An Architecture for Motion Capture Based Animation

FERNANDO WAGNER DA SILVA[1,2]
LUIZ VELHO[1]
PAULO ROMA CAVALCANTI[2]
JONAS GOMES[1]

[1]IMPA–Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina, 110,
22460 Rio de Janeiro, RJ, Brazil
{nando,lvelho,roma,jonas}@visgraf.impa.br

[2]LCG - Laboratório de Computação Gráfica, COPPE - Sistemas / UFRJ
21945-970, Rio de Janeiro, RJ, Brazil, Caixa Postal 68511

Abstract. This paper proposes an architecture for motion capture based animation systems, which works with several data formats and uses the building block paradigm for motion processing operations. Also, a user interface is proposed to perform an intuitive visualization of the animation main elements. A prototype system has been implemented, based on the presented concepts, and its operation is discussed.

keywords: motion capture, computer animation, motion control, animation systems, GUI paradigm.

1 Introduction

Recently, the crescent demand for powerful and intuitive animation systems has led to the development of new techniques, giving the animator more versatility to build complex animations.

The Motion Capture technique provides tools for real-time animation, with extremely realistic results. The widespread use of motion capture techniques is in part due to the low cost of modern capturing systems and also to the demand of different application areas such as special effects and home entertainment.

Although it has been studied since the beginning of the 80's [1] [2], the present utilization of motion capture is restricted to a direct mapping of animation parameters. In other words, the movements captured from live subjects are mapped directly on a virtual actor, and then the animation is displayed. In spite of its value, this use is very limited and do not exploit all the potential of the motion capture process.

Lately, however, tools for analysis, manipulation and reuse of captured data have been proposed. This makes motion libraries more valuable for a wide class of animators.

The application of these techniques are unlimited, from the development of computer games [4] to the production of computer-generated choreographies.

In this work[1], we propose an architecture for motion capture based animation systems. Our goal is to embody a set of tools for analysis, manipulation and reuse of motion captured data, overcoming some limitations inherent to the process.

The architecture was designed to serve as a test bed for new techniques, and also work as a robust converter between the most popular motion capture data formats.

Section 2 of this paper discusses some technological aspects of motion capture systems, pointing out some limitations of the process. In Section 3, we present a method for 3D Euler angle extraction, that is used to generate relative angles. Section 4 discusses a classification for motion operations. In Section 5, we present the architecture, together with the description of a user interface for the system. Section 6 presents a prototype implementation, developed under the methodology of our architecture. Finally, conclusions and future work are given in Section 7.

2 Background

There is a large diversity of motion capture hardware available nowadays, from simple mechanic devices to sophisticated optical systems.

Mechanical systems [5] are composed of potentiometers (or sliders) that measure the position or orientation of joints in an object. Its similarity with conventional stop-motion techniques, that are widely used in movie production, allows a natural migration of traditional animators, thus increasing the popularity of this technique. However, the realism of mechanically captured motions still depends, in great part, on the ability and patience of

[1] Additional info available at *http://www.visgraf.impa.br/Projects/mcapture*

the animator.

Systems based on magnetic technology are probably the most popular ones. Both positional and angular data of the joints of a real subject are captured, using a set of sensors that measure the magnetic field generated by a source. Their main advantage is the possibility of real-time animation of virtual characters, thus offering to the TV industry new possibilities in the field of virtual sets [6].

Some drawbacks of this technology are the sensitivity to metals in the capturing area - which introduces some noise into the final data; the high level of encumbrance - due to the great number of cables attached to the actor; and the sampling rate - too low for fast sport motions.

Optical systems are based on high contrast video imaging of retro-reflective markers, that are placed on objects whose motion is being recorded. This technique provides high sampling rates, but the recorded motion data must be post-processed using computer vision tracking techniques [7].

In the tracking process, the centroids of markers are matched in images from pairs of cameras, using a triangulation to compute the positional data of these markers in 3D space. This process introduces artifacts (offsets) into the final data. Some disadvantages of the optical process are the occlusion of one or more markers during the capturing session, the lack of angular data, and the sensitivity to background light and reflective objects.

Hybrid systems [8], that combine both magnetic and optical technologies are being developed, but are not yet commercially available. An interesting comparison between motion capture systems can be found in [9] and [10].

