HAL
open science

Système d'animation d'objets virtuels : De la modélisation à la normalisation MPEG-4

Marius Preda

To cite this version:

Marius Preda. Système d'animation d'objets virtuels : De la modélisation à la normalisation MPEG-4. Informatique [cs]. Université René Descartes - Paris V, 2002. Français. NNT : . tel-00273236

HAL Id: tel-00273236

https://theses.hal.science/tel-00273236v1

Submitted on 14 Apr 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITE RENE DESCARTES - PARIS V
Centre Universitaire des Saints-Pères
UFR DE MATHEMATIQUES ET INFORMATIQUE

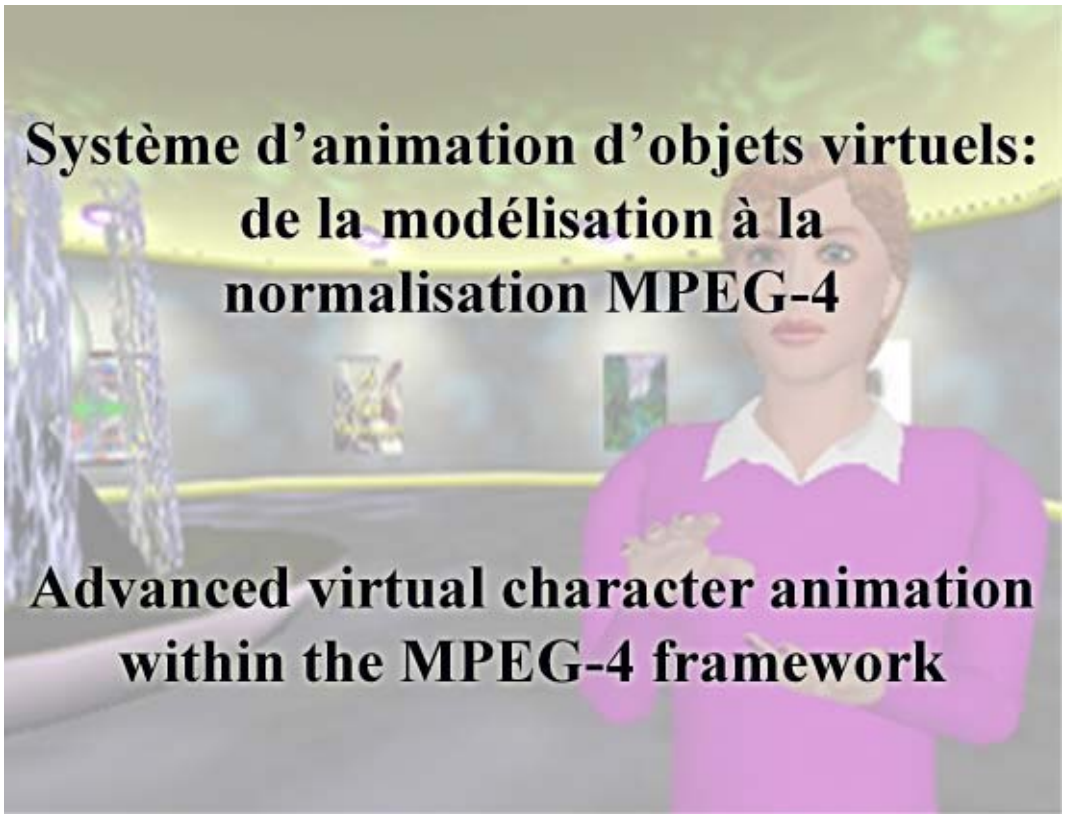
Thèse présentée en vue de l'obtention du grade de Docteur de l'Université RENE DESCARTES - Paris V

Discipline : Sciences de la Vie et de la Matière
Spécialité : Mathématiques – Informatique

Par

Marius PREDA

Sujet de la Thèse :



Soutenue le 19 décembre 2002, devant le jury composé de :

Monsieur le Professeur	Georges STAMON	Président
Monsieur le Professeur	Fernando PEREIRA	Rapporteur
Monsieur le Professeur	Jean SEQUEIRA	Rapporteur
Madame le Professeur	Françoise PRÊTEUX	Directeur de Thèse
Monsieur le Docteur	Gérard MOZELLE	Examinateur

Acknowledgements

This work could not have been successfully carried out without the continuous help and support of many people during more than three years.

I would like first to express my deep thanks to Professor Françoise Prêteux, Head of the ARTEMIS Project Unit at INT, who has supervised this thesis. The quality of the training that I have benefited, the many advice and efficient help that she generously offered me all along these years, and the trust that she granted me by promoting me on the international scene, have been valuable assets for which I am deeply grateful and indebted to her.

I would like to express my respect and very special thanks to Professor Georges Stamon from Université Paris V-René Descartes, who has accepted to preside my Ph.D. Committee.

I express my gratitude to Professor Fernando Pereira from *Instituto Superior Técnico* who has co-reviewed my work. I warmly thank him for his numerous and relevant comments which have greatly contributed to improve the content of this manuscript.

I am also grateful to Professor Jean Sequeira, Head of the LXAO Group of the LSIS Department at Université de la Méditerranée, Aix-Marseille II, who has accepted to co-review this dissertation despite his multiple responsibilities. I am very happy to have the opportunity to meet him again.

I express to Doctor Gérard Mozelle, Head of Digital Broadcasting Research and Standardization activities at Thomson, my deep thanks and respect for having generously shared with me his industrial expertise.

It is with a great interest and pleasure that I have shared the enriching adventure of standardization with my colleagues of the SNHC (Synthetic and Natural Hybrid Coding) Group of MPEG. Thanks to Alain Mignot, Michael Steliaros, Gauthier Lafruit, Alexandru Salomie, Francisco Moran-Burgos, Eric Delfosse, Mahnjin Han, James Kim, Patrick Gioia, et Mikael Bourges-Sevenier for their international support!

I also want to say to all Partners of the ViSiCAST European project how much I have appreciated our stimulating cooperation: many thanks to Mark Wells, Head of Televirtual, for a ever beautiful Visia; to Nick Tanton, Project Manager at BBC, for real-scale MPEG-4 Video experiments; and to Christoph Dosch and Werner Brückner from IRT for this great world première during the International Broadcast Convention when a MPEG-4 avatar-supported sign language broadcast transmission was achieved.

In conjunction with this international influence, the ARTEMIS Project Unit has obviously acted as a favorable environment for the blossoming of my activities. I would like naturally to thank:

- Doctors Nicolas Rougon, Catalin Fetita et Titus Zaharia, Associate Professors at INT, for their complicity and conviviality,
- Mrs. Nicole Teste and Evelyne Taroni, for their administrative and friendly support, as well as for their good mood and humor,
- Doctor Marius Malciu, with whom I shared long summer or winter nights.

Since present and future rely on past, I can not forget to witness all my respect and gratefulness to Professor Vasile Buzuloiu from University POLITEHNICA of Bucarest, who has allowed me to lay solid foundations for this thesis.

Finally, I could not end this page without heartily thanking Andrea for her constant moral support and the infinite patience she has witnessed to me in all circumstances.

To Andrea
To my family

Contents

Contents *1*

List of figures *3*

List of tables *5*

Abbreviations *7*

0 Context and Objectives *1*

1 Chapter One. Virtual Character Animation *7*

1.1 Virtual Character Modelling **9**
1.1.1 Polygonal meshes **9**
1.1.2 Parametric representations **10**
1.1.3 Implicit representations **12**
1.1.4 Building 3D virtual characters **13**

1.2 Virtual Character Animation **15**
1.2.1 Kinematic approaches **17**
1.2.2 Dynamic approaches **21**
1.2.3 Motion-based approaches **21**
1.2.3.1 Motion capture systems 21
1.2.3.2 Motion editing systems 22

1.3 Conclusion **23**

2 Chapter Two. MPEG-4 Face and Body Animation: Specifications, Implementation and Performance Analysis *25*

2.1 Overview of the MPEG-4 FBA specifications **27**

2.2 Creating an FBA compliant avatar from a static 3D model **34**
2.2.1 3D model segmentation **34**
2.2.2 Virtual Human Modelling authoring tool **39**

2.3 BAPs editor: the Artemis Animation Avatar Interface **40**
2.3.1 Key value-based animation **42**
2.3.2 End-effector driven animation **43**

2.4 Performances analysis of the FBA framework in a client-server application **45**
2.4.1 Evaluation framework **45**
2.4.2 Compression efficiency **48**
2.4.3 Realistic animation **50**

2.5 Conclusion **53**

3 Chapter Three. Virtual Character Animation: the Skeleton, Muscle and Skin (SMS) Framework *55*

3.1 Synthetic object deformation: toward a unified mathematical model **57**

3.2 Skeleton, Muscle and-Skin-based modelling and animation **58**
3.2.1 Bone and muscle controllers for animating an articulated object **58**
3.2.2 Bone-based modelling and animation **60**

3.2.2.1 Bone influence volume ... 60
3.2.2.2 Bone transformation and induced effects on the skin ... 61
3.2.3 Muscle-based modelling and animation ... 62
3.2.3.1 Muscle influence region ... 62
3.2.3.2 Muscle deformation and induced effects on the skin ... 63

3.3 Skeleton, Muscle and Skin nodes specification ... 66
3.3.1 SBBone Node ... 66
3.3.2 SBMuscle Node ... 68
3.3.3 SBSegment Node ... 70
3.3.4 SBSite Node ... 70
3.3.5 SBSkinnedModel Node ... 71

3.4 Skeleton, Muscle, and Skin animation stream ... 73
3.4.1 Animation principle and resource representation ... 73
3.4.2 Animation parameter representation ... 73
3.4.3 Temporal frame interpolation ... 74
3.4.4 Animation frame ... 75

3.5 Experimental results ... 76

3.6 SMS versus FBA ... 79

3.7 The SMS framework in the MPEG-4 Context ... 80

3.8 Summary ... 83

4 Chapter Four. MPEG-4 at Work: Sign Language Transmission over Digital Television and Web-based Networks ... 85

4.1 Introducing sign language communication ... 86

4.2 Analysed solutions for sign language communication ... 86
4.2.1 MPEG-4 video object-based encoding of a natural signer ... 87
4.2.2 MPEG-4 Video object-based encoding of a virtual signer ... 95
4.2.3 MPEG-4 Virtual Character Animation (FBA and BBA) ... 96
4.2.4 Comparative evaluation ... 96
4.2.4.1 Equipment (hardware and software) required at the producer's side ... 97
4.2.4.2 Real-time capabilities of producing the content ... 98
4.2.4.3 Reusability of the content ... 98
4.2.4.4 Terminal capabilities ... 99
4.2.4.5 User interaction with the content ... 100
4.2.4.6 Minimum bandwidth required for the transmission of the content ... 100
4.2.4.7 Comparison summary ... 114
4.2.5 Implementation in a real system: the ViSiCAST project ... 116

4.3 Summary ... 117

5 Conclusions and future work ... 119

A.1 Appendix A. Related Publications ... 123
A.1.1 Book chapters ... 124
A.1.2 Journal papers ... 124
A.1.3 Conference papers ... 124
A.1.4 Patents ... 124
A.1.5 Tutorials ... 125
A.1.6 Technical reports ISO ... 125

A.2 Appendix B. XML compliant format for BBA ... 129

6 References ... 135

List of figures

Figure 1.1. Different geometry complexity levels for virtual characters. 9
Figure 1.2. Single NURBS surface a) and b) and patched NURBS surfaces c) and d). 11
Figure 1.3. Metaballs blending. 12
Figure 1.4. Metaball-based model: detail on human right arm designed by Roberto Campus [Campus]. 13
Figure 1.5. Metaball-based model: detail on human heart [Mari]. 13
Figure 1.6. Full human body scanner. Courtesy of Cyberware. 14
Figure 1.7. Virtual humanoid animation, for a segmented character (a) and b)), and for a seamless character (c) and d)). 16
Figure 2.1. Examples of standardized key-points for the "face" object. 27
Figure 2.2. The Body *scene graph hierarchy.* 28
Figure 2.3. Standardized rotation planes for the avatar body. 28
Figure 2.4. Abduction of the right arm. 29
Figure 2.5. Flexion of the left arm. 30
Figure 2.6. Standardized rotation axes attached to shoulder, elbow and wrist. 31
Figure 2.7. Standardized rotation axes attached to fingers. 31
Figure 2.8. Frame predictive-based method block diagram. 32
Figure 2.9. DCT-based method block diagram. 32
Figure 2.10. Avatar initial position effects on the animation result. a) correct initial position, b) incorrect initial position, c) and d) results obtained from the same animation parameter set. 33
Figure 2.11. a) Seamless body. b) FBA Body object: a hierarchical segmented character. 34
Figure 2.12. Segmenting upper arm of the 3D model: selecting first JC. a) first point of the JC, b) second point of the JC, c) automatically generated segment between the two points, d) automatically generated JC. 36
Figure 2.13. Segmenting upper arm of the 3D model: selecting the second JC. a) first point of the JC, b) second point of the JC, c) automatically generated segment between the two points, d) automatically generated JC. 37
Figure 2.14. Segmenting upper arm of the 3D model: geodesic dilation. a) two JC, b) random initialization between the two JC, c) geodesic dilatation, d) segmentation result. 38
Figure 2.15. Virtual Human Modelling Authoring Tool. The Segmentation Module control dialog. 39
Figure 2.16. Virtual Human Modelling Authoring Tool. The Building Module control dialog. 40
Figure 2.17. The ARTEMIS Avatar Animation Interface with the forward kinematics BAP editing control dialog. 41
Figure 2.18. The ARTEMIS Avatar Animation Interface used for interactive tracking of gestures in image sequences. 42
Figure 2.19. The ARTEMIS Avatar Animation Interface with the "Interpolation" control dialog for the case of two key-frames. 43
Figure 2.20. Successive steps for Cyclic-Coordinate Descent algorithm for a two bones (arm-forearm) IK system. 45
Figure 2.21. Animation parameters encoder and the UDP server interface. 46
Figure 2.22. The ARTEMIS Avatar Animation Interface with the "UDP client" control dialog. 46
Figure 2.23. "Applause" sequence: 2 avatars with different morphometries and animated with the same animation parameters. 47
Figure 2.24. Experimental results for sign "B". Predictive (dense bullets) vs DCT BAP coding schemes. Distortion vs Q (I), Bit-rate vs Q (II) and Distorsion vs bit-rate (III). 49
Figure 2.25. Frame from animation data set. 50
Figure 2.26. Finger animation: without BDTs (a), b), c), d)); with BDTs (a'), b'), c'), d')) – BDTs interpolation: key frames (a') and d')), interpolated BDTs (b') and c')). 52
Figure 3.1. Forearm bone influence volume. 61
Figure 3.2. The muscle influence volume (in transparent green). 63
Figure 3.3. Influence of the translation of a control point on the NURBS shape.. 64
Figure 3.4. Influence of weighting a control point (here, B_3) on the NURBS shape. 65

Figure 3.5. Influence of the knot sequence on the curve shape. 65
Figure 3.6. The fields of the SBBone node. 66
Figure 3.7. The fields of the muscle node. 69
Figure 3.8. The fields of the SBSkinnedModel node. 71
Figure 3.9. Principle of the analysor (illustration for one parameter). 78
Figure 3.10. Extract from the "MPEG-4 Requirements, version 15 (La Baule revision)". 81
Figure 4.1. Natural signer: the "Florea" sequence. 88
Figure 4.2. Binary segmentation mask (the signer - white - and the background - black) for the "Florea" sequence. 89
Figure 4.3. Natural signer: the "Carlos" sequence. 90
Figure 4.4. Binary segmentation mask (the signer - white - and the background - black) for the "Carlos" sequence. 91
Figure 4.5. The main steps of the head-and-hands segmentation procedure ("Florea" signer). 92
Figure 4.6. The main steps of the head-and-hands segmentation procedure ("Carlos" signer). 93
Figure 4.7. Binary mask for the NSV3O method for the "Carlos" sequence. 94
Figure 4.8. Virtual character rendering: a) normal and b) modified (no lights and no textures) appearance model. 96
Figure 4.9. I, P, B structure of the Group of Planes (GOP). 101
Figure 4.10. Bitrate [kbps] versus Q. 103
Figure 4.11. Bitrate [kbps] versus frame-rate. 104
Figure 4.12. Bitrate [kbps] versus Q. 106
Figure 4.13. Bitrate [kbps] versus frame-rate. 107
Figure 4.14. Bitrate [kbps] versus Q. 109
Figure 4.15. Bitrate [kbps] versus frame-rate. 110
Figure 4.16. Multimedia Player. 115
Figure 4.17. Synoptic description of a broadcast architecture ensuring sign language transmission into a standardized framework. 115
Figure 4.18. Snapshots of ViSiCAST broadcast system. 116

List of tables

Table 2.1. Bit-rates for the frame predictive-based and DCT-based coding schemes. Results for signs "A" to "L". Q denotes the global quantization value. 48
Table 2.2. Bit-rates [kbps] for the DCT-based coding scheme. Q denotes the global quantization value. 50
Table 3.1. Bit-rates [kbps] corresponding to frame predictive-based and DCT-based encoding schemes applied to "A" to "J" signs data. Q denotes the global quantization value. The frame-rate is set to 10 fps. 77
Table 3.2. Bit-rates [kbps] corresponding to frame predictive-based and DCT-based encoding schemes coupled with temporal interpolation, applied to "A" to "J" signs data. Q denotes the global quantization value. The frame-rate is set to 10 fps. 78
Table 3.3. Main FBA and SMS features. 79
Table 4.1. Scores with respect to the "Equipment required at the content producer's side" criterion. 98
Table 4.2. Scores with respect to the "real-time capabilities of producing the content" criterion. 98
Table 4.3. Scores with respect to the "reusability of the content" criterion. 99
Table 4.4. Scores with respect to the "terminal capabilities" criterion. 100
Table 4.5. Scores with respect to the "user interaction with the content" criterion. 100
Table 4.6. NSV: compression results with respect to the quantization step (example for "Florea" sequence"). 102
Table 4.7. NSV: compression results with respect to the time sampling (example for "Florea" sequence"). 103
Table 4.8. NSV: compression results with respect to the image resolution. 104
Table 4.9. NSV2O: compression results with respect to the quantization step. 106
Table 4.10. NSV2O: compression results with respect to the time sampling. 106
Table 4.11. NSV2O: compression results with respect to the image resolution. 107
Table 4.12. NSV3O: compression results with respect to the quantization step. 109
Table 4.13. NSV3O: compression results with respect to the time sampling. 109
Table 4.14. NSV3O: compression results with respect to the image resolution. 110
Table 4.15. Encoding overload for NSV3O+C. 111
Table 4.16. SSV: compression results with respect to the quantization step. 113
Table 4.17. Scores with respect to the "required bandwidth" criterion. 114

Abbreviations

3D	Three Dimensional
3DMC	3D Mesh Coding
AC	Alternative Components (in frequency representation)
AFX	Animation Framework eXtension
ASL	American Sign Language
BAP	Body Animation Parameters
BBA	Bone based Animation
BIFS	Binary Format for Scenes
BDP	Body Definition Parameters
BDT	Body Deformation Tables
CIF	Common Intermediate Format
CCD	Cyclic Coordinate Descent
DC	Direct Comonent (in frequency representation)
DCT	Discrete Cosinus Transform
FAP	Face Animation Parameters
FBA	Face and Body Animation
H Anim	Humanoid Animation
IP	Internet Protocol
ISO	International Standardization Organization
MPEG	Motion Picture Expert Group
MPEG-4	Multimedia Standard ISOIEC 14496
MPEG-2	Audio-Video Standard ISOIEC 13818
MPEG-2 TS	MPEG-2 Transport Stream
NSV	Natural Signer Video
NSV*i*O	Natural Signer Video with *i* Objects
NURBS	Non uniform rational B Splines
QCIF	Quarter Common Intermediate Format
SMS	Skeleton, Muscle and Skin
SNHC	Synthetic and Natural Hybrid Coding
SSV	Synthetic Signer Video
VRML	Virtual Reality Modeling Language
XML	eXtensible Markup Language
XMT	eXtensible MPEG 4 Textual Format

Context and Objectives

The current multimedia information society brings up new challenging issues in terms of production, distribution, and access of multimedia content. New forms of representation of the content are continuously developed and more and more content becomes available. In a society that cares about each of its members, new technologies can be used for example to address issues related to the distant communication and content access for the hearing impaired community.

We are presenting in this introduction the requirements of such multimedia framework, able to offer a solution to the deaf community using virtual characters. Then, we describe the advantages of providing such framework in terms of applications interoperability, support on the network and terminal devices level, data compression and synchronization, sharing and exchanging the content. These requirements make us review the existing integrated multimedia frameworks. Focusing on two open and international specifications, namely MPEG-4 and VRML, we analyse the performances and capabilities of each, related to their generic functionalities. We then present the structure of this dissertation.

We are currently witness to the explosion of multimedia content consumption. Broadcast productions overflow real life with images and sounds on thousand of channels throughout the world. Furthermore, a new age started with the development of the Internet. Nowadays, it is not only a source of written information, but a huge multimedia network where the distinction between user and content creator is not clearly stated.

The economic as well as social ramifications of this boom of the information technology society have been identified by the EC [1]. Recommendations have been written for the promotion of the interoperability of multimedia application developments and their equal access to all citizens. This intent is specifically expressed when requesting that the multimedia content must be accessible also for people with audio-visual handicaps. Among them, the hearing impaired community is one of the early ones to benefit from these directives, especially where new technologies are concerned. As a communication means, this community developed, a gesture-based language called sign language. Offering them access to the content through their natural language means to provide with a translation between sound information and sign language. Taking for as example the digital television case, broadcasters now have an obligation to have at least 3% of their programs translated in sign language. To have a human translator on the set of every show would prove unrealistic in view of the rabid growth of the sector. Then, automated translation means between oral communication and sign language do need to be developed. To give full satisfaction such systems should not only translate but also be able to display a synthetic visual representation of the sign language. The majors difficulties are to be found in the linguistic complexity of both sign and oral languages, but also in the implementation of the gesture performing representation of a virtual human character.

Therefore, the system to be developed needs to cope with heavy constraints in terms of realistic animation of virtual characters, compatibility with networked environments (both broadcast and webcast), as well as compact representation for low bit rate transmission.

The complexity of the problems that the implementation of such system addresses let us think that such application should be considered in an integrated framework, where production, transmission and consumption of multimedia content are to be analysed. The objective of our research, is to propose a framework for virtual character animation with the following requirements:

- it should enable virtual character definition and animation,
- in order to ensure the visual confort in the case of sign language content, the static representation of the above-mentioned model should be realistic; *i.e.*, the model should support geometries and textures as complex as required,
- in order to ensure the comprehension of the virtual character gesture, the animation of the above-mentioned models should be realistic; in particular, animating the model should not produce seams at the joints level and muscle-like deformations should be supported,
- the animation parameters should be easy to produce; the representation of the animation parameters should be consistent with respect to motion capture systems; editing tools for these parameters should be easy to build,

[1] EC – European Commission

- the animation parameters representation should be compact and streamable, thus allowing animation transmission on low bit-rate networks; furthermore, the stream should be easily multiplexed with other media streams already available,
- the animation speed should be appropriate in order to ensure the smoothness of the motion; with respect to sign language content, a frame-rate of 15 fps is the minimum acceptable,
- the virtual character model should be able to co-exist, in a 3D scene with other kinds of media such as video, sounds and 2D/3D graphic objects,
  the animation should be synchronized with other playable media existent in the same scene,
- the model definition and animation should be possible on low performance terminals, *i.e.*, scalability to the terminal should be considered.

Creating, animating and sharing virtual characters require unified data formats. If some animation industry leaders try, and sometimes succeed [3DSM, Maya99], to impose their own formats in the computer (PC) world, the alternative of an open standard is the only valid solution when hardware products are to be build. A standard is all the more needed and awaited when applications and services require the support of various activity field actors such as content creators, developers, service providers (broadcasters, ISP), terminal manufacturers, or IPR holders.

The need for interoperability and the existence of open standards is concerning more and more legislative structures accros Europe.

The EU Framework Directive on Telecommunication Networks and Services [EC02-a] calls for interoperability of digital television services.

The European Council summit in Barcelona in March 2002 called for "…the Commission and Member States to foster the development of open platforms to provide freedom of choice to citizens for access to applications and services of the information society, notably through digital television...". It went on to invite the Commission to present "a comprehensive analysis of remaining barriers to: the achievement of widespread access to new services and applications of the information society through open platforms in digital television…" [EC02-b].

The eEurope 2005 Action Plan states: "By supporting the emergence of alternative access platforms, such as digital television or 3 G mobile systems, the new action plan will further facilitate e-inclusion, also for people with special needs" [EC02-c].

Current works to provide 3D applications with a unified and interoperable framework are materialized by 3D graphics interchange standards such as VRML[2] [VRML] and multimedia 2D/3D standards such as MPEG-4 [ISO99]. Each one addresses, more or less in a coordinated way, the virtual character animation issue. In the VRML community, the H-Anim[3] group released three versions of their specifications (1.0, 1.1 and 2001), while the SNHC[4] sub-group of MPEG[5] released two versions: MPEG-4 Version 1 allows face animation and MPEG-4 Version 2 allows body animation. In MPEG-4

[2] VRML - Virtual Reality Markup Language
[3] H-Anim – Humanoid AnimationWorking Group
[4] SNHC - Synthetic and Natural Hybrid Coding
[5] MPEG - Motion Picture Expert Group

the specifications dealing with the definition and animation of avatars are grouped under the name FBA[6]. The next section analyses the main similarities and differences of these two frameworks.

The VRML standard deals with a textual description of 3D objects and scenes. It focuses on the spatial representation of such objects, while the time behaviour is less supported. The major mechanism for supporting animation consists in defining it as an interpolation between key-frames.

The MPEG-4 standard, unlike the previous MPEG standards, does not only cope with highly efficient audio and video compression schemes, but also introduces the fundamental concept of media objects such as audio, visual, 2D/3D natural and synthetic objects to make up a multimedia scene. As established in July 1994, the MPEG-4 objectives are focused on supporting new ways (notably content-based) of communicating, accessing and manipulating digital audiovisual data [Pereira02]. Thus, temporal and/or spatial behaviour can be associated with an object. The main functionalities proposed by the standard address the compression of each type of media objects, hybrid encoding of the natural and synthetic objects, universal content accessibility over various networks and interactivity at the end-user. In order to specify the spatial and temporal localisation of an object in the scene, MPEG-4 defines a dedicated language called BIFS (Binary Format for Scenes). BIFS inherits from VRML the representation of the scene, described as a hierarchical graph, and some dedicated tools such as animation procedures based on interpolators, events routed to the nodes or sensor-based interactivity. However, BIFS introduces some new and advanced mechanisms such as compression schemes to encode the scene, streamed animations, integration of 2D objects and advanced time control.

