Virtual character animations from human body motion by automatic direct and inverse kinematics-based mapping

EAI Endorsed Transactions
on Creative Technologies
Research Article ICST.ORG

Virtual Character Animations from Human Body Motion by Automatic Direct and Inverse Kinematics-based Mapping

Andrea Sanna, Fabrizio Lamberti, Gianluca Paravati, Gilles Carlevaris, Paolo Montuschi
Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino 10129, Italy

Abstract

Motion capture systems provide an efficient and interactive solution for extracting information related to a human skeleton, which is often exploited to animate virtual characters. When the character cannot be assimilated to an anthropometric shape, the task to map motion capture data onto the armature to be animated could be extremely challenging. This paper presents two methodologies for the automatic mapping of a human skeleton onto virtual character armatures. Kinematics chains of the human skeleton are analyzed in order to map joints, bones and end-effectors onto an arbitrary shaped armatures. Both forward and inverse kinematics are considered. A prototype implementation has been developed by using the Microsoft Kinect as body tracking device. Results show that the proposed solution can already be used to animate truly different characters ranging from a Pixar-like lamp to different kinds of animals.

Keywords: virtual character animation, automatic armature mapping, motion capture, graph similarity, forward kinematics, inverse kinematics

Received on 16 April 2014, accepted on 05 May 2014, published on 27 February 2015

doi: 10.4108/ct.2.2e6

1. Introduction

Several techniques are used to animate virtual characters. One of the most used is based on virtual skeletons (named rigs), which are related to meshes to be animated and deformed. The animator can move bones of the virtual skeleton in order to move mesh vertices. The so called keyframing method uses forward and inverse kinematics techniques to fix a set of key frames (poses), which will be automatically interpolated by the animation program [1][2]. The keyframing approach has been often outperformed by motion capture solutions [3]. Mocap solutions allow to directly gather the motion of the animator: motion data can been used both to animate interactively a virtual character and recorded and then applied to the character.

The keyframing approach allows animators to author animations that would be difficult or impossible to act out. On the other hand, complex actions usually need a lot of time (and skilled animators) to be produced by keyframing. Motion capture systems can gather complex motion data in a very accurate and smooth way, thus allowing animators to create very realistic (human) movements. Mocap systems are, in general, very expensive and data representing the animators' movements might be hard to be used for animating non-anthropometric characters (even if a solution to partially cope with this latter limitation has been proposed in [4]).

The goal of this paper is to automatically map the skeleton of the animator onto the armature of any virtual character, thus relaxing any possible constraints related to the armature topology to be animated. This task has been manually addressed in a number of previous works, such as [5] [6] and [7]. Manual association (mapping) can be a time consuming and difficult task, since only skilled animators are generally able to immediately identify the *best* match.

This work proposes and compares two different solutions that aim to find an efficient mapping of the human skeleton onto virtual character armatures. Exploiting graph similarity criteria, user preferences, motion constraints and information about the armature topology, the skeleton of the animator can be mapped onto the character armature bones both by a direct mapping and by a mapping of the end-effectors. The first solution (already presented in [8]) can be used when the character is animated by forward kinematics and the number of bones of the virtual character does not exceed the number of bones tracked by the mocap system. On the other hand, end-effectors are considered when the character is animated by Inverse Kinematics (IK); an end-effector is a bone on which the movement is applied, then the position of all the other bones

*Corresponding author Andrea Sanna. Email: andrea.sanna@polito.it

of the kinematics chain is updated automatically by an IK solver in order to place the end-effector in the selected position. Users can choose whether control a character by forward or inverse kinematics and then accept/reject/modify the association that has been automatically found.

The current implementation uses armatures defined in Blender [9] and the Microsoft Kinect sensor [10] as motion capture device. Nonetheless, the proposed methodology is general and it could be easily extended to any other tracking system. A mapping between the human skeleton tracked by the Kinect and the Blender armature allows the devised methods to translate the local movements of human skeleton bones into translations and rotations of the associated armature bones. In this way, a real-time markerless motion capture system for digital puppetry with generic characters is actually implemented.

The paper is organized as follows: Section 2 briefly reviews other approaches that have been designed to map a human skeleton onto a virtual character armature. Section 3 presents the first solution (which implements a direct mapping in forward kinematics) and shows how the similarity scores in the mapping matrix are computed. Section 4 proposes some applications of the matching algorithm to non-anthropometric virtual characters. On the other hand, Sections 5 and 6 present the solution based on the inverse kinematics and some examples, respectively. User feedbacks are presented in Section 7 where an assessment of both solutions is presented. In order to help reader to better understand automatic mapping algorithms, some mathematical steps, applied to a reference character, are also presented in the Appendix.