Finally, a problem that arises from the great diversity of motion capture hardware and technologies is the great number of motion data formats, which reduces significantly the compatibility of animation systems.

3 A Method for 3D Euler Angle Extraction

As cited before, one of the main disadvantages of optical systems is that they capture only positional data of joints. Angular data is extremely important because it can be mapped onto a "position-independent" skeleton hierarchy, giving more freedom to the animator.

In an animation system with motion capture facilities, each joint may have basically two types of angular data: absolute angles and relative angles. The first type is useful to execute a direct visualization of the captured motion, but positional data is still necessary for placing the joints in space, at each frame of the animation.

Relative angles are useful to create complex animations, allowing an easier modification of motion parameters. The positional information of the first frame is used to place the joints in the correct position in space (and also to estimate the length of the limbs), and then the animation is driven only by the joint angles. For example, a rotation applied to the shoulder joint will propagate to all joints of its sub-tree, i.e., the elbow and the wrist. The entire skeleton structure can be moved in space using the positional information of the hips, for example. Besides, techniques like keyframing and inverse kinematics can also be incorporated as new features, since they can be adapted to work in a motion capture environment.

To calculate these angles, we developed an algorithm based on geometry. Traversing the topological structure of the skeleton, our algorithm calculates both absolute and relative angles, for each joint of the structure.

Absolute angles are obtained by projecting the links over the coordinate planes lying on the proximal[2] joint. For each plane, the projected vector is then normalized and its angle with respect to the current axis is calculated (see Figure 1).

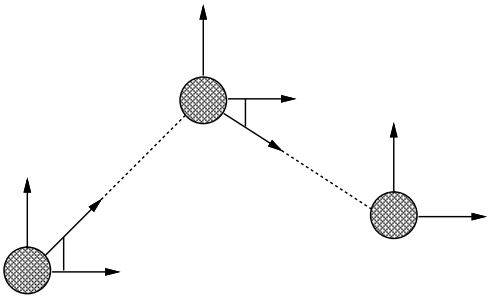

Figure 1: absolute angle calculation

Relative angles are also retrieved by projecting the linked structure over the coordinate planes (Figure 2). For each link we calculate a unit vector formed by the proximal and distal joints. To avoid ambiguity, we established that links are related in a clockwise manner. Using this rule, the projected linked structure in Figure 3 will have the appropriate angles, as shown in Figure 4.

Using the previously calculated vectors of the links, grouped in pairs, we can calculate the angle between them using $\alpha = \cos^{-1}(\overrightarrow{A_iA_{i-1}}_N \cdot \overrightarrow{A_iA_{i+1}}_N)$ (see Figure 4).

However, in some cases, the desired angle is not α, but $\overline{\alpha} = 360^\circ - \alpha$ (in Figure 3, angle number 3). This case is expected and occurs when the link angle is greater than 180°. To solve this problem, we use a simple and fast criteria to know whether a point is on the left side of an oriented segment or not. If a point c is on the left of the

[2]The terms *proximal* and *distal* will be used to describe positions as "near" and "distant" from the point of origin.

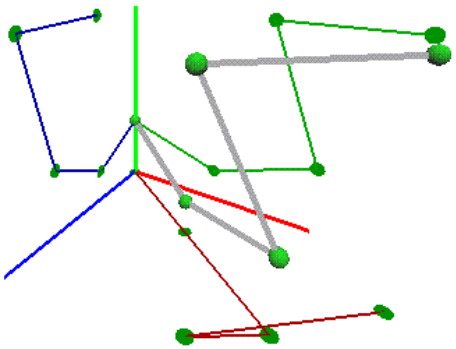

Figure 2: linked branch projection over the coordinate planes

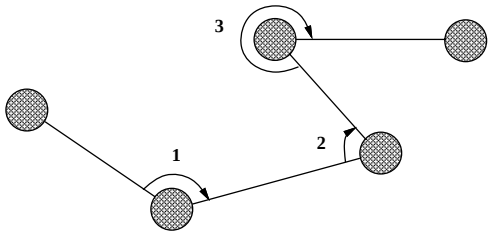

Figure 3: rule to avoid ambiguity in relative angle calculation

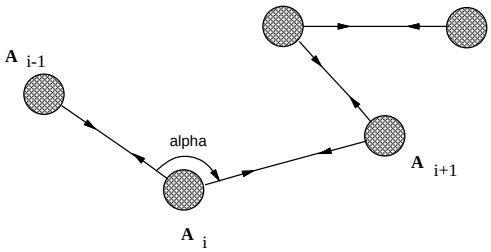

Figure 4: grouped vectors disposition and angle extraction

segment determined by (a, b), the triple (a, b, c) forms a counterclockwise circuit (Figure 5). Then an area based algorithm [11] returns the signed area determined by a, b and c, i.e., positive if counterclockwise, and negative if clockwise.