In terms of functionalities related to virtual characters, both VRML and MPEG-4 standards define a set of nodes in the scene graph to allow for a representation of an avatar. However, only the MPEG-4 SNHC specifications deal with streamed avatar animations. A major difference is that an MPEG-4 compliant avatar can coexist in a hybrid environment and its animation can be synchronized with other types of media objects, while the H-Anim avatar can only exist in a VRML world and must be animated by VRML generic, usually non-compressed, animation tools.

With the large collection of tools it provides, the MPEG-4 standard can be the ideal candidate with respect to some of the requirements listed at the beginning of this section. As the MPEG-4 standard already has a framework dealing with the animation of human-like models, so-called FBA, we first propose to evaluate this framework. As some of the requirements are not fulfiled by the FBA, especially those related to realistic animation and representation of a generic model, we propose a new framework, so-called Skeleton, Muscle and Skin (SMS). The presentation of this work is structured in four chapters.

A comprehensive view of the evolution of 3D virtual character-related techniques is presented in Chapter 1. This brief overview is structured according to the two major components of an animation system, that is 1) modelling, and 2) animation. These components are strongly interconnected and application-dependent, as we will show afterwards. As our interest focuses on methods used in a production environment, special attention is paid to automatic methods to build and animate a virtual character. That is why the full-body geometry scanners and motion capture systems are also presented.

[6] FBA - Face and Body Animation

The second chapter describes the Face and Body Animation framework within the MPEG-4 standard in terms of specifications, tools implementation and performances analysis. The first section of the second chapter introduces the basic concepts used by the MPEG-4 standard to define a collection of multimedia objects and their relationships. In this general framework, the FBA specific mechanism for defining a virtual body in an MPEG-4 scene is presented. The FBA specifications do not mandate any specific method to obtain a real description of a 3D human body and its associated animation parameters. To address this first issue, we propose, a novel semi-automatic approach to get a MPEG-4 compliant body representation from a 3D mesh model of a human. This semi-automatic approach based on a segmentation algorithm is implemented as a part of the Virtual Human Modelling (VHM) authoring tool. In order to address the second issue, we present a dedicated authoring tool named the ARTEMIS Animation Avatar Interface (3AI), implemented to facilitate the editing and extraction of animation parameters. This authoring tool includes various interpolation schemes and one inverse kinematics method. The 3AI authoring tool also provides high-level functionalities like (1) composition of multimedia scenes made of natural data (images, video) and synthetic body representation and (2) calibration of a 3D body model according to anthropometrical characteristics. Then, we address a practical evaluation of both the compression and the animation performance of the FBA framework. Using the two above-mentioned authoring tools, an FBA compliant avatar and several animation data sets corresponding to the American Sign Language alphabet were created. A complete server-client communication system was then set up to test the complete application from the encoding of the animation parameters and their transmission over an IP (Webcast) and MPEG-2 (Broadcast) networks to their decoding and rendering of the associated human model. Results are presented and analysed in terms of trade off between the quality of the animation and the necessary bit-rate. These results underline performances and limitations of the FBA framework. In terms of applications, the original contribution described in this chapter consists in an end-to-end testing platform related to FBA content production, transmission and consumntion. In terms of methodology the original contributions are related to the FBA compliant content production, namely the geometry segmentation algorithm and the FBA animation parameters creation tools (interpolation, inverse kinematics).

The goal of the third chapter is to introduce a new framework for virtual character animation. The framework is built on a generic deformation model that we introduce in the first section. The model consists of a controller with a geometric support, a volume embedding the geometric support and a measure of the affectedness of the controller in the volume. With respect to these elements, we classify the main deformation tools reported in the literature. Then, we propose two declinations of the model which offer efficient control of the geometrical support and appropriate volume specification: bone and muscle controllers. We consider these tools as the core of the SMS framework and we show how they can be used to define and animate generic virtual characters. In order to represent generic virtual characters into a 3D scene, we provide the SMS architecture with the scene graph nodes. We then address the issue of efficient animation parameters representation by (1) enriching the framework with interpolation and inverse kinematics support and (2) proposing to adapt two compression methods (frame predictive-based and DCT-based). The results in terms of bit-rate, that we present and analyse, allow addressing animation on low bit-rate networked environment. A comparative study between the SMS and FBA frameworks, states the advantages of the SMS one. Finally, we describe how we have promoted the SMS framework into the MPEG-4 standard. The original contributions presented in this

chapter have to be considered in three classes. In terms of methodologie, a unified mathematical model of deformation mechanisms is proposed, and two instances of this model, the bones and muscle controllers are detailed. Within the second class, of applications, two compression methods for the animation parameters have been integrated. Finally, the SMS framework has been promoted to the MPEG-4 standard by providing it with a scene graph structure and a animation stream syntax compatible with the MPEG-4 structure.

There are several applications where synthetic data must coexist with natural data and moreover, the synchronisation is required. As previously mentioned, one of such application is the transmission of sign language related content over IP and MPEG-2 Transport Stream. This system is presented in Chapter 5. One of the requirements of this application is that the supplementary bandwidth needed for transmitting the sign language content must be less than 100 kbps for broadcast and less than 56 kbps for webcast. We will describe six possible solutions for sign language transmission over networks. The six solutions are classified with respect to the data format used to represent the content at different link of the chain: production, transmission and consuming. Thus, we decline video-based solutions, 3D-solutions and a hybrid (video and 3D) solution. With respect to some key-criteria we will analyse each solution, highlighting the specific advantages and limitations. By performing compression tests with each method, we will present and discuss the results with respect to the quality/bit-rate. The orinal contribution in this chapter consists in the analysis of several methods for transmitting sign language content by enphasising the advantages and limitations of each one with respect to the content production, transmission and consumntion.

Finally, concluding remarks and perspectives of future works are reported in the last chapter.

Chapter One

Virtual Character Animation

The first 3D virtual human models were designed and animated with the help of computers in the 60's. Since then, virtual character models have become more and more popular, so as to be available to growing numbers of users for all kinds of purposes. From simple and easy-to-control models used in video games [Activision, EArts], to more complex virtual assistants for commercial [Lat] or news web sites [Seonaid], or to the new stars of virtual cinema [WdPixar], television [Vandrea] or advertising [EveSolal], the 3D character models industry is now booming.

Furthermore, the steady improvements of the distributed network area in terms of bandwidth capabilities, bit rate performances or advanced communication protocols have lead to the

emergence of 3D communities [Blaxxun] and immersion experiences [Thalmann00] in distributed 3D virtual environments.

Hereafter, we present a comprehensive view of the evolution of 3D virtual character-related techniques. This brief overview is structured according to the two major components of an animation system, that is 1) modelling, and 2) animation. As our interest is on methods used in a production environment, special attention is paid to automatic methods for building and animating a virtual character. Full body geometry scanners and motion capture systems are presented and discussed.

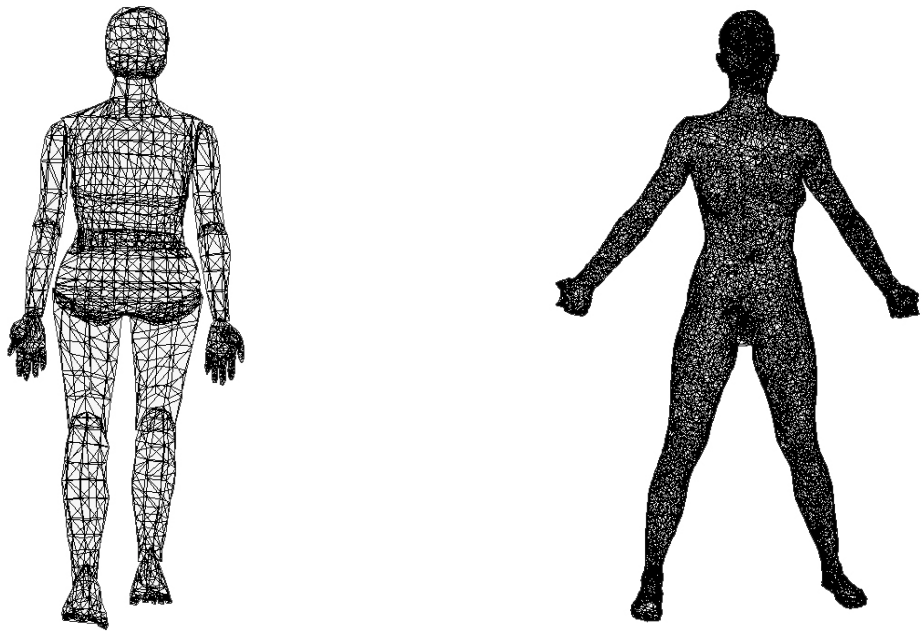
1.1 Virtual Character Modelling

Virtual character modelling consists in specifying the geometry and appearance properties (colour, material, texture) of a synthetic model. According to the chosen type of synthetic model geometry, designing a virtual character can be done either using a segmented-based approach or within a seamless framework. The so-called *segmented* character is defined as a hierarchical collection of 3D objects, referred to as segments. The so-called *seamless* virtual character is geometrically defined as a single and continuous mesh. The most relevant techniques used to specify the model geometry refer to surface-based representations and more specifically to 1) polygonal meshes, 2) parametric equations, and 3) implicit representations.

1.1.1 Polygonal meshes

The polygonal surface-based technique consists of explicitly specifying the set of planar polygons which compose the 3D object surface, and connecting them together [Foley92]. Two types of information have to be kept about an object, namely, purely *geometric* information (coordinates of vertices) and *topological* information (which gives details on how the geometric entities relate to each others).

Historically, the polygonal surface-based technique is the first method introduced in computer graphics and remains the basic tool used at the rendering stage by all other surface representation techniques [OpenGL, Kovach]. Its main advantage lies in its capability to describe all complex surfaces, with smoothness conditioned by the number of polygons used. Figure 1.1 shows an example of two characters with different geometry complexities.



a) b)

Figure 1.1. Different geometry complexity levels for virtual characters.
a) 3400 triangles, b) 25000 triangles.

However, in a real-time animation framework, increasing the number of polygons of a virtual model may significantly deteriorate the animation's performances. In this case, the adopted solution consists in deriving a lower resolution version of the virtual character, then animating/deforming this version, and eventually recovering the full-resolution version using a subdivision surface method [Catmull78, Loop87, Zorin00].

1.1.2 Parametric representations

Non-planar parametric representations of a surface are widely used in computer graphics because of the easy computation and simple mathematical manipulation. A non-planar surface patch, *i.e.* an elementary curved surface entity, is defined as the surface traced out with the two parameters $(u,v)\in[0,1]^2$, in a two-parameters representation:

$$P(u,v)=[x(u,v),y(u,v),z(u,v)]. \tag{1.1}$$

According to the curve degree, the patch surface can be of a linear, cardinal, B-spline or Bézier type [Farin]. The patch-based modelling method aims to produce patches trailing the curve profile(s) and to stitch them together in order to build complex surfaces. Specific constraints have to be introduced in order to properly build a patch-based virtual character. Special care is recommended to (1) completely cover a joint with a unique patch surface, and (2) ensure the continuity of the mesh at the patch borders where adjacent frontiers should be identically deformed. Mathematically, the patch-based surface representation is the limit surface obtained through convergence of an iterative surface subdivision procedure.

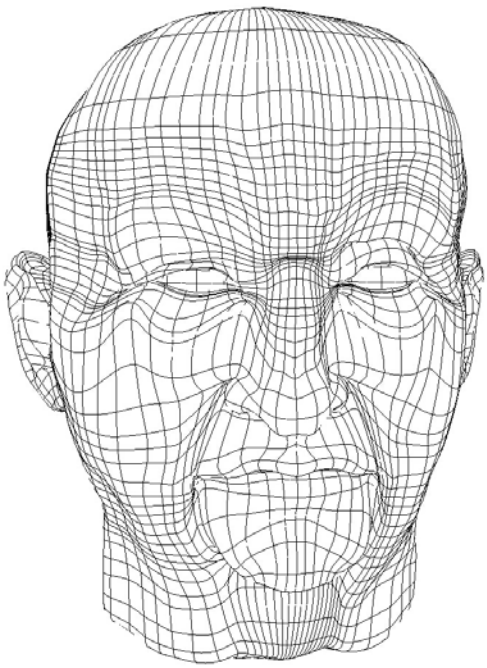
NURBS-based surface representation [Piegl97] is an extension of the B-spline patch obtained by weighting the influence of each control point. A NURBS surface of degree (*p,q*) is defined as:

$$P(u,v)=\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)\cdot N_{j,q}(v)\cdot w_{i,j}P_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)\cdot N_{j,q}(v)\cdot w_{i,j}}, \tag{1.2}$$

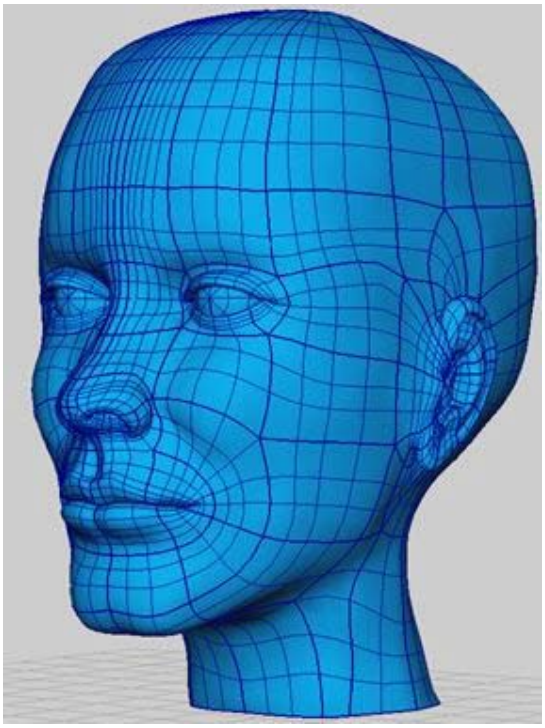
where $N_{i,j}$ denotes the B-spline basis functions, $P_{i,j}$ the control points, and $w_{i,j}$ the weight of $P_{i,j}$. In this case, a non-uniform weighting mechanism increases the flexibility of the representation, which then becomes appropriate to model a wide range of shapes with very different curvature characteristics, while the number of control points is kept to a minimum.

As an example of the ability of NURBS-based representation to model complex shapes, Figure 1.2 a) and b) shows a head model represented by using a single NURBS surface. However, the number of control points involved in this type of representation may be important, which raises several limitations for real-time animation. Decreasing the number of control points requires to model a 3D object by patching several NURBS surfaces as illustrates Figure 1.2 c) and d).

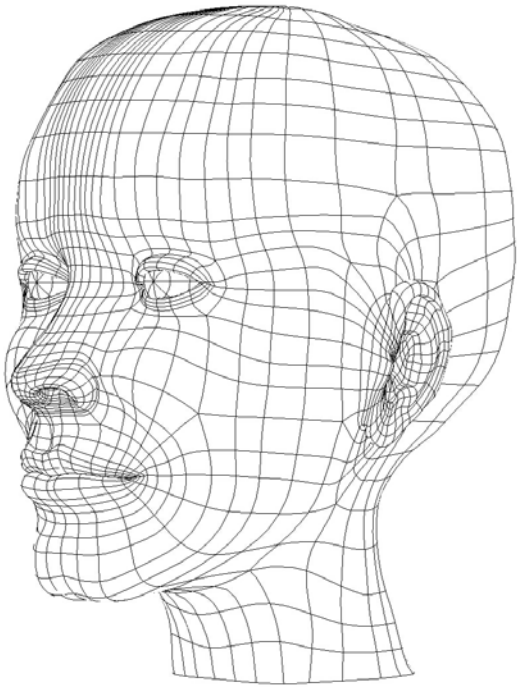
a)



b)



c)



d)

Figure 1.2. Single NURBS surface a) and b) and patched NURBS surfaces c) and d). Courtesy of Jeremy Birn (part of “NURBS Head Modeling” tutorial [Birn]).

1.1.3 Implicit representations

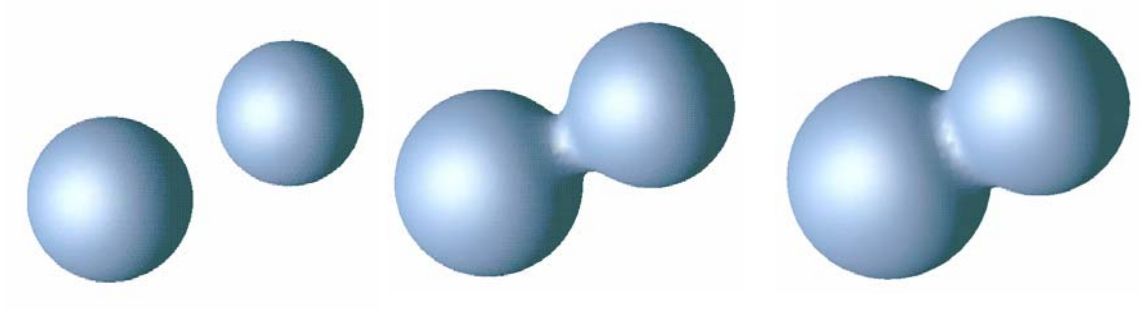
The implicit representation of the geometry of a 3D object is a modelling technique using the so-called metaballs [Blinn82, Wyvill86], also known as blobby objects. A metaball can be viewed as a particle surrounded by a density field. The density assigned to the particle (its influence) decreases with the distance. An implicit model is inferred by considering an isosurface through this density field - the higher the isosurface value, the nearer it will be to the particle. The key to using metaballs is the equation specifying the influence of an arbitrary particle on an arbitrary point in space. Blinn [Blinn82] used exponentially decaying fields for each particle:

$$C(r) = b * \exp(-ar)\,, \tag{1.3}$$

while Wyvill *et al.* [Wyvill86] defined a cubic polynomial based on the radius of influence R of a particle and the distance r from the centre of the particle to the field location considered:

$$C(r) = 2\frac{r^3}{R^3} - 2\frac{r^2}{R^2} + 1\,. \tag{1.4}$$

The powerful aspect of metaballs lies in the way they can be combined. By simply summing the influences of each metaball at a given point, we get a very smooth blending of the spherical influence fields (Figure 1.3) thus allowing for a realistic, organic-looking shape representation.



a) b) c)

Figure 1.3. Metaballs blending.

By combining metaballs-based surfaces, the designer can achieve high quality representations of real anatomical structures as shown in Figure 1.4 for an example of a human right arm and in Figure 1.5 for a human heart. However, because the control of implicit surfaces is ensured by the equation coefficients, intuitive animation of such surfaces is hard to achieve. While the global effects as scaling are easy to perform, local deformations are usually obtained by a combination of locally-defined implicit functions. Furthermore, mapping textures on such geometry is a difficult task. Usually, the models designed with this technique have their colours specified at the vertex level or are covered by plain colours.

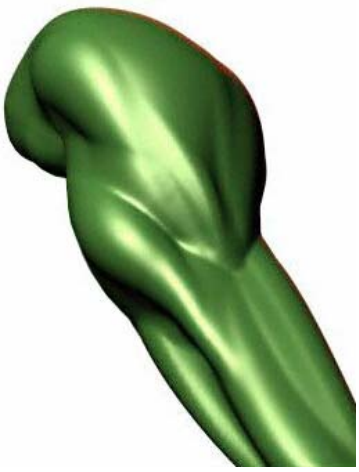


Figure 1.4. Metaball-based model: detail on human right arm designed by Roberto Campus [Campus].

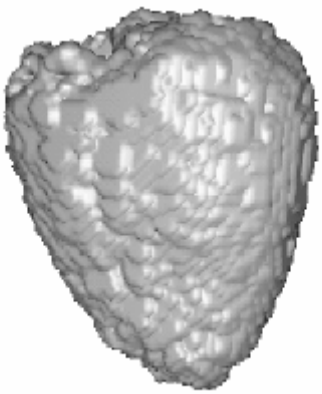


Figure 1.5. Metaball-based model: detail on human heart [Mari].

In practice, the choice of one of the above three approaches for 3D object representation depends on the targeted application and is a trade-off between speed requirements for real-time animation and quality of the object rendering. For example, a character involved in a computer gaming environment is usually modelled with simple polygons, while in the case of a movie, characters can be either modelled as complex polygonal surfaces using subdivision surface techniques or directly defined as NURBS surfaces. Conversely, virtual character animation by means of implicit surfaces remains difficult because of the complexity of manipulating the control coefficients to obtain authoring free-style animation. In addition, a real-time rendering is hard to achieve due to the time-consuming rasterisation process required for complex objects. The use of metaballs is thus limited to off-line productions such as in the movie industry [WD97].

From the previous remarks, we are focusing in our research on meshed models and we present in the following the basic approaches used in practice for building a virtual character.

1.1.4 Building 3D virtual characters

Two different courses can be chosen from in order to build a virtual character according to the researched appearance of the virtual character (cartoon-like or realistic), and depending on the technology available to the designer.

On the one hand, the designer can build interactively the model's anatomical segments and set up the model hierarchy. Several authoring tools and geometry generating mechanisms make it possible to model a virtual character in a relatively short time [3DSM, Maya]. The main drawback of this method is that the result is strongly dependent on the designer's artistic skills and experience. In addition, this procedure is tedious and time-consuming.

On the other hand, a faster and proven method is the use of 3D scanners. Contrary to Computer Aided Manufacturing, the aim of 3D scanning is to create an electronic representation of an existing object, capturing its shape, colour, reflectance or other visual properties. In its principle,

3D scanning is similar to a number of other important technologies (like photocopying and video) that quickly, accurately and cheaply record useful aspects of physical reality.

The scanning process can be structured according to the following steps: acquisition, alignment, fusion, decimation and texturing. The first step aims at capturing the geometric data of the 3D object by using a dedicated scanning device (Figure 1.6). Depending on the type of scanner used, the execution of this phase can vary considerably. Either a single scanning is enough to capture the whole object, or series of partial scans (called range maps) are needed, each of them covering a part of the object. In the latter case, range maps taken from different viewpoints have to be aligned, which is the task of the second step. This procedure can be completely automatic if the exact position of the scanner during each acquisition is known. Otherwise, a manual operation is needed to input the initial placement, then the alignment is performed automatically. Once aligned, the partial scans need to be merged into a single 3D model ("fusion" step).

Figure 1.6. Full human body scanner. Courtesy of Cyberware.

Because 3D scanners provide a huge amount of data, a "decimation" step is required. For an effective use of the model, one has to reduce the size of the acquired geometric information, especially of the less significant parts of the object (a sample rate of 0.25 mm can be useful for very complex sections, but not on flat surfaces). Decimation software can be based on edge collapsing [Ronfard96] and error-driven simplification [Schroeder92].

The last step, "texturing", is not mandatory for applications such as simulations in a virtual environment but, for a large array of objects, additional information about the real appearance of the object must be provided. This is usually achieved by texturing the final model, using pictures taken during acquisition. The pictures are first aligned to the geometry (manually or automatically) and then mapped to the model.

Once the virtual character has been created, one should be able to change its postures in order to obtain the desired animation effect. The following section addresses the problem of virtual character animation and presents the main approaches reported in the literature.

1.2 Virtual Character Animation

Virtual character animation is directly related to the model type used in generating the character, that is, *segmented* or *seamless*. Indeed, in the first case, the model is defined as a hierarchical collection of segments, while in the second case, it is composed of a single mesh. These particularities have to be taken into account in the developed animation approaches.

Animating a segmented character consists in applying affine transformations to each segment. The main advantages of such a simple technique are real-time capabilities on basic hardware configurations and intuitive motion control. However, such an animation procedure results in seams at the joint between two segments (Figure 1.7 a) and b)). This undesirable effect, in the case of cartoon-like animations, can be more or less side-stepped using *ad hoc* methods, such as introducing spheres at the joint level or 3D objects to mask the joints. However, more realistic animations require handling local deformations.

Animating a seamless character consists in applying deformations at the skin level. The major 3D mesh deformation approaches can be classified into the following 5 categories:

- *Lattice-based* [Maestri99]. A lattice is a set of control points, forming a 3D grid, that the user modifies in order to control a 3D deformation. Points falling inside the grid are mapped from the unmodified lattice to the modified one using smooth interpolation.
- *Cluster-based* [Maestri99]. Grouping some vertices of the skin into clusters enables to control their displacements by using the same parameters.
- *Spline-based* [Bartels87]. Spline and, in general, curve-based deformations allow deforming a mesh with respect to the deformation of the curve.
- *Morphing-based* [Blanz99]. The morphing technique consists in smoothly changing a shape into another one. Let us mention that such a technique is very popular for animating virtual human faces from pre-recorded face expressions.
- *Skeleton-based*. [Lander99]. The skeleton is a hierarchical structure and the deformation properties can be defined for each element of this structure.

The first four categories are used in animating specific objects as eyes (lattice), and face expressions (morphing), and are more or less supported by the main animation software packages. The last category, more and more encountered in virtual character animation systems, introduces the concept of *skeleton*.

The skeleton of a virtual character is a set of semantic information (usually, the skeleton is not displayed) composed in a hierarchical structure of elementary entities called bones. The seamless character is deformed by applying rigid transformations onto the skeleton *bones*. These transformations induce displacements of a subset of the skin vertices. This technique avoids seams

at joint level (Figure 1.7. c) and d)). The skeleton-based animation concept will be described in detail in Chapter 3.