2. Background

Several works are known in the literature that propose solutions to use a performer as a sort of controller for animating virtual characters. Anthropometric limbs are interactively controlled by inverse kinematics in [11]; in this case, only sub-parts of the whole skeleton can be animated independently, thus this approach might be unable to cope with constraints that require the whole body to be animated. All body parts are considered in [12], where the main goal is to map the movements made by a performer onto an animated character by only considering constraints on the end-effectors, thus animating only by the inverse kinematics. An extension targeted to the control of non-anthropometric characters is proposed in [13]; an intermediate skeleton with less degrees of freedom is used and the remaining degrees of freedom are computed analytically in [13].

The above works basically relies upon specific representations of motion (e.g., through simplified skeletons). A comparable approach is also used in [14], where a data structure especially dedicated to motion adaptation is proposed. Moreover, in all the works considered, the focus is mainly on human-like animation.

Generic-shape characters are considered in a recent and impressive work [15]. The solution proposed in [15] allows animators to directly set a mapping between their own skeletons and the mesh of the object to be controlled. The character mesh is segmented from the background; then, the body of the animator is *embedded* to position vacated by the object. By a vocal command, limbs of the animator are attached to the parts of the mesh the body overlaps. These attachments serve as constraints for the deformation model that is inspired by the Embedded Deformation method proposed in [16].

Another solution proposing the animator as a direct controller of a virtual character has been presented in [17]. As [18], the solution proposed in [17] combines different feature mapping functions together with an intelligent blend scheme in order to generate highly realistic output motions. In particular, some features of the animator body are directly used by a direct mapping to animate the character, whereas another stage of motion classification allows the system to identify frames of pre-set clips of the virtual character to be blended with the direct mapping motion.

The approach proposed in [19] is tailored for non-humanoid characters. The motion of the animator is gathered and analyzed. The basic idea is to compare, by a statistical mapping function, a small set of corresponding key poses in order to define a mapping between the motion of the animator and the virtual character. Therefore, this approach is not completely automatic as a set of poses for the character has to be defined; moreover, it is not aimed to obtain a real-time puppetry animation.

This paper extends and improves the work presented in [8]. In [8], only an automatic mapping for armature controlled by forward kinematics was presented: this part is reviewed in Sections 3 and 4; on the other hand, the solution presented in [8] is not able to cope with armatures composed by a number of bones larger than the animator's one. The present manuscript tackles this issue and presents a second automatic mapping algorithm, which is based on the animation of end-effectors by inverse kinematics. Moreover, both automatic mapping techniques are evaluated in this paper; user tests have been performed in order to gather subjective and objective user feedbacks.

3. The first solution: a direct skeleton mapping

The first technique to map the skeleton of the animator onto an armature of a virtual character is shown in Figure 1. Basically, different blocks contribute to

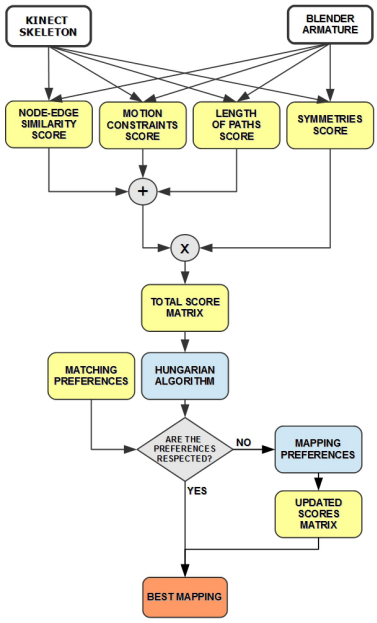

Figure 1. The first mapping algorithm described by a flow-chart.

compute a *score matrix*; coefficients of this matrix are used to automatically map the human skeleton onto the Blender armature. A set of parameters/constraints are considered:

- topology details (e.g., node-edge similarity);
- motion constraints (related both to the animator and the character to be animated);
- length of kinematic chains;
- symmetries of the armature.

Moreover, user preferences can be exploited to *force* some mappings, thus possibly overriding the above mentioned criteria (e.g., arms might be favorite with respect legs, and so on).

3.1. Graph representation

The first step of the mapping algorithm considers both the human and the virtual character skeletons. The Microsoft Kinect used as mocap device in this work provides tracking data containing information related to the center of mass and the position of each of the twenty joints of the captured skeleton. Moreover, a status information is associated with each joint: it indicates whether the joint position is being tracked or inferred (which happens when the Microsoft Kinect cannot see this point and tries to accurately *guess* it on the basis of information from previous frames and neighboring joints). Joints tracked for the skeleton are split into three main sections:

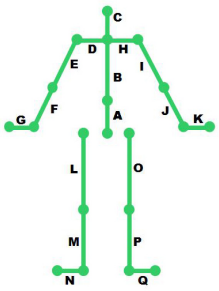

Figure 2. The skeleton extracted by the Kinect application: 20 joints are tracked.

1. the central area, containing the head, the neck, the spine and the hip center;
2. the arms, containing for each arm the shoulder, the elbow, the wrist and the hand;
3. the legs, containing for each leg the hip, the knee, the ankle and the foot.