Using this "leftness" criteria we are able to calculate the correct angles of the linked structure: if the triple (A_{i-1}, A_i, A_{i+1}) has a negative area, then we take $\overline{\alpha} = 360^\circ - \alpha$, and the process continues, for each triple

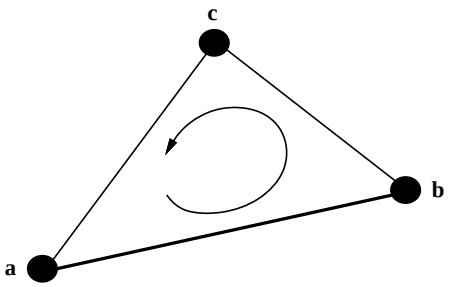

Figure 5: point c is on the left of the segment $\overline{ab}$ if (a, b, c) forms a counterclockwise circuit

until reaching the end-effector [3] of the linked branch.

Finally, the dimensions of actor's limbs are also estimated, measuring the 3D distance between the proximal and distal joints of each link.

This procedure is used as a pre-process for motion captured data from optical systems.

4 A Classification for Motion Operations

We can make an analogy between our system and a CSG modeling system, where basic primitives (in our case, the motions) are combined through operations like union, intersection and difference. In our paradigm, however, the (motion) operations are filtering, blending and concatenation. They can be classified in three types: unary, binary and n-ary.

Unary operations have one motion as operand, and are useful to modify special parameters of the motion (e.g., a filtering operation over selected joints) or even to modify the entire motion (e.g., warping the motion curves [17]).

Binary operations have two operands, and its main purpose is to join or group different motions, creating a longer one. Examples of binary operations are concatenation, cyclification [18] and transition. Note that although concatenation may be applied to several motions in sequence, it can be carried out locally as an operation between two motions.

Binary operations have many interesting applications, from computer fight games to virtual reality cooperative environments [19].

The last type of motion operations, n-ary, deals with two or more operands. Motions can be totally or partially blended, generating new interesting types of movements. In the case of partial blending of two motions, one can choose to apply a walk motion to the legs and hips of a skeleton, while letting the torso and arms execute a dance

[3] The term *end-effector* is often used in robotics, referring to the last joint of an articulated chain.

motion.

We can also group the existing motion operations in three types, according to the method of modification and / or combination they perform in their operands. They are:

- **Filtering**

Filtering operations can be applied to the joint curves of a motion to reduce noise, producing smoother results.

In [12], Williams use a multiresolution filtering method to decompose the motion into frequency bands. He showed that high-frequencies contain the details of the motion, whereas low-frequencies contain general, gross motion patterns. In a practical example a walk motion was processed, extracting a basic "walking" factor and a "qualitative" factor, like brisk.

It seems that most digital filtering techniques are suitable to use with captured motion data.

- **Concatenation**

Concatenation operations can be used to create longer animations. Smooth changes between different motions are achieved through interpolation of end of the first motion with the beginning of the second motion.

Direct concatenation can be used as well, but for non-cyclic[4] motions it will generate a discontinuity at the transition.

Transitions between motions are made interpolating the joint curves parameters over an interpolation interval. In [18], an approach using spacetime constraints and inverse kinematics was used, generating seamless and dynamically plausible transitions between motion segments.

- **Blending**

Blending operations are normally used to combine special characteristics of different motions. For example, two kinds of walk motion may be combined to produce a new one, blending the joint curves of both motions. Using this approach, it is possible to create a whole family of different motions, just varying the blend factor between the curves.

In blending operations, there must be special attention to motion synchronization and reparametrization. Synchronization between motions can be achieved using time-markers, which act as kinematic constraints, matching important events in both motions that will be combined and performing reparametrizations when needed. Without these tools, motion blending is useless.