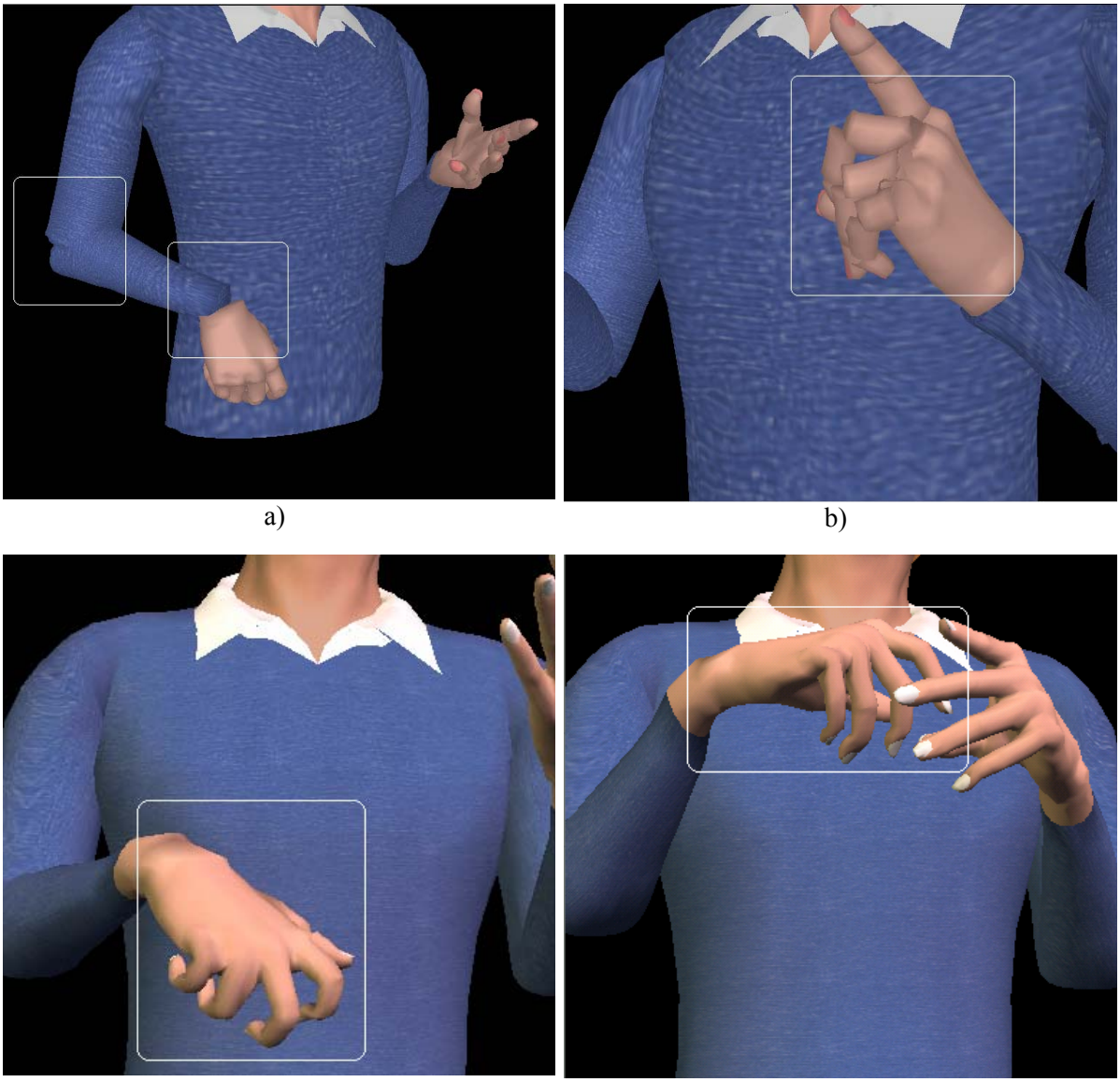


Figure 1.7. Virtual humanoid animation, for a segmented character (*a)* and *b)*), and for a seamless character (*c)* and *d)*).

To design the virtual character skeleton, an initialisation stage is necessary: the designer has to specify the influence region of each bone of the skeleton as well as a measure of influence. This stage is mostly interactive and recursively repeated until the desired animation effects are reached. Deforming a seamless character remains a complex and time-consuming task, requiring dedicated hardware. However, the increasing achievements in computer graphics hardware contribute to promote the skeleton-based animation technique as a "standard" in 3D applications [3DSM99, Maya99, Kalra98]. The current trend is in favour of dividing the virtual character into seamless sub-parts and to deform each one independently of the others. This strategy helps to fulfil the real-

time requirements and to unify the animation approaches for segmented and seamless virtual characters.

In this respect, three main approaches for skeleton-based virtual character animation can be distinguished in the literature, namely *kinematic*, *dynamic* and *motion-based*. They are presented in the following sections.

1.2.1 Kinematic approaches

The kinematic approaches take into account critical parameters related to the virtual character properties such as position (orientation) and velocity.

In kinematic manipulation techniques, one of the working solutions is to directly control the relative geometric transformation of each bone of the skeleton. This approach, also called Forward Kinematics (FK), is a very useful tool for the designer [Watt92] to manipulate the postures of virtual characters. In this method, critical parameters correspond to the geometric transformation applied to each bone of the skeleton.

An alternative approach is to fix the location in the world coordinates for a specific level of the skeleton, so-called *end-effector* (*e.g.* the hand for a human avatar), and to adjust accordingly the geometric transformation of its parents in the skeleton. With this method, also called Inverse Kinematics (IK), the critical parameters correspond to the geometric location of the end-effector.

Considering a skeleton-based virtual character, its posture is given by a set of parameters including the position and the orientation of each bone in the space. These parameters, associated with the set of joints (bones articulations) in the skeleton, form the so-called state vector of the skeleton.

Given a subset of interconnected joints in a virtual character skeleton and its associated state vector q, the forward kinematics approach consists in specifying q over the time. In this way, at each moment, the state of a bone in the chain can be computed. The state x (position and orientation, $x = (P_x, P_y, P_z, O_x, O_y, O_z)^T$) of the end-effector is then expressed as a function of the state vector q:

$$x = f(q). \tag{1.5}$$

The inverse kinematics issue consists in finding the joint state vector q if the end-effector is in a specific position and orientation, x:

$$q = f^{-1}(x). \tag{1.6}$$

Using IK to change the posture of a virtual character has the following advantages:

- for the designer, it represents a powerful manipulation tool, reducing the time to set key postures;
- with respect to the animation transmission, the set of target position of the end-effector is more compact than the geometric transformations for all the components of the link.

As solving the IK problem consists in solving a non-linear equations system, in most cases being a redundant system, the task is not easy and the solution is not unique. The various methods reported in the literature can be categorised into *analytical* and *iterative* methods. However, methods combining the two approaches have been reported for solving the IK problem in the case of a human-like arm and leg [Tolani00]. An overview of the major approaches is presented hereunder:

i) **Analytical method**. When the number of joints is reduced, the non-linear equation system can be directly solved. The method has first been proposed in robotics [Paul81, Craig89] and then adopted in computer graphics [Korein85, Tolani00]. Its limitation lies in the fact that for a particular articulated structure, there may be no analytical solution.

ii) **Iterative methods**. Inspired by the Newton-Raphson method for the resolution of non-linear equations [Ortega70], the linearization around an initial configuration has been proposed [Whitney69]. Iterative methods solve the inverse kinematics problem by using a sequence of steps leading by increment to a better solution for the joint angles. The goal is to minimize the difference between the current and the desired position of the end-effector. In this category, the most developed approaches rely on Jacobian matrix as well as on optimization and minimization methods. On the one hand, pseudo-inverse and transpose of Jacobian have been investigated. On the other hand, gradient-based, genetic algorithm and cyclic coordinate descend approaches have been proposed. Let us present each method and comment on their advantages and limitations.

i) *Jacobian inverse*

The Jacobian J is the multidimensional extension of the single variable differentiation [Watt92]. There is a relationship between the Cartesian space of the end-effector x and the joint space of the joint angles q: the Jacobian transforms the differential angle variations to differential motions of the end-effector [McKerrow91]:

$$\dot{x} = J(q)\dot{q} \,. \tag{1.7}$$

The vector $\dot{x}$ contains the linear and rotational velocity components of the end-effector $\dot{x} = (dP_x, dP_y, dP_z, dO_x, dO_y, dO_z)^T$ and $\dot{q}$ represents the time derivative of the state vector (rotational velocity for each joint). In order to obtain $\dot{q}$, the following operation should be performed:

$$\dot{q} = J^{-1}(q)\dot{x} \,. \tag{1.8}$$

In a simple representation, the Jacobian columns (a column corresponds to one joint) are the partial derivative of the end-effector with respect to each joint as follows:

$$J = [J_1 J_2 ... J_n], where\ J_i = \begin{bmatrix} \partial P_x / \partial q_i \\ \partial P_y / \partial q_i \\ \partial P_z / \partial q_i \\ \partial O_x / \partial q_i \\ \partial O_y / \partial q_i \\ \partial O_z / \partial q_i \end{bmatrix}. \quad (1.9)$$

Performance measures for various methods applied to obtain the Jacobian can be found in [Orin84]. In order to solve Equation (1.8), the inverse of the Jacobian is required. [Maciejewski90] outlined the following problems when inverting the Jacobian: (1) usually the Jacobian is not square, (2) singularities may occur (Jacobian determinant equals 0), and (3) an infinite number of solutions for the same end-effector configuration are possible (the Jacobian is degenerated). In the first case, the Jacobian pseudoinverse provides the best approximation which gives a least squares solution of minimum norm. [Golub65] defined the pseudoinverse using Singular Value Decompostion (SVD). Other methods to define the pseudoinverse can be found in [Rao71]'s comprehensive book. The singularity problem has been solved by [Nakamura86] using the so-called "singularity robust inverse".

ii) Jacobian transpose

To overcome the computation cost of Jacobian inverse, [Wolovich84, Welmen93 and Siciliano96] propose to use the transpose of the Jacobian instead of its inverse. The idea is based on the principle of virtual works and generalised forces [Paul81]. The virtual work can be enunciated respectively

- as the product between the force F acting on the end-effector and the distance it is displaced,

- or as the product between angular displacement of a joint and the torque τ affecting the joint:

$$F\dot{x} = \tau\dot{q}\ . \quad (1.10)$$

By transposing (1.10) and substituting $\dot{x} = J(q)\dot{q}$, we obtain:

$$\tau = J^T F\ . \quad (1.11)$$

Equation (1.11) is used to define a new control law to find an iterative solution of Equation (1.5):

$$\dot{q} = J^T(q)K(x_d - x), \quad (1.12)$$

where $x_d - x$ is the error, *i.e.* the difference between the desired position x_d and the current position x . K is a positive defined weight matrix. The advantages of the described algorithm are that no numerical singularity is met, since only forward kinematics is used, and the algorithm is computationally efficient in comparison to a

numerical inversion of the Jacobian. However, because of its poor convergence properties, especially near a solution, the number of steps required to reach the solution may be much higher than with pseudoinverse-based techniques.

iii) *Optimization and Minimization Methods*

Optimization is a well-known field in numerical mathematics, widely reported in the literature [Walsh75]. In the framework of IK, the difference between the goal position of the end-effector and its current state, expressed as a function of joints, can be considered as the measure to be minimized:

$$E(q) = (P - x(q))^2 . \tag{1.13}$$

In the computer graphics field, [Zhao89] and [Zhao94] report the use of an optimization method, [Goldfarb 69], for the manipulation of an articulated figure: a non-linear function (describing the degree of satisfaction of all constraints) is minimized under a set of linear equality and inequality constraints describing joint limits. This method is applied to manipulate a human-like articulated figure [Phillips90]. Genetic algorithms have recently been used to tackle general optimization problems, and they have also been applied to inverse kinematics [Khwaja98]. Generally, genetic programming-based methods are very slow and finding an adequate solution cannot be guaranteed (in a reasonable time). However, this method is useful for more complex and independent motion controls as well as animations (*e.g.* walk, jump).

iv) Cyclic Coordinate Descent

The Cyclic Coordinate Descent (CCD) method is a special case in the family of minimization methods. Its principle consists in applying the *E(q)* minimization (1.13) to each joint separately [Eberly01, Welman93], in successive order, from the most distant segment to the base segment. The difference between this method and the previous one is that only one joint variable is modified at each step. Following the fact that only one joint variable is changing along the minimization process, an analytic solution could be used. The interesting features of the CCD method are that there is no need for matrix inversion and it is free of singularities. In most cases, only a few iterations are enough to achieve a sufficient precision and therefore this method could be used in real-time applications.

The relatively small number of parameters used with the IK approach makes it an appropriate technique to be considered when a low bit-rate animation is targeted. We will address this issue in Chapter 2 by evaluating an IK method operating with the standardized FBA animation parameter set and in Chapter 3 by including in the generic developed framework (node definition and animation parameters) the IK-related information.

1.2.2 Dynamic approaches

Dynamic approaches refer to physical properties of the 3D virtual object, such as mass or inertia, and specify how the external and internal forces interact with the object. Such physics-based attributes have been introduced since 1985 in the case of virtual human-like models [Armstrong85, WilhelmS85]. Extensive studies [Badler95, Boston98] on human-like virtual actor dynamics and control models for specific motions (walking, running, jumping, etc.) [Pandy90, Pandy99 and Wooten98] have been carried out. Recently, Faloutsos *et al.* [Faloutsos01] proposed a framework making it possible to exchange controllers (*i.e.* a set of parameters) to drive a dynamic simulation of the character. The controller evolution is obtained by using the goals of the animation as an objective function. The results are physically plausible motions. Even if some positive steps have been achieved for specific motions, to simulate dynamically articulated characters displaying a wide range of motor-skills is still a challenging issue.

1.2.3 Motion-based approaches

Computer animation of virtual characters and interactive virtual environments require that a source generating a motion be available. The first animation technique, called animation from observation, has been copied from cartoons animation studios: a trained animator draws the draft of the character in the key postures and some other animators "interpolate" in-between. When adopting such an animation scheme in the case of a 3D virtual character, the menial work is left to the computer. Indeed, the key-frame-based interpolation technique is a wide-spread method in 3D applications and fully computerized. Furthermore, the creation of the key-frames may also be assisted by computers, a lot of progress was achieved in the areas of inverse kinematics and procedural motion. The major limitation, however, lies in the fact that a lot of skill and much toil are needed in order to get realistic and convincing motions. To overcome this constraint, motion capture systems were developed in the late 70's. While the animation results are highly realistic, re-using the captured data (*e.g.* animating another character with a different morphometry, adding motion details, changing motion paths, etc.) requires additional editing procedures. The next two sections present key aspects in both motion generating approaches: *motion capture* and *motion editing* systems.

1.2.3.1 Motion capture systems

The motion capture technique [Menache00] consists in tracking and recording the position (and the orientation) of a set of markers placed on the surface of a real object. Usually, the markers are positioned at the joints. The markers positions, expressed in the world coordinate system, are then converted into a set of geometric transformations for each joint [Badler93, Hirose98, Molet99].

Motion capture technologies are generally classified into *active* and *passive* sensor-based capture according to the nature of the sensors used. With an active sensor-based system, the signals to be processed are transmitted by the sensors while, in a passive sensor-based system, they are acquired by light reflection on the sensors. With respect to the nature of the sensors, the active sensor-based systems can be one of the following: mechanic-, acoustic-, magnetic-, optic- and inertial-based.

One of the earliest methods, using active mechanical sensors [Faro] is a prosthetic system. This is a set of armatures attached all over the performer's body and connected with a series of rotation and linear encoders. Reading the status of all the encoders allows for the analysis of the performer's postures.

The so-called acoustic method [S20sd] is based on a set of sound transmitters attached to the performer's body. They are sequentially triggered to emit a signal and the distances between transmitters and receivers are computed from the time needed for the sound to reach the receivers. The 3D position of the transmitter, and implicitly of the performer's segment, is then computed by using triangulation procedures or phase information.

The most popular method for motion capture is based on magnetic fields [Ascension, Polhemus]. Such a system is made of one transmitter and several magnetic-sensitive receivers attached to the performer's body. The magnetic field intensity is measured by the receivers so that the location and orientation of each receiver are computed.

More complex active sensors are based on fiber optics. The principle consists in the measure of the light intensity passing through the flexed fiber optics. Such systems are usually used to equip data-gloves [Vpl89].

The last method using active sensors is based on inertial devices, such as accelerometers, small devices which measure the acceleration of the body part they are attached to [Aminian98].

When using active sensors, the performer is burdened with a lot of cables, limiting his motion freedom. In this context, the recent development of motion capture systems using wireless communication are very promising [Ascension, Polhemus].

The second class of motion capture techniques uses passive sensors. One camera, coupled to a set of mirrors properly oriented, or several cameras allow for the 3D posture reconstruction from these multiple 2D views. To reduce the complexity of the analysis, markers (light reflective or LEDs) are attached to the performer's body. The markers are detected on each camera view and the 3D position of each marker is computed. However, occlusions due to the performer's motions may get in the way. Additional cameras are generally used in order to reduce the loss of information and ambiguities.

Since 1995, computer vision-based motion capture has become an increasingly challenging issue when dealing with the tracking, posture computation and gesture recognition problems in the framework of human motion capture. Several analysis methods have been reported [Moeslund01] but their limitations do not allow us to consider computer vision-based motion capture as a mature and accurate motion capture technique.

1.2.3.2 Motion editing systems

Motion editing systems can be split in two classes: 1) the tools that allow the designer to produce a motion from scratch and, 2) the tools for the clean-up of captured motion parameters.

The first class includes the usual methods to specify the position of objects at a very dense time sampling [Lutz20], thus creating the motion's continuity illusion. Such methods became highly

evolved with the development of the cartoons animation industry [Thomas81]. The technique, still used in 2D cartoons animation, has the drawbacks of being tedious as well as requiring special skills in order to create convincing motions by using such dense series of successive individual postures [Lasseter87]. With this method, computers are now used to reduce the production time and required efforts as the interpolation between key-frames and the computation of some joint values from a minimal set imposed by the designer (solving inverse kinematics problem) can be automated.

In the second class, existing motions are re-used to create new ones. From a library of motions, adaptation techniques have been proposed by using motion editing tools. Techniques from the signal processing field have been applied to manipulate animated motions. [Brudelin95] introduces a displacement mapping in order to affect an existent motion. A technique based on warping the animation parameter curves is presented in [Witkin95], where the designer creates the set of key-frames constraints, which are then used to derive a smooth deformation, thus preserving the structure of the captured motion. Interpolation and extrapolation techniques have been widely reported in the literature: in the frequency domain, Fourier analysis techniques are used [Unuma95], while in the spatial domain, either one-dimensional [Wiley97, Guo96] or multidimensional [Rose98] techniques are adopted. Many editing systems address the user-specified constraints problem [Witkin88].

In order to produce optimal motions, the non-linear optimization problem can be solved by employing optimal control techniques [Brotman88]. A space-time control system that allows the user to guide interactively a numerical optimization process to find an acceptable solution in a reasonable time has been developed [Cohen92]. Better performances for motion editing can be achieved through a simplification of the system by ignoring the physics-related aspects from the objective function and constraints [Gleicher97]. Furthermore, this technique has been applied to re-target motions to different virtual characters [Gleicher98]. Hierarchical wavelet representation is used to automatically add motion details [Liu94] and to generate smooth transition between motions [Rose96].

1.3 Conclusion

In this chapter, we have presented a brief overview of the 3D virtual character-based animation techniques. This overview, structured according to the two major components of an animation system, 1) *modelling* and 2) *animation*, showed the strong interconnection and application dependency between virtual character definition and animation tools. Core technologies being available to model and animate characters, real advances can be made towards the use of 3D virtual character-based animation in real life applications like the creation of communities populated by 3D citizens and immersion experiments as predicted in [Thalmann00]: *"Networking coupled with highly interactive technology of virtual worlds will dominate the world of computers and information technology"*.

One of the first complete *Virtual Reality* systems, called *Virtual Environment Operating Shell* (VEOS) [Bricken94], was developed by the University of Washington. An increasing number of *Virtual Reality* systems have almost at the same time been proposed for special application fields or

according to special architecture constraints [Division93, Carlsson93, Macedonia94] and MASSIVE [Greehalgh95]. VLNET [Capin97, Panzic97] is one of the firsts *Virtual Reality* systems providing realistic human representations but this system was not designed to take into account compression related issues appearing in distributed environment as well as in broadcast or Webcast environment, topic that will be elaborated in the next chapter.

Chapter Two

MPEG-4 Face and Body Animation: Specifications, Implementation and Performance Analysis

In the previous chapter, it was established, independently of any compression related matters, that seamless model are better suited for realistic animation tasks than segmented one. Taking into account the limited bandwidth capabilities available for transmission on most networks as well as and the desirability of a multimedia integrated framework, we focus, in this chapter, on the first attempts to standardize both animation and compression of human models.

Initial work was conducted by the VRML group for generic 3D models in 1997 [VRML]. Based on Interpolator node, the VRML approach allows animating a model between two key frames by interpolating, for example, the position of each vertex in between its original and final position. The specificity of human model animation was then investigated by the H-Anim Group [HAnim], which defined the human body as a number of *segments* connected to each others by *joints*. Animation of this segmented model could then be further refined by locally altering the positions of some vertices. This approach was then extended by the SNHC group within the MPEG-4 standardization process and the so-called Face and Body Animation (FBA) framework was adopted at the beginning of 1999.

Because the FBA specification is the only one providing both an animation and a compression framework and despite the fact that it relies on a segmented model, we have considered it as a starting point for our approach to build a virtual character communication system.

The first section of this chapter introduces the basic concepts used by the MPEG-4 standard to define a multimedia scene composed of 2D and 3D objects together with spatial and temoral behaviours. In this general framework, the FBA specific mechanism to define a virtual body in an MPEG-4 scene is introduced. Since the manners for obtaining the content are various and quikly evolve in time, the FBA specifications do not mandate any specific methods to obtain a real description of a 3D human body and its associated animation parameters. Defining only the representation format, the specifications allow a free developement of content creation. To address this first issue, we propose, in Section 2, a semi-automatic approach to get a MPEG-4 compliant body representation from a 3D mesh model of a human. This semi-automatic approach based on a segmentation algorithm is implemented as part of the Virtual Human Modelling (VHM) authoring tool. In order to address the second issue, we present in Section 3 a dedicated authoring tool named ARTEMIS Animation Avatar Interface (3AI), implemented to ease the editing and extraction of animation parameters. This authoring tool includes various interpolation schemes and one inverse kinematics method. The 3AI authoring tool also provides high level functionalities like (1) the composition of multimedia scene made of natural data (images, video) and synthetic body representation, and (2) the calibration of a 3D body model according to anthropometrical characteristics. Section 4 focuses on a practical and real evaluation of both the compression and the animation performances of the FBA framework. Using the two above-mentioned authoring tools, an FBA compliant avatar and several animation data sets corresponding to the American Sign Language alphabet were created. A complete server-client communications system was then set up to test the complete application from the encoding of the animation parameters, their transmission over an IP (Webcast) network to their decoding and the rendering of the associated human model. Results are presented and analysed in terms of trade off between quality of the animation and the required bit-rate. These results underline the performances as well as the limitations of the FBA framework.

2.1 Overview of the MPEG-4 FBA specifications

A key concept in the MPEG-4 standard is the definition of the scene, where text, 2D and 3D graphics, audio and video data can (co)exist and (inter)act. A scene is represented as a tree where each object in the scene is the instantiation of a node or a set of nodes. Compression representation of the scene is done through the Binary Format for Scene (BIFS) specification [ISOIEC01]. Special transformations and grouping capabilities of the scene make it possible to cope with spatial and temporal relationships between objects. In order to support face and body animation, two sets of dedicated nodes have been created: the Face node and the Body node.

The structure of the Face node allows the geometric representation of a face as a collection of meshes. Animations can be performed at a high level, using a standardized number of expressions and visemes as well as at a low level. In this case, a standardized number of key points (84), corresponding to the human features (e.g. middle point of upper lip) are defined on the face surface (Figure 2.1). The animation is performed by deforming the mesh in the key point neighbourhoods. Face animation with MPEG-4 has already been presented and discussed in detail [Doenges97, Escher98, Lavagetto99]. The animation of the human body within the MPEG-4 standard is less documented in the literature [Preda99, Preda01, Seo00] and will be addressed in the rest of this chapter.

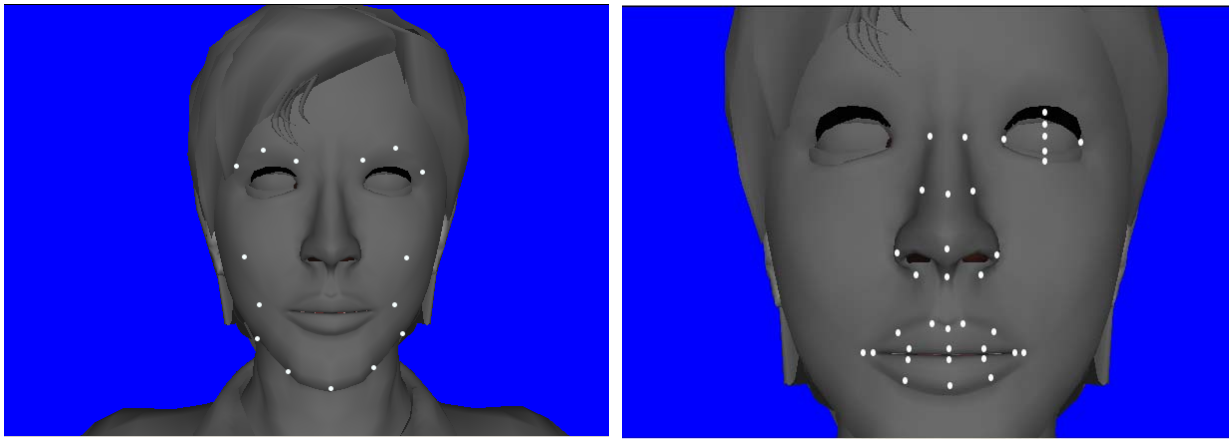


Figure 2.1. Examples of standardized key-points for the “face” object.

A virtual body object is represented in the scene graph as a collection of nodes (the two fundamental ones being BDP and BAP) grouped under the so-called Body node (Figure 2.2).

The BDP (Body Definition Parameters) node controls the intrinsic properties of each anatomical segment of the avatar body. It includes information related to the avatar body representation as a static object (bodySceneGraph node), deformation behaviour (bodyDefTables and bodySegmentConnectionHint nodes) [ISOIEC01],. BDP is actor-specific; hence the complete morphology of an actor can readily be altered by overriding the current BDP node. Geometrically, the static definition of the virtual body object is a hierarchical graph consisting of nodes associated with anatomical segments and edges (this representation could be compressed using the MPEG-4 3D Mesh coding algorithm [ISOIEC01]) defining subpart relationships, grouped under the bodySceneGraph node. The MPEG-4 virtual avatar is defined as a segmented virtual character, using the H-Anim V2.0

[HAnim] nodes and hierarchy: Humanoid, Joint, Segment, and Site nodes. For a complete scene graph structure of an H-Anim avatar one can refer to [HAnim].