The skeleton extracted by the Kinect application is shown in Figure 2. The Kinect application is connected by a socket to the Blender Game Engine (BGE): a Python script receives joint data and translates them into valid transformations for controlling the armature of the virtual character. A bone-to-bone mapping is performed; in other words, the position/rotation of a bone tracked by the Kinect is used to set up the position/rotation of a well defined bone of the character armature (for more details about the software architecture see [5]).

Blender armature is also analyzed: the starting point is the root bone. This step describes the armature as a sort of graph where bones are associated to nodes/vertices and relations between bones are considered as arcs/edges. The graph can be represented by an adjacency matrix [20]; given a graph G_A, $G_A = G(V_A, E_A)$ where V_A are the vertices and E_A are the edges, if the cardinality of V_A is n_a, then the adjacency matrix A of this graph is a $n_a \times n_a$ matrix in which entry $[A]_{ij}$ is equal to 1 if and only if $(i, j) \in E_A$, 0 otherwise. The adjacency matrix of an undirected graph will always be symmetric. Another useful graph representation is obtained by means of pair of matrices called edge-source matrix A_s and edge-terminus matrix A_t. This representation allows self-loops to be considered in the graph [20]. Let $s_A(i)$ denote the source of edge i, and let $t_A(i)$ denote the terminus of edge i. Then A_s and A_t can be defined as follows:

$$[A_s]_{ij} = \begin{cases} 1 & \text{if } s_A(j) = i \\ 0 & \text{else} \end{cases}$$

$$[A_t]_{ij} = \begin{cases} 1 & \text{if } t_A(j) = i \\ 0 & \text{else} \end{cases}$$

The graph representation given by A_s and A_t has the following properties:

- the adjacency matrix A is equal to $A_s A_t^T$;
- $A_s A_s^T$ is equal to a diagonal matrix D_{A_s} with the out-degree (i.e., the number of outgoing edges) of node i in the i-th diagonal position;
- $A_t A_t^T$ is equal to a diagonal matrix D_{A_t} with the in-degree (i.e., the number of incoming edges) of each node in the corresponding diagonal entry.

3.2. Node-edge similarity scores

Describing armatures as graphs allows the automatic procedure to assign a similarity score between pairs of nodes belonging to two different graphs (e.g, armatures); in this way, a match between two bone configurations (i.e., the skeleton extracted by the Kinect application and the Blender armature) is possible from a mathematical point of view. The applied similarity criterium considers both node and edge similarity as proposed in [20].

Given two graphs G_A and G_B, a simple way to give a definition of an edge score is: an edge in G_B is like an edge in G_A if their source and terminal nodes are similar, respectively. A is the adjacency matrix of G_A and B the adjacency matrix of G_B. D_{A_s}, D_{A_t} and D_{B_s}, D_{B_t} are the diagonal matrices containing the out-degree and the in-degree values of every node in G_A and G_B, respectively.

By iterating an certain number of times (usually, a satisfactory convergence is obtained with 11 iterations [20]) equation (1), a $n \times m$ scores matrix X is obtained, where n is the total number of bones in the Blender armature and m is the number of bones in the Kinect skeleton.

$$x_k \longleftarrow (A \otimes B + A^T \otimes B^T + D_{A_s} \otimes D_{B_s} + D_{A_t} \otimes D_{B_t}) x_{k-1}. \quad (1)$$

The symbol $\otimes$ represents the Kronecker's matrix product, k the k-th iteration and x_k a column vector obtained by concatenating the columns of the scores matrix X. The iteration method presented well recognizes nodes that are very similar and provides good results if one of the two graphs is a subgraph of the other one.

A topological similarity is not sufficient to denote how two skeletons/armatures are similar, thus other parameters have to be taken into account.

3.3. Motion constraints scores

Bone motion constraints are another important parameter to be considered in the score matrix generation: the mapping technique should select pairs of bones that exhibit similar degrees of freedom. For example, it is easier to control a virtual segment exhibiting a high degree of freedom by means of a hand rather than with the spine. Constraint similarity is taken into account by assigning a *penalty* proportional to the different degree of freedom of each pair of bones; moreover, if two bones have completely different movement types, their score is set in the matrix X in such a way that they cannot be matched. Motion constraints scores are added to node-edge similarity scores (see Figure 1).

3.4. Length of paths scores

Length of kinematics chains are another parameter to be taken into account: sub-chains of similar length of the human skeleton and of the virtual armature character should have a greater mapping score than not similar long paths. A sort of *bonus* is added (see Figure 1) to the score of all those bone pairs that share the same position in a chain starting from the bone root. The bonus is added to enhance the probability of mapping a bone of the virtual character, place in a well given position of the chain, onto a bone placed at the same (or similar) position in the human skeleton. Updating the matrix of scores according to this criterion brings another advantage: the probability of mapping sequential bones in a chain by respecting the natural rank order is implicitly increased.

