# Automatically Mapping Human Skeletons onto Virtual Character Armatures

Andrea Sanna, Fabrizio Lamberti, Gianluca Paravati, Gilles Carlevaris, and Paolo Montuschi

Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino 10129, Italy
{andrea.sanna,fabrizio.lamberti,gianluca.paravati,
gilles.carlevaris,paolo.montuschi}@polito.it
http://www.polito.it

**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 a novel methodology for the automatic mapping of a human skeleton onto virtual character armatures. By extending the concept of graph similarity, joints and bones of the tracked human skeleton are mapped onto an arbitrary shaped armature. A prototype implementation has been developed by using the Microsoft Kinect as body tracking device. Preliminary results show that the proposed solution can already be used to animate truly different characters such as a Pixar-like lamp, a fish or a dog.

**Keywords:** virtual character animation, automatic armature mapping, motion capture, graph similarity.

## 1 Introduction

The animation of virtual characters is an exciting and challenging task. The mesh describing the shape of a character is linked to a set of bones usually named armature (rigging). The manipulation of the armature allows the user to animate the character. The traditional approach 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]. This approach has been often outperformed by motion capture solutions [3]. In this case, animators motions are tracked and can be recorded to a computer and then applied to the characters or directly used to make interactive animations.

Each method has its advantages and drawbacks. On the one side, keyframing can produce animations that would be difficult or impossible to act out. However, complex actions can be both very difficult and time consuming to reproduce. On the other side, motion capture can reproduce in a very accurate, fast and smooth way a variety of human (and animal) movements. Nonetheless, capture

M. Mancas et al. (Eds.): INTETAIN 2013, LNICST 124, pp. 80–89, 2013.

systems are, in general, very expensive and motion data could be hard to use for animating different kinds of characters (though a solution to partially cope with this latter limitation has been proposed in [4]).

The present paper addresses the problem to match the human skeleton of the animators (which act out the scene as if they were the characters to be animated) on a generic armature in an automatic and efficient way. In a number of previous works, such as [5] [6] and [7], the above association was implemented manually. Unfortunately, manual association can be a time consuming and difficult task, since only skilled animators are generally able to immediately identify the *best* match.

The proposed solution aims to find an efficient mapping of the human skeleton onto the virtual character armature, by exploiting an extended graph similarity criterion. In short, the matching algorithm analyzes and transforms a skeleton into a graph describing its constituting parts and the connections among them. Several parameters are taken into account, namely armature topology, general user preferences, symmetry and motion constraints. The above parameters are used to compute a similarity matrix, which is then exploited to associate each bone in the considered armature to the *most similar* bone in the human skeleton. Users can either accept the association that has been automatically found or modify the proposed bone mapping according to their own needs.

The current implementation uses armatures defined in Blender [8] and the Microsoft Kinect sensor [9] 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 method to translate the local movements of a human skeleton bone into translations and rotations of the related armature bones. In this way, a 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 proposed solution and, in particular, 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.

## 2 Background

One of the first attempts to interactively control anthropometric limbs by inverse kinematics is proposed in [10]. However, this method is only meant for controlling sub-parts of the skeleton independently. Hence, it is not always able to cope with constraints that require the whole body to be animated. The issue of controlling all the body parts is tackled in [11]. Here, the goal is to map the movements made by a performer onto an animated character by only considering constraints on the end-effectors. An extension targeted to the control of non-anthropometric characters is proposed in [12]. In this latter work, an intermediate skeleton with less degrees of freedom is used and the remaining degrees of freedom are computed analytically.

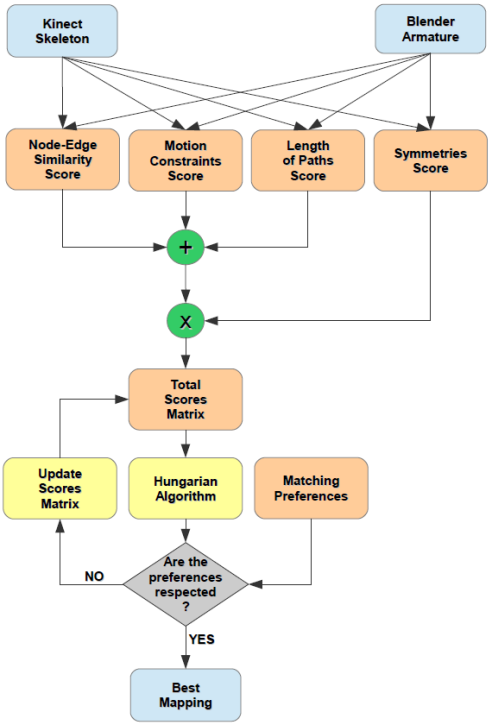


**Fig. 1.** Flow-chart of the mapping algorithm

The above works basically relies upon specific representations of motion (e.g., through simplified skeletons). A comparable approach is also used in [13], 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.