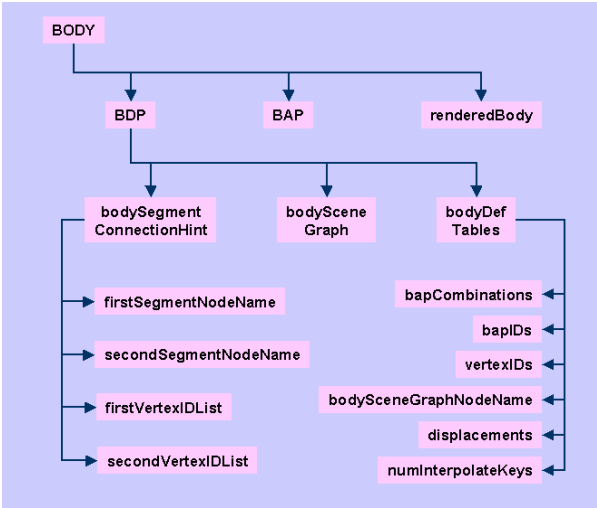
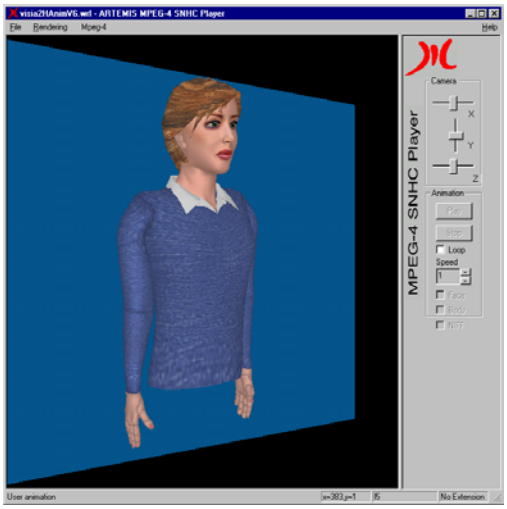
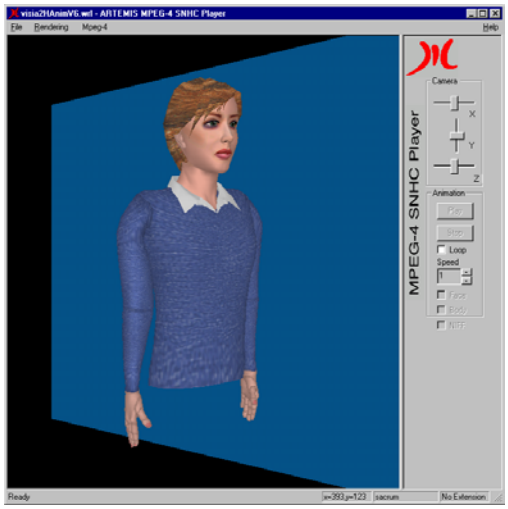


Figure 2.2. The Body scene graph hierarchy.

The BAP (Body Animation Parameters) node defines the animations parameters as extrinsic properties of an anatomical segment, *i.e.* its 3D pose with respect to a reference frame attached to the parent segment. The orientation of any anatomical segment is expressed as the composition of elementary rotations, namely twisting, abduction and flexion. Here, 296 angular joint values are enough to describe any 3D posture of a virtual avatar. The angular values are specified against the local 3D coordinate system of the anatomical segment. The origin of the local coordinate system is defined as the gravity centre of the joint contour common to the considered anatomical segment and its parent. The rotation planes are specified (Figure 2.3, Figure 2.4 and Figure 2.5) and anatomical segment rotation axes are standardized (Figure 2.6 and Figure 2.7). Contrary to BDPs, BAPs are meant to be generic, *i.e.* independent or poorly dependent of the avatar geometry.

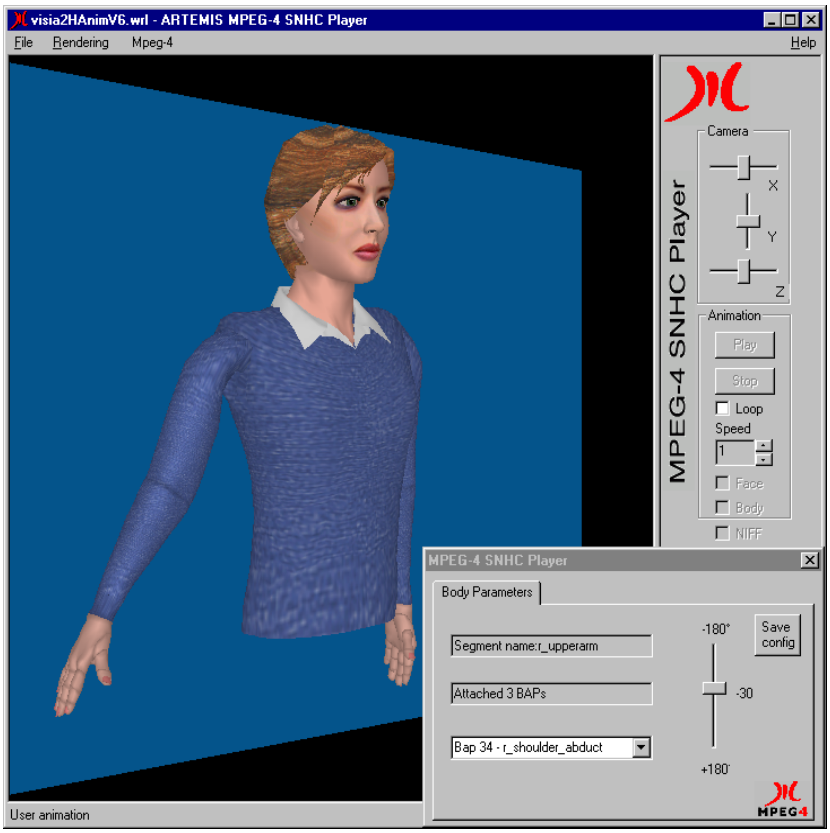


a) Coronal plane (CP).

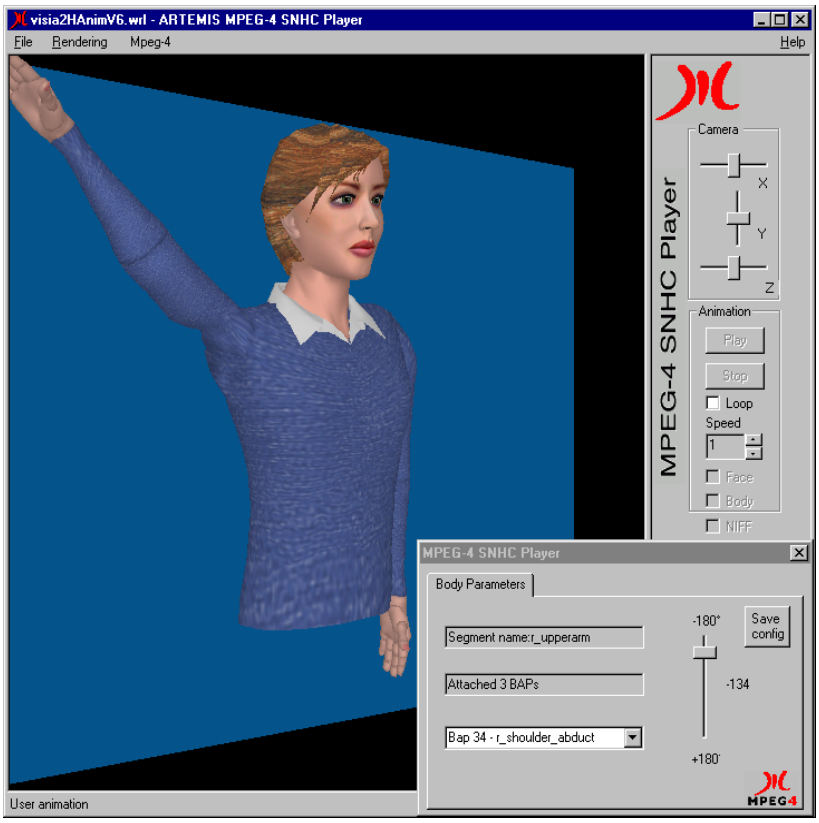


b) Sagittal plane (SP).

Figure 2.3. Standardized rotation planes for the avatar body.

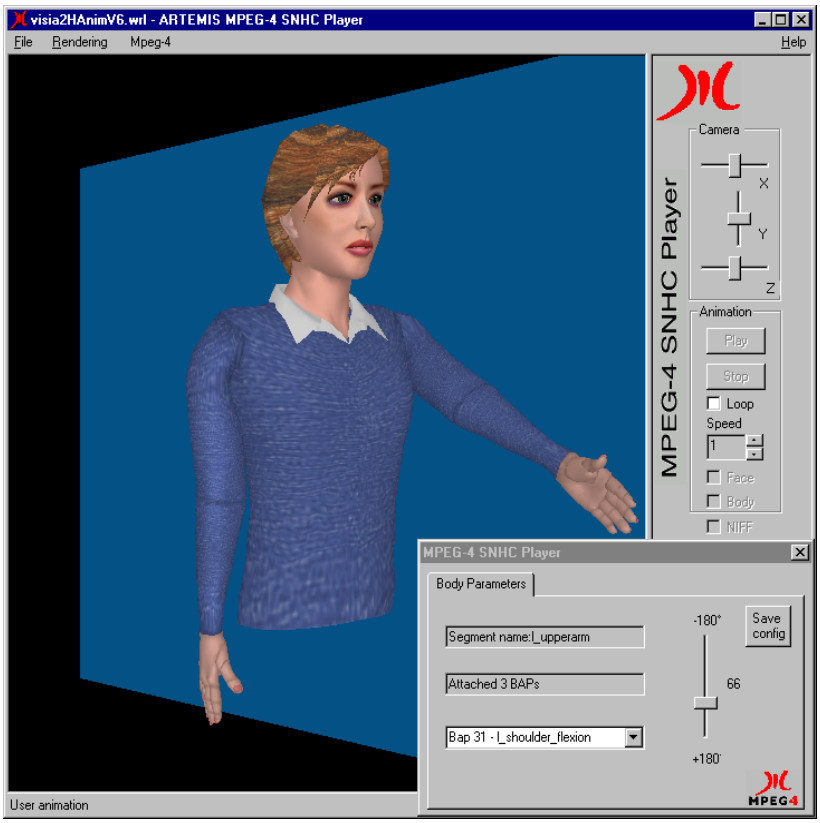


c) Right arm motion in CP (BAP #34 =-30°).

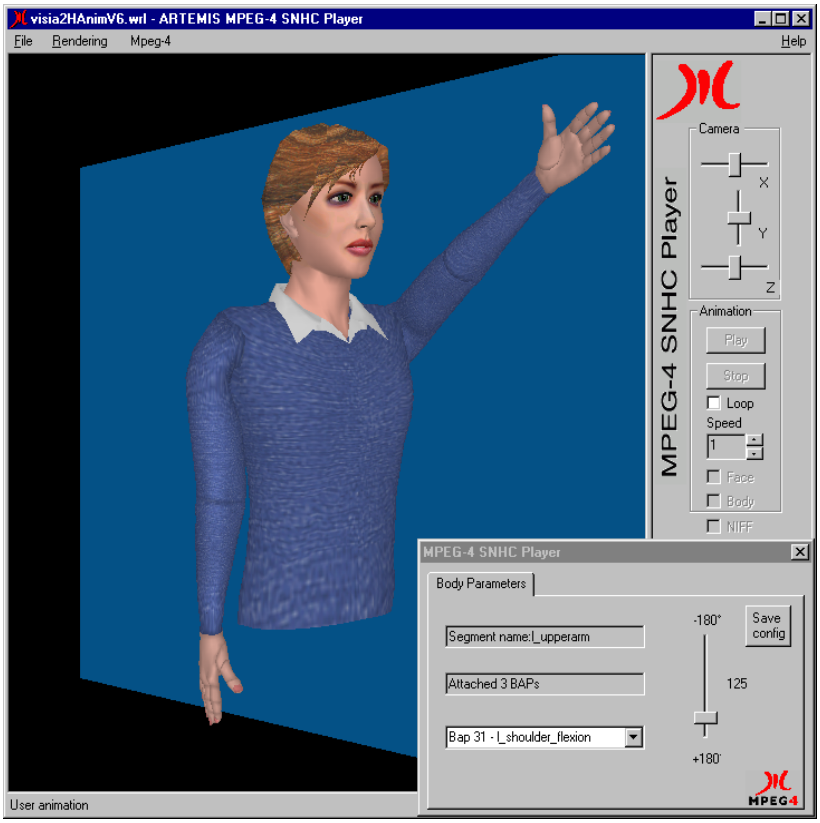


e) Right arm motion in CP (BAP #34 =-134°).

Figure 2.4. Abduction of the right arm.



d) Left arm motion in SP (BAP #31=66°).



f) Left arm motion in SP (BAP #31=125°).

Figure 2.5. Flexion of the left arm.

We illustrate in the following figures (Figure 2.6 and Figure 2.7) the rotation axes for arm, forearm and fingers. For a complete definition of the axes associated to all body segments, one can refer to [ISOIEC01].

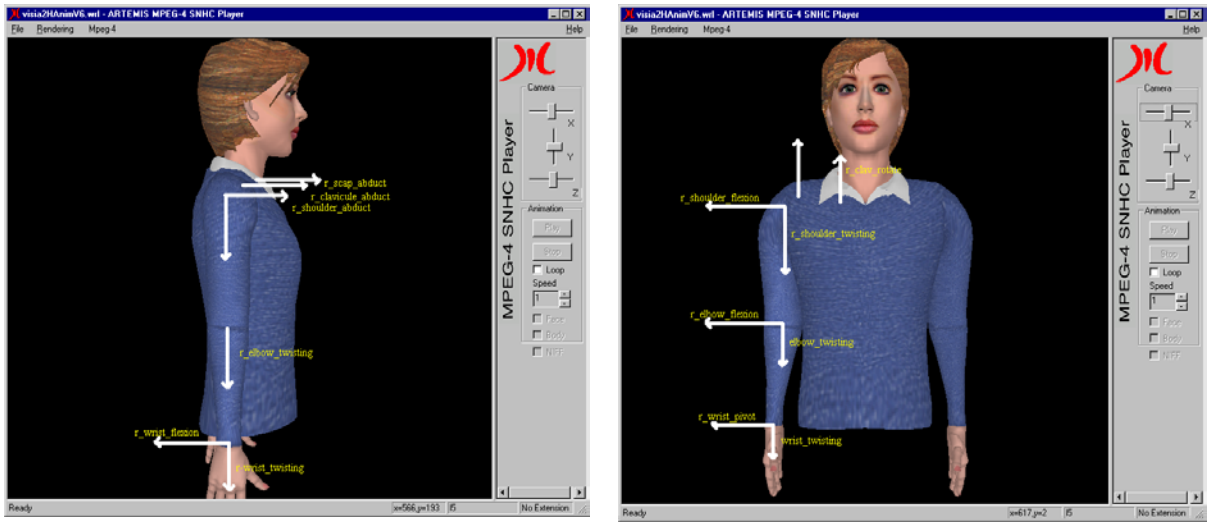


Figure 2.6. Standardized rotation axes attached to shoulder, elbow and wrist.

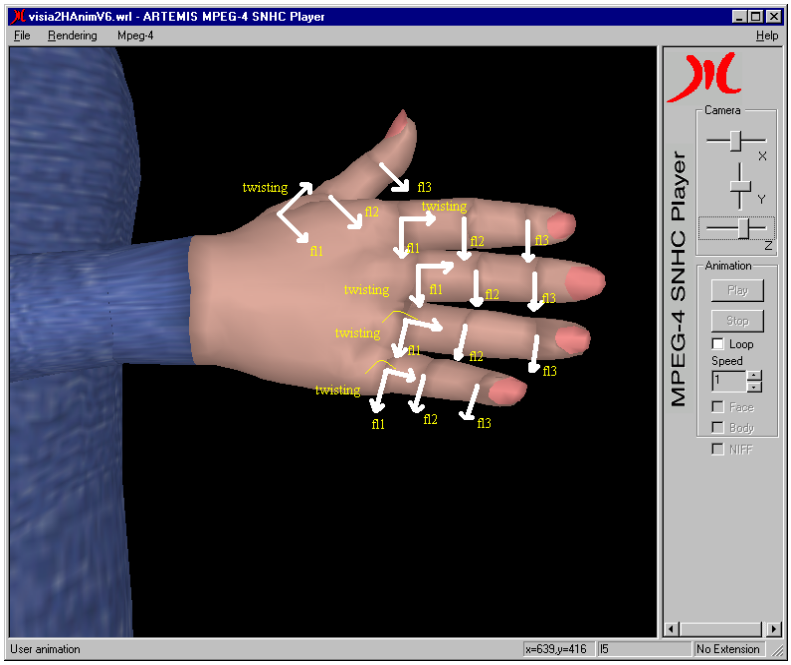


Figure 2.7. Standardized rotation axes attached to fingers.

The FBA specifications provide two encoding methods (predictive and DCT-based). In the first method, BAPs are coded with a predictive coding scheme. For each parameter to be coded in frame *n*, the decoded value of this parameter in frame *n-1* is used as a prediction. Depending on precision requirements on BAPs, different quantization step sizes could be applied. They consist of a local (BAP specific) step size and a global one (used for bit-rate control). The quantized prediction error is then encoded by arithmetic coding. Taking into account the fact that natural human motions are constrained

by physical properties, the range of each BAP is limited to a specific interval. Using this property, the coding efficiency is increased. The block diagram of this method is illustrated in Figure 2.8.

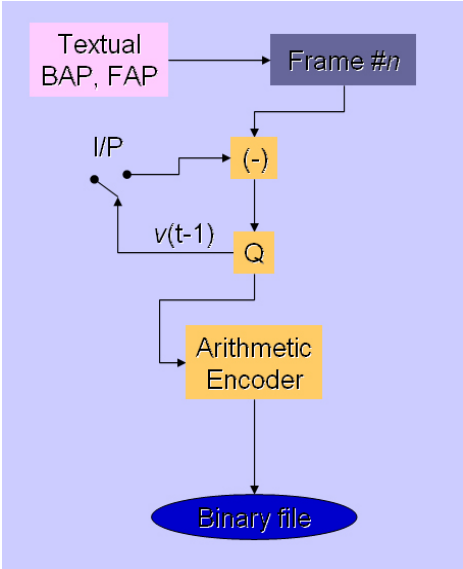


Figure 2.8. Frame predictive-based method block diagram.

The DCT-based coding method splits BAP time sequences into BAP segments made of 16 consecutive BAP frames. Three steps are necessary to encode each BAP segment: (1) determination of the 16 coefficient values using discrete cosine transform (DCT), (2) quantizing and coding the AC coefficients and (3) predictively coding and quantizing the DC coefficients. The global quantization step Q for the DC coefficients can be controlled and the AC coefficients quantization step is set to 1/3 from Q. The continuous component coefficient (DC) of an intra coded segment is encoded as it is and, for an inter-coded segment, the DC coefficient of the previous segment is used as a prediction of the current DC coefficient. The prediction error and alternative component coefficients (AC), (for both inter and intra coded segments), are coded using Huffman tables. The block diagram of the DCT-based method is illustrated Figure 2.9.

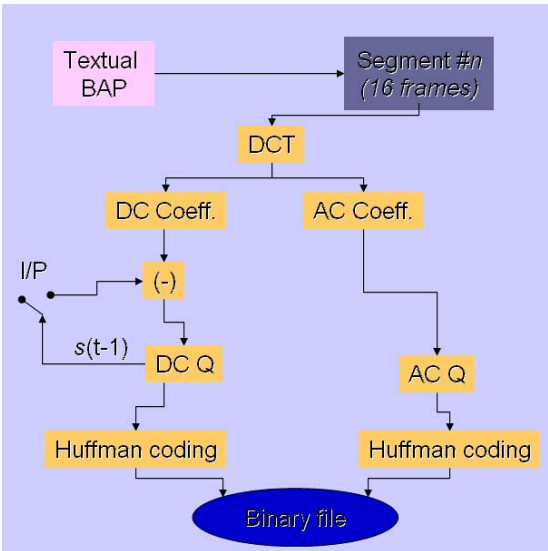
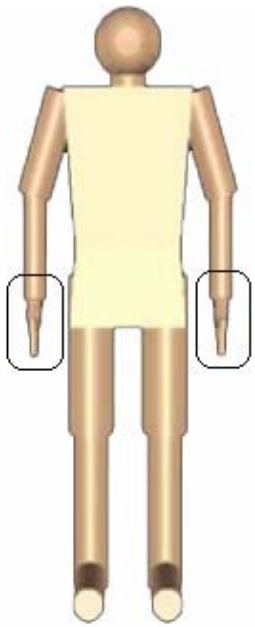


Figure 2.9. DCT-based method block diagram.

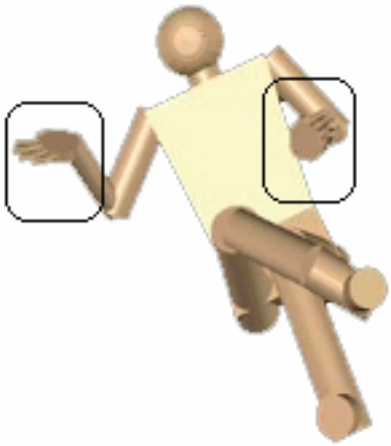
An important concept to define the complete compressed representation of an animated avatar is to split into separate streams the representation of the avatar as a static object and the animation of the avatar. The goals of such an approach are (1) to avoid to transmit the avatar geometry when the bandwidth is limited and (2) to be able to re-use the same animation content for different avatars. Typically, for a sign language communication system, it could be expected that children would prefer cartoon-like avatars and adults more realistic and "serious" looking characters.

Figure 2.10 a) and b) shows different initial position for an avatar (details are shown on the wrist orientation level). When the same set of parameters is applied to both avatars, different effects are obtained as shown in Figure 2.10 c) and d). In order to ensure a correct and consistent interpretation of the animation set on different avatars, the standard specifies the initial position of the character to be the one depicted in Figure 2.10 a).



a) Michael Miller's "solder" avatar.

b) Cindy Reed-Ballreich's "Nancy" avatar.



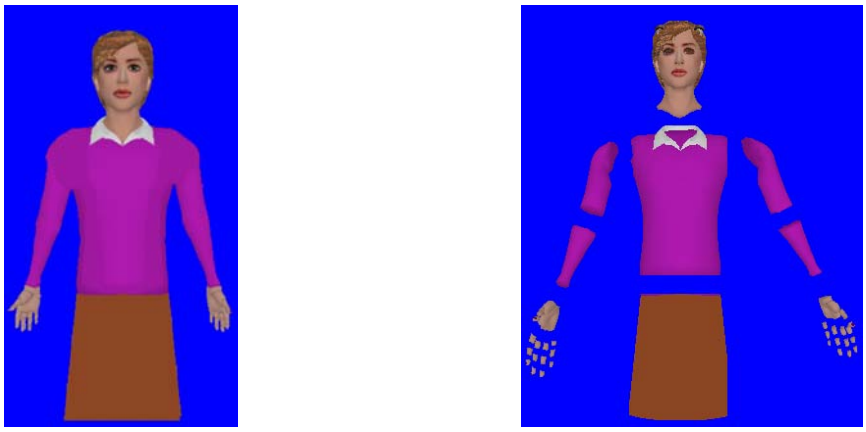
c)

d)

Figure 2.10. Avatar initial position effects on the animation result. a) correct initial position, b) incorrect initial position, c) and d) results obtained from the same animation parameter set.

2.2 Creating an FBA compliant avatar from a static 3D model

The FBA specifications do not mandate any specific methods to get a real description of a 3D human body. Two different courses can be chosen from in order to build a virtual character depending on how cartoon-like or realistic this virtual character must be as well as depending on the technology available to the designer. On the one hand the designer can build interactively the model anatomical segments and set up the model hierarchy. In this method, describing the information about anatomical segments and their hierarchy is part of designing the virtual character. The main drawback is that the final result is strongly dependent on the designer's artistic skills and talent. On the other hand a faster and proven method is to use 3D scanners to acquire the 3D model. In this case, as the scanned data are raw geometric data without semantic information about the anatomical segments, getting an articulated version compliant with the FBA specifications, requires segmenting the original 3D mesh model into separate set of 3D object with anatomical meaning and from them on to build an FBA compliant articulated model. In the rest of this section we propose first a segmentation approach and then a particular implementation of the complete procedure within the Virtual Human Modelling authoring tool (VHM).



a) b)

Figure 2.11. a) Seamless body. b) FBA Body object: a hierarchical segmented character.

2.2.1 3D model segmentation

Splitting a VRML model (and in general an *IndexedFaceSet*-based model) into anatomical subparts can be performed by a supervised polygonal mesh propagation algorithm. Automatic and semi-automatic segmentation procedures of a human like model into anatomical segments have been reported in the literature. The earlier works were aiming at the extraction of body measurements needed for clothes' mass production. Pargas *et al.* developed a software package where sliced body scan data can be edited and the body measurements are then manually extracted [Pargas97]. In the attempts to automate measurements extractions, the body landmarks were only roughly positioned. Therefore, the error margin on these measurements was significant. For the same application, Jones *et al.* focused on the torso part of the body [Jones95]. A set of cross section slices at the level of the key anatomical landmarks on the torso is manually selected. These slices are fitted and then interpolated to generate a NURBS approximation. This technique is a good trade-off between data size and surface detail representation. The approach remains particularly suited to clothes' design, however, the

process needs a lot of manual operations even though some attempts were made to separate automatically the torso from the upper part of the arms [Li97].