3.5. Symmetries scores

Armatures can exhibit some kind of symmetry; for instance, parts of the body of animals (e.g., legs) can be subdivided into small groups. On the other hand, a main symmetry in the human skeleton can be obtained by splitting left parts from right parts. This kind of symmetry can be present also in Blender armatures where bones are labeled by the suffix ".L" or ".R" to denote left and right parts with respect to the spine, respectively. By searching for the bones containing ".L" or ".R" in their names, it is possible to force a mapping of these bones onto the corresponding parts of the human body. The skeleton tracked by the Kinect application is already split in five groups: body, left arm, right arm, left leg and right leg. Depending on the bone type in Blender, the search space is reduced to a few groups that compose the Kinect skeleton. In this way, the left part of an armature will be always mapped onto the left arm or the left leg of the animator and the right part of the armature onto the right arm or leg. Scores can assume the following values: 0 (to avoid mapping a bone in the left part onto a bone in the right part and vice versa), 1 (to leave unchanged the score of bones not related to symmetries, like the spine bones) and 2 (to force left part bones to be mapped onto left

part bones and vice versa). Symmetries scores multiply the previously obtained scores recorded in the matrix.

3.6. Evaluation component: Hungarian algorithm

In order to identify the best matching between two graphs, the node pairs with the highest scores in the matrix, according to a certain evaluation criterion, have to be found. This problem is known as the maximum weight bipartite graph matching problem. A common algorithm for identifying such a maximum weight is the Hungarian algorithm, described in [21]. In the proposed technique, the Hungarian algorithm is applied to the scores matrix to obtain a matching between the nodes of the graph representing the Blender armature and the nodes of the graph representing the Kinect skeleton, which maximizes the sum of squared matched scores. Thus, for each bone in the Blender armature, the Kinect bone it will be mapped onto is obtained.

3.7. Preferences

Preliminary tests that considered all the above mentioned criteria in order to generate the score matrix outlined how the resulting mapping can propose users to use some parts of the Kinect skeleton (e.g., the shoulders) which are very difficult to control for animating characters. This issue has been tackled by adding user's preferences to the overall mapping strategy.

If some armature bones are mapped onto the arms, forearms or hands are preferred to the shoulders. Of course, any other preferences could be coded in the matrix. Furthermore, the user can choose to avoid the mapping of the possible unused parts belonging to a mapped arm. The same approach is applied for the legs. After the application of the preferences, the Hungarian algorithm is used again to assign all the other unused bones.

4. Experimental results: forward kinematics

Three non-anthropometric characters (a Pixar-like lamp, a fish and a dog) have been used to the the first mapping technique. Blender armatures have been mapped onto the human skeleton extracted by the Kinect application and results are shown in Figures 3, 4 and 5, respectively. The kinematics chain of the lamp is mapped part onto two bones of the left arm (bones 1 and 2) and part onto three bones of the right arm (bones 3, 4 and 5). Arms are selected here, since their preferences score is the highest one (see Section 3.7). A completely different mapping has been obtained for the fish. In this case, it can be noticed how the symmetry (see Section 3.5) related to the fins is taken into account for mapping bones 4 and 5 on the right arm and bones 6 and 7 on the left arm. Since bones

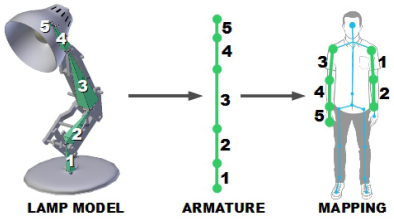

Figure 3. Mapping of the human skeleton onto the armature used to animate a Pixar-like lamp (a video is available at http://youtu.be/iEuT7pzdMqw).

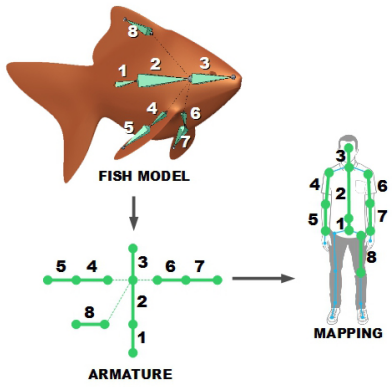

Figure 4. Mapping of the human skeleton onto the armature used to animate a fish (a video is available at http://youtu.be/v1mMtAgFhzQ).

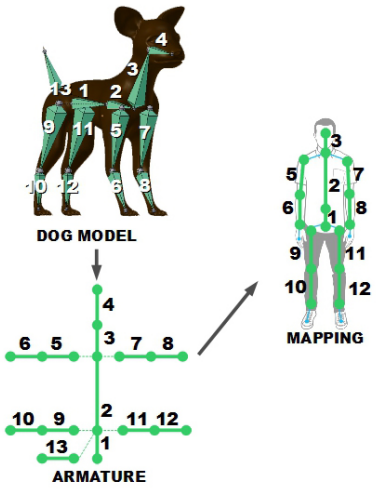

Figure 5. Mapping of the human skeleton onto the armature used to animate a dog (a video is available at http://youtu.be/cQIKeOAwN24).