Note that concatenation can be interpreted as a particular case of blending where little or no overlapping occurs.

[4] Perfect cyclic motions are almost impossible in captured motions, due to measurement errors and normal human variation in the capture process.

5 The Proposed Architecture

The motivation of the proposed architecture includes three main objectives:

- to provide a set of tools for motion manipulation and analysis.
- to allow the production of high-quality complex animations, using reusable motion libraries.
- to compensate technological limitations of motion capture hardware.

The framework is composed of basic modules: *input*, *processing* and *output*, each one responsible for a specific set of tasks (Figure 6). These modules are supported by a graphical user interface.

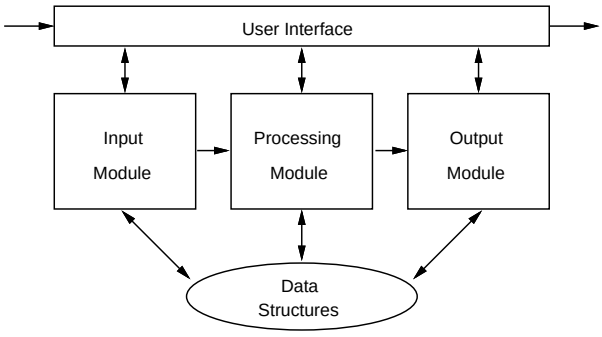

Figure 6: framework of the architecture

The data structures of the architecture represents two entities: an actor and motions.

The actor is treated as a skeleton. Its topology is represented by a graph formed by joints and links. Its geometry is represented by series of connected limbs. This description is adequate to be used in a motion capture animation system, since it reflects the structure of an articulated figure. For data acquisition, markers are attached at the joints of a live performer (the real actor).

At the programming level, the actor is represented using a modified version of Zeltzer's APJ (*Axis Position Joint*) structure [13], adapted to work with motion captured data.

Motions are best represented as curves in time. Normally, the captured data consists of marker's positional and / or angular variation, sampled by the capture hardware during the number of frames required to complete the actor's performance. This description is used for each degree of freedom (DOF) of the actor.

5.1 Input Module

This module focuses on problems concerning the interpretation and pre-processing of motion data.

The first step before loading a motion file is to specify the skeleton where the motion will be mapped. Because there are several file formats available, there must be a way to define different skeletons, each one appropriated to receive the data from a specific motion capture file format.

To maintain compatibility, we created skeleton definition files (*SDF*), that relates different file formats with the internal default skeleton definition. In other words, the default skeleton description provided in the architecture change its state, according to the incoming motion data format.

Sometimes, the input data will have missing information. This usually happens in optical systems, when the cameras cannot track one or more markers due to occlusion. In this case, linear interpolation can "fill the hole", but the best approach is to use prediction filters[5], with biomechanical constraints, to compute the joint behavior in the "hole" region.

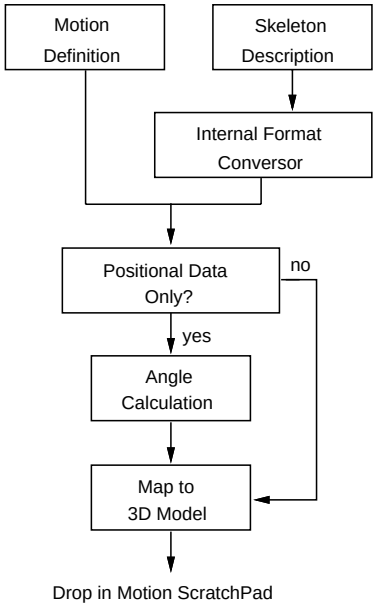

Figure 7: input module framework

5.2 Processing Module

This module comprises the set of tools for motion analysis, manipulation and reuse. These tools were described in Section 4.

The goal is to provide efficient ways to modify the original captured data, generating new classes of motions that inherit the aliveness and complexity typical to the capture process. Moreover, this module was designed to allow the integration of new motion processing techniques, therefore being extensible, as shown in Figure 8.

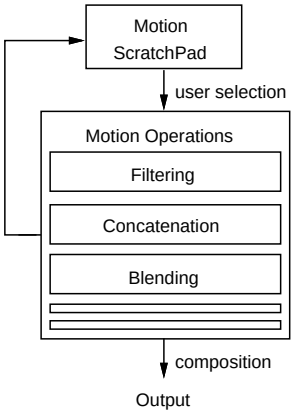

Figure 8: processing module framework

5.3 Output Module

As outlined before, motion captured data portability is important to improve the flexibility of animation production.

