-shaped robotic interface is exploited both as an input device (using sensors embedded in robot's arms) and output device (through vibrotactile feedback).

The effectiveness of TUIs for character animation has been demonstrated in scenarios ranging from home entertainment to professional productions. As a matter of example, in [41], a low cost arm-shaped device with embedded sensors based on an Arduino board is used to animate an articulated arm with the same DOFs, whereas in [42], the authors illustrate the design of the sophisticated dinosaur-shaped instrumented armatures that were used to produce many of the stop motion shots of *Jurassic Park*.

2.3 Reconfigurable Tangible Interfaces

Despite the advantages associated with the use of TUIs, the fixed structure of solutions above could represent a severe limitation to their applicability in general-purpose animation scenarios [5]. Hence, a number of solutions have been proposed, based on both instrumented and non-instrumented components that can be assembled in many ways to create the desired configuration. Reconfigurable TUIs were initially exploited for 3D modeling [43] and, more recently, started to be used for 3D animation [44]. Interfaces above suffer from the limitation of being unaware of the actual topology built by the user, which could represent an important hint capable to simplify the mapping step [8].

One of the first examples of an hub-and-strut construction kit that is capable to capture both the topology and geometry of the model being built based on information communicated by individual bricks is reported in [45]. The interface was developed as a general-purpose modeling tool for exploring physical chemistry, mechanical robotics, etc. As such, it was not able offer fine-grained control required for animation. Nonetheless, the authors demonstrated its applicability for controlling simple digital puppets. This idea has been further explored in [8], where the effectiveness of a modular and reconfigurable topology-aware TUI for general purpose character articulation is validated through a user study with unskilled users. The interface is created by assembling hot-pluggable instrumented components with accurate sensing capabilities, which makes the system suitable both for rapid prototyping and for precise posing. Given the compact size of the components used, the animators could possibly recreate the whole structure of the character to be animated. This way, the process of retargeting deformations of the physical assembly onto the virtual character could be simplified. However, even though schematics of the interfaces have been released as open hardware, compactness of the design actually makes the interface quite costly and hard to recreate, especially for unskilled users. With this system, animators could decide to animate a complex armature in parts, by reusing the same tangible prop on different parts. However, the mapping process still requires to work with a mouse & keyboard interface, and retargeting results might not be as intuitive as in the case of a complete armature.

In [9], the solution in [8] is extended to cope with the control of virtual characters that contain tens or hundreds of DOFs with a TUI composed of a small set of elements through a retargeting strategy based on rig simplification. The idea is to find a simple geometric skeleton (intended as a subset of the original rig) which can be paired with the given set of tangible elements. Based on the number of available elements, the number of DOFs manipulated at the same time can be adjusted, thus controlling posing accuracy. The method is dependent on a preparatory step, in which a number of sample poses need to be defined in order to let the system prioritize DOFs to be controlled. Hence, results are influenced by the quality and size of the sample set.

Another limitation of solutions presented so far is that they are designed to collect only relative measurements. Thus, to control, for instance, character position and orientation, other interfaces should be used in separate animation steps. Some attempts to address the above issue have been made already, although under simplified conditions. For instance, in [17] and [46] rigid objects are manipulated and tracked to animate corresponding 3D models in a virtual set, but only unarticulated tangible devices are considered.

By taking into account advantages and drawbacks of the above solutions, the approach proposed in this paper combines general-purpose consumer-level hardware with an automatic mapping process designed to support the assembly of the tangible device, with the aim to help the animator create, based on available bricks, the best topology possible for the particular character to be animated. Rather than relying on a rig simplification like in [9], the devised approach aims to produce an intuitive configuration of a limited set of tangible components that allows the animator to control, in separate steps, all the original DOFs of a rigged virtual character by still working with a small number of physical controls. The tangible prop and the animator who is handing it are immersed within an affordable motion capture environment designed to let both skilled and unskilled users animate 3D virtual characters in a natural way by using keyframing as well as performance-driven techniques.

3 PROPOSED SYSTEM

In the following, the designed animation system will be introduced, by providing also some implementation details.

3.1 Architecture Overview

The proposed architecture is illustrated in Fig. 1. The animator interfaces with the *Animation software* (right) through an *Interaction agent* (middle), which is in charge to gather interactions originating from multiple *Input devices* (left).

The architecture has been designed for extensibility. Thus, for instance, the *Input devices* block is meant to group interaction systems based on different sensing technologies (cameras, microphones, motor encoders, etc.) whose output could be used to operate the software and create the animation. In this work, three types of input interfaces were considered, namely, a *Tangible interface*, a *Body tracking interface* and a *Speech interface*, handled by the *Interaction agent* through the *Input devices manager* block.

The *Tangible* and *Body tracking interface* blocks are used as sources of position and orientation data to be mapped onto
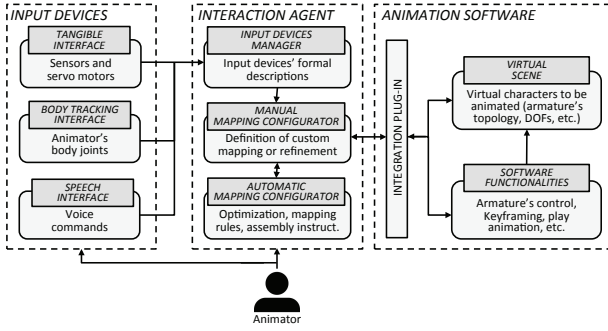
Fig. 1. Architecture of the proposed animation system.

specific transformations of the character to be animated. The *Speech interface* block is exploited to recognize voice commands that are used to control the *Animation software*.

The *Animation software* hosts information concerning the *Virtual scene* to be animated, i.e., about 3D objects and their deformers (at present, animation control has been experimented with rigid transformations, on generic objects, and armature-based deformations, on rigged characters, though in the future it could be extended to other scenarios, e.g., relying on shape keys, drivers, etc.). This block implements a number of *Software functionalities* that need to be activated by the animator, e.g., to select character's parts, insert keyframes, etc. An *Integration plug-in* block, which is tailored to the specific software used, is responsible for making the above information available to the designed system.

Information collected by the *Input devices manager* and the *Integration plug-in* is exploited to define a mapping between the set of available interface elements, i.e., servo motors, sensors and body-tracked joints, and the DOFs of the character's armature to be animated (later referred to as the *target armature*). Mapping can be set up manually by the animator by means of the *Manual mapping configurator* block, which provides a set of GUIs for directly controlling the assignment of each individual servo motor, sensor and tracked joint to a specific DOF in the target armature.

Unfortunately, since character's target armature may be made up of a significant number of bones, it could be difficult for an animator to ensure that the proper mapping is used. This is especially true when the number of available interface elements is not enough for controlling all the character's DOFs at one time. In these situations, in order to optimize the affordance in using the considered input interfaces, mapping should take into account multiple factors, like the similarity between the topology of the target armature and the possible assemblies of available tangible bricks or tracked body parts, the DOFs (and related ranges) of character's bones and of interface elements, etc. To deal with this issue, an *Automatic mapping configurator* has been devised, which receives information collected by the above blocks and is in charge to propose a solution to the problem of controlling all the DOFs of the target armature through a possibly-limited set of interface elements by minimizing the assignment costs for all the (DOF, element) pairs.

The animator can choose to work with the *Tangible interface* and the *Body tracking interface* either in a separate or combined way. In the latter case, the animator

could benefit at the same time from the affordances offered by the tangible elements as well as from the availability of absolute positioning information and the possibility to adopt performance-driven animation obtained through body tracking. Both forward and inverse kinematics are supported, by either mapping input data directly onto virtual character's bones or indirectly onto end-effectors.

Fig. 2 illustrates the above process for a lamp character with a small number of DOFs. Starting from the target armature and available interface elements reported on the left, the *Automatic mapping configurator* would create a single set of mapping rules and the tangible elements assembly shown in the middle (mapping could then be manually refined). On the right, an animator using the tangible prop (whose shape, in this case, mimics that of the character) and body pose to control the target armature is depicted.

When the number of available interface elements is lower than the number of DOFs to be controlled, the configurator would explore all the possible decompositions of the target armature in so-called *partitions*. An example is given in Fig. 3. Here, the interface elements configuration optimizing the overall assignment could be used to control only one of the above partitions at a time, with specific mapping rules for each partition. The configurator would create a vocabulary of voice commands to be pronounced for selecting a partition (and the related set of mapping rules), as well as for activating selected *Software functionalities*. At present, the system can be programmed to handle commands for inserting/deleting/copying/pasting a keyframe, moving in the timeline, enabling/disabling continuous keyframing (for performance-driven animation), playing the animation created, etc. As done in other works (e.g., in [47]), in the future voice commands could be used to control more complex functionalities (e.g., to retrieve recorded actions from pre-defined animation libraries).

When tangible elements are considered, mapping rules are exploited also to create step-by-step instructions required for assembling servo motors and sensors included in the handed prop, thus making it easier to use the system for both skilled and unskilled animators. The procedure developed for generating instructions is presented in Appendix A (included as supplemental material and available online[4]. A video showing the automatic mapping process from the selection of the virtual character to the assembly of the tangible interface is available for download[5]. Source code of the devised system is also available on GitHub[6].