1, 2 and 3 are topologically similar to the spine of the human skeleton, they are mapped onto it. The computation of mapping scores is detailed, for this character, in the Appendix. The dog armature is the most complex one among the three considered. Again, because of topological similarity, the spine of the dog is

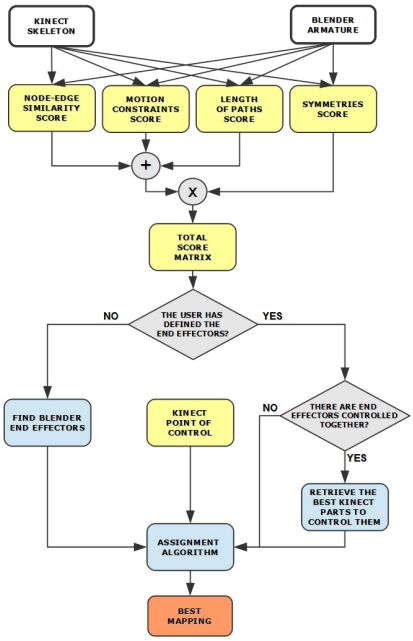

Figure 6. Flow-chart of the second mapping algorithm.

mapped onto the spine of the human skeleton (bones 1, 2 and 3). The symmetry is used to map the left parts of the skeleton onto the left parts of the armature and vice versa. Moreover, still because of topological similarity, the arms are mapped onto the front paws (the tail makes the similarity score for the back paws, with respect the skeleton arms, lower than the front paws). In this case, the user can choose to obtain a mapping also for the head and the tail or, as shown in the Figure 5, to leave these two bones unmapped as the five main kinematics chains of the skeleton have been already (even if not completely) used.

5. The second solution: a direct mapping of the end-effectors

The solution shown in Section 3 presents a main drawback: it is impossible to cope with armatures including a number of bones larger than the one provided by the tracking device. Unmapped bones cannot be animated (e.g., the dog's head of the previous example: the bone N. 4 is unmapped and the dog's head cannot be animated). In this case, forward kinematics cannot be used for the mapping, unless the target armature is reduced by collapsing pairs of bones. In order to control a chain of bones by the inverse kinematics it is necessary to consider the end-effectors (EEs) and update, by a IK solver, positions and orientations of all the other bones of the chain. Multiple solutions are generally possible and constraints are used to "help" the solver in order to find the best solution. Therefore, before to start the mapping step, all possible end effectors for an armature model have to be found. The armature is explored by starting from the root bone; following all possible chains it is possible to obtain all the paths forming the whole armature. Each of these paths should be animated in order to provide a realistic motion, thus one end effector has to be identified for all of the paths. The best choice is to take the terminal bone of a chain as the end effector, as moving the terminal all the other bones of the chain will be animated.

When the end-effectors have been identified, a suitable mapping for a human animator has to be found. It has been assumed that the skeleton provided by the Microsoft Kinect could have a maximum of five control points (CPs) able to drive kinematics chains: the head, the two hands and the two feet. At the next step, the best association between end-effectors of the character and control points of the animator has to be determined. Through the use of similarity scores calculated by the graph similarity theory, it is possible to evaluate each combination EE-CP, thus identifying the highest score. The similarity score is not only the similarity between two bones, but it also depends on all the other similarity scores of the bones connected to them (as illustrated in the coupled node-edge scoring of the graphs similarity theory). It results to be very suitable for character animation, as the algorithm aims to maximize the similarity of the human part (corresponding to a control point) with a chain of bones (corresponding to an end-effector). When the best scored combination is found, the corresponding control point is labeled as "no longer available" and the algorithm iteratively considers the next best pair EE-CP. When the control points are terminated, two events can occur: 1. there are not other EEs to be mapped, that is, the mapping is complete. 2. other EEs are not mapped and the 5 control points have been already assigned. The event 2 can be managed only by grouping two (or more) EEs under the influence/control of a *master* bone. In this case, the algorithm has to be able to retrieve bones controlling multiple EEs; this can happen when an animator wants to force the animation of multiple model parts with a single EE (see for instance Figure 8 where both wings are controlled by a master bone). In this case, the score of a master bone is determined by averaging scores of the EEs it controls. The complete flow chart of the second mapping algorithm is shown in Figure 6.

6. Experimental results: inverse kinematics

This section shows the application of the second mapping algorithm to three non-anthropomorphic characters (a seagull, a snake and an ostrich). Moreover, for the seagull model, both the application without master bones and with a master bone controlling both

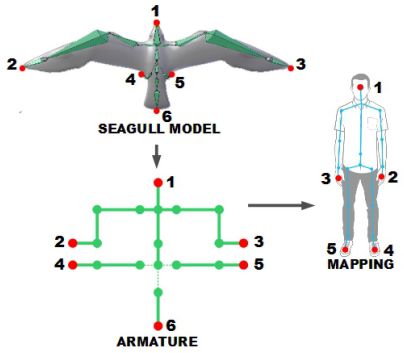

Figure 7. Mapping of the human skeleton onto the armature used to animate a seagull model(a video is available at http://youtu.be/KL55NuRKQAk).