Most data formats can be converted without great effort, since they use the same technology. However, there are data formats with different markers arrangement or number. In this case, skeleton conversions are plausible, yet not always possible.

One alternative is to specify a universal data format that accepts most existing features of professional systems. In that way, motion libraries are easier to be maintained. They can be improved with new different motions, coming from various sources.

It is also necessary that the processed animation could be rendered frame by frame using the system, or even piped to professional rendering systems, like RenderMan [21] and PovRay [14].

5.4 Interface Module

One major problem on most animation systems is that they do not provide a concise description of basic entities, operations and concepts. The functionality of our architecture would be limited by a conventional user interface.

To complete the architecture description, we introduce an interface that represents the basic structure presented in the previous sub-sections. This interface is part of the prototype system, that will be described in Section 6.

We decided to adopt an interface paradigm used in post-production video workstations [15]. Motions are

[5] Personal communication, Lance Williams, Apple Computer, 1996.

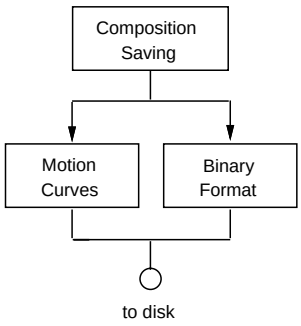

Figure 9: output module framework

represented by a horizontal bar, using a frame ruler associated with it. This visual description gives a precise spatial and temporal perception of the movement.

The user interface is composed of various graphical objects that are organized in panels and represent higher level operations. For a more detailed description of the interface paradigm used in the prototype system, please refer to [3].

6 The Prototype System

In this section we present a prototype system, implemented according to the proposed architecture.

This system works with motion captured data, using the processing module to create new motions, expanding the existing library.

Special attention was given to the system's GUI, which uses a dynamic approach, with several windows sharing information at different levels. The program is controlled by a loop that change the interface contents according to user interaction.

In this system, work with motions is straightforward: the user can select several motions from a existing library, and put them in a scratchpad. With a few commands, the user is able to apply different motion operations, with all necessary information available within his visual field.

Initially, the set of operations consists of filtering, concatenation, blending and transition. One of the main objectives of the system is to continuously integrate new motion tools, expanding the animator's possibilities and creating new motions with the existing tools. In that way, motion selection, cropping, cut and other higher level operations can be added to the initial set.

Figure 10 shows a snapshot of a typical system usage. Note the interface objects that represent the motion curves as a sampled signal (1), and the skeleton graph description (2).

Window 3 shows the Motion ScratchPad, a graphical object created to help user interaction with the system. Acting as a motion organizer, the ScratchPad provides useful information and a global perception of the motions placed on it.

Motion operations also have their own graphical objects. For example, window 4 shows a concatenation operation between several motions.

The playback of animations is executed in window 5. A control panel is integrated with it, providing controls for interactive playback as used in video recorders.

6.1 Implementation Issues

The architecture and prototype system presented in this work were implemented in the programming language C, using a SGI Indigo 2 graphic workstation as the base platform. We employed OpenGL for rendering and XForms [20] for the basic GUI generation. The advanced GUI objects were designed and implemented separately, and then added to the Forms library.

Due to OpenGL's rendering facilities and to the dynamic interface control used in the system, a real-time frame rate is achieved during the playback of animations (about 15 frames/sec in a SGI Indigo 2). The prototype system was also tested in the Linux and RISC 6000 platforms, also with good frame rates.

7 Conclusions and Future Work

This paper presented an architecture for motion capture based animation systems. Using the building block paradigm, motions can be combined or modified to create new motions and longer animations.

The architecture deals with some technological problems inherent to the capture process, providing various animation tools. Also, a GUI was proposed to offer a conceptually correct visualization of the animation elements.

A prototype system was built, based on the proposed architecture and interface, with promising results that encourage us to improve it.