3.2 Implementation

As said, the *Tangible interface* block has been currently implemented through a collection of off-the-shelf Lego Mindstorms EV3 bricks included in the core (#5003400) and expansion (#45560) sets, encompassing medium and large servo motors and touch, infrared, ultrasonic and gyro sensors. User's interaction gathered by the above elements is collected by one or more intelligent brick components, which are programmed to transfer numerical readings to

4. https://www.dropbox.com/s/ua8jddtjrqa9tht/appendix.pdf
5. https://youtu.be/shiZ1i2uAJU
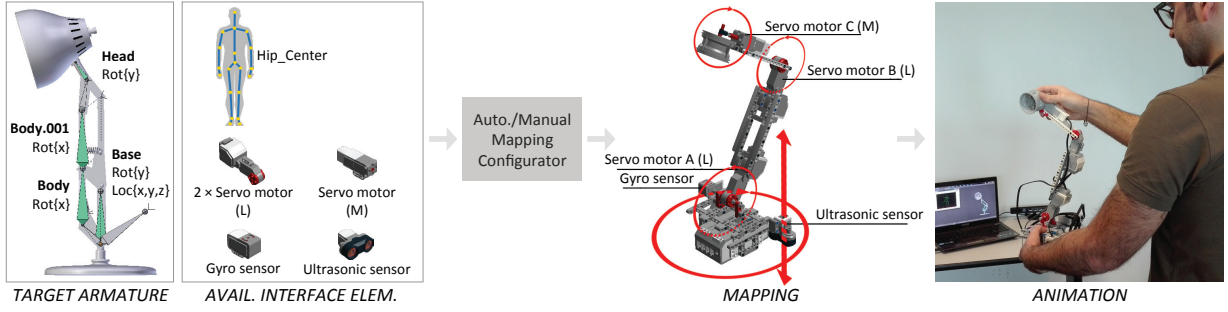6. https://github.com/grainsgroup/tui-nui-char-anim

Fig. 2. Definition of the mapping rules to control the target armature of a lamp character with a given set of tangible bricks and body-tracked joints.
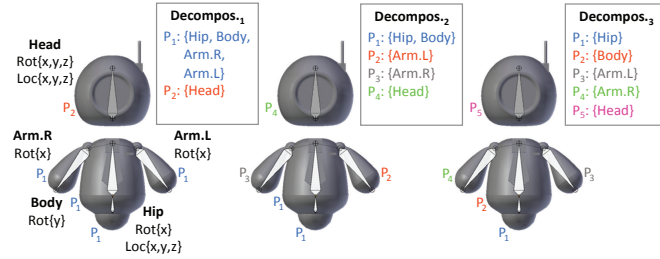

Fig. 3. Decompositions and partitions generated by the *Automatic mapping configurator* for a target armature with 5 bones and 13 DOFs when only 6 interaction elements, (namely 5 servo motors/sensors plus the Kinect-tracked hip joint) are available. In the various decompositions, a bone may be assigned to a different partition (indicated by $P_\#$)

the *Interaction agent* in JSON over a Wi-Fi connection implemented using a third-party API[7] (with Wi-Fi, a sampling rate of 10Hz was achieved, and latency was maintained, on average, around 370ms, well below what was experienced over USB). In the future, intelligent bricks could also be used to drive the servo motors and block them into a given position. Other instrumented construction kits may be explored as well. The *Body tracking interface* is presently based on data coming from a Microsoft Kinect device, which has been used to gather a 20-joint representation of the animator's body. Microsoft Kinect could be easily replaced by other technologies for real-time skeleton tracking. The *Speech interface* block has been developed using the Microsoft Speech Platform[8] library. Finally, in this work, Blender has been used as *Animation software*, and the plug-in implemented in Python (whereas the *Interaction agent* has been written in C#). However, extension of the methodology to other scriptable animation suites would be rather straightforward.

4 AUTOMATIC MAPPING

In this section, the approach designed to let the *Automatic mapping configurator* block of Fig. 1 help the animator in the configuration of the interfaces to be used for controlling the pose of a rigged character in the 3D space is presented.

4.1 Optimization Procedure

As said, mapping between interface elements and character armature's parts is determined by an optimization procedure, whose steps are illustrated in the following sections.

7. https://github.com/BrianPeek/legoev3
8. https://msdn.microsoft.com/en-us/library/jj127858.aspx

4.1.1 Selection of the Mapping Mode

The first step consists in asking the animator to choose between a *TUI-*, *NUI-* or *TUI+NUI-based mapping mode*. The TUI-based mapping mode will exploit data coming from the *Tangible interface* block in Fig. 1, i.e., from servo motors and sensors available for assembling the tangible interface. The NUI-based mapping will rely on data coming from the *Body tracking interface* block (i.e., from the Kinect-tracked body joints). In principle, the first two mapping modes should be considered as alternative, mainly because many Kinect-tracked joints would be occluded during manipulation of the tangible interface, and because it could be generally uncomfortable for the animator to use, e.g., his or her arms, to pose the character while handing the interface itself. Nonetheless, in order to increase the number of DOFs that can be controlled simultaneously, in the TUI+NUI-based mapping mode the Kinect-tracked hip joint is automatically added to the set of available interface elements, as its use would not affect the manipulation of the tangible prop. The animator can decide to exclude the sensors (in this case, in the TUI+NUI-based mapping mode, the hip joint would be neglected, as it would be assimilated to a sensor). The animator can also decide to make the mapping consider the presence of possible symmetries in the character to be animated, and whether to control position and orientation DOFs together (*P&O control*) or separately (*P|O control*).

4.1.2 Representation of the Target Armature

The mapping process needs to gather, from the *Animation software*, topological information on the target armature. For each bone, the name, the position in the kinematics chain, the bones connected to it, possible copy-location/rotation constraints, and its DOFs are retrieved. Information obtained is recorded into a graph-based representation, where nodes are associated with armature bones, whereas edges express relations between bones. For instance, in Fig. 3, node names as well as position and orientation DOFs for the bones in the considered target armature are reported.

Depending on the complexity of the target armature, the same set of interface elements (i.e., body joints for the NUI-based mapping mode and/or assembly of servo motors, sensors and hip joint for the TUI-/TUI+NUI-based mapping modes) may have to be mapped to one or more subsets of bones. Hence, a partitioning step is required, whose goal is to determine all the ways in which the nodes of the target armature's graph can be grouped so that each partition contains nodes with a total number of DOFs that is lower or

equal to the maximum number of DOFs that can be handled simultaneously with the available interface elements.

To this aim, the number of DOFs provided by the available interface elements is first determined. When the P|O control modality is selected, position and orientation DOFs are counted separately (to account for the fact that servo motors and gyro sensors natively provide orientation DOFs, whereas ultrasonic sensors and the Kinect-tracked hip joint provide position DOFs), and partitioning is firstly performed by considering only orientation DOFs. For the P&O control modality no distinction is made, as each interface element is assumed to provide generic DOFs. Partitioning is controlled by two parameters, *MaxBonesInPartition* and *MaxDofsInPartition*. The first parameter indicates the maximum number of bones that the partition can contain, and ranges between 1 and partition size. The second parameter refers to the maximum number of DOFs for the partition. The lower bound is defined by the bone with the highest number of DOFs, whereas the higher bound is given by the number of available interface elements. For the P|O control modality, this value is determined by considering the bone with the highest number of position or orientation DOFs, depending on the type of partitioning being performed; for the P&O control modality, its value is derived from the bone for which the sum of position and orientation DOFs is maximum. If the minimum value of *MaxDofsInPartition* is greater than the number of available interface elements, then there is at least one bone whose DOFs will have to be split among different partitions (and this fact would have an impact on intuitiveness of the resulting mapping, especially for bones with three orientation DOFs). In the partitioning, the two parameters are varied to generate alternative partitions (referred to as partial decompositions).

Partitioning is executed on each of the weakly-connected components of the graph (2, in the example in Fig. 3, identified by bones with different shades), by initiating a depth-first search on all the nodes in the graph component and creating a new partition when the bone visited has no relation with the last node added to current partition or when the maximum number of bones/DOFs has been reached for the partition. Before passing to the next component, the values of *MaxBonesInPartition* and *MaxDofsInPartition* are adjusted until the lower bounds are reached, and the graph is visited again. The above step is repeated for all the nodes in the component, by removing possible duplicates.

Partial decompositions obtained for all components are finally combined through a Cartesian product, thus getting all the graphs decompositions, i.e., sets of partitions, to be considered in the mapping. When splits (i.e., bones with more than one children) are found, partitioning can consider the possible presence of symmetries. If a symmetry is identified (parent bone with children conventionally marked with .L/.R, in Blender's notation) and the animator has chosen to consider them, only partitions that include both parent and children bones or partitions where parent and children are all separated are created. Otherwise, .L or .R bones are discarded. Bones with copy-location/rotation constraints are neglected, as well as as bones belonging to an inverse-kinematic chain (in this case, only the end-effector is considered for the next processing steps).