One of the most recent and impressive approaches known in the literature to animate generic-shape characters by the skeleton of the animator is reported

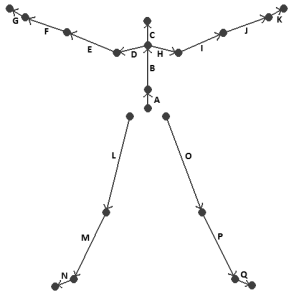


**Fig. 2.** Human skeleton extracted by the tracking application via the Microsoft Kinect

in [14]. The solution proposed allows the animator to directly manipulate the mapping of the skeleton onto 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 [15].

## 3 The Proposed Solution: Skeleton Mapping

Figure 1 shows the flow-chart of the mapping algorithm, which will be described in details in the following sections. From the chart, it can be easily noticed how different blocks actually contribute to determine the score matrix that is used to map the human skeleton onto the Blender armature. Such blocks consider armature topology details (e.g., node-edge similarity), motion constraints, length of kinematic chains and symmetries. Moreover, user preferences can be exploited to *force* some mappings, thus possibly overriding other criteria.

### 3.1 Graph Representation

The first step of the matching algorithm analyzes both the skeleton produced by the tracking device and the Blender armature of the virtual character to be animated.

As mentioned in the Introduction, the Microsoft Kinect is used as capture device in this work. Tracking data contain information about the center of mass and the position of each of the twenty joints of the captured skeleton, along with the status of each joint. Status information 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 based on information from previous frames and neighboring joints). Joints tracked for the skeleton are split into three main sections:

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.

Figure 2 shows the skeleton extracted by the Kinect application. A socket connection is created between the Kinect application and a Blender Python script controlling the execution of the program inside the Blender Game Engine (BGE). The script constantly receives user's skeleton data from the Kinect application, computes the necessary transformations required and applies them to the armature to be controlled (for more details about the software architecture see [5]).

Blender armature is explored starting from the root bone and a mathematical description is generated for it. In particular, bones are associated to graph nodes/vertices and relations between nodes are mapped to graph arcs/edges.

The graph can be represented by an adjacency matrix [16]; 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 [16]. 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

The approach presented above uses an iterative procedure to assign a similarity score between pairs of nodes belonging to two different graphs, thus allowing a match between the two bone configurations (i.e., the skeleton extracted by the Kinect application and the Blender armature). In the similar way as [16], the matching strategy is based on the coupled node-edge method. This method returns similarity scores considering not only the node similarity scores, but also edge similarity.

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 a certain number of times (usually, a satisfactory convergence is obtained with 11 iterations [16]) 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.

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

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.

Nonetheless, the above score is not sufficient to denote the similarity between the two armatures. Hence, other parameters (beyond the graph topologies) need to be taken into account in order to propose the user an efficient and accurate bone mapping.

### 3.3 Motion Constraints Scores

Another parameter to be considered in the mapping process is represented by the motion constraints related to each bone: two bones (one of the human skeleton and one of the Blender armature) exhibiting similar degrees of freedom should be preferred by the mapping technique. 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. These constraints are taken into account while updating the scores matrix $X$ by assigning a *penalty* to bones with degrees of freedom that are different. In particular, a value proportional to the existing difference is applied. If two bones have completely different movement types, their score is set 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

Another criteria considered to update the matching scores is represented by the length of kinematic chains: a long path of connected bones should be mapped onto a similarly long path. 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 enhances the probability that a bone in the Blender armature, placed in a certain position, will be mapped onto a bone placed in the same 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 may exhibit one or more symmetries. This behavior is easy to verify in models like animals, where some parts of the body (like the legs) are symmetric and they could be subdivided into small groups. 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. Usually, a Blender animator marks a bone in the left or right parts of the skeleton by adding the suffix ".L" or ".R" to the end of the bone's name, 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. To this purpose, the

Kinect skeleton is 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. This approach allows the proposed technique to always map the left part of a model onto the user's left arm or onto the user's left leg, and vice versa. 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 [17]. 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

After having performed several tests by exploiting all the previous scores, it was realized that, in many cases, the approach proposed correctly chose some parts of the Kinect skeleton like the shoulders, but these segments were indeed difficult to use for controlling and 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

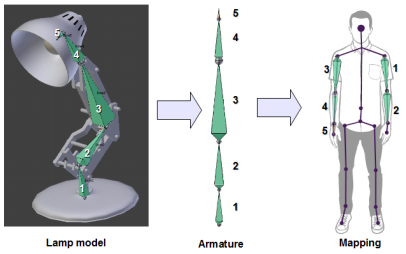


**Fig. 3.** Mapping of the human skeleton onto the armature used to animate a Pixar-like lamp (a video is available at `http://130.192.5.7/intetain2013/lamp.avi`)

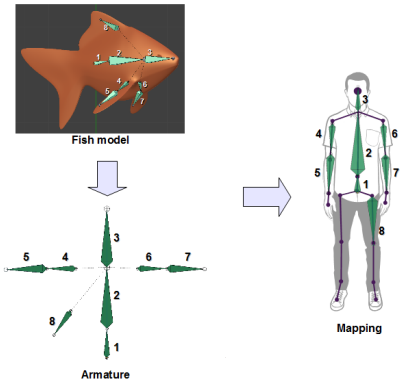


**Fig. 4.** Mapping of the human skeleton onto the armature used to animate a fish (a video is available at `http://130.192.5.7/intetain2013/fish.avi`)