Nurre considered the automatic segmentation of the human body into its functional parts [Nurre97]. He approximated the body structure by a six-stick template representing the head, the two arms, the two legs and the torso. The goal was to segment the body into six segments corresponding to these parts. The approach combines global shape description, namely moment analysis and local criteria of proximity, which are derived from the pre-knowledge of the relative body parts positioned in a standard posture (standing body with arms held at the sides). The range data is organised into slices of data points. The horizontal slices are stacked vertically and the data points are assigned to the different body parts according to the slice's topology and its position in the body, e.g. a slice having two separated closed curves must represent points measured at the level of the legs. A slice consisting of three closed curves must belong to the torso and arms area, a slice with two joined closed curves is assumed to correspond to the transition between the legs to the torso (at the level of the groin). Dekker *et al.* improved Nurre's work, with some efforts to enhance the localization of the key landmarks of the human body [Dekker98]. For instance they differentiated the slices circumference in the torso area to locate the waist position. Additionally, in a recent work [Douros99], they improved the model of the torso using a B-spline approximation. Ju *et. al.* [Ju00] proposed an automatic method, first slicing the model transversally (parallel to the horizontal plane) then segmenting the body surface by using the contour information of the slices. While the designer's intervention is not necessary, the method has its limitations as the model must be in a dedicated pose and all the joint planes obtained are horizontal. Typically, the joint between the arm and shoulder is not correctly identified.

In order to overcome such limitations, we now propose a semi-automatic procedure, giving, with minimal intervention from the designer, an acceptable segmentation. The different steps of this procedure are illustrated in Figure 2.12, Figure 2.13 and Figure 2.14. The principle is to create vertex and face adjacency lists of the sub-meshes associated with anatomical segments. The algorithm follows 3 steps. The first one, the initialization step, aims to create interactively a sorted list of mesh vertices, the so-called Joint Location Control Vertices (JLCVs). The designer has to select between 3 and 7 vertices on the mesh, at each joint level (Figure 2.12. a, b, and Figure 2.13 a, b). The second step generates automatically the related joint vertices from the JLCVs, making up the so-called Joint Contour (JC). A 3D propagation-based method including angular constraints is applied to any couple of two successive JLCVs, (v_i, v_j), (the last JLCV is coupled with the first one). For an arbitrary vertex v_i, we call its first order neighbourhood the set of all vertices which are directly connected to v_i. For each vertex v belonging to the first order neighbourhood of v_i, the angle (vv_iv_j) is computed. The vertex v_{min} that minimizes the angular measure is selected. The procedure is iterated by replacing v_i by v_{min} and stops as soon as there exists v (one order neighbour of the current v_i) such that $v= v_j$. The set of selected vertices will form the connection line (CL) between the initial v_i and v_j. By concatenating all the CLs, we obtain the JC associated with a joint (Figure 2.12. d and Figure 2.13 d). The last stage constructs the anatomical segment related to two adjacent JCs. The same propagation procedure is applied to an arbitrary pair of vertices v_a and v_b, each one belonging to a JC (Figure 2.14 a). An arbitrary vertex v_k belonging to the CL obtained with respect to v_a and v_b is selected (Figure 2.14 b). A 3D geodesic reconstruction (iterative elementary geodesic dilation) is applied to v_k with respect to the

mesh surface limited by the two JCs (Figure 2.14 c). Finally, the anatomical segment is the sub-mesh made of the two JCs and the reconstructed component from v_k (Figure 2.14 d).

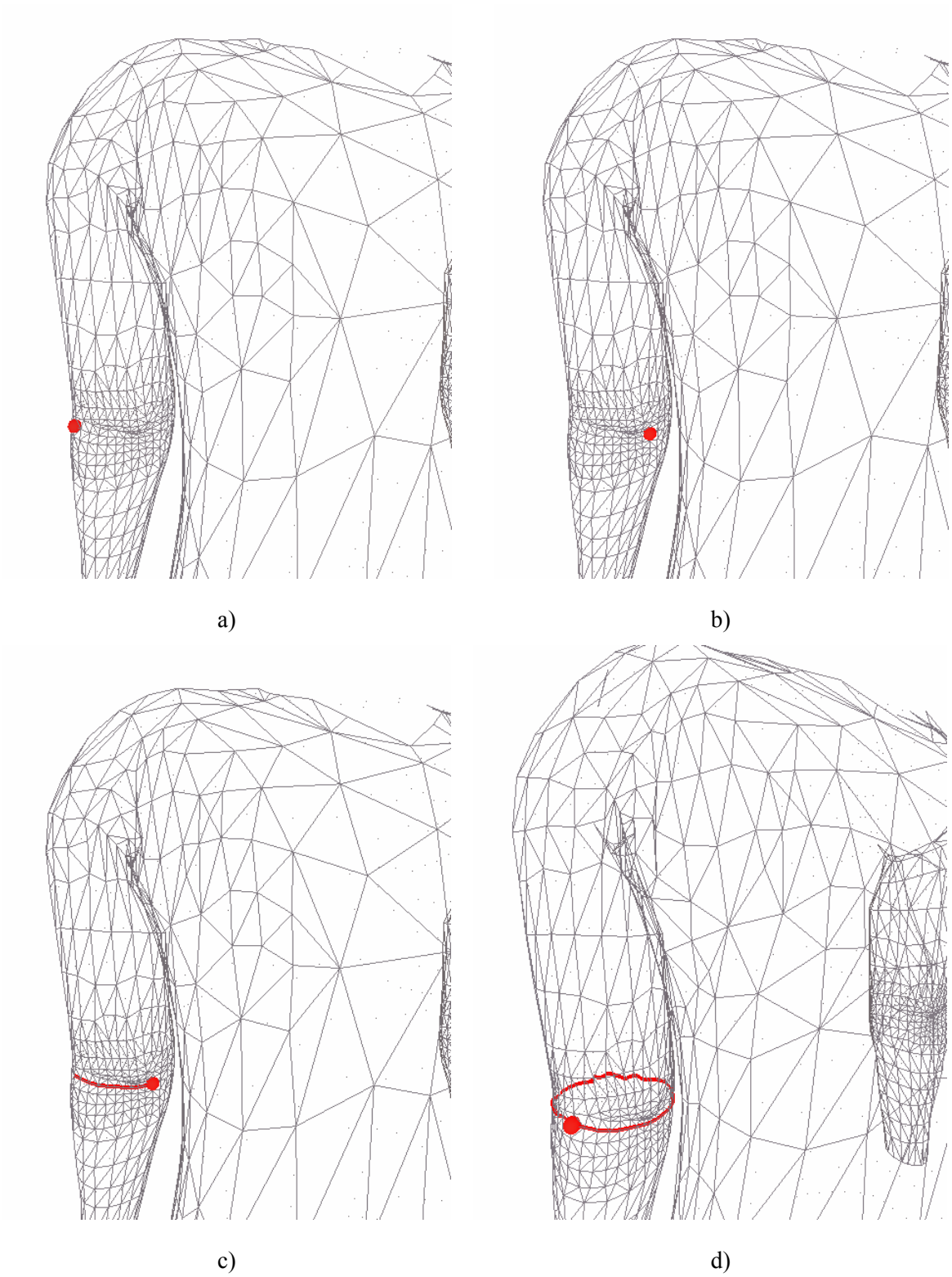


Figure 2.12. Segmenting upper arm of the 3D model: selecting first JC.
a) first point of the JC, b) second point of the JC,
c) automatically generated segment between the two points, d) automatically generated JC.

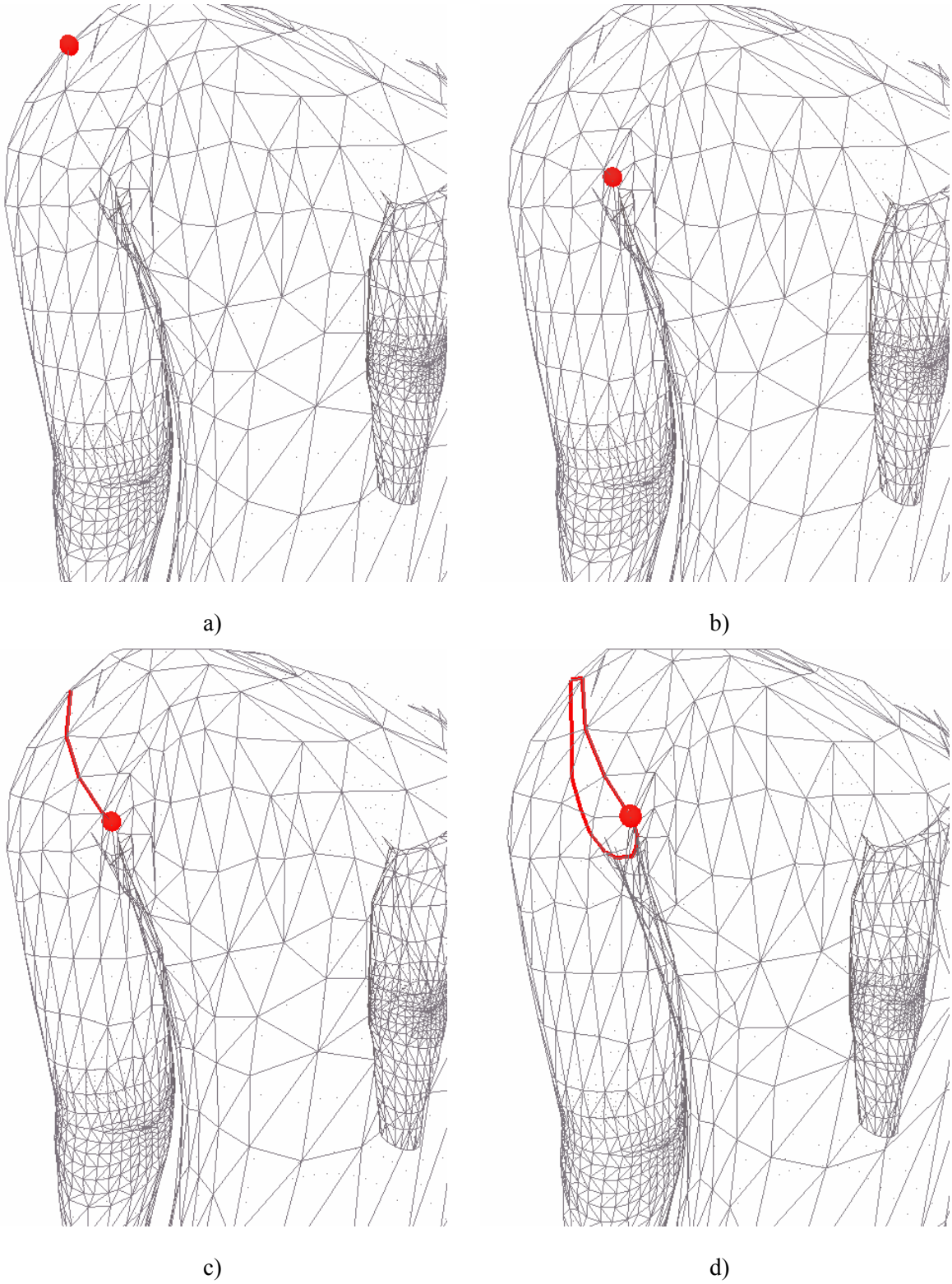


Figure 2.13. Segmenting upper arm of the 3D model: selecting the second JC.
a) first point of the JC, b) second point of the JC,
c) automatically generated segment between the two points, d) automatically generated JC.

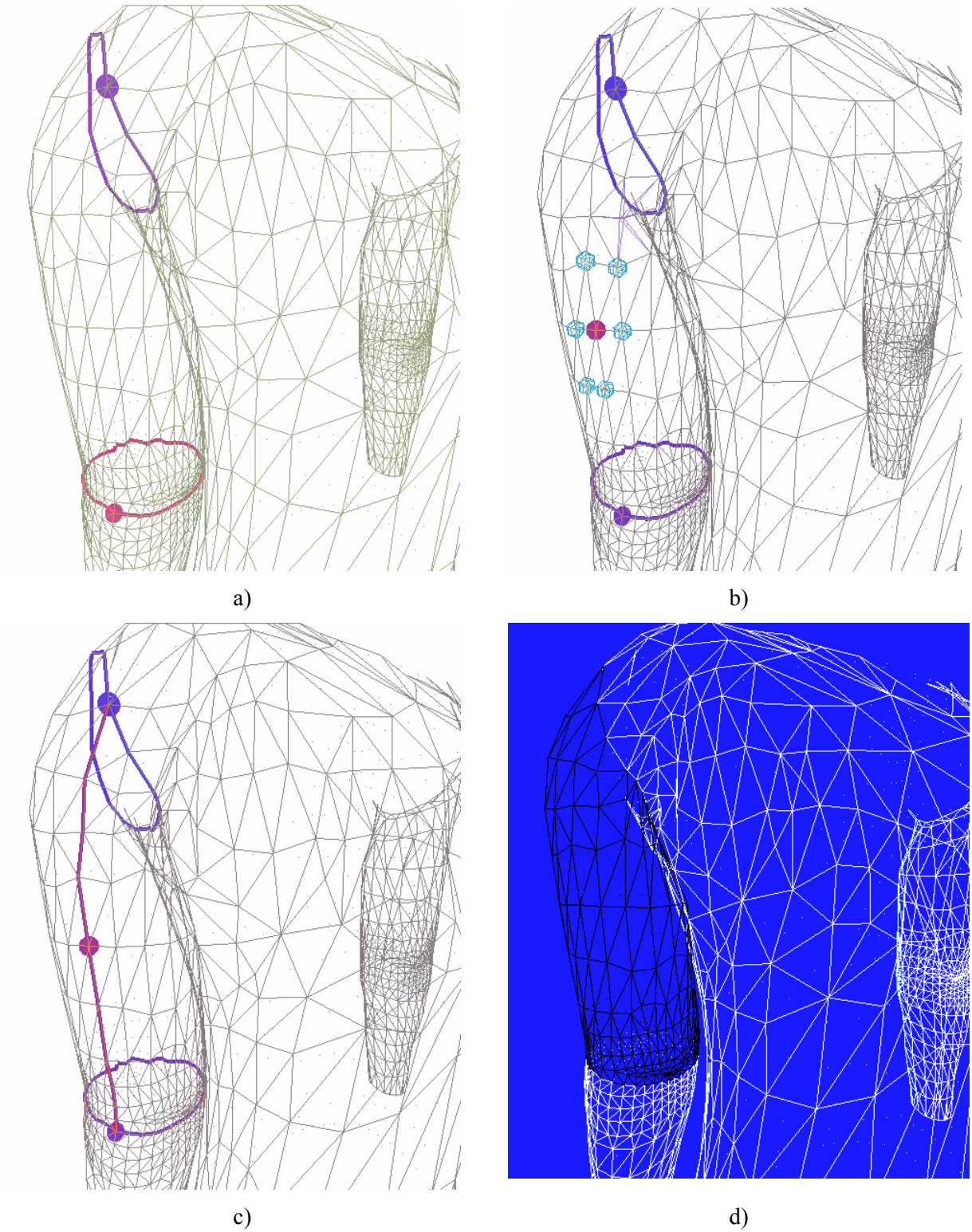


Figure 2.14. Segmenting upper arm of the 3D model: geodesic dilation.
a) two JC, b) random initialization between the two JC, c) geodesic dilatation, d) segmentation result.

2.2.2 Virtual Human Modelling authoring tool

The goal of the Virtual Human Modelling (VHM) authoring tool (available on Windows platforms) is to help a designer to obtain, from a scanned geometric model of a human like avatar, an articulated version, compliant with the FBA specifications.

The authoring tool is made of two parts:

- a *Segmentation Module* to split the original object into a set of 3D objects using the above-mentioned segmentation algorithm. With the *Segmentation Module* presented in Figure 2.15, the designer selects for each anatomical segment a set of points on the two segment joints (the Joint Contour are called "borders" in Figure 2.15) and the module extracts the 3D object between the two joints to obtain an anatomical segment. Then the designer can save the 3D object corresponding to the anatomical segment into an indexed data base.

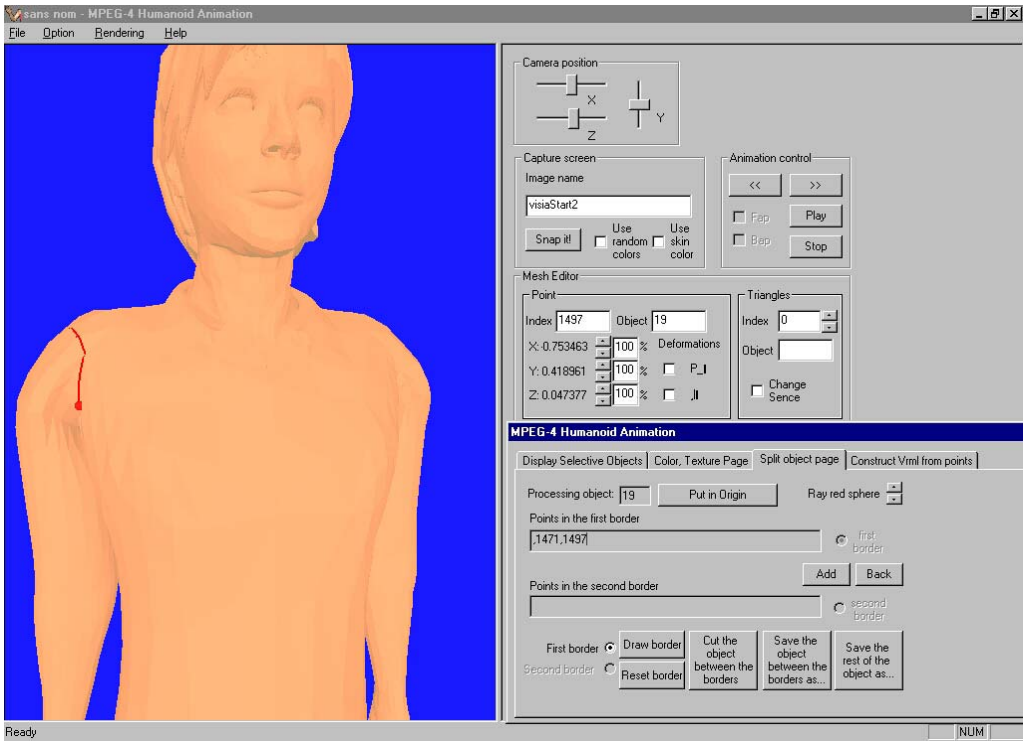


Figure 2.15. Virtual Human Modelling Authoring Tool. The Segmentation Module control dialog.

- a *Building Module* to build the articulated character by using the FBA predefined hierarchy. Once all the 3D objects associated with the anatomical segments are available, the *Building Module* is used to set up the parent-child property of the anatomical segments. The user interaction is simple and intuitive (Figure 2.16): the module loads from the indexed database all the anatomical segments and associates them to the FBA standard segments. In Figure 2.16, the active segment is illustrated in mesh representation and the rest of the body in dots. In green there are illustrated the rotation centers for each anatomical segment. The application also proposes, for each anatomical segment, its related joints and the designer has to select the joint

between the current segment and its parent. In order to allow for complex meshes (several 3D objects composing the same anatomical segment) the application has an option to load a so-called "extended segment". The name of such a segment is not specified in the FBA standard and it will be attached (with respect to its geometric transformation) to a standard segment. Finally, the virtual humanoid can be saved as an FBA compliant avatar.

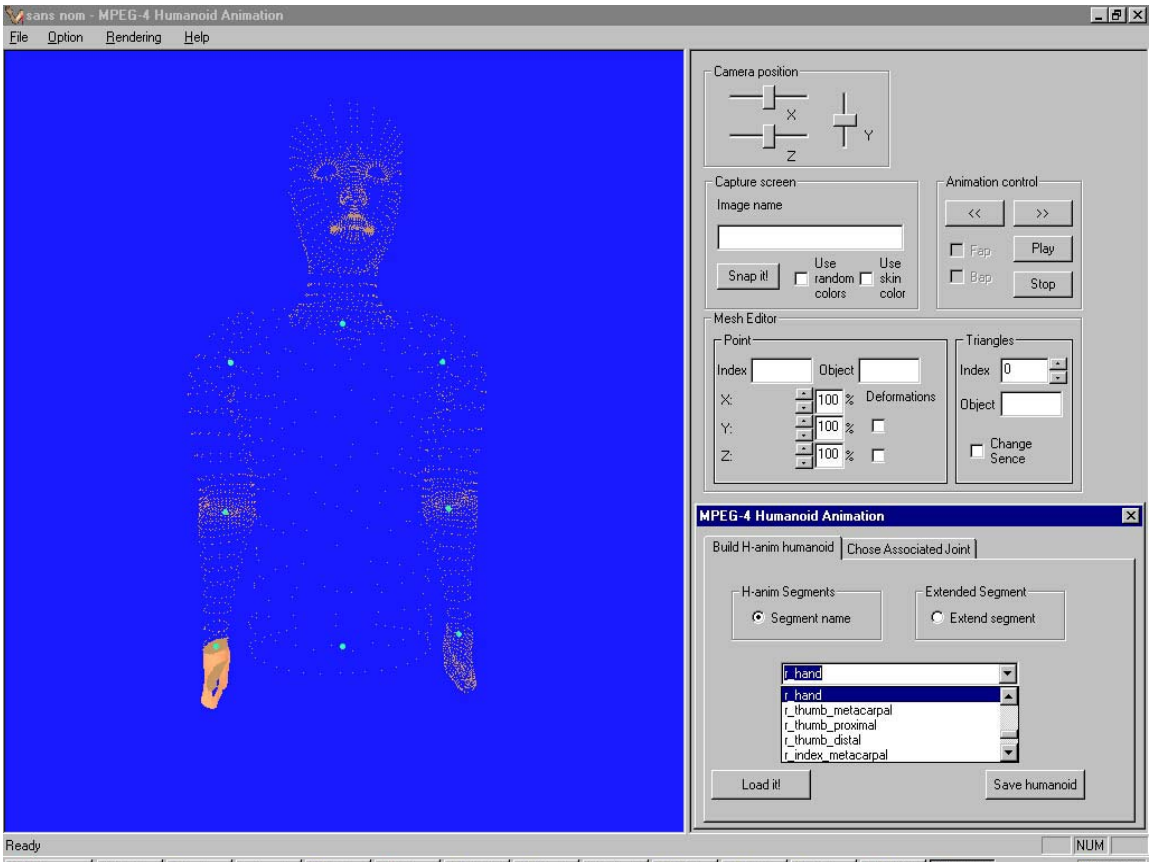


Figure 2.16. Virtual Human Modelling Authoring Tool. The Building Module control dialog.

2.3 BAPs editor: the Artemis Animation Avatar Interface

The Artemis Animation Avatar Interface (3AI) is a C++ user-friendly interface, available on X11/Motif and Windows (Figure 2.17) platforms.

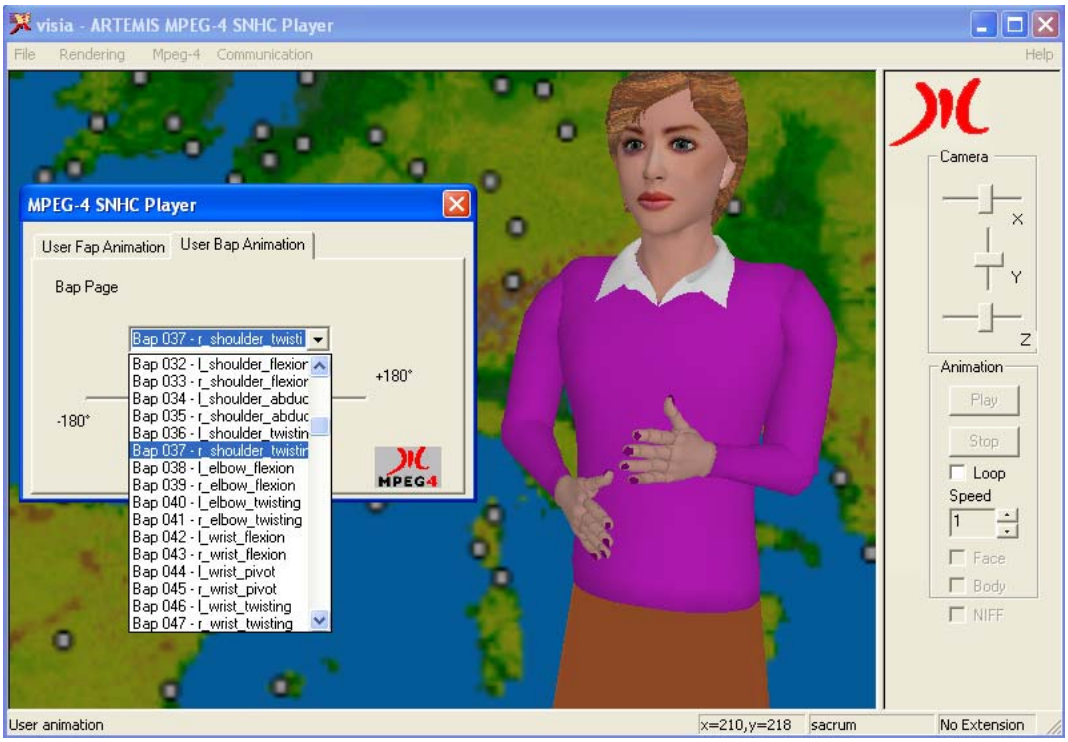


Figure 2.17. The ARTEMIS Avatar Animation Interface with the forward kinematics BAP editing control dialog.

As basic functions of the interface are:

- loading an FBA compliant avatar;
- 3D composition of objects such as images, video sequences, human body models or anatomical part models (hand, arm), and 3D scenes;
- calibration of a 3D body model according to the anthropometric characteristics of the actor in a video sequence (dimensions of the palm, length of the fingers ...);
- interactive extraction of BAPs to describe any posture or corresponding to the posture shown in the video sequence (see Figure 2.18). This feature of the 3AI authoring tool has been used to generate the MPEG-4 BAPs data set for the alphabet letters used in American Sign Language (ASL) [Prêteux98].
- BAPs editing through selection of key-frames and end-effector positioning;
- animation of the virtual human model according to a BAPs file source and network resource (*e.g.* UDP server).