For instance, assuming that the TUI+NUI-based map-

ping mode has been selected and 5 tangible bricks, e.g., 3 large and 1 medium servo motors and 1 ultrasonic sensor (i.e., 5 DOFs) plus the Kinect-tracked hip joint (i.e., 3 DOFs) are available, for the considered target armature the algorithm would generate the 3 decompositions in Fig. 3 (with symmetries, otherwise 8 decompositions would be created).

4.1.3 Problem Formulation

As said in Section 3, in the devised formulation automatic mapping is modeled as an assignment problem, where the goal is to find the minimum cost association between the bones of each partition of the target armature and the available interface elements. By referring to possible configurations of interface elements in terms of candidate *source armatures* (i.e., collection of bones, as for the target armature) and describing them as graphs, costs can be expressed through entries c_{ij} of a matrix C, where row i represents a bone of the target armature in a specific partition, whereas column j describes the bones of a given source armature.

4.1.4 Source Armatures Generation

When the animator chooses the NUI-based mapping mode, there is just one source armature, in this work defined by the bones of the Kinect skeleton. When the TUI- or the TUI+NUI- based mapping modes are selected, the set of source armatures needs to be first generated. The generation process passes through the creation of so-called *source armature templates*, i.e., armatures that can be exploited to control the DOFs of the target armature but whose bones do not have any particular interface elements assigned (yet).

Source armature templates generation begins with the calculation of all the permutations with repetition of length n (where n is equal to the number of available interface elements) of the three-axis set (x, y, z), resulting in 3^n possible DOF sequences. If the P|O control modality has been selected, the number of occurrences of the above DOF sequences in target armature's partitions is computed for each orientation decomposition, by considering alternative representations based on Proper Euler and Tait-Bryan angles. For the P&O control modality, although partitions explored contain bones with both position and orientation DOFs, position DOFs are neglected since it is not necessary to map the specific interface element controlling a given DOF on an exact axis (for instance, a servo motor mounted along the x axis could easily control a position DOF along the y axis). DOF sequences with the highest number of occurrences are selected (as they are those that can control all the orientation DOFs of the corresponding target armature), and unused DOFs removed (for the P|O control modality) or interpreted as generic, or "don't care" $(-)$ DOFs (for the P&O control modality). For instance, for the two partitions in the first decomposition of Fig. 3, DOF sequence produced would be $(-, -, -, x, y, x, x)$. The process is illustrated in Fig. 4.

DOF sequences are then converted into source armature templates by combining DOFs in a sequential way or introducing splits (if splits are present in the partitions of the decomposition). For the same source armature template, a number of *alternatives* are generated since a particular DOF sequence can represent one or more bones, each with one or more DOFs (when a split is found, alternatives with and without symmetric children are considered). A sample
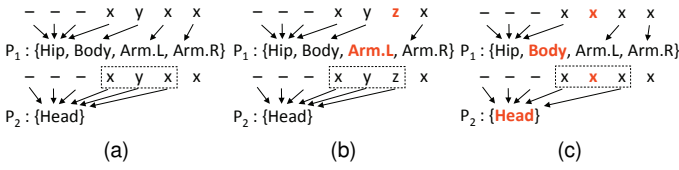
Fig. 4. Determination of the DOF sequences with the highest number of occurrences in the two partitions of the first decomposition of Fig. 3: (a) sequence with the highest number of occurrences, (b) sequence whose DOFs do not match all the DOFs of P_1, and (c) sequence whose DOFs do not match all the DOFs of neither P_1 nor P_2. Non-matching DOFs are marked in bold and red. Dashed boxes indicate the particular Proper Euler/Tait-Bryan representation considered.

subset of the source armature templates that would be generated for the DOF sequence $(-, -, -, x, y, x, x)$ is illustrated in Fig. 5a. Alternatives generated for sequential and split armature templates by considering other associations of the selected DOFs to the same bone are reported in Fig. 5b and Fig. 5c, respectively. Split source armature templates in Fig. 5a would not be generated if symmetries are ignored and partitions do not contain splits without symmetries.

Source armatures are finally generated by considering all the possible associations between bones in the alternative source armature templates and interface elements available for assembling the tangible prop (in the P&O control modality, the set of elements include the Kinect-tracked hip joint).

4.1.5 Solution to the Assignment Problem

As shown in previous works (e.g., [8], [27]), the assignment problem modeled as illustrated in the previous sections is solved by applying the Hungarian algorithm [50] on matrix C. When the P|O control modality is selected, orientation DOFs are first considered, followed by position DOFs. In the P&O modality, no distinction is made between DOF types.

The algorithm calculates the cost of mapping a particular source armature onto a partition of the target armature, by choosing the minimum cost obtained for the various mappings. This cost is later referred to as *partition cost*. If the target armature is made up of multiple partitions, a so-called *decomposition cost* is calculated as the sum of partition costs for all the partitions of a given decomposition. In the P|O control modality, decomposition cost is computed for all the (source armature, orientation decomposition) pairs. The pair with the minimum cost defines the source armature and the configurations to be used for controlling orientations. If the TUI- or TUI+NUI-based mapping modes are used and there are position DOFs to be controlled, the source armature obtained is enriched with interface elements which are not already part of it, and the Hungarian algorithm executed on the partitions in the position decompositions (in the P&O control modality this step is not performed, as the decomposition cost already considers position and orientation DOFs). This way, mapping configurations allowing to control all the DOFs of the target armature are finally produced (the presence of .L/.R bones possibly neglected in the partitioning step because of symmetries is now taken into account).

From the configurations created it is possible to determine the interface elements to which the DOFs of the source armature template have been assigned and, for tangible

bricks, the way they have to be assembled. To simplify the assembly, each servo motor/sensor is assumed to be included in an *elementary* block that can be connected through a "standard" (for this paper) set of mounting points to any other block, on any given axis (by possibly generating a split). More details are provided in Appendix A.

Each configuration is linked to a voice command , which can be later modified. For instance, the configurations created by the TUI+NUI-based mapping mode for the target armature in Fig. 3 with selected interface elements are shown in Fig 6.

4.2 Metrics

Each entry in matrix C is computed as the sum of the costs calculated by the metrics reported in the following sections and expressed in the [0,1] range. Metrics have been designed to evaluate, both in objective and subjective terms, the impact of assigning a bone belonging to a source armature to a bone belonging to a partition of the target armature. Metrics rely on tabulated values, which have been collected in Appendix B (and used for all the experiments reported).

4.2.1 Node Similarity

This metric calculates a topological similarity score between the bones in a source armature and in a partition of the target armature by comparing their graph-based representations (G_T and G_S, respectively) using the measure based on structural similarity of neighborhoods proposed in [48]. A score vector x_k is obtained by iterating the equation:

$$x_k \leftarrow (T \otimes S + T^T \otimes S^T + D_{T_I} \otimes D_{S_I} + D_{T_O} \otimes D_{S_O}) \cdot x_{k-1} \quad (1)$$

where the symbol $\otimes$ represents the Kronecker's matrix product, $\leftarrow$ indicates a vector normalization in the [0,1] range (the Frobenius norm is used, in this case), T and R are the adjacency matrices of G_T and G_S, whereas D_{S_I}, D_{T_I}, D_{S_O} and D_{T_O} are the diagonal matrices containing the out-degree and the in-degree values for every node in G_T and G_S. Number of iterations has been set to 11 as in [27]. Initial condition x_0 can be chosen arbitrarily, since no prior information about node similarity is available. Hence, it is conventionally set to be the all-ones vector [48].

At each iteration, normalization sets a similarity equal to 1 for entries in the score vector corresponding to nodes with the same position in the graph structure. When computation is complete, a score matrix N is obtained by concatenating the resulting score vector. Finally, n_{ij} entries in N are converted to the corresponding c_{ij} entries used to initialize the cost matrix C by computing $c_{ij} = 1 - n_{ij}$. It is worth remarking that, in NUI-based mapping mode, the weight of this metric is increased by considering that source armature is already available (the Kinect skeleton), and higher intuitiveness is expected to be achieved by matching its topology with that of the target armature. An example of NUI-based mapping is given in the Appendix.