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

This section presents the application of the devised mapping approach to three non-anthropometric characters (a Pixar-like lamp, a fish and a dog), whose Blender armatures have been mapped onto the human skeleton extracted by the Kinect application. 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 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 that is available at `http://130.192.5.7/intetain2013/appendix.pdf`. The dog armature is the most complex one among the three considered. Again, because of topological similarity, the spine of the dog is 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

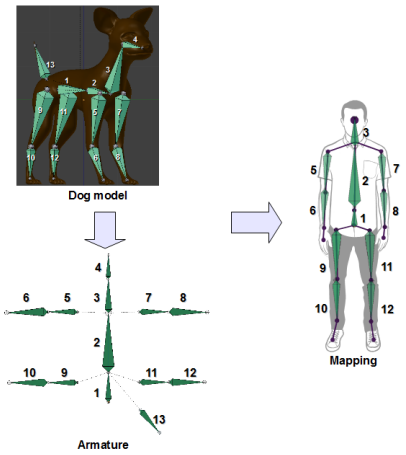


**Fig. 5.** Mapping of the human skeleton onto the armature used to animate a dog (a video is available at `http://130.192.5.7/intetain2013/dog.avi`)

the five main kinematics chains of the skeleton have been already (even if not completely) used.

## 5 Conclusion and Future Works

This paper presents a novel automatic procedure 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. The proposed solution is 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, kinematic chains length, user preferences and so forth. The mapping between the armatures is static, meaning that the proposed algorithm assigns direct relationships among bones based on the analysis of the topologies of the two armatures. 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.

Future works will be mainly aimed to gather a larger number of user feedbacks (currently, the system is being tested by a few students of the Computer Animation course of the Master of Science degree in Computer Science at Politecnico di Torino), in order to evaluate how the proposed mapping is close/far to/from the statistically best solution. The approach could/should be improved and extended in order to let it cope with armatures including a number of bones that is larger than the one provided by the tracking device. In this case, forward kinematics could not be used for the mapping, unless the target armature is reduced by collapsing pairs of bones.

Currently, the proposed method favours the static relations between bones by exploiting morphological features of the armature to be animated; indeed, less importance is assigned to the kinetic model (degree of freedom of the bones, range of movements). Thus, future works will also consider and study in depth the kinetic mapping between the armatures to improve the translation between human movements into those of the target model to be animated.

**Acknowledgments.** The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the manuscript.

## References

1. Burtnyk, N., Wein, M.: Computer generated key frame animation. Journal of the Society of Motion Picture and Television Engineers 8(3), 149–153 (1971)
2. Burtnyk, N., Wein, M.: Interactive skeleton techniques for enhancing motion dynamics in key frame animation. Communication of the ACM 19(10), 564–569 (1976)
3. Menache, A.: Understanding motion capture for computer animation and video games. Morgan Kaufmann, New York (2000)
4. Gleicher, M.: Retargeting motion to new characters. In: Proceedings of the ACM Siggraph 1998, pp. 33–42 (1998)
5. Sanna, A., Lamberti, F., Paravati, G., Domingues Rocha, F.: A Kinect-based Interface to Animate Virtual Characters. International Journal of Multimodal User Interfaces, 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. The Blender project, `http://www.blender.org`
9. The Kinect web site, `http://www.xbox.com/kinect/`
10. Tak, S., Young Song, O., Ko, H.S.: Spacetime sweeping: An interactive dynamic constraints solver. In: Proceedings of the Computer Animation, p. 261. IEEE Computer Society (2002)
11. Shin, H.J., Lee, J., Shin, S.Y., Gleicher, M.: Computer puppetry: An importance-based approach. ACM Trans. Graph. 20(2), 67–94 (2001)
12. Monzani, J.S., Baerlocher, P., Boulic, R., Thalmann, D.: Using an intermediate skeleton and inverse kinematics for motion retargeting. Computer Graphics Forum 19(3) (2000)
13. Kulpa, R., Multon, F., Arnaldi, B.: Morphology-independent representation of motions for interactive human-like animation. Computer Graphics Forum 24, 343–352 (2005)
14. Chen, J., Izadi, S., Fitzgibbon, A.: KinÊtre: Animating the world with the human body. In: ACM SIGGRAPH 2012 Talks, pp. 39–144 (2012), doi:10.1145/2343045.2343098
15. Sumner, R.W., Schmid, J., Pauly, M.: Embedded deformation for shape manipulation. In: Proceedings of the ACM Siggraph 2007 (2007)
16. Zager, L.: Graph Similarity and Matching. Department of Electrical Engineering and Computer Science, Massachussets Institute of Technology (2005)
17. Andràs, F.: On Khun's Hungarian method - a tribute from Hungary. Egervàry research Group on Combinatorial Optimization Technical report, TR-2004-14 (2004), `http://www.cs.elte.hu/egres/tr/egres-04-14.pdf`