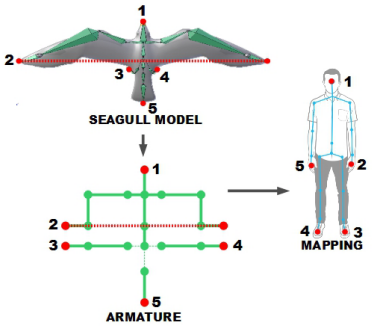

Figure 8. Mapping of the human skeleton onto the armature used to animate a seagull model with master bone controlling two end-effectors (a video is available at http://youtu.be/qaqP3za2dJM).

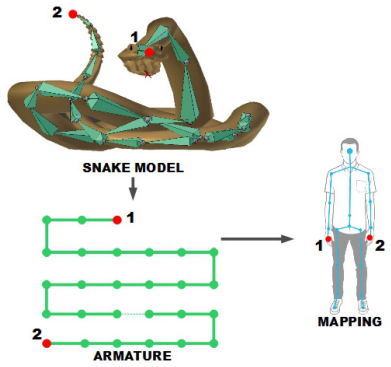

Figure 9. Mapping of the human skeleton onto the armature used to animate a snake model (a video is available at http://youtu.be/1ROz0mNgt8g).

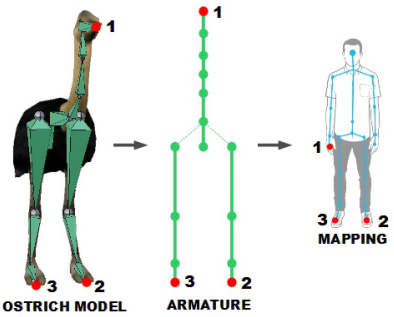

Figure 10. Mapping of the human skeleton onto the armature used to animate an ostrich model (a video is available at http://youtu.be/sM_ZiHVHFtY).

the EEs of the wings have been tested. Results are shown in Figure 7, 8, 9 and 10, respectively. In the seagull armature (see Figure 7) there are six different chains of bones and six corresponding EEs. This armature is very similar to the human one, apart the presence of an extra segment (the tail). The proposed system maps the head (end-effector 1) onto the human head, the two wings (EE 2 and 3) onto the two arms, the paws (EEs 4 and 5) onto the legs and the tail (EE 6) is unmapped. A different mapping is obtained considering the seagull model where a master bone to control both wings has been inserted (Figure 8). In this case the five end effectors are mapped onto each animator control point. The head of the seagull (EE 1) is assigned to the animator head, the master bone controlling the wings is mapped onto one hand, the two paws (EEs 3,4) are matched with the two legs and the tail (EE 5) is animated with the other hand. A completely different mapping is obtained for the snake model. The armature is formed by two chains of bones: one for the body and the head and one for the tail. The resulting mapping is as follows: the snake head (EE 1) is associated with the right hand and the snake tail (EE 2) is mapped onto the left hand. The last character is an ostrich model. Its armature configuration is not complex and it includes two chains of bones for the legs and a long chain composed by many bones for the body, the neck and the head. The two paws (EEs 2 and 3) are mapped onto the two animator's legs, whereas the body-neck-head part (EE 3) is controlled by the right hand of the animator.

7. User feedbacks

In order to validate proposed automatic mapping algorithms, 10 students of the Computer Animation course of the Master of Science degree in Computer Science at Politecnico di Torino have performed some tests. The main goal was to compare the proposed mapping strategies with manually selected mappings, thus focusing the analysis on how good and useful are the proposed hints and how automatic mappings can help the user to efficiently animate virtual characters.

Two different models have been chosen for the tests. The lamp model has been selected to be animated by forward kinematics, whereas the ostrich model to be animated by inverse kinematics. The two models have been chosen for their particular armature

configurations and consequently for the large number of mappings that could be associated to each model; this allows students to find the mapping that better fits their preferences. Students had to set their own mappings between the Kinect skeleton and the armature of virtual characters, then they were asked to animate models by using the chosen mapping and subsequently to retry the animation by using the automatic mapping. Students had to animate the lamp and the ostrich reproducing some very simple motions presented by two reference videos.

After the animation step, students evaluated some characteristics of mappings such as: intuitiveness, usability, difficulty to be retrieved and so on. Moreover, the time necessary to manually define a mapping has been also measured. Tests have shown as an average time of 48s has been measured for the lamp and 33s for the ostrich. Taking into account that these models are composed by a limited number of bones, the initial step of mapping definition might be much more time consuming for complex model armatures. Furthermore, a careful mental effort and a good model analysis were required to students in order to reach a satisfactory mapping.