4.2.2 DOF Coverage

This metric is designed to penalize the assignments between bones that have different DOFs. Given a bone b_T from a
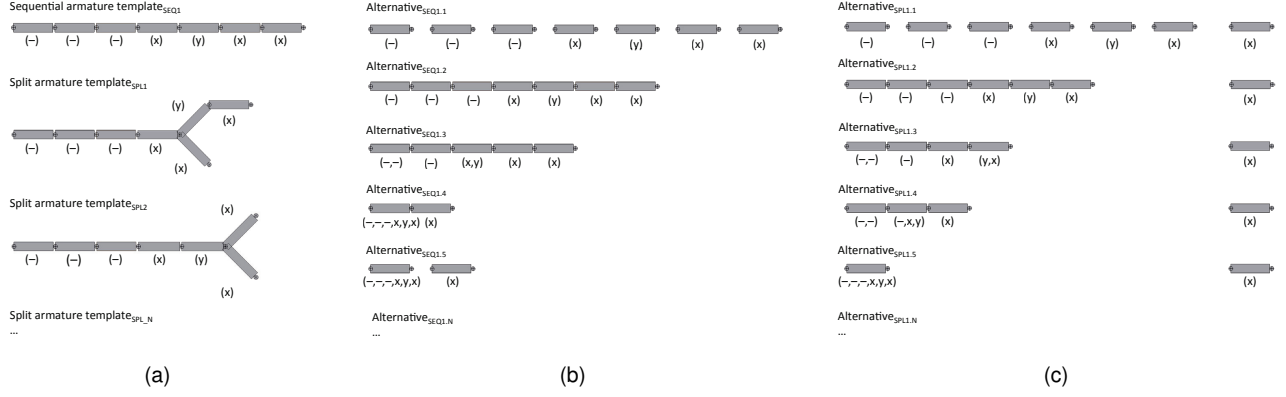
Fig. 5. Subset of (a) sequential and split source armature templates that would be generated for the DOF sequence $(-, -, -, x, y, x, x)$ in Fig. 4, (b) alternatives for the sequential armature template, and (c) alternatives for of the split armature template.
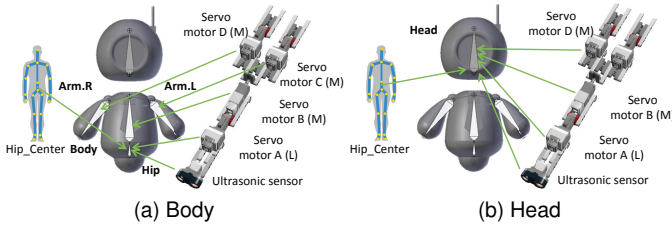

(a) Body   (b) Head

Fig. 6. The two configurations created by the automatic procedure to control the (a) body and (b) head of the armature in Fig. 3 using the TUI+NUI-based mapping mode by working with 3 large and 1 medium motors, 1 ultrasonic sensor and the Kinect-tracked hip joint (P&O modality). Arrows indicate element-to-bone mapping for each configuration.

partition of the target armature and a bone b_S from the source armature, it is calculated as:

$$DOC(b_T, b_S) = 1 - \frac{controllable\_dofs(b_T, b_S)}{dofs} \quad (2)$$

where the function $controllable\_dofs()$ returns the number of DOFs in b_T that can be controlled by b_S, whereas $dofs$ is the total number of DOFs of b_T. In the computation, the alternative representations of b_T based on Proper Euler and Tait-Bryan are considered.

4.2.3 Component Range

This metric, which penalizes the use of components with a small operating range, is defined as:

$$COR(b_T, b_S) = \sum_{i=0}^{dofs} \frac{range\_cost(dof_i, c_i)}{dofs} \quad (3)$$

where dof_i is the i-th DOF of b_T, c_i is the component associated with that DOF and dof has the same meaning of eq. (2). Function $range\_cost()$ is defined as:

$$range\_cost(dof_i, c_i) = 1 - \frac{range(dof_i, c_i)}{max\_range} \quad (4)$$

where function $range()$ is designed to return the operating range of the the given component for the particular DOF, and max_range defines the maximum range for the components used. A description of the $range()$ function and of values returned is reported in the Appendix (Tables 1–3).

4.2.4 Component Annoyance

This metric measures the counter-intuitiveness of using a given component to control a specific DOF as:

$$COA(b_T, b_S) = \sum_{i=0}^{dofs} \frac{annoyance\_cost(dof_i, c_i)}{dofs} \quad (5)$$

where function $annoyance\_cost()$ returns a cost value describing annoyance for the considered (DOF, component) pair. Values were empirically defined, as reported in the Appendix (Table 4).

4.2.5 Position in Chain

This metric penalizes the assignment of bones with different positions in the two kinematic chains, and is defined as:

$$PIC(b_T, b_S) = \frac{|level(b_T) - level(b_S))|}{max\_level} \quad (6)$$

where function $level()$ return the distance, in terms of number of bones, between the bone passed as parameter and the root, whereas max_level is the maximum level that can be reached by visiting the target armature. For the purpose of computing the metric, the Kinect skeleton is considered as divided into 5 chains, as illustrated in Table 5 of the Appendix (the first bone of each chain is the root).

4.2.6 Symmetry

This metric assigns a cost $SYM(b_T, b_R) = 0$ if bones b_T and b_R have the same symmetry, 1 if they have opposite symmetry, and 0.5 if a bone has no symmetry and the other has either .R or .L symmetry. Bones in the central chain of the Kinect skeleton have no symmetry assigned.

4.2.7 Partition Count

This metric penalizes the creation of a large number of configurations (i.e., the generation of a possibly high cognitive load for the user, with many commands and mappings to remember) and is defined as:

$$PAC(b_T, b_S) = \frac{partitions(b_T)}{max\_partitions} \quad (7)$$

where $partitions(b_T)$ returns the number of partitions in the specific decomposition the bone belongs to, whereas $max\_partitions$ is the number of partitions in the decomposition with the largest number of partitions.
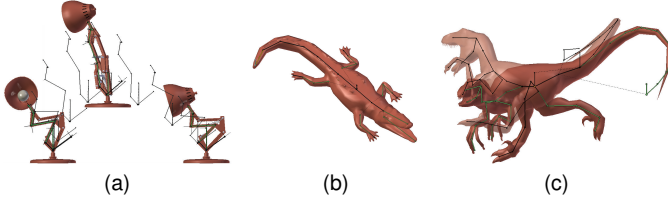
Fig. 7. Case studies: (a) lamp$_{ref}$ task, where 3 poses have to be recorded in as many keyframes by also positioning the character in the virtual space, (b) crocodile$_{ref}$ task, whose goal is to replicate a single character's target pose (shown) with a medium complexity armature, and (c) dyno$_{ref}$ task, featuring a combination of the above goals for a more complex character. In the dyno$_{free}$ task (not shown), the same character in (c) is used, but there is no predefined pose to recreate.

5 RESULTS

In this section, experimental observations that were carried out to assess the effectiveness of the designed solution will be presented. The experimental setup will be first illustrated. Then, performance metrics will be defined and the results of objective measurements discussed. Lastly, the outcomes of questionnaire-based subjective evaluations will be analyzed.

5.1 Case Studies

Experiments considered four case studies characterized by an increasing complexity, which served to separately investigate different features of the proposed system.

The focus of the first case study was on the production of a simple animation by recreating 3 predefined reference poses. The lamp character introduced in Section 3 was selected (characterized by 4 moving bones and 7 DOFs), in order to intentionally keep the complexity of the armature low. The number of interface elements to be used was chosen to make the TUI+NUI mapping create a single configuration for the whole target armature, as shown in Fig. 2. Character's pose needed to be controlled both in terms of armature articulation as well as absolute positioning in the virtual space, in order to match the poses illustrated in Fig. 7a. Voice commands were used to advance in the timeline and to control the insertion of keyframes. In the following, this case study will be referred to as the $lamp_{ref}$ *task*. A video is available for download[9].

The second case study was aimed to analyze the articulation of a character with a medium complexity armature (16 moving bones, 24 DOFs) using both direct and inverse kinematics. The crocodile character illustrated in Fig. 7b (in the pose to be recreated) was chosen, and the TUI+NUI-based mapping mode selected by working with 2 large servo motors and 1 ultrasonic sensor (plus the Kinect-tracked hip joint). As shown in Fig. 8, the automatic mapping process created 6 different configurations, each activated by a voice command and controlling a subset of the target armature's bones. In the following, this case study will be referred to as the $crocodile_{ref}$ *task*. A video is available for download[10].

Based on the findings obtained from the two scenarios above, two additional case studies were designed by combining the characteristics of the previous ones and adding further complexity/diversity. A much more complex dyno
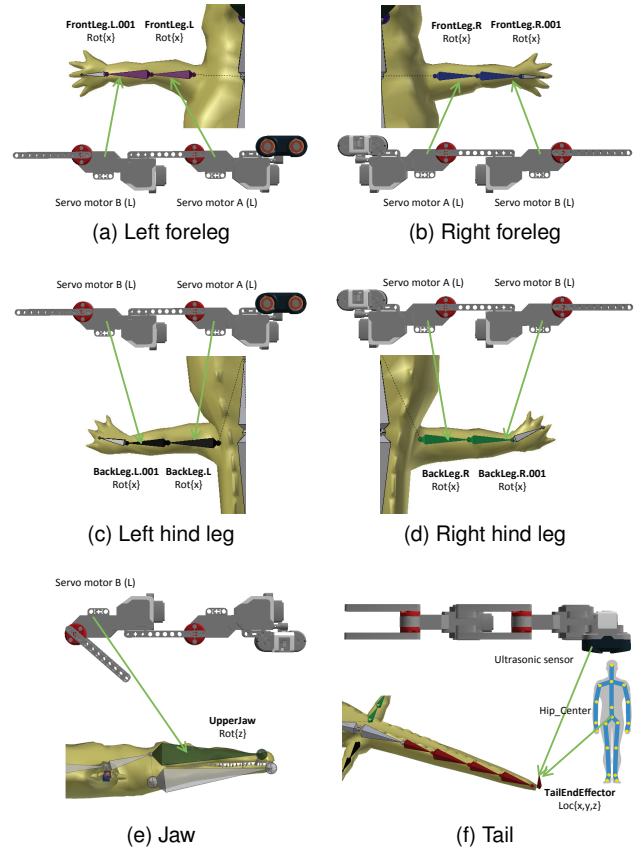

Fig. 8. Configurations created for the target armature selected for the crocodile$_{ref}$ task, and mapping on the interface elements. Configurations are activated by the given (user-defined, in this case) voice commands.

character was selected (38 moving bones, 88 DOFs), by considering a set of interface elements including 3 large and 1 medium servo motor plus the Kinect-tracked hip joint. As illustrated in Fig. 9, with the TUI+NUI mapping 13 configurations were created (bones-configuration associations and corresponding voice commands are shown using colors).