7.1 Future Work

We plan to expand the flexibility or our system. Future work include:

- insertion of keyframing and inverse kinematics modules, improving system's flexibility.
- combine motion capture with procedural animation [24] and behavioral animation [25] [27]. In the first case, captured motions could act as a guide for procedural objects. In the second case, the behavioral

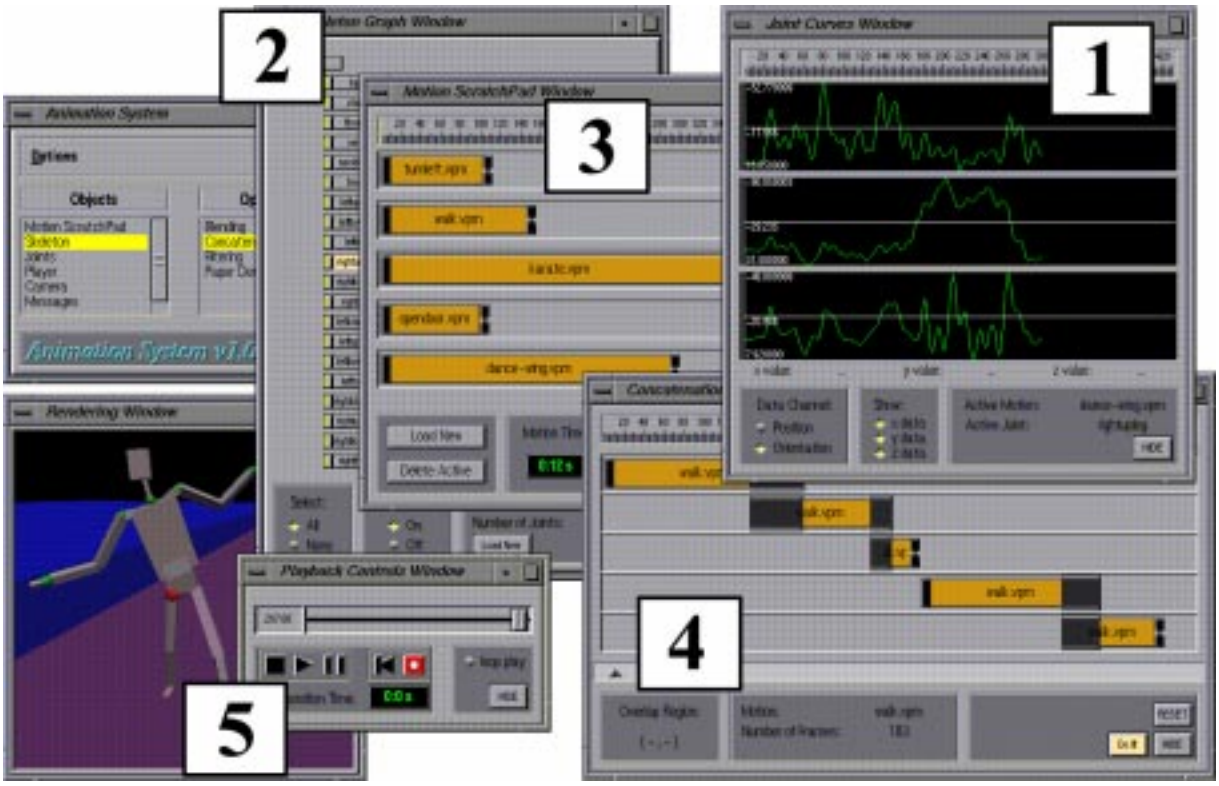

Figure 10: Snapshot of the prototype system.

functions could control the processing module, combining and modifying captured motions to improve the visual quality of the animation.

- combine motion capture with sound. In this case, the time-markers could be useful to synchronize the key moments in the motion with the temporal description of the sound.

- implementation of other advanced motion operations ([12], [18], [17]), comparing their results and extracting conclusions and suggestions for improvements and / or new techniques.

8 Acknowledgements

The authors would like to thank Viewpoint Datalabs, Inc. [22] and Biovision, Inc. [23] for access to motion capture data, and to the Brazilian Council for Scientific and Technological development (CNPq) for the financial support. This research has been developed in the laboratory of VISGRAF project at IMPA and at LCG/UFRJ, as part of the Master programme of the first author. This project is sponsored by CNPq, FAPERJ, FINEP and IBM Brasil. Also thanks are due to the reviewers for their valuable comments.