For the lamp model test, for which students were asked to find a direct mapping for each bone belonging to the armature, exactly half of students have preferred the automatic mapping (the technique proposed in Section 3), whereas the other half have preferred their personal mapping in terms of ease of use. Then, it was asked students to evaluate the intuitiveness of mappings providing a grade in a range between 1(lowest rate) to 7(highest rate). It was found that the students have better evaluated the automatic mapping (an average rate of 4,7) respect their mapping (an average rate of 4,2). For the ostrich test, to be animated by inverse kinematics, students were asked to find a mapping for the three possible end-effectors (head, left paw, right paw). In this case, only two students have preferred their mapping, whereas the others have preferred the automatic one (the technique presented in Section 5) or have given an equal preference. Also the intuitiveness of the suggested mapping was better evaluated than the one chosen by the students (5,6 for the automatic mapping versus 5,3 for the manual mapping).

Another interesting factor to be analyzed is how many manual mappings were similar (or exactly the same as) the suggested one. Considering the two tests aggregated, about the 22% of students selected a mapping completely identical to the one automatically provided, whereas the 89% have identically defined the mapping of at least one bone (or end-effector). Generally, the 56% of the bones to be mapped have received the same point of control than the one automatically provided. In conclusion, it was studied the user satisfaction and the personal opinion about the automatic mapping. Approximately all the students have said that the automatic mapping could be useful for animating a model and about the 55% of them has said that would have defined a different mapping after the usage of the automatic one.

8. Conclusion and future works

This paper presents two automatic procedures to map the human skeleton captured by a tracking system (the Microsoft Kinect, in the proposed implementation) onto the armature of a virtual character to be animated. Proposed solutions are able to efficiently tackle issues related to the control of non-anthropometric characters by taking into account topological similarity scores as well as other parameters such as motion constraints, kinematics chains length, user preferences and so forth. Both skilled animators and, above all, an audience with little or no experience in computer animation could take advantage of the devised approach. The step of mapping the animator's skeleton onto the armature of the character to be animated is efficiently automated, thus reducing time losses and frustrating attempts.

User feedbacks have been also gathered and they show how proposed solutions well fit user expectations, thus they can be considered as a valuable alternative to manual mappings, which require both time to be correctly defined and skills of an expert animator.

Several other open problems need to be further investigated. At the moment, all approaches known in the literature try to migrate the human motion toward a corresponding/similar motion of the virtual character; on the other hand, other forms of mapping could be considered in order to move toward a context awareness animation.

Appendix A. Mapping process for the fish character

This Appendix provides the details of the mapping process for the fish character. Figure A.1 shows the two matrices produced by the node-edge similarity and the motion constraints blocks, respectively. Matrices have $n = 7$ rows (corresponding to the number of bones in the Blender armature for the fish) and $m = 17$ columns (corresponding to the number of bones in the human skeleton). A value in position i, j of the node-edge similarity scores matrix represents the similarity between bone i in the Blender armature and bone j in the Kinect skeleton. The motion constraints score matrix penalizes bones by a value that is proportional to the difference between their degrees of freedom; for instance, bone 2 in the fish armature cannot have the same type of rotational movement of bone 1 in the Kinect skeleton. Hence, the mapping of this pair of bones is penalized by a factor -0.067. The left matrix of Figure A.2 represents the length of paths scores, which

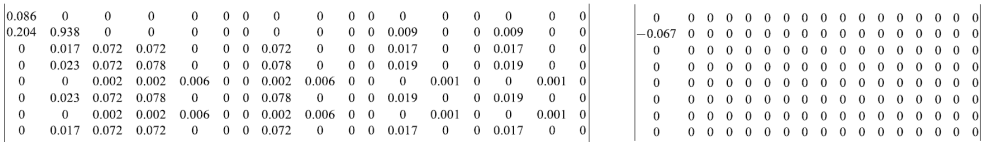

Figure A.1. The left matrix shows the node-edge similarity scores, whereas the right matrix holds the motion constraints scores.

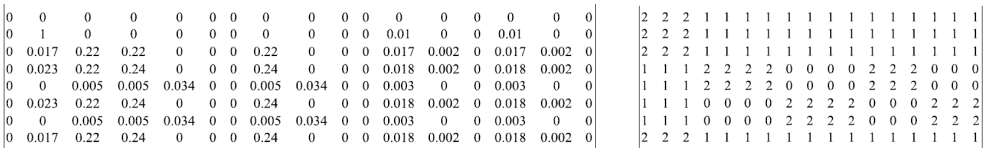

Figure A.2. The left matrix contains the symmetries scores, whereas the right matrix hosts the length of paths scores.