Similarly to the lamp$_{ref}$ task, in the third case study character's armature had to be controlled to match the 2 reference poses in Fig. 7c. In the following, this case study will be referred to as the $dyno_{ref}$ *task*. A video is available for download[11].

In the last case study, in order to investigate a scenario closer to real usage conditions, the constraint to produce the animation by recreating reference poses was removed. Animators were asked to create at least 5 different poses from scratch, by moving (not necessarily in all the keyframes) the various character's parts (legs, arms, torso, head and tail). The whole set of voice commands could be used. This case study will be later referred to as the $dyno_{free}$ *task*. A video showing an example of walk cycle animation is available[12].

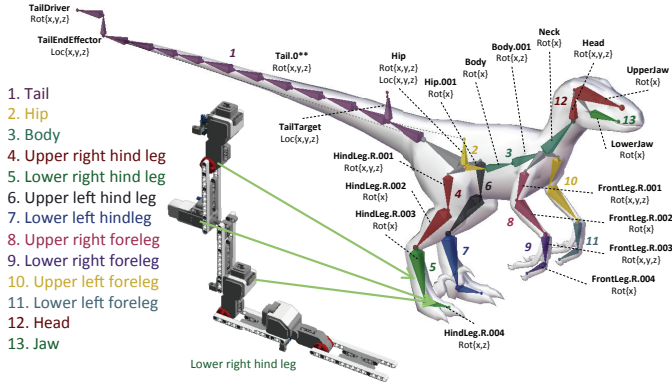More mapping examples obtained with different characters and other settings are reported in Appendix C.

---

9. https://youtu.be/h-4GBxjtvgU
10. https://youtu.be/ODfcjeJAaiU
11. https://youtu.be/Lfs0MUohxd8
12. https://youtu.be/DozMaDil4y4

Fig. 9. Configurations created for the target armature used in the dyno_{ref} and $\text{dyno}_{\text{free}}$ tasks. Mapping onto the interface elements for the configuration activated by the voice command "lower right hind leg" is shown.

5.2 Experimental Setup and Procedure

Experiments involved 52 volunteers (32 for the first and second task, 20 for the third and fourth task), aged between 20 and 42 years, recruited among university students and staff with different backgrounds. Specifically, half of the subjects were considered as skilled users, as they attended at least a computer animation course and/or they have already used one or more animation suites. The remaining subjects were considered as newbies. Thanks to the involvement of both kinds of users, it was possible to assess the potential of the proposed system in a wide spectrum of usage conditions.

Each volunteer was asked to carry out the tasks by using both the interfaces created by the proposed system and the mouse & keyboard (M&K) Blender's interface. To compensate for a possible learning effect, half of the subjects were asked to start with the proposed interface, half of the subjects with the reference one. All the subjects underwent a preliminary training on the use of the tangible and natural interfaces. Additionally, first-time users were introduced to the use of the main functionalities of the M&K interface, as well as to the basics of computer animation needed for completing the tasks. During the training, subjects were left free to pose other characters with armature topologies analogous to the ones used in the experiments, in order to get acquainted with the interaction means.

Volunteers were asked to work on the various tasks by starting with the character in the rest pose. Before moving any character's part, each subject had to register the current pose of the tangible and natural interfaces to the rest pose of the (currently articulated part of the) virtual character, by visually matching the reference and target armatures. This way, a direct and straightforward perception of the current character's pose could be built, as if the given set of virtual bones were in the animator's hands and body. Once a match was found, a voice command had to be issued to reset the association between data provided by interface elements and target armature's bones. This way, bones' position and orientation DOFs could be controlled using relative changes in the tangible interface and in the animator's skeleton.

After having registered the rest pose, the steps involved in the test procedure were as follows:

1) adjust the orientation and/or position of the con-

trolled armature's parts;
2) change the active set of controlled bones;
3) insert a keyframe;
4) move to another frame in the timeline and iterate.

Not all of the steps were actually performed in all the tasks (e.g., last step was not needed in the $\text{crocodile}_{\text{ref}}$ task).

For the first three tasks, similarly to [5] and [8], no time limit nor minimum accuracy threshold were set a priori. That is, volunteers were left free to decide when to consider the task as completed, either because they felt that pose has been replicated in a proper way, or because they were not able to improve it further. However, in the execution of the first two tasks, the system was programmed to inform them through an audio signal when the distance between the current and target pose has gone below a given threshold. In the execution of the third task the notification was disabled, in order to evaluate its possible impact. In the fourth task, since there was not a specific goal to reach, it was up to the volunteer deciding when to consider the animation as ready.

5.3 Performance Indicators

Evaluation was carried out both in objective and subjective terms. This section describes the performance indicators that have been exploited in the objective evaluation of the first three tasks, in which a measurable goal was set (namely, recreating the predefined reference poses). Indicators have been obtained by extending those used in [8] to account for the fact that, with the proposed system, the animator can perform also absolute positioning.

The first indicator is the *completion time* (T_C), which takes into account the time needed to complete the required task, i.e., to finalize the character's pose for a given frame or set of keyframes, depending on the task. When using the proposed system, each task started with a voice command. When using M&K, a mouse click was required. Task completion was identified by the insertion of the last keyframe. Time required for moving in the timeline was neglected.

The second indicator is the *pose distance (D)*, which measures the proximity of the current pose reached by the animator's controlled character with respect to the reference one. It represents the complementary measure of pose accuracy and it varies between 0%, when the two poses perfectly overlaps, and 100%, when the distance between them is equal to the one measured when interaction began. This metric is computed, for a given keyframe, by averaging the normalized Euclidean and angular distances between each bone of the two character' armatures (L_2 norms), i.e., as:

$$D = \frac{1}{2} \cdot \sum_{i=0}^{n} \left(\frac{\delta_i}{\Delta} + \frac{\theta_i}{\Theta} \right) \qquad (8)$$

where n is the number of bones, whereas δ_i and θ_i are the Euclidean and angular distances between the i-th bones of the two armatures. Δ and Θ are the normalization factors for the Euclidean and angular distances, defined as:

$$S = \sum_{i=0}^{n} \delta_i^*, \ \ \Theta = \sum_{i=0}^{n} \theta_i^* \qquad (9)$$

where δ_i^* is the Euclidean distance between the initial (rest) position of the i-th character's bone and its target position,
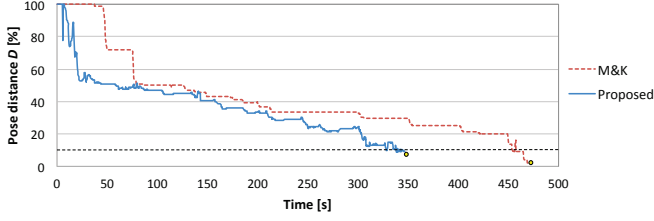
Fig. 10. Example of amount of work for a user carrying out the dyno$_{ref}$ task (only one frame is shown). Horizontal dashed line represents the 10% threshold on pose distance for audio notification (not used here).