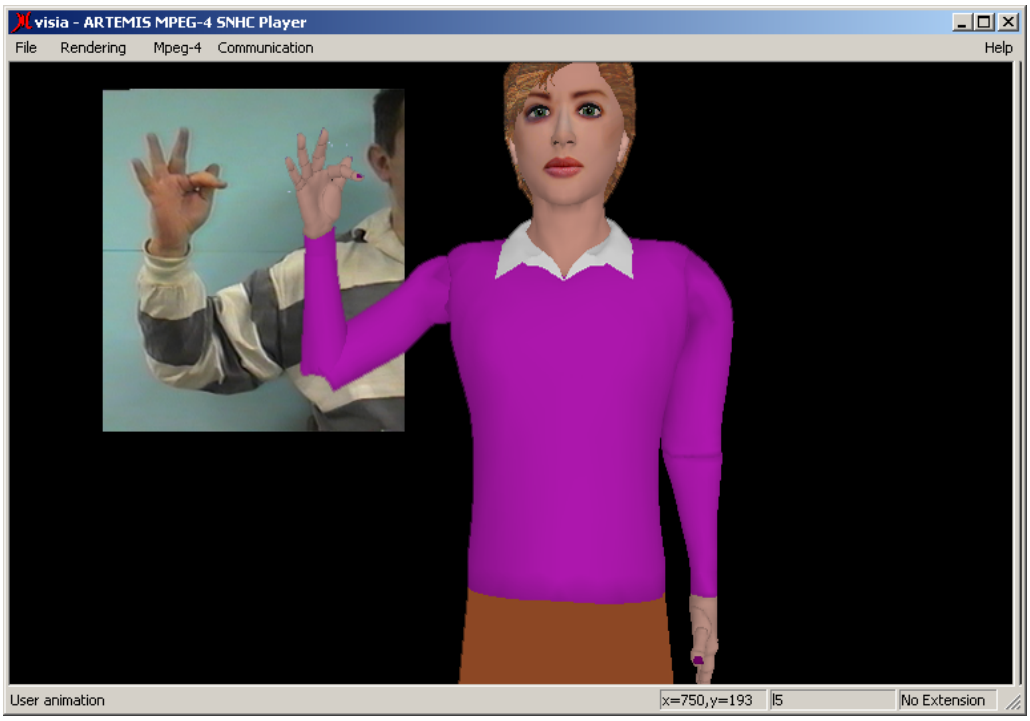


Figure 2.18. The ARTEMIS Avatar Animation Interface used for interactive tracking of gestures in image sequences.

In the rest of this section, we focus on the process of generating BAPs. Although the proposed interface is simple and straightforward to use, the amount of time needed to produce quality content is important. In order to reduce this time, we propose two mechanisms for the automatic generation of a temporal or spatial subset of parameters from some key parameters. The first mechanism is a temporal interpolation, reducing the action of the animator to some key frames, and the second one is an inverse kinematics method, reducing the action of the animator to specifying some relevant parameters in the key frame.

2.3.1 Key value-based animation

In order to animate an avatar, the MPEG-4 standard mandates the specification of the entire set of animation parameters for each frame. Usually, high level motion description systems like sign language notation systems define movements by specifying either the initial and final positions, in the case of simple motions, or the initial position and a trajectory, in the case of complex motions. It is also possible to describe a motion by specifying several key frames. In this subsection, we address the issue of automatically translating a high level animation description defined by key-frames to BAPs parameters. The principle consists in computing automatically the BAPs associated with the intermediate frames. The BAP file contains the explicit values of the joint angles and in order to compose the entire transformation of an anatomical segment the axis is obtained from the standard specifications. Hence, the key issue is the rotation interpolation. Basic linear interpolating techniques, directly applied in the angular space, have been proven to generate bad results [Wagner] leading to unnatural motions. Therefore, we propose to use a more elaborated interpolation scheme.

If more than two key frames are available, a high-order interpolation scheme like spline-based method could be used. Therefore, we have selected as a first approach a 3 degree spline interpolation [Bartels87].

When two key frames only are available, we propose a second approach in three steps as follows:

1) convert the BAP parameters into a quaternion representation,

2) apply a linear interpolation scheme in the quaternion space, and

3) convert back the quaternion representation of parameters into a BAP representation.

The choice of using the quaternion space is based on a mathematical result, which shows that linear interpolation between two quaternions results in a *continuous* rotation around a fixed axis.

The performances of the interpolation techniques have been first evaluated in the editing tool framework (Figure 2.19). Visual inspection of the interpolated sign language gestures shows that the quaternion–based interpolation generates realistic and fluid motions if appropriate key frames have been selected. The spline-based method performs visually better results, since the continuity forced in the key-frames produces a smooth and natural motion. On the other hand, a higher order of the spline curve involves more complex computations.

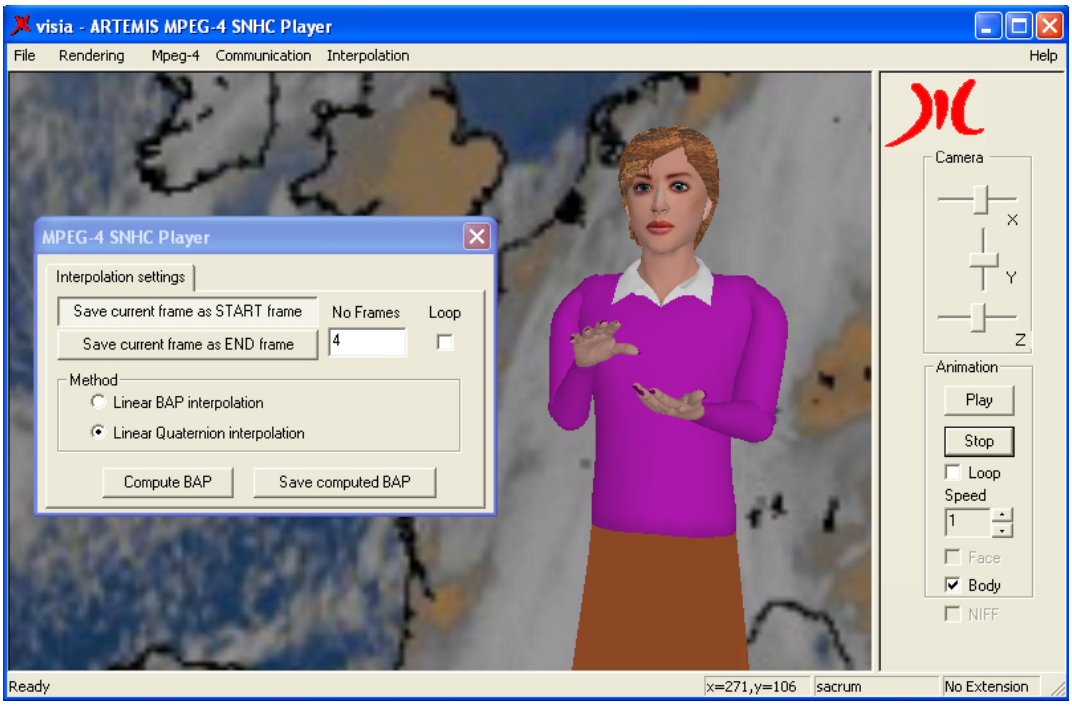


Figure 2.19. The ARTEMIS Avatar Animation Interface with the "Interpolation" control dialog for the case of two key-frames.

2.3.2 End-effector driven animation

The FBA specifications do not allow for the animation of the virtual body by specifying only a subset of the animation parameters. However, many applications address the issue of generating avatar

animation by specifying only relevant subparts of the virtual body. The rest of the animation parameters are computed during the animation stage. For example, in the case of a sign language description system, the main interest is focused on hand motions. As the arms, forearms and body posture are of relatively small importance; they are not explicitly described in the sign language notations, but obtained during the animation.

The complete set of BAPs has to be available in order to use a MPEG-4 player (FBA profile compliant); the MPEG-4 standard does not allow to specify only a sub-set of these parameters and to compute the rest at the user terminal. In this context, we propose here, an automatic method to generate the arms, forearms and body animation parameters starting only from the 3D hand location. We have developed a generic approach to compute the posture of an articulated model by specifying the localization of the end-effectors. This generic approach uses the animation techniques known as inverse kinematics, technique which exploits the biomechanical properties of the avatar.

An appropriate method to solve the inverse kinematics problem for real time animation is the so-called Cyclic-Coordinate Descent (CCD) method [Eberly01, Welman93]. It is based on a heuristic method which has been proposed to find quickly an initial feasible solution for a standard minimization-based algorithm. The method attempts to minimize position and orientation errors by varying one joint at a time. Each iteration involves a single traversal of the chain from the most distal link to the base. Each joint variable q_i is modified in turn to minimize an objective function. As a solution is obtained for each joint, the end-effector position and orientation and the rest of the body are updated immediately to reflect the change. Thus, minimization problem to be solved at any particular joint incorporates the changes made to more distal joints during the current iteration.

Let us consider that the current end-effector position P_c is

$$P_c = (x_c, y_c, z_c)^T \tag{2.1}$$

and that the current orientation O_c of the end-effector is specified by the three orthonormal rows of the rotation matrix

$$O_c = (u_{1c}, u_{2c}, u_{3c})^T . \tag{2.2}$$

The end-effector can be placed as close as possible to some desired position P_d and orientation O_d to find the joint vector q which minimizes the error measure:

$$E(q) = E_P(q) + E_O(q), \tag{2.3}$$

which is the sum of positional error measure

$$E_p(q) = \|P_d - P_c\|^2, \tag{2.4}$$

and orientation error measure

$$E_O(q) = \sum_{j=1}^{3} ((u_{jd} \cdot u_{jc}) - 1)^2 \ . \tag{2.5}$$

At each joint, the original n-dimensional optimization problem is reduced to a one-dimensional problem only involving the joint variable q_i. The current end-effector position and the rest of the body are updated to reflect the change before proceeding with the next joint (the parent of the current joint in the hierarchical structure). Once arrived at the root of the structure, if the goal is not achieved, a supplementary pass is performed. By limiting the number of passes, the method can give an appropriate solution even when the goal cannot be reached. During the entire process, BAPs are easily deduced and generated automatically.

By applying the method without imposing any constraints to the joint values can result in unrealistic postures of the avatar. To avoid these undesirable results we have used a set of ranges for all BAPs as specified in [ISOIEC01]. This ranges are provided by the MPEG-4 specifications in order to efficiently address the compression of the animation parameters and they reflects the limits of the joint angles for a human body. In the CCD implementation, when the solution of the analytical equation is out of range, the joint value is initialized with the appropriate value in the range (min or max).

The results obtained for a set of simple and complex motions with hands as end-effector (Figure 2.20) show that the performances of this method (in terms both of computation complexity and stability performances) are well adapt for real time animation applications.

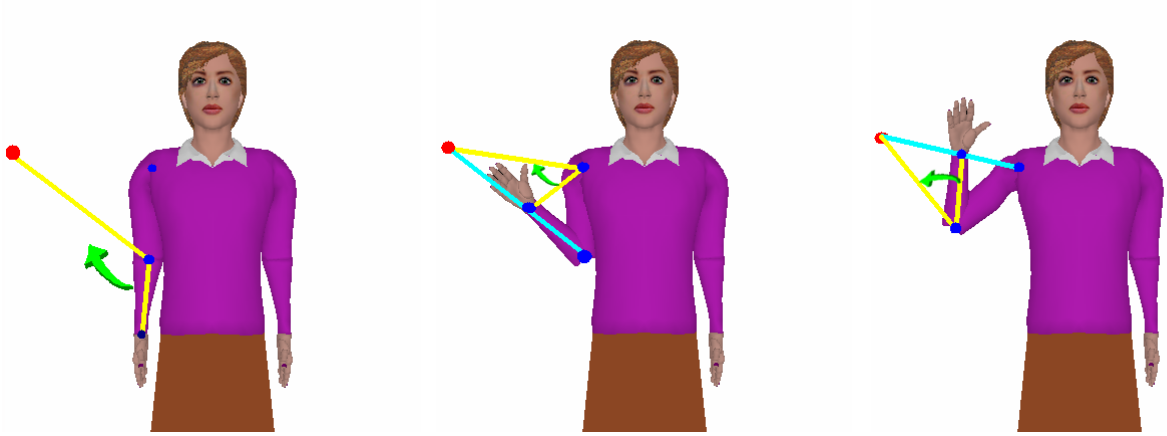


Figure 2.20. Successive steps for Cyclic-Coordinate Descent algorithm for a two bones (arm-forearm) IK system.

2.4 Performances analysis of the FBA framework in a client-server application

This Section focuses on a practical evaluation of both the compression and the animation performances of the FBA framework.

2.4.1 Evaluation framework

Using the two above-mentioned authoring tools, an FBA compliant avatar and several animation data sets corresponding to the American Sign Language alphabet were created. A complete server-client communication system was then set up to test the complete application from the encoding of the

animation parameters, their transmission over an IP (Webcast) network to their decoding and the rendering of the associated human model. In order to test the transmission we have built an UDP server (Figure 2.21) and enriched the 3AI platform with an UDP client (Figure 2.22). The server can be configured to send the animation data to a single IP address or to broadcast it to the entire sub-network.

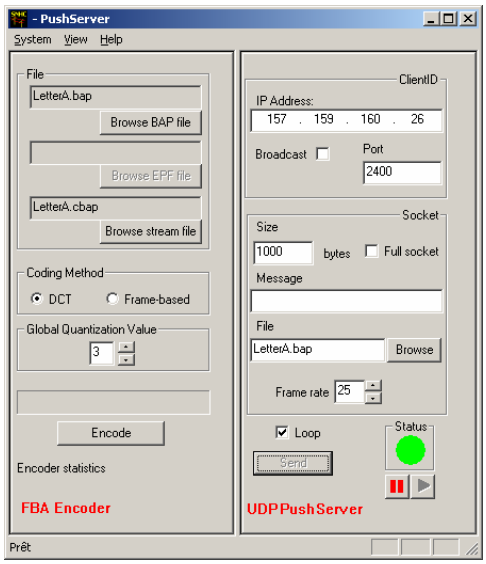


Figure 2.21. Animation parameters encoder and the UDP server interface.

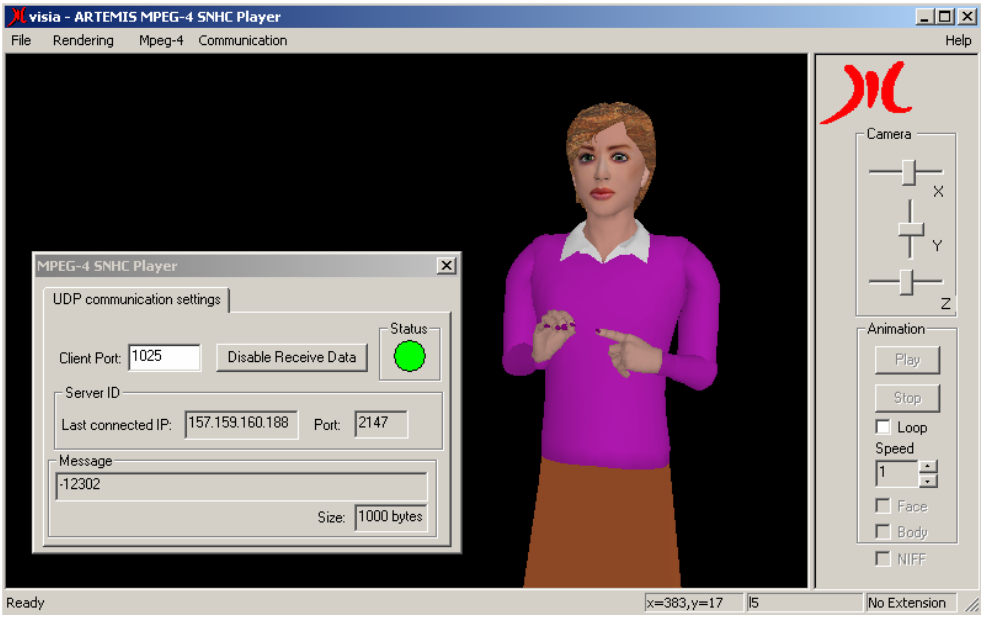


Figure 2.22. The ARTEMIS Avatar Animation Interface with the “UDP client” control dialog.

Two application scenarios are possible for providing the user terminal with a human model. The first one is by building a native (default) avatar into the terminal and the second consists in transmitting the avatar at the beginning of the animation session.

With MPEG-4, the avatar mesh and attributes can be transmitted with respect to one of the following scenarios:

i) non-compressed,

ii) as the Face and Body nodes are part of the scene, the BIFS compression can be used. As BIFS provides a generic compression mechanism, the results are not optimized for avatar geometry;

iii) using the dedicated 3DMeshCoding (3DMC) algorithm provided by MPEG-4.

While a non-compressed mesh (geometry and topology) corresponding to a virtual character is usually between 400 and 600 bits per vertex, the BIFS representation is between 100 and 300 bits per vertex, and the 3DMC representation is between 20 and 30 bits per vertex. Note that these values are obtained by considering both the geometry (the set of vertices) and the topology (the set of triangles) of the avatar.

In all three cases, the model geometry and attributes are transmitted once. We are interested now in analysing the condition when the default avatar (already built in the user terminal) is not valid. We call the default avatar a valid one, with respect to a set of animation data, if the animation results are those suited by the creator of the animation parameters.

Since the MPEG-4 FBA specifications does not impose geometric measures for avatars, when dealing with different morphometry of the avatars anatomical segments undesirable effects may rise. The different lengths of the segments such as arms, forearms or fingers may produce, during the animation, for one avatar the correct result and for another crossing effects between anatomical segments. Figure 2.23 shows an example using the same animation parameter set on two different avatars. As standard lengths for the anatomical segments are not specified by the FBA framework, the

a) “Dan” avatar. b)”Visia” avatar.

Figure 2.23. “Applause” sequence: 2 avatars with different morphometries and animated with the same animation parameters.

invariance of the model definition and the animation parameters is not complete and a re-targeting procedure must precede the animation stage.

We are addressing in the next section the animation parameters compression and transmission.

2.4.2 Compression efficiency

We have tested the MPEG-4 animation parameters encoding schemes on BAPs data corresponding to all the letters of the American Sign Languege (ASL) alphabet.. In ASL, all the letter signs are performed with one hand and the avatar body position and orientation do not change over time. In this experiment, the animation frame rate is 10 frames per second, which is, according to hearing impaired people, the lowest frame rate acceptable for a sign language transmission. The distortion between the original and the decoded sequences is defined as the mean square error of the BAP vectors:

$$D = \frac{1}{No_frames} \sum_{i=0}^{No_frames} \left\| BAP_i^{(o)} - BAP_i^{(d)} \right\|^2 , \qquad (2.6)$$

where $BAP_i^{(o)}$ and $BAP_i^{(d)}$ are the original and the decoded BAP vectors of the frame i, respectively. Table 2.1 presents the compression results, in kbps, for “A” to “J” letters signs, obtained by applying the DCT and the frame-based method, respectively, as well as the distorsion, computed by using the Equation (2.6).

Sign	Q=1		Q=2		Q=4		Q=8		Q=16		Q=31	
	P	DCT	P	DCT	P	DCT	P	DCT	P	DCT	P	DCT
A	1.32	2.80	1.30	2.42	1.25	1.83	1.22	1.54	1.18	1.14	1.14	0.86
B	1.41	2.61	1.35	2.22	1.34	1.64	1.32	1.36	1.32	1.08	1.29	0.81
C	1.80	3.65	1.78	3.15	1.71	2.24	1.68	1.87	1.63	1.41	1.58	1.10
D	1.45	2.66	1.42	2.30	1.38	1.69	1.35	1.40	1.32	1.01	1.31	0.74
E	1.75	3.23	1.71	2.85	1.65	2.13	1.62	1.74	1.57	1.30	1.53	1.00
F	1.37	2.04	1.33	1.83	1.31	1.36	1.28	1.12	1.26	0.88	1.25	0.67
G	1.81	2.83	1.75	2.43	1.72	1.76	1.67	1.47	1.63	1.12	1.57	0.84
H	1.78	2.83	1.74	2.39	1.70	1.70	1.65	1.40	1.63	1.08	1.58	0.76
I	1.53	2.70	1.49	2.39	1.46	1.73	1.41	1.45	1.39	1.10	1.38	0.81
J	1.37	2.20	1.33	1.89	1.31	1.34	1.27	1.11	1.24	0.82	1.21	0.59

Table 2.1. Bit-rates for the frame predictive-based and DCT-based coding schemes. Results for signs "A" to "L". Q denotes the global quantization value.

Figure 2.24 shows the results obtained with different quantization steps size for both encoding methods in the case of sign letter “B”. These results are analysed by studying the bit rate as a function of the distortion. The encoding methods allow to control the bit rate by modifying the global

quantization step (Figure 2.24 II). Figure 2.24 I illustrates the distortion *versus* quantization step diagram. Figure 2.24 III results from the combination of Figure 2.24 I and Figure 2.24 II.

Do observe that DCT curve (Figure 2.24 III) decreases, while the frame-based method curve is almoast constant. When dealing with a wide range of bit rates, the DCT-based method has to be used. Since DCT-based method uses a 16 frames temporal buffer, an animation occurs. For sign language application, this delay introduces a small non-synchronisation, but does not affect the message comprehension. In the case of applications that require near loss-less compression and exact synchronization of the animation with another media, the use of the frame predictive-based method is recommended. In order to increase the efficiency of the arithmetic encoder, the MPEG-4 FBA specifications standardize a set of ranges for each animation parameter. The global quantization step is used here for scaling the value to be encoded in the corresponding range. Each animation parameter is encoded with the same number of bits inside this range. If the obtained scaled value is outside of the range, a higher quantization step has to be used. This behaviour of the frame-based encoder is also proved by the non-invariance of the bitrate with respect to the global quantization step, as illustrated in Figure 2.24.

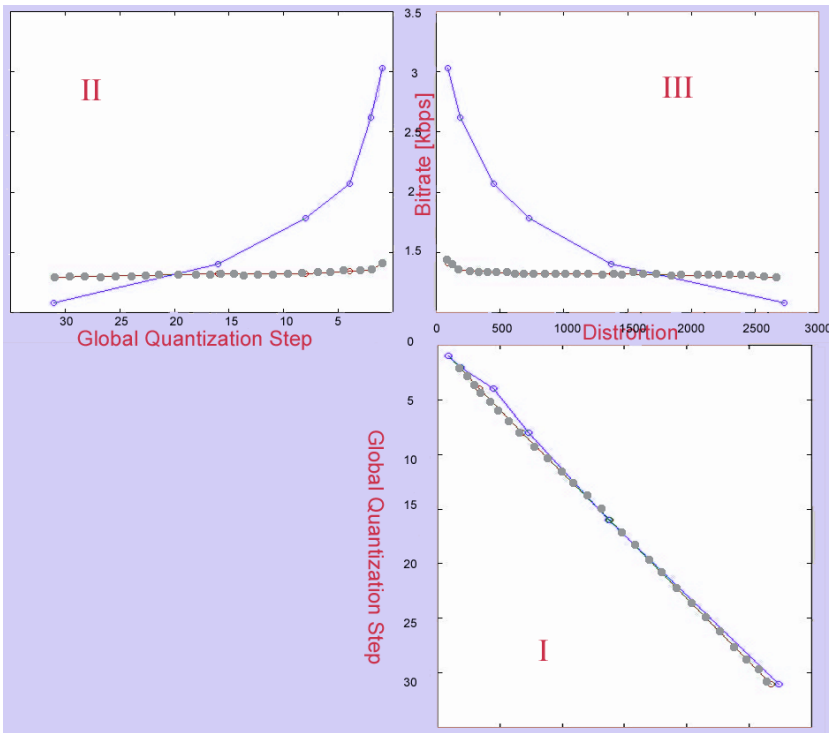


Figure 2.24. Experimental results for sign "B". Predictive (dense bullets) *vs* DCT BAP coding schemes. Distortion *vs* Q (I), Bit-rate *vs* Q (II) and Distorsion *vs* bit-rate (III).

We have tested the FBA compression techniques on a more complete data set representing real sign language content. In this sequence, both arms and the body of the avatar are animated. The frame-rate for this content is 20 fps. Figure 2.25 shows a frame from this data set.

Figure 2.25. Frame from animation data set.

When using the frame predictive-based method, a quantisation value bigger than 4 has to be used and the obtained bit-rate is close to 6.2 kbps. The compression results for the DCT-based method are presented in Table 2.2.

Sign	Q=1	Q=2	Q=4	Q=8	Q=12	Q=12	Q=24	Q=31
Bitrate[kbps]	12,08	10,41	8,5	7,29	6,25	5,41	3,95	3,33

Table 2.2. Bit-rates [kbps] for the DCT-based coding scheme. Q denotes the global quantization value.

The low bit rate, less than 15 kbps, obtained by compressing the animation parameters, while keeping visual degradation at a satisfactory level, allows animation transmission in a low bit-rate network.

2.4.3 Realistic animation

We analyse in this section the FBA framework performances related to the realistic representation of the avatar. Such requirement is of major importance when dealing with a sign language communication system.

In order to provide realistic 3D deformations and to avoid the seams at the joint level, (effects induced by the animation), the MPEG-4 FBA includes a deformation modelling tool achieved by instantiating *Body Deformation Tables* (BDTs). BDTs address flowing motions by specifying a list of vertices of the 3D model as well as their local displacements as functions of BAPs [ISOIEC01]. Let us

demonstrate the BDT concept to deform the shape of a finger at the joint between ring finger proximal and middle phalanges. We instantiate the BodyDefTables corresponding to these anatomical segments:

```
BodyDefTable {
        bodySceneGraphNodeName     "l_ring_proximal"
        bapIDs [143]
        vertexIds          [ 50, 51, 52, 53, 54]
        bapCombinations          [100, 200, 300, 450, 500]
        displacements
[ -0.11 0.12 0, 0.1 0 0, 0.1 0 0, 0.1 0 0, 0.1 0 0, 0.1 0 0,
-0.25 0.25 0, 0.22 0 0, 0.22 0 0, 0.22 0 0, 0.22 0 0, 0.22 0 0,
-0.35 0.37 0, 0.3 0 0, 0.3 0 0, 0.3 0 0, 0.3 0 0, 0.3 0 0,
-0.4 0.47 0, 0.41 0 0, 0.41 0 0, 0.41 0 0, 0.41 0 0, 0.41 0 0,
-0.52 0.6 0, 0.48 0 0, 0.48 0 0, 0.48 0 0, 0.48 0 0, 0.48 0 0]
}
BodyDefTable {
        bodySceneGraphNodeName     "l_ring_middle"
        bapIDs [ 143]
        vertexIds [ 0, 1, 2, 3]
        bapCombinations [100, 200, 300, 450, 500]
        displacements
[ -0.1 0 0, -0.09 0 0, -0.08 0 0, -0.1 0 0,
-0.22 0 0, -0.2 0 0, -0.21 0 0, -0.22 0 0,
-0.29 0 0, -0.28 0 0, -0.3 0 0, -0.3 0 0,
-0.35 0 0, -0.37 0 0, -0.39 0 0, -0.4 0 0,
-0.4 0 0, -0.4 0 0, -0.45 0 0, -0.45 0 0]
}
```

In this case, the BDTs refer to the deformation of the segments *l_ring_proximal* and *l_ring_middle* as functions of BAP #143, *l_ring-flexion2* (See [ISOIEC01] for segments and BAPs names). The vertices with indices 50, 51, 52, 53 and 54 on surface *l_ring_proximal* are deformed. The displacements for vertex 50 are (-0.11 0.12 0), (-0.25 0.25 0), (-0.35 0.37 0), (-0.4 0.47 0) and (-0.52 0.6 0) for the BAP *l_elbow_flexion* values 100, 200, 300, 450 and 500, respectively. By controlling the shape at the vertex level, muscle-like deformations can be made. Note that the vertices displacements are to be provided for BAP key values. Realistic deformations can be possible if the deformation tables are large enough, with an important number of animation parameter key values and an important number of affected vertices (usually up to 30).