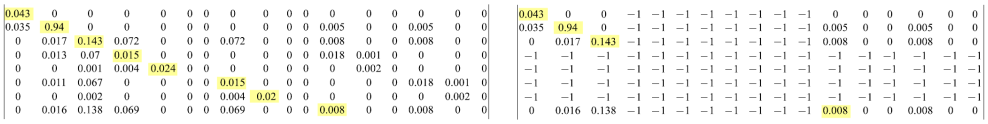

Figure A.3. The resulting scores matrix (left) and the updated scores matrix (right) after the application of preferences.

add a bonus to bone pairs that share the same position in a chain. The three matrices above are added together and then multiplied by the symmetry scores matrix (see the right matrix of Figure A.2). The symmetries scores matrix avoids (values 0) the matching of bones placed in armature parts with a different symmetry and amplifies the values of bones in the correct parts (values 2). The highlighted values in the final scores matrix are those selected by the Hungarian algorithm to produce the mapping (see the left matrix of Figure A.3). In the last step, bones are matched by also considering preferences. The remaining bones are mapped by applying the Hungarian Algorithm to an updated matrix. In fact, after the evaluation of preferences, the final scores matrix is updated by setting positions corresponding to mapped bones to a negative value. Highlighted values are those selected at the second application of the Hungarian Algorithm. The right matrix of Figure A.3 shows the updated scores matrix.

References

[1] Burtnyk, N. and Wein, M. (1971) *Computer generated key frame animation* Journal of the Society of Motion Picture and Television Engineers vol. 8, no. 3, pp. 149-153.

[2] Burtnyk, N. and Wein, M. (1976) *Interactive skeleton techniques for enhancing motion dynamics in key frame animation* Communication of the ACM vol. 19, no. 10, pp. 564-569.

[3] Menache, A. (2000) *Understanding motion capture for computer animation and video games* Morgan Kaufmann, New York.

[4] Gleicher, M. (1998) *Retargeting motion to new characters* In Proceedings of the ACM Siggraph'98, pp. 33-42.

[5] Sanna, A. and Lamberti, F. and Paravati, G. and Domingues Rocha, F. (2013) *A Kinect-based Interface to Animate Virtual Characters* International Journal of Multimodal User Interfaces. vol. 7, pp. 269ï£¡279, DOI 10.1007/s12193-012-0113-9.

[6] The Bloop project, http://dm.tzi.de/research/hci/bloop

[7] The Brekelmans Jasper web site, http://www.brekel.com

[8] Sanna, A. and Lamberti, F. and Paravati, G. and Carlevaris, G. and Montuschi, P. (2013) *Automatically mapping human skeletons onto virtual character armatures* Proceedings of the 5th International Conference on Intelligent Technologies for Interactive Entertainment, 2013, pp. 80-89.

[9] The Blender project, http://www.blender.org

[10] The Kinect web site, http://www.xbox.com/kinect/

[11] Tak, S. and Young Song, O. and Ko, H.S. (2002) *Spacetime sweeping: An interactive dynamic constraints solver* In Proceedings of the Computer Animation, IEEE Computer Society, pp. 261.

[12] Shin, H.J. and Lee, J. and Shin S.Y. and Gleicher M. (2001) *Computer puppetry: An importance-based approach* ACM Trans. Graph. 20, 2:67ï£¡94.

[13] Monzani, J.S. and Baerlocher, P. and Boulic, R. and Thalmann, D. (2000) *Using an intermediate skeleton and inverse kinematics for motion retargeting* Computer Graphics Forum 19, 3.

[14] Kulpa, R. and Multon, F. and Arnaldi, B. (2005) *Morphology-independent representation of motions for interactive human-like animation* In Computer Graphics Forum, vol. 24, pp. 343ï£¡352.

[15] Chen, J. and Izadi, S. and Fitzgibbon, A. (2012) *KinÊtre: animating the world with the human body*

ACM SIGGRAPH 2012 Talks, pp. 39:1-39:1, DOI 10.1145/2343045.2343098.

[16] Sumner, R.W. and Schmid, J. and Pauly, M. (2007) *Embedded deformation for shape manipulation* In Proceedings of the ACM Siggraph'07.

[17] Seol, Y. and O'Sullivan, C. and Lee, J. (2013) *Creature Features: Online Motion Puppetry for Non-human Characters* Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2013, pp. 213-221.

[18] Ishigaki, S. and White, T. and Zordan, V.B. and Liu, C.K. (2009) *Performance-based Control Interface for Character Animation* ACM Trans. on Graph. vol. 28, no. 3.

[19] Yamane, K. and Ariki, Y. and Hodgins, J. (2010) *Animating Non-humanoid Characters with Human Motion Data* Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 169-178.

[20] Zager, L. (2005) *Graph Similarity and Matching* Department of Electrical Engineering and Computer Science, Massachussets Institute of Technology.

[21] Andràs, F. (2004) *On Khun's Hungarian method - a tribute from Hungary. Egervàry research Group on Combinatorial Optimization* Technical report, TR-2004-14. http://www.cs.elte.hu/egres/tr/egres-04-14.pdf