(a) Compl. time (T_C)   (b) Pose distance (D)   (c) Am. of work (W)

(d) Compl. time (T_C)   (e) Pose distance (D)   (f) Am. of work (W)

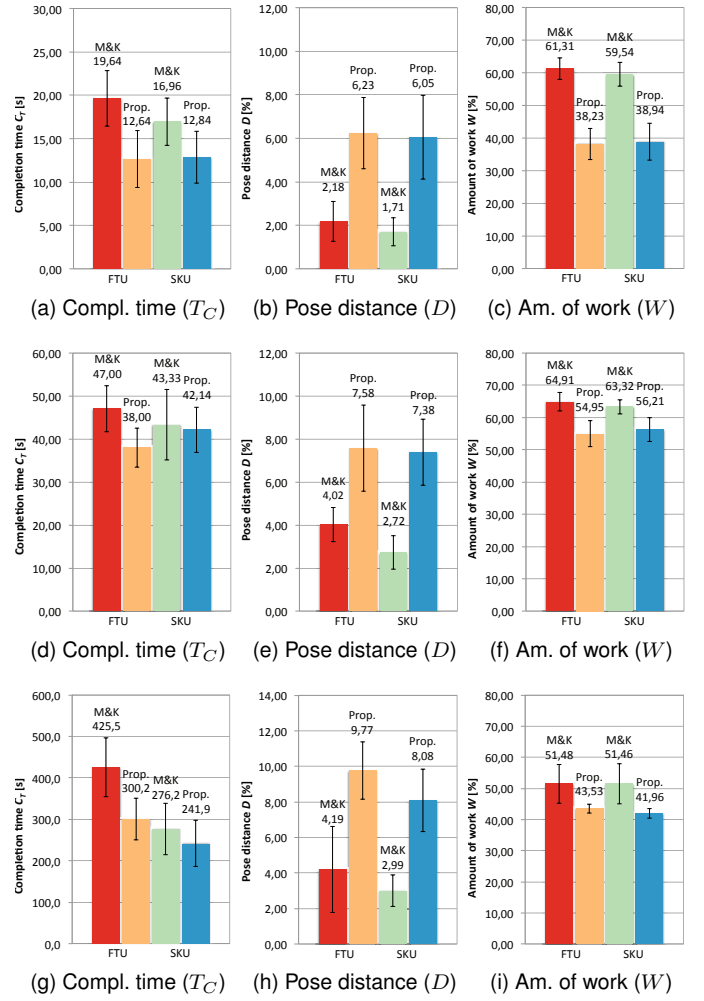(g) Compl. time (T_C)   (h) Pose distance (D)   (i) Am. of work (W)

Fig. 11. Results in terms of operation speed (completion time), accuracy (pose distance) and amount of work for the lamp$_{ref}$ (a–c), crocodile$_{ref}$ (d–f) and dyno$_{ref}$ (g–i) tasks (first-time users, FTU, and skilled users, SKU).

whereas θ_i^* is the absolute angular displacement between the initial (rest) and the target pose computed as the final angular position relative to its initial position. When Δ or Θ are equal to zero, the contribution of the corresponding term is neglected and averaging avoided.

The last indicator, named *amount of work (W)*, provides a measure of the work necessary to create an animation as similar as possible to the reference, and is defined as:

$$W = \frac{1}{2 \cdot T} \int_0^t \sum_{i=0}^n \left(\frac{\delta_i(t)}{\Delta} + \frac{\theta_i(t)}{\Theta} \right) dt \qquad (10)$$

where $t = 0$ represents the starting time of the test, whereas $t = T$ is the maximum time at which the minimum pose distance is reached with the two interfaces. Basically, W gives the flavor of how simple or complex it is reaching a given animation goal [8]. Small values are to be interpreted as an indication of a quick pose updating, which allows the animator to rapidly reach a pose as similar as possible to the target one, whereas large values denote a slow convergence to the target pose. Similar considerations made about Δ and Θ for eq. (8) also apply to eq. (10). Amount of work for a given user carrying out the dyno$_{ref}$ task with the proposed and the M&K interfaces is represented by the area under the two curves illustrated in Fig. 10.

5.4 Objective Evaluation

Results obtained by calculating the T_C, D and W indicators on experimental observations can be interpreted in different ways, depending on the type of users who actually carried out the given tasks. Hence, in the following, first-time users will be first considered. Then, results for skilled users will be discussed. Lastly, an intergroup analysis will be performed.

Fig. 11 reports the results in terms of operation speed (completion time), pose accuracy (pose distance) and amount of work obtained by both groups in the execution of the lamp$_{ref}$ (Fig. 11a-c), crocodile$_{ref}$ (Fig. 11d-f) and dyno$_{ref}$ (Fig. 11g-i) tasks for the proposed and the M&K interfaces. Results have been calculated by averaging indicator values on all the keyframes.

For all the tasks, first-time users benefited from the use of the proposed interface both in terms of operation speed and amount of work. Statistical significance was studied through paired student t-tests. Completion time with the proposed interface was, on average, significantly lower than using M&K (35% faster, $p = 8.11 \cdot 10^{-8}$, for the lamp$_{ref}$ task, 20% faster, $p = 3.4 \cdot 10^{-3}$, for the crocodile$_{ref}$ task, and 30% faster, $p = 1.0 \cdot 10^{-3}$, for the dyno$_{ref}$ task).

The same considerations are valid also for the amount of work, which in all the cases was, on average, significantly lower with the proposed interface than with the M&K one ($p = 1.86 \cdot 10^{-19}$, $p = 1.56 \cdot 10^{-3}$, and $p = 3.6 \cdot 10^{-3}$). Benefits above were paid in terms of accuracy, which resulted to be lower for the proposed interface than for the M&K one ($p = 1.64 \cdot 10^{-8}$, $p = 7.32 \cdot 10^{-4}$, and $p = 4.14 \cdot 10^{-2}$).

Results for skilled users suggest that, overall, the proposed interface was never slower than the M&K one. In the crocodile$_{ref}$ and dyno$_{ref}$ task, it was not possible to find a statistically-significant difference between the two means ($p = 0.81$, and $p = 0.14$), whereas in the lamp$_{ref}$ task, the proposed interface performed 25% faster than the M&K one ($p = 5.76 \cdot 10^{-4}$). Results also confirm that the amount of work is again lower with the proposed interface than with the M&K one ($p = 2.18 \cdot 10^{-15}$, $p = 4.71 \cdot 10^{-3}$, and $p = 3.09 \cdot 10^{-2}$), though accuracy is lower as well (consideration made for first-time users still apply). In the lamp$_{ref}$ task, the benefit coming from the use of the proposed interface was more evident, due to the fact that the combination of different input modalities allowed animators to simultaneously adjust both the armature's shape as well as character's position in the virtual space (not considered in

the other two tasks), which is not possible using M&K.

Considering operation speed, it can be observed that the gain obtained with proposed interface by skilled users is lower than for first-time users: skilled users were faster than first-time users in completing the tasks with the M&K interface as they were accustomed to work with it. Moreover, by focusing on operation speed with the proposed interface, it can be observed that differences between skilled and first-time users were lower than with M&K. These facts explain why skilled users benefited less from the proposed system and confirm that the training phase was effective in clearing the differences between the two groups. It is worth observing that results are particularly relevant for skilled users, since they were able to complete the assigned tasks in almost the same time with the two interfaces, despite their previous knowledge of the M&K one.

Concerning pose distance, skilled users were always more accurate than first-time users when working with the M&K interface. With the proposed interface, this consideration is valid only for complex armatures. For low-medium complexity armatures, users reached similar accuracy levels. This can be due both to the fact that differences between groups are flattened when using the proposed interface as well as to the limited sensitivity of the tangible bricks used [49]. It is worth noticing that the presence of the audio notification was not found to have an influence on completion time or accuracy. Volunteers continued articulating the charactering for 13%, and 5% of the overall posing time in the $lamp_{ref}$ and $crocodile_{ref}$ tasks (without the notification), and for 4% for the $dyno_{ref}$ task (with the notification).

An interesting observation can be made by comparing results in terms of amount of work obtained by first-time users with the proposed interface and skilled users with the M&K one. Results show that first-time users were able to complete both the tasks with a lower overall effort when using the proposed interface. When completion times are similar with the two interfaces but amount of work is lower with the proposed one, it means that users were satisfied with the reached poses approximately after the same time, but with the proposed interface they spent less time to approach the target pose. Basically, the proposed interface allows animators to quickly draft a rough pose for the character at the cost of a lower accuracy in the final pose.

5.5 Subjective Evaluation

For the first three tasks, volunteers were asked to judge the usability of the proposed interface through a questionnaire based on the ISO 9241-400 standard, which is aimed at evaluating ergonomic and human factors for physical input devices used in interactive systems. Scores pertaining perceived accuracy and operation speed, as well as physical effort, mental effort and intuitiveness for the M&K and the proposed interfaces were collected in a seven-point Likert scale. Comments about pros and cons of the two interfaces were additionally recorded. The 32 volunteers involved in the $lamp_{ref}$ and $crocodile_{ref}$ tasks filled in a single questionnaire for the whole experience. The same questionnaire was administered again to the remaining 20 volunteers who participated in the $dyno_{ref}$ task. For the $dyno_{free}$ task, since the animations created by the volunteers
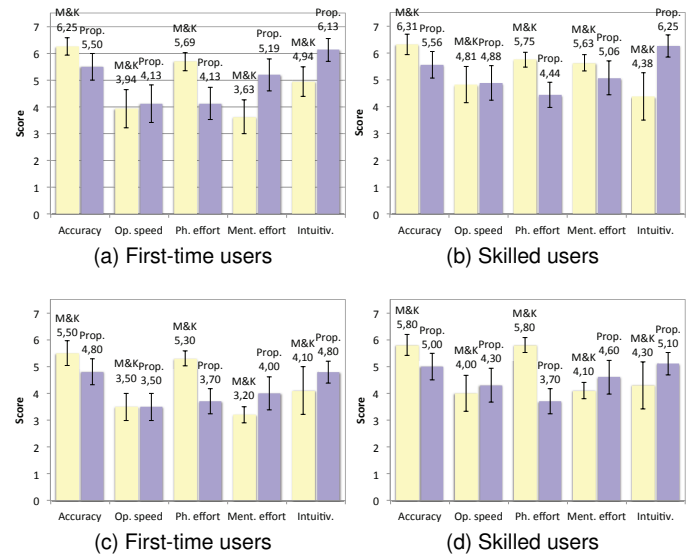

Fig. 12. Subjective evaluation of the proposed and M&K interfaces upon completion of (a, b) the $lamp_{ref}$, $crocodile_{ref}$ and (c, d) $dyno_{ref}$ tasks. For physical and mental effort, higher scores indicate a lower effort (better).

could not be compared, it was chosen to collect their overall preference for either the proposed or the M&K interface.