In order to obtain the vertices displacements for all the animation parameters value set, the standard allows a BDT interpolation method using the reference BDTs associated with key frames. Figure 2.26 shows the results of the BDT-based interpolation technique in the case of a simple finger flexion movement.

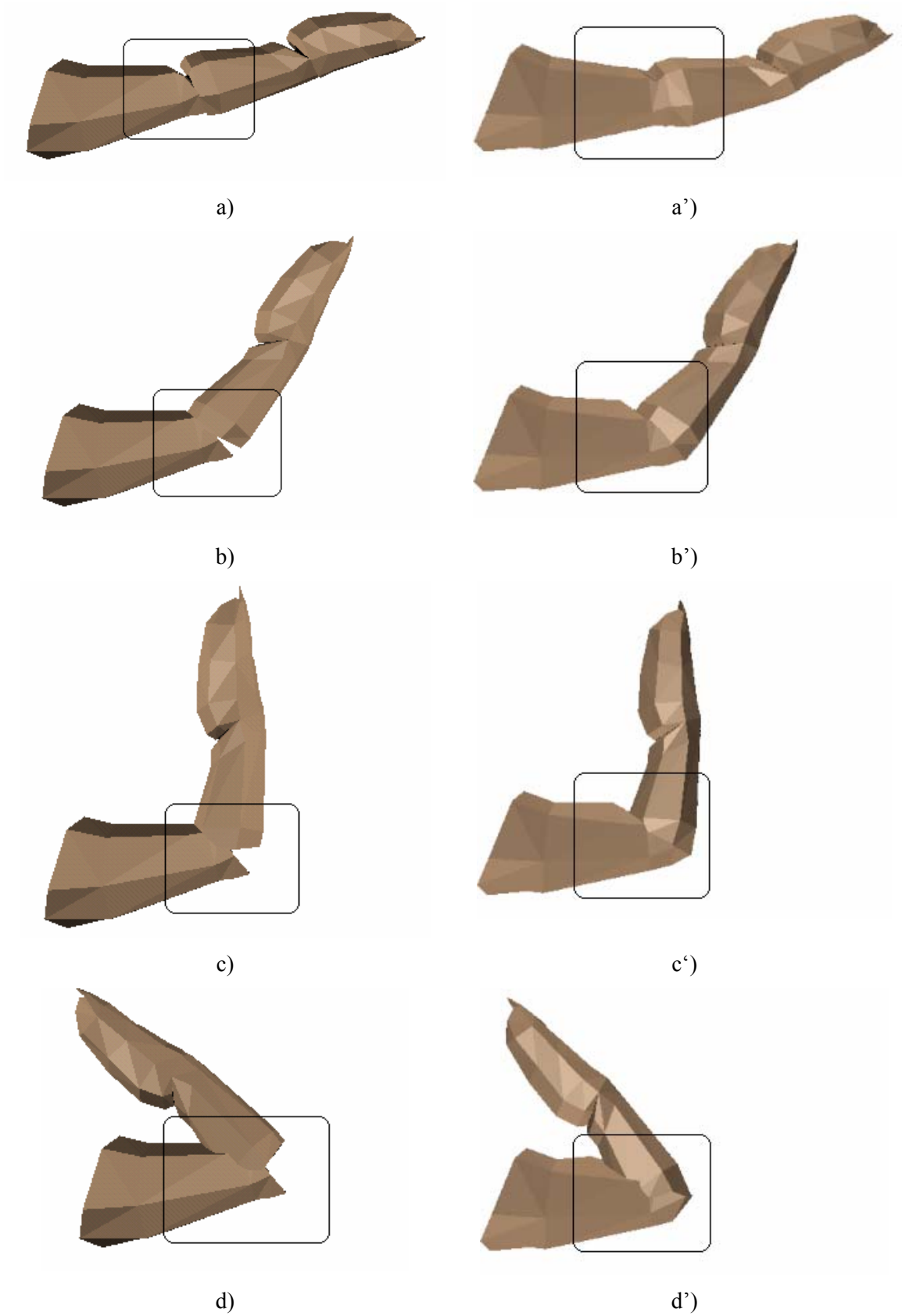


Figure 2.26. Finger animation: without BDTs (*a)*, *b)*, *c)*, *d)*); with BDTs (*a')*, *b')*, *c'), d')*) – BDTs interpolation: key frames (*a')* and *d')*), interpolated BDTs (*b')* and *c')*).

There are two major limitations related to the deformation table mechanism. The first one is related to the deformation table generation. There are no automatic procedures to directly build the tables. A possible indirect method is to generate, with the same set of animation parameters and for the same avatar, the animation in two different ways: one using the segmented body as defined in the FBA and one using any available method in the literature regardless of its processing requirements. By selecting a number of key frames and by analysing the difference between the positions of the same vertex in the two animations, one can obtain the deformation tables. The limitation consists, then, in the obligation to animate the avatar with another method.

The second limitation of the FBA framework consists in the poor support for the deformation table interpolation schemes and the non existence of compression mechanisms to reduce the size of the tables. In order to achieve realistic animation, the tables must have an important number of entries in terms of affected vertices. As the only interpolation scheme supported is linear interpolation, the tables must have an important number of entries in terms of key frames. A possible solution is to support high level interpolation schemes as proposed in [Guttierez02], or to allow compression mechanism for the BodyDefTable node. None of these two solutions are supported by the FBA specifications.

In conclusion, providing realistic animation in the FBA framework is possible through the BodyDefTable mechanism. However, to generate the deformation tables requires additional animation methods and the size of the deformation tables can make up to 50% of the size of the entire model, depending on the requested deformation accuracy.

2.5 Conclusion

The MPEG-4 FBA specifications for virtual human characters offer a basic framework for animation, which allows the representation of the avatar as a segmented virtual character. Furthermore, the motion parameters representation is very compact and makes it possible, in a compressed form, to get very low bit rate streaming animations in networked environments.

We have presented in this chapter an evaluation framework for the FBA with respect to

- the interdependence between the model definition and the animation parameters,
- the animation compression performances and
- the realistic animation capabilities.

Since the FBA do not mandate a specific method to obtain the avatars and the animation parameters, we have proposed tools to produce these data. A semi-automatic segmentation algorithm allows to segment a human model obtained from 3D scanner. Then, an authoring tool builds the hierarchical structure of the character. Once the avatar static structure is available, an animation editing software, enriched with time and spatial interpolation/extrapolation mechanisms, allows to generate the animation parameters. The performances in terms of compression and realistic animation were then presented and discussed.

The poor capabilities that FBA provide in terms of easy and efficient handling of mesh deformation, led us to propose an integrated framework, based on a seamless representation of the virtual character and a set of mesh deformation controllers. The next chapter introduces this framework and shows its performances with respect to FBA.

Chapter Three

Virtual Character Animation: the Skeleton, Muscle and Skin (SMS) Framework

The purpose of this chapter is to introduce an augmented animation framework for virtual character animation. This framework is founded on a generic deformation model that we introduce in the first section of the chapter. The proposed model relies on a deformation controller defined by means of a geometric support, an influence volume around this support and a measure of affectedness within the influence volume. With respect to these elements, a classification of the main deformation techniques reported in the literature is presented. In the following, we introduce two instances of our deformation model which offer efficient control of the geometrical support and appropriate volume

specification: (1) bone controller and (2) muscle controller. Our original contribution consists in proposing the Skeleton, Muscle and Skin (SMS) framework, builded around these concepts (bone and muscle). We show how they can be used to define and animate generic virtual characters. In order to embed generic virtual characters into a 3D scene we provide the SMS architecture with the scene graph nodes. We then address the issue of an efficient representation of the animation parameters by (1) enriching the animation capabilities with temporal interpolation and inverse kinematics support and (2) adapting two data compression techniques to the SMS animation data, namely predictive- and DCT-based. Experimental results are presented and discussed. A quantitative and qualitative analysis of the obtained results shows that the SMS framework offers a good support to perform the transmission of a realistic virtual character animation over low-bit-rate networks. A comparative study between the SMS and FBA frameworks, states the advantages of the former.

Finally, we describe how we have promoted the SMS framework in the MPEG-4 standard.

3.1 Synthetic object deformation: toward a unified mathematical model

The key issue pointed out in the previous chapter refers to realistic animation that FBA tools can not efficiently achieve. One of the main reasons comes from considering the avatar as a segmented mesh and performing the animation by applying a rigid geometric transformation to each segment. In order to overcome this limitation, here we consider the object as a seamless mesh and animate it by means of deformations. In the following, we introduce a unified mathematical model for synthetic object deformation.

Let $M(\Omega)$ be a seamless mesh, where $\Omega = \{v_0, v_1, ... v_n\}$ is the set of the mesh vertices and let $(\Omega_i)_i$ be a family of non-empty subsets of Ω. A local deformation function $\varphi_i : \Omega \to \mathbf{R}^3$ makes it possible to move a vertex $v \in \Omega_i$ into the new position expressed as $v + \varphi_i(v)$. Here, φ_i is extended from Ω_i to Ω as the null function, *i.e.* $\forall v \in \Omega \setminus \Omega_i, \varphi_i(v) = 0$. Note that the family $(\Omega_i)_i$ is not necessarily a partition of Ω. In particular, $\bigcup_i \Omega_i$ can be a strict subset of Ω (some vertices may remain unchanged) and for two distinct subsets Ω_i and Ω_j, the intersection $\Omega_i \cap \Omega_j$ can be nonempty. The deformation satisfies the superposition principle, *i.e.* the deformation induced by both φ_i and φ_j at a vertex v belonging to $\Omega_i \cap \Omega_j$ is expressed as the sum $\varphi_i(v) + \varphi_j(v)$.

In order to achieve a compact description and an efficient implementation of a deformation model, let us introduce the notion of *deformation controller*, which is defined as a triplet made of:

- the support S associated with a n dimensional (nD) geometric object ($n \in \{0, 1, 2, 3\}$),
- an influence volume $V(S)$ associated with S,
- the affectedness measure μ, defined on $V(S)$ and characterizing the intrinsic deformation properties of the influence volume.

A family $(C_i = \{S_i, V(S_i), \mu_i\})_i$ of controllers is said to be associated with a mesh $M(\Omega)$ if and only if the following relationships are fulfilled:

1. $\forall i \in \{0, 1, ..., n\}\ \Omega_i = \Omega \cap V(S_i)$
2. it exists a mapping ψ_i from Ω_i to S_i such that any vertex $v \in \Omega_i$ is linked to a set of elements of S_i.

Applying an (affine or not) transformation T_i to C_i is equivalent to define a deformation function φ_i on Ω_i such that:

$$\forall v \in \Omega_i, \varphi_i(v) = \mu_i(v) \sum_{\xi_k \in \psi_i(v)} \omega_k \left[T_i(\xi_k) - \xi_k \right], \tag{3.1}$$

where μ_i is the affectedness measure associated with C_i and ω_k is a weighting coefficient.

In practice, the transformation T_i is applied to the controller and is propagated with the influence volume $V(S_i)$ according to the affectedness measure μ_i, which plays the role of a weighting function. When $\psi_i(v)$ is reduced to a single element, Equation (3.1) is simplified as:

$$\forall v \in \Omega_i, \varphi_i(v) = \mu_i(v)\left[T_i(\psi_i(v)) - \psi_i(v)\right]. \tag{3.2}$$

This controller-based deformation framework enables to unify the different deformation techniques reported in the literature with respect to the dimension of the controller support. Typically, the most representative technique of a volumic controller-based approach is the lattice-based deformation model. In this case, the 3D grid is considered as the controller volume. The 1D controller-based approach covers most of the deformation techniques currently used, namely: deformation tables (a particular case being described in Chapter 2), spline-based, and skeleton-based. The 0D controller principle is used in the case of cluster-based and morphing-based approaches.

In practice, choosing an appropriate controller results from a trade-off between:

1. the complexity of representing the controller, directly linked to its dimension, and
2. the distribution of mesh vertices affected by the controller, specifically by choosing the most appropriate influence volume.

An optimal balance is obtained by using a 1D controller. The support of the controller is thus easy to control (the number of parameters is small) and the corresponding influence volume covers a large class of configurations. Let us now apply such an approach for animating an articulated virtual character in the case of two specific 1D controllers, namely a segment and a curve defined as a NURBS, referred to as bone- and muscle-based deformation, respectively.

3.2 Skeleton, Muscle and-Skin-based modelling and animation

3.2.1 Bone and muscle controllers for animating an articulated object

An articulated virtual character and generally, an articulated synthetic object, also called kinematics linkage, is composed of a set of rigid links which are connected at the joints. With respect to the modelling classification presented in Section 1.1, one can distinguish the following configurations:

- in the case of a segmented virtual character, a rigid link corresponds to the envelope of an anatomical segment;
- in the case of a seamless virtual character, a rigid link is associated with each anatomical bone of the skeleton.

In order to define the static 3D posture of an articulated virtual character, including geometry, colour and texture attributes, the approach here proposed consists in considering the entire virtual character as a single 3D mesh referred to as *skin*.

During the animation stage, the anatomical bones can only be affected by rigid transformations and cannot be locally deformed. Realistic animations require however local skin deformations, for simulating muscular activity effects. In order to fulfil this requirement, we propose to attach curve-based entities at an arbitrary level of the skeleton.

We call Skeleton, Muscles and Skin (SMS) the framework for animating virtual characters by using the above-mentioned bones- and muscles-based deformation controllers.

Two issues are addressed by the SMS framework. The first one deals with the definition of the skinned model as a static model described by its geometry and its appearance. In order to perform this task, a hierarchical skeleton and a collection of muscles are introduced. The second issue deals with the animation of articulated models together with a compressed representation of the animation parameters.

Defining a SMS virtual character requires to specify a set of static attributes (geometry, texture, etc.) as well as a deformation model. From a geometric point of view, an SMS synthetic model is represented in such a way that the set of vertices which belong to the skin of the virtual character is specified as a unique list. In order to define various appearances at different levels of the SMS virtual character, distinct shapes can be concatenated provided that all the shapes composing the skin share the same list of vertices. This type of representation avoids to get seams on the skin during the animation stage, while preserving the possibility to define various sets of colour, texture and material attributes at different levels of the skin.

The animation behaviour of a skinned model is defined by means of skeleton and muscle layers together with their properties.

The skeleton is a hierarchical structure built from bones. A bone is defined as a segment of length l by means of:

1. a geometric transformation of the bone, defined with respect to its parent in the skeleton hierarchy,
2. a model of influence of the bone movement on the surface of the articulated model, and
3. inverse kinematics constraints.

A muscle layer is a collection of individual muscles, each one being attached to a bone. Specifically, it is defined by means of:

1. a curve represented as a NURBS which can be deformed, and
2. a model of influence of the curve deformation on the skin of the model.

Each bone and each muscle has influence on the skin. Thus, by changing a bone position or orientation or by deforming the muscle curve, the skin vertices belonging to the associated influence volume will be displaced accordingly. Here, to define the skinned virtual character consists in specifying for each bone and for each muscle an associated influence volume, *i.e.* the subset of the affected skin vertices together with the related measure of affectedness as a set of weighting coefficients. The influence can be directly specified vertex by vertex by the designer of the synthetic model, or computed before performing the animation. In the first case, the list of affected vertices and the weighting coefficients are included in the bone/muscle definition. In the second case, distinct approaches are proposed for

bones and muscles, respectively. The two following sub-sections present different strategies for computing the bone and the muscle influence volume, respectively.

3.2.2 Bone-based modelling and animation

3.2.2.1 Bone influence volume

The bone influence volume is defined as the support of the affectedness measure μ. Here μ is expressed as a family of functions $(\mu_d)_{d\in[0,l]} \cup \mu_{0^-} \cup \mu_{l^+}$. μ_d is defined on the perpendicular plane located at distancee d from the bone origin. The support of μ_d is partitioned into three specific zones ($Z_{d,in}$, $Z_{d,mid}$ and $Z_{d,ext}$) by two concentric circles characterised by their respective radius r_d and R_d (Figure 3.1.). μ_d is then defined as follows:

$$\mu_d(x) = \begin{cases} 1, & x \in Z_{\text{int}} \\ f\left(\dfrac{\delta(x, Z_{ext})}{R_d - r_d}\right), & x \in Z_{mid}, \\ 0, & x \in Z_{ext} \end{cases} \tag{3.3}$$

where $\delta(x, Z_{ext})$ denotes the Euclidean distance from x to Z_{ext} and $f(\cdot)$ is a user specified fall-off to be chosen among the following functions: $x^3, x^2, x, \sin(\frac{\pi}{2}x), \sqrt{x}$ and $\sqrt[3]{x}$. This set of functions allows a large choice for designing the influence volume and ensures the generality of the model.

The affectedness measure μ_{0^-} (respectively μ_{l^+}) is defined in the same manner but using two half-spheres of radius r_0 and R_0 (respectively r_l and R_l) as illustrated in Figure 3.1.

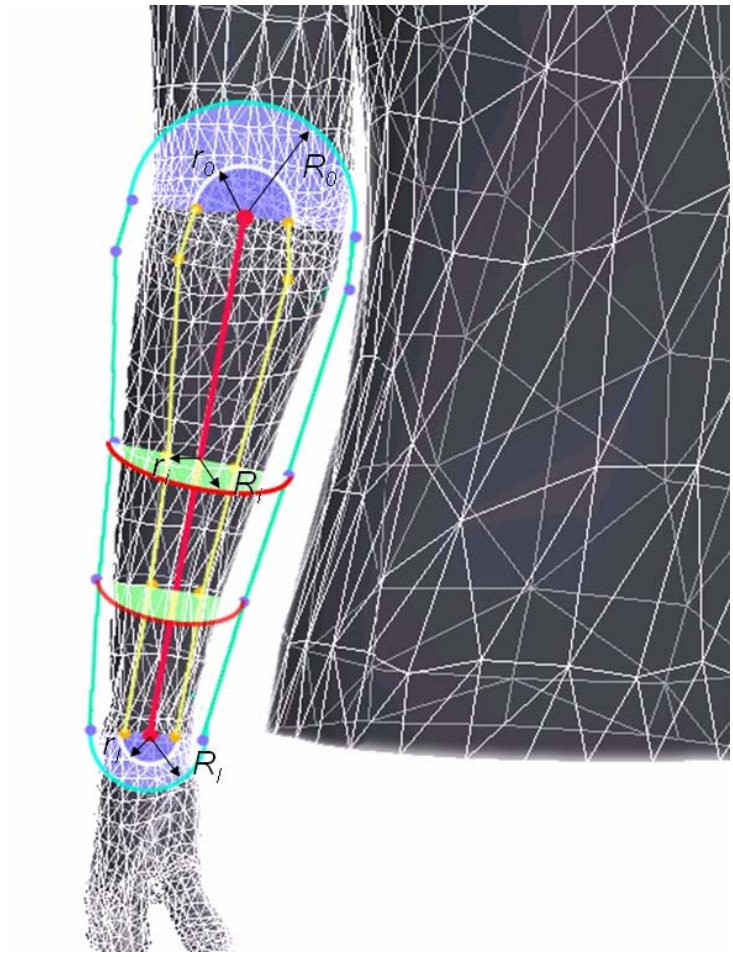


Figure 3.1. Forearm bone influence volume.

The bone influence volume being defined, animating the virtual character consists in deforming its mesh by translating its vertices according to the bone transformation. Let us detail the mathematical formulation of the bone transformation.

3.2.2.2 Bone transformation and induced effects on the skin

Here only affine transformations are applied to the bone controller. In virtual character animation, the most widely used geometric transformation consists in changing the orientation of the bone with respect to its parent in the skeleton hierarchy. Thus, the bone can be rotated with respect to an arbitrary axis. However, when special effects are needed, the bone can also be translated. For instance, in cartoon-like animations, thinning and thickening the skin envelope are frequently used. For such effects, the bone transformation must contain a scale component specified with respect to a predefined direction.

The general form of the geometric transformation of a bone b_i is expressed as a 4×4 element matrix T_i obtained as follows [VRML]:

$$T^w i = TR^w b_i \cdot R^w b_i \cdot S^w b_i \tag{3.4}$$

where $TR^w b_i$, $R^w b_i$, $S^w b_i$ give the bone translation, rotation and scale, respectively, expressed in the world coordinate system.

In practice, the computations are performed in the local coordinate system attached to the bone. In order to perform the scaling with respect to a predefined direction, the matrix SR_b performs a pre-orientation of the bone. Thus, in this system, the T_i transformation is expressed as:

$$T_i = TR_{b_i} \cdot C_{b_i} \cdot R_{b_i} \cdot SR_{b_i} \cdot S_{b_i} \cdot (SR_{b_i})^{-1} \cdot (C_{b_i})^{-1} \tag{3.5}$$

where matrix C_b allows to pass from the bone local coordinate system to the world coordinate system. For the sake of flexibility and compactness, the general transformation matrix is expressed in the bone node definition as separated components, as shown below (notations according to the MPEG-4 formulation):

exposedField	SFVec3f	center	0 0 0	#corresponding to C_b
exposedField	SFRotation	rotation	0 0 1 0	#corresponding to R_b
exposedField	SFVec3f	translation	0 0 0	#corresponding to TR_b
exposedField	SFVec3f	scale	0 0 0	#corresponding to S_b
exposedField	SFRotation	scaleOrientation	0 0 1 0	#corresponding to SR_b

Here, the animation consists in updating, at each frame, the components *translation*, *rotation*, *scale*, *scaleOrientation* and *center*.

Once the bone geometric transformation is computed, it is possible to compute the new position of the skin vertices in the bone influence volume according to formula (3.2).

The SMS framework does not limit the number of bones used for animating a virtual character. Consequently, local deformation effects can be obtained by creating and animating additional bones. If such an approach can be used with a certain success for simulating limited local deformation, the efficiency may strongly decrease when considering realistic animations. To address this kind of animation, we introduce in the next section the muscle-based deformation.

3.2.3 Muscle-based modelling and animation

3.2.3.1 Muscle influence region

The muscle influence volume is constructed as a tubular surface generated by a circle of radius r moving along the NURBS curve (see Figure 3.2.)

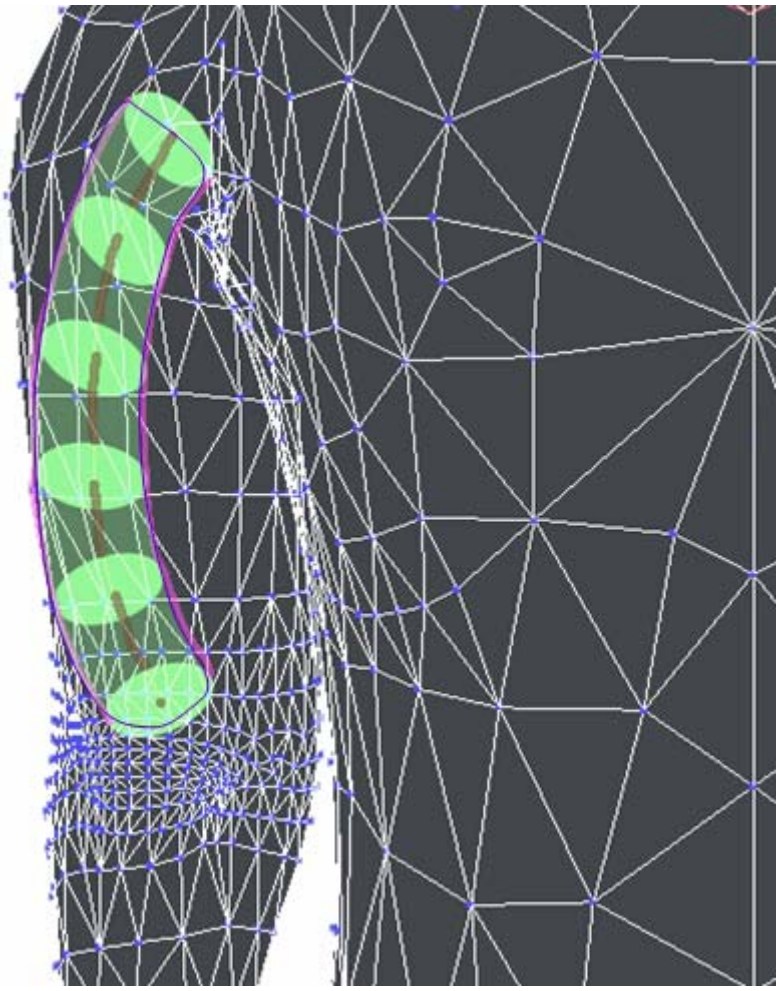


Figure 3.2. The muscle influence volume (in transparent green).

The affectedness function is then defined as follows:

$$\mu(v) = \begin{cases} 0 & \delta(v_i, \psi(v_i)) > r \\ f\left(\dfrac{r - \delta(v_i, \psi(v_i))}{r}\right) & \delta(v_i, \psi(v_i)) \leq r \end{cases}, \tag{3.6}$$

where δ denotes the Euclidean distance, $f(\cdot)$ is to be chosen among the following functions: $x^3, x^2, x, \sin(\frac{\pi}{2}x), x^{1/2}$ and $x^{1/3}$, and ψ is the function assigning to v its correspondent point on the muscle curve.