9 References

[1] GINSBERG, C. M., *Human Body Motion as Input to an Animated Graphical Display*, Master Thesis, Massachusetts Institute of Technology, May 1983.

[2] MAXWELL, D. R., *Graphical Marionette: A Modern day Pinocchio*, Master Thesis, Massachusetts Institute of Technology, June 1983.

[3] SILVA, F., VELHO, L., CAVALCANTI, P. AND GOMES, J., A New Interface Paradigm for Motion Capture Based Animation Systems. In *Proceedings of the 8th EUROGRAPHICS Workshop on Computer Animation and Simulation - CAS'97*.

[4] FX Fighter - Motion Capture vs. Keyframing Page. *http://www.im.gte.com/FXF/fxfwhat2.html*

[5] DYER, S., MARTIN, J., ZULAUF, J., Motion Capture White Paper. Technical Report. Silicon Graphics, December 12, 1995.

[6] *Virtual Sets*. Silicon Studio Features. December 1995.

[7] AZARBAYERJANI, A., WREN, C., Real-Time 3D Tracking of the Human Body. In *Proceedings of IMAGE'COM 96*, Bordeaux, France, May 1996.

[8] Character Motion Systems. In *Computer Graphics (SIGGRAPH'94)*, Course no. 9.

[9] MULDER, S., Human Movement Tracking Technology. Hand Centered Studies of Human Movement Project, Simon Fraser University. Technical Report 94-1, July 1994.

[10] SILVA, F, Movimento Capturado - Introdução á Tecnologia. Relatório Interno - Laboratório VISGRAF, IMPA, Abril de 1997.

[11] O'ROURKE, J., *Computational Geometry in C*. Cambridge University Press, 1994.

[12] WILLIAMS, L., BRUDELIN, A., Motion Signal Processing. In *Computer Graphics (SIGGRAPH'95 Proceedings)*(August 1995), pp. 97-104.

[13] ZELTZER, D. AND SIMS, F., A Figure Editor and Gait Controller for Task Level Animation. In *Computer Graphics (SIGGRAPH'88), Course Notes*, no. 4, 164-181.

[14] Persistence of Vision Ray-Tracer. *http://www.povray.org*

[15] Turbo Cube - User's Guide. IMIX Company.

[16] SILVA, F., CAVALCANTI, P., Animações em Tempo Real Utilizando Movimentos Capturados. In *Proceedings of SIBGRAPI'96, IX Brazilian Symposium of Computer Graphics and Image Processing*, pp. 333-334, 1996.

[17] WITKIN, A. AND POPOVIC, Z., Motion Warping. In *Computer Graphics (SIGGRAPH'95 Proceedings)*(August 1995), pp. 105-108.

[18] COHEN, M., ROSE, C., GUENTER, B. AND BODENHEIMER, B., Efficient Generation of Motion Transitions Using Spacetime Constraints. In *Computer Graphics (SIGGRAPH'96 Proceedings)*(August 1996), pp. 147-154.

[19] OZ Virtual. *http://www.oz.com/ov*

[20] Xforms Home Page, *http://bragg.phys.uwm.edu/xform*

[21] Pixar's Renderman, *http://www.pixar.com/renderman*

[22] Viewpoint Datalabs, *http://www.viewpoint.com*

[23] Biovision Inc., *http://www.biovision.com*

[24] PERLIN, K., Realtime Responsive Animation with Personality. In *IEEE Transactions on Visualization and Computer Graphics*, Vol 1, No.1, March 1995.

[25] TERZOPOULOS, D. ET AL., Artificial Fishes with Autonomous Locomotion, Perception, Behavior and Learning, in a Physical World. In *Proceedings of the Artificial Life IV Workshop*, MIT Press (1994).

[26] WITKIN, A. AND KASS, M., Spacetime Constraints. In *Computer Graphics (SIGGRAPH'88 Proceedings)*(August 1988), pp. 159-168.

[27] COSTA, M. AND FEIJÓ, B., An Architecture for Concurrent Reactive Agents in Real-Time Animation. In *Proceedings of SIBGRAPI'96, IX Brazilian Symposium of Computer Graphics and Image Processing*, pp. 281-288. 1996.

[28] AMAYA, K., BRUDELIN, A., CALVERT, T., Emotion from Motion. In *Proceedings of Computer Animation (CA'96)*, 1996.