Results obtained after the execution of the first two tasks are reported in Fig. 12a-b. Results for third task are given in Fig. 12c-d. For the sake of readability, scores have been mapped on a better-to-worse (7-to-1) scale and averaged among the participants. Results can be better analyzed by considering first findings and implications valid for both kinds of users. Then, evidences valid for only first-time users and, lastly, for skilled users can be discussed.

Overall, the analysis of subjective scores for these tasks suggests that the proposed interface is perceived as significantly more intuitive than the M&K one ($p = 0.001$ for the first two tasks, $p = 0.0074$ for the third one). Based on answers to open questions, the proposed interface was perceived as more intuitive thanks to a greater awareness of the character to be controlled and to the affordance of the interface. Intuitiveness is paid with a higher physical effort, motivated by the size of the tangible assembly and by the standing pose required to track the animator's skeleton, when used. Lastly, for both kinds of users, the M&K interface is perceived as significantly more accurate, as observed in the objective evaluation.

First-time users judged the proposed interface significantly better from the point of view of the mental effort required (strong evidences, since $p = 0.0008$ and $p = 0.015$), whereas they perceived the two interfaces comparable from the point of view of the operation speed ($p = 0.08$ and $p = 1.0$). This latter result contrasts with objective measurements, which indicate lower completion times for the proposed interface. Motivation is probably linked to the limited sensitivity of the technology used for the tangible interface, which did not allowed animators to reach the target pose with the desired accuracy. As seen in Fig. 10, the pose distance with the proposed interface quickly drops down when drafting the rough pose for the given keyframe, but then the animators continued to adjust the pose trying

to get closer to the target without actually improving notice-ably the pose distance. Hence, they probably perceived this process as long as with the M&K interface.

Skilled users perceived the proposed interface as fast as the M&K one ($p = 0.69$ and $p = 0.467$), confirming objective observations. No statistically significant difference was found concerning mental effort ($p = 0.13$ and $p = 0.36$). This can be considered as a positive result, as skilled users were supposed to be already accustomed to the M&K inter-face, but not to the proposed one.

Results obtained for the fourth task are aligned with those reported above. In fact, 7 out of the 10 fist-time users expressed their preference for the proposed interface. As expected, based on objective evaluation and on findings from the first three tasks, this inclination was not observed with skilled users, whose preference was equally distributed between the two interfaces. Notwithstanding, as said, given their better knowledge of the M&K interface this result could be considered as rather encouraging.

6 CONCLUSION AND FUTURE WORK

This paper presented a system that enables the animation of virtual characters through keyframing and performance-driven techniques by combining reconfigurable tangible and natural user interfaces in a unified pipeline. To broaden the audience of possible users and include also newbies, the system has been designed to automatically build both the instructions for aiding animators to assemble the tangible interface and the retargeting rules between input interfaces and the particular virtual character to be animated. The devised automatic mapping procedure is capable to take into account the number and type of available interface elements and the topology and DOFs of the character's rig.

Subjective tests revealed that, thanks to the perceived affordances, animating with the tangible and natural in-terfaces used by the proposed system is significantly more intuitive than with M&K, at the cost of an increased physical effort. Improved intuitiveness is confirmed by objective measurements, which also suggest that the proposed sys-tem allows both skilled and first-time users to pose and animate virtual characters faster than by using M&K, thanks also to the possibility to simultaneously control character's pose and location in the 3D space. Results of experimental observations indicated that the improvements in terms of efficiency in creating animations were greater for first-time users than for skilled users, and that first-time users work-ing with the interfaces created by the proposed system were able to operate even faster than skilled users working with M&K, though reached a lower accuracy in the final pose.

Future work will be devoted to foster a tighter inte-gration of tangible and body tracking-based interfaces by dealing with occlusion issues using wearable technologies. Moreover, better uses of active interface elements, e.g., ex-ploiting servo motors to reload configurations, inform the animator of rig constraints, etc. will be explored. The limited accuracy of affordable off-the-shelf input elements used will be also tackled by introducing in the speech interface new functionalities allowing the animator to fine-tune the mapping during animation. Lastly, efforts will be put in trying to improve the process of registering the interface

elements with the character's parts they are mapped onto, e.g., by exploiting size-relations between physical and vir-tual components as well as by adopting immersive virtual reality and augmented reality technologies.

REFERENCES

[1] P.C. DiLorenzo, "Premo: DreamWorks Animation's New Approach to Animation," *IEEE Comp. Gr. & App.*, vol. 35, no. 4, pp. 14-21, 2015.
[2] J. Lin, T. Igarashi, J. Mitani, M. Liao, and Y. He, "A Sketching Interface for Sitting Pose Design in the Virtual Environment," *IEEE Trans. on Vis. and Comp. Graph.*, vol. 18, no. 11, pp. 1979-1991, 2012.
[3] J. Chen, S. Izadi, and A. Fitzgibbon, "KinEtre: Animating the World with the Human Body," in. *Proc. 25th Annual Symp. on User Interface Software and Technology*, pp. 435-444, 2012.
[4] M. Dontcheva, G. Yngve, and Z. Popovic, "Layered Acting for Character Animation," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 409-416, 2003.
[5] W. Yoshizaki, Y. Sugiura, A.C. Chiou, S. Hashimoto, M. Inami, T. Igarashi, Y. Akazawa, K. Kawachi, S. Kagami, and M. Mochimaru, "An Actuated Physical Puppet as an Input Device for Controlling a Digital Manikin," in *Proc. Conf. on Human Factors in Computing Systems*, pp. 637-646, 2011.
[6] L. Leite, "Virtual Marionette," in *Proc. Int. Conf. on Intelligent User Interfaces*, pp. 363-366, 2012.
[7] M. Kyto, K. Dhinakaran, A. Martikainen, and P. Hamalainen, "Improving 3D Character Posing with a Gestural Interface," *IEEE Computer Graphics and Applications,* in press.
[8] A. Jacobson, D. Panozzo, O. Glauser, C. Pradalier, O. Hilliges, and O. Sorkine-Hornung, "Tangible and Modular Input Device for Character Articulation," *ACM Transactions on Graphics*, vol. 33, no. 4, art. 82, 2014.
[9] O. Glauser, W.C. Ma, D. Panozzo, A. Jacobson, H. Otmar, and O. Sorkine-Hornung, "Rig Animation with a Tangible and Modular Input Device," *ACM Transactions on Graphics*, vol. 35, no. 4, 2016.
[10] H.J. Shin, J. Lee, S.Y. Shin, and M. Gleicher, "Computer Puppetry: An Importance-based Approach," *ACM Transactions on Graphics*, vol. 20, no. 2, pp. 67-94, 2001.
[11] D. J. Sturman, "Computer Puppetry," *IEEE Computer Graphics and Applications*, vol. 18, no. 1, pp. 38-45, 1998.
[12] T.H.D. Nguyen, T.C.T. Qui, K. Xu, A.D. Cheok, S.L. Teo, Z. Zhou, A. Mallawaarachchi, S.P. Lee, W. Liu, H.S. Teo, L.N., Thang, Y. Li, and H. Kato, "Real-Time 3D Human Capture System for Mixed-Reality Art and Entertainment," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 6, pp. 706-721, 2005.
[13] J. Lazlo, M. van de Panne, and E. Fiume, "Interactive Control for Physically-based Animation," in *Proc. 27th Annual Conf. on Computer Graphics and Interactive Techniques*, pp. 201-208, 2000.
[14] Y. Wu, T. Lu, and J. Song, "A Real-Time Mesh Animation Frame-work Using Kinect," in *Proc. 14th Pacific-Rim Conf. on Multimedia*, pp. 245-256, 2013.
[15] H. Rhodin, J. Tompkin, K.I. Kim, K. Varanasi, H.P. Seidel, and C. Theobalt, "Interactive Motion Mapping for Real-time Character Control," *Computer Graphics Forum*, vol. 33, no. 2, 2014.
[16] H. Ishii, and B. Ullmer, "Tangible Bits: Towards Seamless Inter-faces between People, Bits and Atoms," in *Proc. Conf. on Human Factors in Computing Systems*, pp. 234-241, 1997.
[17] R. Held, A. Gupta, B. Curless, and M. Agrawala, "3D Puppetry: A Kinect-based Interface for 3D Animation," in *Proc. 25th Annual Symp. on User Interface Software and Technology*, pp. 423-434, 2012.
[18] M.W. Chao, C.H. Lin, J. Assa, and T.Y. Lee, "Human Motion Re-trieval from Hand-Drawn Sketch," *IEEE Transactions on Visualization and Computer Graphics*, pp. 729-740, 2012.
[19] M. Kostandov, R. Jianu, W. Zhou, and T. Moscovich, "Interactive Layered Character Animation in Immersive Virtual Environments," in Proc. *ACM SIGGRAPH*, art. 15, 2006.
[20] M. Oshita, Y. Senju, and S. Morishige, "Character Motion Control Interface with Hand Manipulation Inspired by Puppet Mecha-nism," in *Proc. Int. Conf. on Virtual-Reality Continuum and its Ap-plications in Industry*, pp. 131-138, 2013.
[21] S. Ishigaki, T. White, V.B. Zordan, and C.K. Liu, "Performance-based Control Interface for Character Animation," *ACM Transac-tions on Graphics*, vol. 28, no. 3, art. 61, 2009.
[22] L. Leite, and V. Orvalho, "Anim-actor: Understanding Interaction with Digital Puppetry Using Low-cost Motion Capture," in *Proc. 8th Int. Conf. on Adv. in Computer Entertainment Tech.*, art. 65, 2011.

[23] D. Arsenault, and A. Whitehead, "Wearable Sensor Networks for Motion Capture," in *Proc. 7th Int. Conf. on Intelligent Technologies for Interactive Entertainment*, pp. 158-167, 2015.

[24] D. Sturman, "A Brief History of Motion Capture for Computer Character Animation," *SIGGRAPH*, 1994.

[25] C. Larboulette, and S. Gibet, "I Am a Tree: Embodiment Using Physically Based Animation Driven by Expressive Descriptors of Motion," in *Proc. Int. Symp. on Mov. and Computing*, pp. 1-8, 2016.

[26] I. Baran, and J. Popovic, "Automatic Rigging and Animation of 3D Characters," *ACM Trans. on Graphics*, vol. 26, no. 3, art. 72, 2007.

[27] A. Sanna, F. Lamberti, G. Paravati, G. Carlevaris, and P. Montuschi, "Automatically Mapping Human Skeletons onto Virtual Character Armatures," in *Proc. 5th Int. Conf. on Intelligent Technologies for Interactive Entertainment*, pp. 80-89, 2013.

[28] Y. Seol, C. O'Sullivan, and J. Lee, "Creature Features: Online Motion Puppetry for Non-human Characters," in *Proc. 12th Symp. on Computer Animation*, pp. 213-221, 2013.

[29] C. Barnes, D.E. Jacobs, J. Sanders, D.B. Goldman, S. Rusinkiewicz, A. Finkelstein, and M. Agrawala, "Video Puppetry: A Performative Interface for Cutout Animation," *ACM Transactions on Graphics*, vol. 27, no. 5, art. 124, 2008.

[30] T. Igarashi, T. Moscovich, and J.F. Hughes, "Spatial Keyframing for Performance-driven Animation," in *Proc. of the Symp. on Computer Animation*, pp. 107-115, 2005.

[31] W. Graham, "The story of Waldo C. Graphics," *Course Notes: 3D Character Animation by Computer, ACM SIGGRAPH*, pp. 65-79, 1989.

[32] B. Robertson, "Motion Capture Meets 3D Animation," in *On the Cutting Edge of Technology*, pp. 1-14, 1993.

[33] G.A. Lee, G.J. Kim, and M. Billinghurst, "Immersive Authoring: What You eXperience Is What You Get (WYXIWYG)," *Communications of the ACM*, vol. 48, no. 7, pp. 76-81, 2005.

[34] D. Avrahami, J.O. Wobbrock, and S. Izadi, "Portico: Tangible Interaction on and Around a Tablet," in *Proc. 24th Annual Symp. on User Interface Software and Technology*, pp. 347-356, 2011.

[35] D. Ninomiya, K. Miyazaki, and R. Nakatsu, "Networked Virtual Marionette Theater," in *Proc. 1st Int. Conf. on Ubi-Media Computing*, pp. 397-401, 2008.

[36] C. Esposito, W.B. Paley, and J.C. Ong, "Of Mice and Monkeys: A Specialized Input Devices for Virtual Body Animation," in *Proc. Symp. on Interactive 3D Graphics*, pp. 109-ff, 1995.

[37] N. Numaguchi, A. Nakazawa, T. Shiratori, and J.K. Hodgins, "A Puppet Interface for Retrieval of Motion Capture Data," in *Proc. Symp. on Computer Animation*, pp. 157-166, 2011.

[38] T.C. Feng, P. Gunawardane, J. Davis, and Jiang, "Motion Capture Data Retrieval using an Artist's Doll," in *Proc. 19th Int. Conf. on Patter Recognition*, pp. 1-4, 2008.

[39] M.P. Johnson, A. Wilson, B. Blumberg, C. Kline, and A. Bobick, "Synpathetic Interfaces: Using a Plush Toy to Direct Synthetic Characters," in *Proc. Conf. on Human Factors in Computing Systems*, pp. 152-158, 1999.

[40] N. Shimizu, N. Koizumi, M. Sugimoto, H. Nii, D. Sekiguchi, and M. Inami, "A Teddy-bear-based Robotic User Interface," *Computers in Entertainment*, vol. 4, no. 3, art. 8, 2006.

[41] A. Uribe-Quevedo, H. Leon, and B. Perez-Gutierrez, "Arm-like Mechanism User Interface for 3D Animation," in *Proc. 13th Int. Conf. on Control, Automation and Systems*, pp. 1463-1467, 2013.

[42] B. Knep, C. Hayes, R. Sayre, T. Williams, "Dinosaur Input Device," in *Proc. Conf. on Human Factors in Comp. Systems*, pp. 304-309, 1995.

[43] R. Watanabe, Y. Itoh, M. Asai, Y. Kitamura, F. Kishino, and H. Hikuchi, "The Soul of ActiveCube - Implementing a Flexible, Multimodal, Three-Dimensional Spatial Tangible Interface," in *Proc. Int. Conf. on Adv. in Computer Entertainment Tech.*, pp. 173-180, 2004.

[44] TH.S. Raffle, A.J. Parkes, and H. Ishii, "Topobo: A Constructive Assembly System with Kinetic Memory," in *Proc. Conf on Human Factors in Computing Systems*, pp. 647-654, 2004.

[45] M.P. Weller, E.Y.L. Do, and M.D. Gross, "Posey: Instrumenting a Poseable Hub and Strut Construction toy," in *Proc. 2nd Int. Conf. on Tangible and Embedded Interaction*, pp. 39-46, 2008.

[46] R. Metoyer, L. Xu, and M. Srinivasan, "A Tangible Interface for High-Level Direction of Multiple Animated Characters," in *Proc. Graphics Interface*, pp. 167-176, 2003.

[47] Adobe Character Animator CC, http://www.adobe.com/products/character-animator.html [Online]. Acc. Dec. 24, 2016.

[48] L.A. Zager, and C. Verghese, "Graph similarity scoring and matching," in *Applied Mathematics Letters*, vol. 21, no. 1, pp. 86-94, 2008.

[49] Lego Mindstorms Education User Guide - http://www.nr.edu/csc200/labs-ev3/ev3-user-guide-EN.pdf

[50] H.W. Kuhn, "The Hungarian method for the assignment problem,", in *Naval Research Logistics*, vol. 2, no. 1-2, pp. 83-97, 1955.

Fabrizio Lamberti (M'02-SM'14) is an Associate Professor at Politecnico di Torino, Italy. He received his M.S. and the Ph.D. degrees in computer engineering from Politecnico di Torino, Italy, in 2000 and 2005. His interests pertain computational intelligence, human-machine interaction, computer graphics, and visualization. He serves as an Associate Editor for IEEE Transactions on Emerging Topics in Computing and for IEEE Consumer Electronics Magazine.

Gianluca Paravati (M'14-SM'16) is an Assistant Professor at Politecnico di Torino, Italy, where he received his B.Sc. and M.Sc. degrees in electronic engineering and Ph.D. degree in computer engineering in 2005, 2007, and 2011, respectively. His research interests include computer graphics, human-machine interaction, and machine learning.

Valentina Gatteschi is a Postdoctoral Research Assistant at Politecnico di Torino, where she received her B.Sc. and M.Sc. degrees in management engineering and the Ph.D. degree in computer engineering in 2005, 2008 and 2013, respectively. Her main research interests are in semantics and natural language processing. She has been involved in several European projects on education.

Alberto Cannavò received his B.Sc. degree from University of Messina, Italy, in 2013. He received his M.Sc. degree in computer engineering from Politecnico di Torino, Italy, in 2015. Since November 2016 he is a Ph.D. student at Politecnico di Torino, where he carries out his research in the areas of computer graphics and human-machine interaction.

Paolo Montuschi (M'90-SM'07-F'14) is a Full Professor in the Department of Control and Computer Engineering at Politecnico di Torino. His research interests include computer arithmetic and architectures, computer graphics, and education. He is an IEEE Computer Society Golden Core member, and serves as Editor-in-Chief of IEEE Transactions on Computers, as the Chair of the Computer Society Technical Achievement Award Committee, as a member of the IEEE Publications Services and Products Board, and the IEEE Products and Services Committee.