3.2.3.2 Muscle deformation and induced effects on the skin

A muscle is designed as a curve together with an influence volume on the virtual character's skin. In order to build a flexible and compact representation of the muscle shape, we use a NURBS-based modelling. Animating the muscle consists here in updating the NURBS parameters. The main

advantages of NURBS-based modelling are the accurate representation of complex shapes. The control of the curve shape is easily addressed. A set of control points and knots, ruling the shape of the curve, can be directly manipulated in order to control the local curvature. In addition, NURBS parameterization is very compact even for complex shapes. For example, approximating a circle by a set of line segments requires tens or thousands of segments to obtain a circle-like shape instead a polygon-like one. Using NURBS instead of line segments, the representation requires only seven control points.

As NURBS theory is widely reported in the literature, we shall not get into details here, the interested reader should report to [Piegl97]. In the following, we illustrate how one can animate a NURBS curve.

In order to change the shape of a NURBS curve, only the positions of the control points have to be modified. Figure 3.3 illustrates the effect of translating an inner control point of a NURBS defined by 12 control points; in green - the initial curve, in red - the deformed curve obtained by translating B_7.

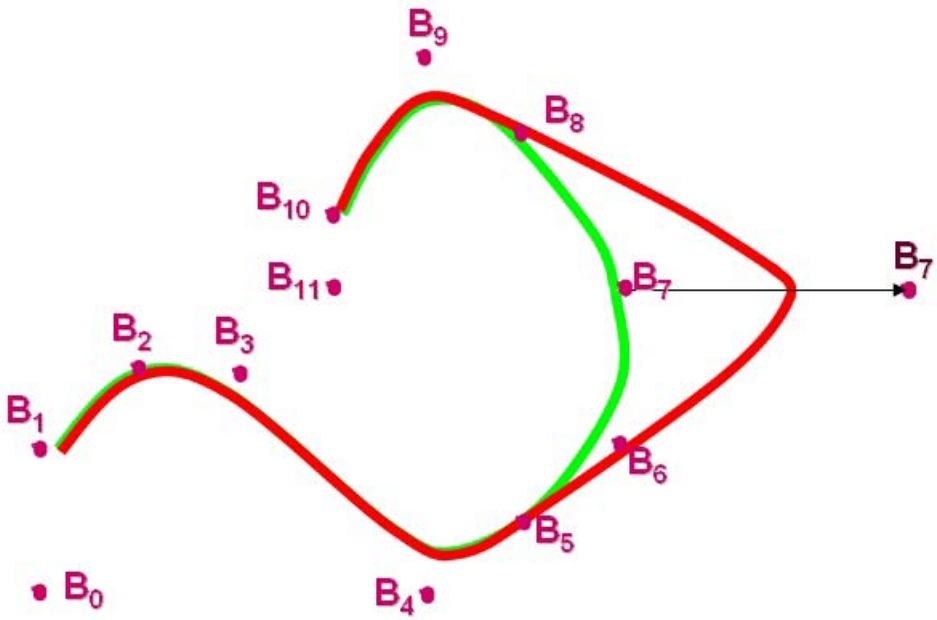


Figure 3.3. Influence of the translation of a control point on the NURBS shape..

To give more importance to a specific control point, a weighting coefficient can be assigned. Such an effect is illustrated in Figure 3.4.; in red - the initial curve w_{B3} =0.25, in blue - the deformed curve w_{B3} =1 and in green - the deformed curve w_{B3} =2.5. A careful selection of the knot sequence allows to model a given angle configuration Figure 3.5; in red an uniform curve, the knot sequence is {0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0}, for the green curve the knot sequence is {0.0, 1.0, 2.0, 3.0, 3.0, 5.0, 6.0, 7.0} and for the blue curve the knot sequence is {0.0, 0.0, 0.0, 3.0, 4.0, 5.0, 6.0, 7.0}.

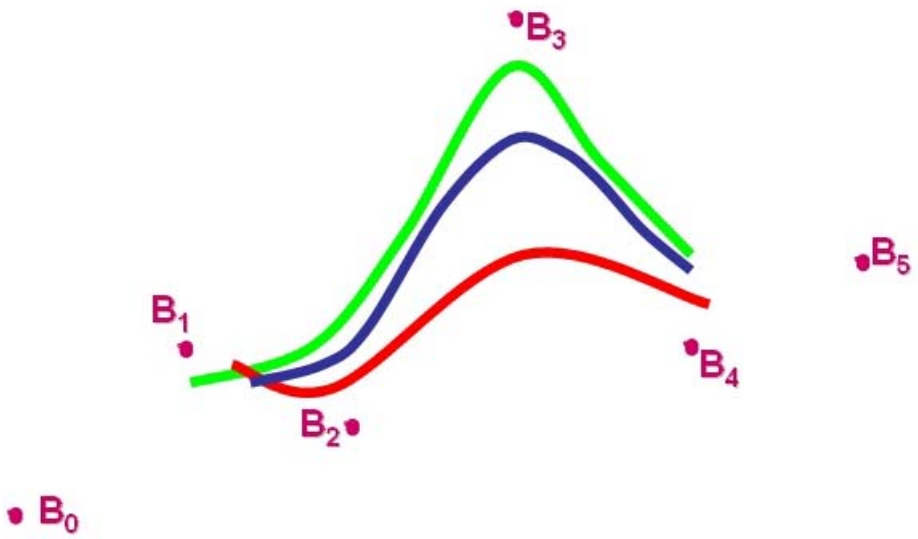


Figure 3.4. Influence of weighting a control point (here, B_3) on the NURBS shape.

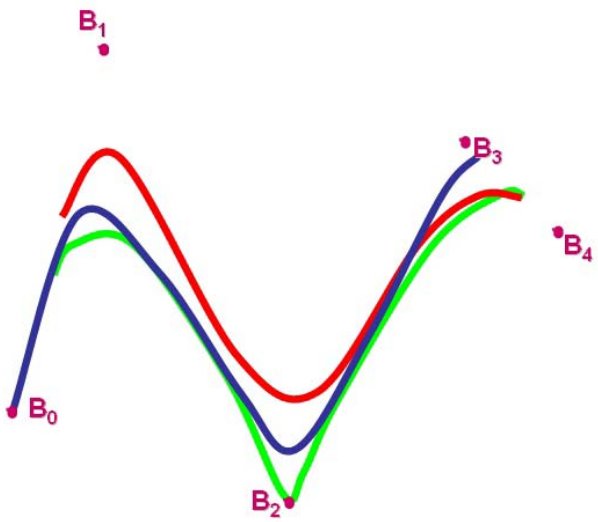


Figure 3.5. Influence of the knot sequence on the curve shape.

NURB-based modelling is fully supported within the SMS framework. Animating a muscle consists in updating the values of its NURBS parameters according to affine or not affine transformation T:

exposedField	MFFloat	knot	[ ]
exposedField	MFVec4f	controlPoint	[ ]

Here, the *controlPoint* component is a vector of control points. Each control point is a vector with 4 elements: 3 for the 3D position (x, y, z) and 1 for the weight (w).

Once the muscle transformation is computed, it is possible to compute the new position of the skin vertices in the muscle influence volume according to Equation (3.2).

3.3 Skeleton, Muscle and Skin nodes specification

One of the main purposes of the SMS framework is to allow the definition and the animation of a virtual character within a 3D scene. In this context, we propose a scene graph architecture to describe the SMS elements. This architecture is build according to the VRML and MPEG-4 scene graph definition rules. Thus, the structure of the proposed architecture relies on the definition of scene graph nodes. In order to easly identify the SMS-related nodes in the scene graph, all the node names start with SB (Skin&Bones) letters.

At the root of the SMS-related node hierarchy, a SBVCAnimation node is defined. The main purpose of this node is to group together a subset of virtual characters of the scene graph and to attach to this group an animation resource (file or stream). An SMS virtual character is defined as a SBSkinnedModel node and it is related to a collection of bones – each one defined as a SBBone node - together with a collection of muscles – defined as SBMuscle nodes. An optimal modelling issue is addressed by defining the SBSegment node, as explained in Section 3.3.3. In addition, the SBSite node allows defining semantic regions in the space of virtual character. These nodes are detailed below. Since SBVCAnimation node is dedicated to integrate the SMS animation stream into an MPEG-4 stream, its structure is detailed in the Section 3.7, where the MPEG-4 integration details are presented.

3.3.1 SBBone Node

The fields of the SBBone node are illustrated in Figure 3.6.

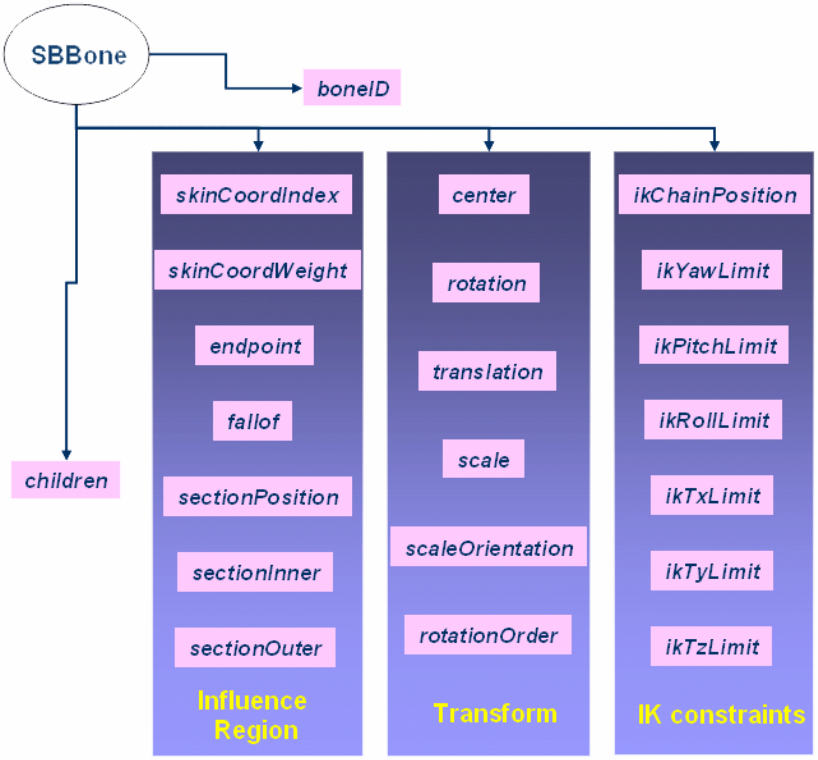


Figure 3.6. The fields of the SBBone node.

The interface of the SBBone node is as follows. The first column refers to the type of the component (field or event), the second one gives the type of the component, the third one indicates the name and the last one indicates the default value ("[]" syntax means that the component is a vector with a non-specified default value).

```
SBBone{
    eventIn        MFNode        addChildren
    eventIn        MFNode        removeChildren
    exposedField   SFInt32       boneID            0
    exposedField   MFInt32       skinCoordIndex    [ ]
    exposedField   MFFloat       skinCoordWeight   [ ]
    exposedField   SFVec3f       endpoint          0 0 1
    exposedField   SFInt32       falloff           1
    exposedField   MFFloat       sectionPosition   [ ]
    exposedField   MFFloat       sectionInner      [ ]
    exposedField   MFFloat       sectionOuter      [ ]
    exposedField   SFInt32       rotationOrder     0
    exposedField   MFNode        children          [ ]
    exposedField   SFVec3f       center            0 0 0
    exposedField   SFRotation    rotation          0 0 1 0
    exposedField   SFVec3f       translation       0 0 0
    exposedField   SFVec3f       scale             0 0 0
    exposedField   SFRotation    scaleOrientation  0 0 1 0
    exposedField   SFInt32       IkchainPosition   0
    exposedField   MFFloat       IkyawLimit        [ ]
    exposedField   MFFloat       IkpitchLimit      [ ]
    exposedField   MFFloat       IkrollLimit       [ ]
    exposedField   MFFloat       IKTxLimit         [ ]
    exposedField   MFFloat       IKTyLimit         [ ]
    exposedField   MFFloat       IKTzLimit         [ ]
}
```

The SBBone node specifies four types of information, namely semantic data, bone geometric transformation, bone-skin influence volume and bone IK constraints. Let us detail each of these components.

During the animation, each bone can be addressed by using its identifier, *boneID*. This field is also present in the animation resource (file or stream). If two bones share the same identifier, their geometric transformations have the same set of parameter values.

The bone-skin influence volume described in Section 3.2.2.1 is implemented as follows: the specification of the affected vertices and the measure of affectedness are obtained by instantiating the *skinCoordIndex* and *skinCoordWeight* fields. The *skinCoordIndex* field enumerates the indices of all skin vertices affected by the current bone. The *skinCoordWeight* field is a list of values of affectedness measure (one for each vertex listed in *skinCoordIndex*). The influence volume specified with respect to a certain number of planes (discretisation step), is computed as follows. The *sectionInner* (respectively *sectionOuter*) field is a list of inner (respectively outer) radii of the influence volume corresponding to different planes. The *sectionPosition* field corresponds to the distance d . The *falloff* field specifies the choice of the measure of affectedness function as follows: -1 for x^3, 0 for x^2, 1 for x, 2 for $\sin(\frac{\pi}{2}x)$, 3 for $\sqrt{x}$ and 4 for $\sqrt[3]{x}$. The location of the bone is specified by the *center* and

endpoint fields. The possible 3D geometric transformation consists of (in this order): (1) (optionally) a non-uniform scale, (2) a rotation with respect to an arbitrary point and axis, and (3) a translation.

The *rotationOrder* field contains information related to the conversion from the quaternion representation, used in the node data, to the Euler angles-based representation, used in the animation resource (file or stream), and *vice versa*.

The IK information related to a bone refers to the position of the bone in a kinematics chain and to the definition of possible movement constraints of this one. Relative to the IK information one can distinguish the following cases:

If the bone is the root of an IK chain, then *IKchainPosition* equals 1. In this case, when the IK scheme is applyed only the bone orientation changes.

If the bone is the last element in the kinematics chain, then *IKchainPosition* equals 2. In this case, the animation stream has to include the final position of the bone (X, Y and Z in the world coordinates).

If *IKchainPosition* equals 3, the bone belongs to the IK chain, but it is not the first nor the last one in the chain. In this case, the position and the orientation of the bone are computed *via* the IK procedure.

Finally, if the bone does not belong to any IK chain (*IKchainPosition*=0), it is necessary to transmit the local transformation of the bone in order to be able to displace this one.

Let us give two simple examples in order to illustrate how the IK information is used during the animation.

If an animation stream contains motion information related to a bone with *IkchainPosition*=1, this information will be ignored.

If an animation stream contains motion information related to a bone with *IkchainPosition*=3, this means that the animator imposes the orientation of the bone and the IK solver will use this value as a constraint.

The IK constraints of a bone are related to its orientation and translation. Thus,

IkyawLimit (respectively *IkpitchLimit* and *IkrollLimit*) field consists in a pair of min/max values, which limit the bone rotation with respect to the X (respectively Y and Z) axis.

IKTxLimit (respectively *IKTyLimit* and *IKTzLimit*) field consists in a pair of min/max values, which limit the bone translation in the X (respectively Y and Z) direction.

The SBBone node is used as a building block in order to describe the hierarchy of the articulated virtual character by attaching one or more child objects. The children field has the same semantic as in MPEG-4. In order to support a dynamic change of the skeleton structure, the SBBone node includes the *addChildren* and the *removeChildren* fields. The global geometric transformation of a given child of a bone is obtained by composing the bone transformation with the parent one.

3.3.2 SBMuscle Node

The fields of the SBMuscle node are illustrated in Figure 3.7.

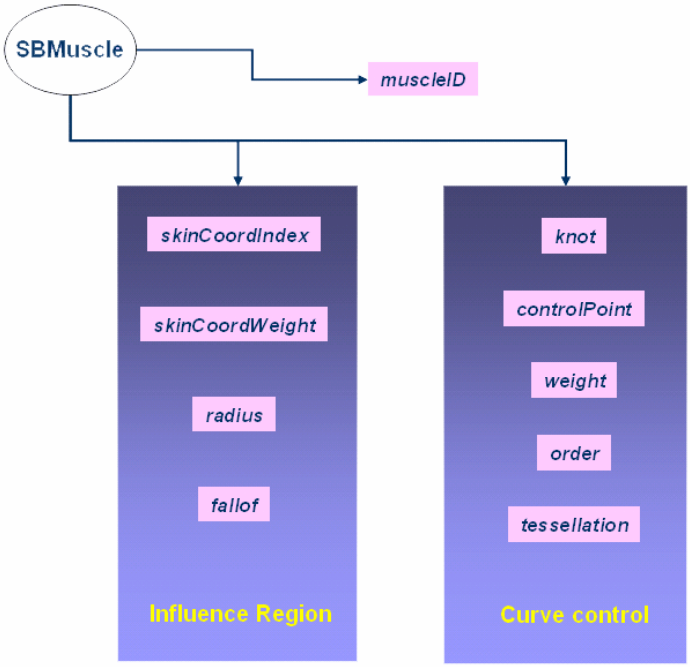


Figure 3.7. The fields of the muscle node.

The syntax of the SBMuscle node is defined as follows:

```
SBMuscle{
    exposedField   SFInt32      muscleID          0
    exposedField   MFInt32      skinCoordIndex    [ ]
    exposedField   MFFloat      skinCoordWeight   [ ]
    exposedField   SFNode       muscleCurve
    exposedField   SFInt32      radius            1
    exposedField   SFInt32      falloff           1
}
```

The SBMuscle node enables to add information relative to local deformation for simulating muscle activity at the skin level. The muscle influence volume described in Section 3.2.3 is supported as follows. Specification of the affected vertices and of the measure of affectedness is performed by instantiating the vertices list (*skinCoordIndex*) and the affectedness list (*skinCoordWeight*). The influence volume is computed by using the *radius* and *falloff* fields. The radius field specifies the maximum distance for which the muscle will affect the skin. The *falloff* field specifies the choice of the measure of affectedness function as follows: -1 for x^3, 0 for x^2, 1 for x, 2 for $\sin(\frac{\pi}{2}x)$, 3 for $\sqrt{x}$ and 4 for $\sqrt[3]{x}$.

The modelling of the muscle curve is based on NurbsCurve structure as defined in [BlaxxunNR]:

```
NurbsCurve {
    Field          MFFloat      knot            [ ]
    Field          SFInt32      order           3
    exposedField   MFVec3f      controlPoint          [ ]
    exposedField   MFFloat      weight          [ ]
    exposedField   SFInt32      tessellation
}
```

3.3.3 SBSegment Node

The SBSegment node syntax is specified as follows:

```
SBSegment{
    exposedField   SFString     name                 “”
    exposedField   SFVec3f      centerOfMass   0 0 0
    exposedField   SFVec3f      momentsOfInertia     [ 0 0 0 0 0 0 0 0 0 ]
    exposedField   SFFloat      mass                 0
    exposedField   MFNode       children       [ ]
    eventIn        MFNode       addChildren
    eventIn        MFNode       removeChildren
}
```

The *name* field must be present, in order to be able to identify the SBSegment during the animation.

The physical properties of a segment are defined by its *mass* (the total mass of the segment), its *centerOfMass* (the location of the center of mass of the segment) and its *momentsOfInertia* (the matrix of the moment of inertia) fields.

The *children* field can be an arbitrary object, including a SBSkinnedModel.

A SBSegment node is a grouping node that we introduce in order to provide the following functionalities:

1. to be able to separate different parts of the skinned model into parts which are independent relative to deformation. Two parts are said to be independent relative to deformation if a geometrical transformation of the first one does not involve skin deformation of the second one and *vice versa*. This is a crucial requirement for the optimization of the animation. For achieving this independence, SBSegment node may contain a SBSkinnedModel node as a child component. To exemplify this concept one can consider the fingers of a humanoid virtual character together with the hand and with a part of the forearm as a unique 3D object, segmented from the rest of the body. The upper part of the forearm and the rest of the body can change the pose of the considered object, but cannot deform it.
2. to be able to attach stand-alone 3D objects on different parts of the skeleton hierarchy. For example, a ring can be attached to a finger; the ring geometry and its attributes are defined outside the skinned model but the ring will have the same local geometrical transformation as the attached bone.

3.3.4 SBSite Node

The syntax of the SBSite node is specified as follows:

```
SBSite{
    exposedField   SFVec3f      center               0 0 0
    exposedField   MFNode       children                     [ ]
    exposedField   SFString     name                         ""
    exposedField   SFRotation   rotation             0 0 1 0
    exposedField   SFVec3f      scale                1 1 1
    exposedField   SFRotation   scaleOrientation            0 0 1 0
    exposedField   SFVec3f      translation          0 0 0
```

```
        eventIn        MFNode        addChildren
        eventIn        MFNode        removeChildren
}
```

The SBSite node indicates a predefined 3D point, not necessarily belonging to the skin, which is usually used to localize an end-effector. This point is obtained by using the *center*, *rotation*, *scale*, *scaleOrientation* and *translation*. The *children* field is used to store any object which can be attached to the SBSegment node.

The SBSite node can be used to:

1. define an "end effector", *i.e.* a location which can be used by the inverse kinematics solver,
2. introduce an attachment point for accessory objects such as clothes, or
3. specify the location of a virtual camera in the reference frame of the SBSegment node.

Any SBSite node is stored in the *children* field of a SBSegment node. Note that, the SBSite node is a specialized grouping node that defines a coordinate system associated to the nodes belonging to its *children* field. This local coordinate system is defined relative to the SBSite node coordinate system. The SBSite node is considered as a specialized grouping node because it can only be defined as a child of an SBSegment node.

3.3.5 SBSkinnedModel Node

The main fields of the SBSkinnedModel node as well as a typical skeleton and skin structure are illustrated in Figure 3.8.

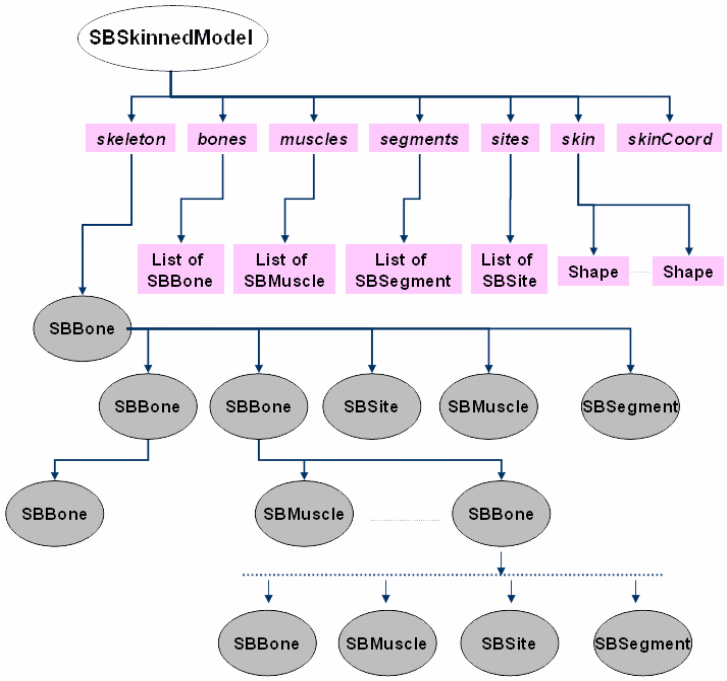


Figure 3.8. The fields of the SBSkinnedModel node.

The SBSkinnedModel node syntax is specified as follows:

```
SBSkinnedModel{
    exposedField  SFString        name                    ""
    exposedField  SFVec3f         center                  0 0 0
    exposedField  SFRotation      rotation                0 0 1 0
    exposedField  SFVec3f         translation             0 0 0
    exposedField  SFVec3f         scale                   1 1 1
    exposedField  SFRotation      scaleOrientation        0 0 1 0
    exposedField  MFNode          skin                    [ ]
    exposedField  SFNode          skinCoord               NULL
    exposedField  SFNode          skinNormal              NULL
    exposedField  MFNode          skeleton                [ ]
    exposedField  MFNode          bones                   [ ]
    exposedField  MFNode          muscles                 [ ]
    exposedField  MFNode          segments                [ ]
    exposedField  MFNode          sites                   [ ]
    exposedField  SFNode          weighsComputationSkinCoord  NULL
}
```

The SBSkinnedModel node is the root used to define an SMS virtual character and it contains the definition parameters of the entire seamless model or of a seamless part of the model. Specifically, this node contains the following data:

- the *name* field specifies the name of the skinned virtual character, allowing thus a simple identification during the animation stage.
- the parameters of a geometric transformation which allows to place the character in the scene. This geometric transformation has a generic affine form and it is specified by the following fields: *center*, *translation*, *rotation*, *scale* and *scaleOrientation*;
- a list *skinCoord* containing the 3D coordinates of all the vertices of the seamless model skin;
- a list of shapes used to build the character skin. The *skin* field consists of a collection of shapes which share the same *skinCoord*. This mechanism allows to consider the model as a continuous mesh and, in the same time, to attach different attributes (*e.g.* colour, texture, etc.) to different parts of the model;
- a specification of the skeleton hierarchy. The *skeleton* field contains the root of the bone hierarchy;
- a list containing all the bones, segments, sites, and muscles belonging to the character. The *bones*, *segments*, *sites* and *muscles* fields contain respectively, the lists of all SBBone, SBSegment, SBSite and SBMuscle nodes, previously defined;
- the *weighsComputationSkinCoord* field describes a given static pose of the skinned model. This pose is used in the initialization step in order to compute the bone and the muscle influence volumes;

3.4 Skeleton, Muscle, and Skin animation stream

3.4.1 Animation principle and resource representation

Animating the SMS model consists in updating, at each frame, the bones and muscles fields associated to their geometric transformations.

To address streamed animation, we consider that the animation data is independent of the model parameters. Thus, the model is transmitted or loaded at the beginning of the animation session and the animation parameters are sequentially transmitted at each frame. We propose two representation techniques for the animation data format: the first one corresponds to a non-compressed (human readable) data. This is useful when editing the animation parameters. In this case the file format is XMT [Kim00] compliant in order to allow easy editing and data exchange. We call this representation "SMS textual". The second representation is a compressed data format. By using appropriate compression schemes, we are able to perform low bit-rate animation transmissions. We call this representation "SMS binary".

Conceptually, both animation data formats use the same representation of the geometrical transform parameters. The next sections describe this representation.

3.4.2 Animation parameter representation

The key point for representing the SMS animation parameters consists in decomposing the geometric transformations into elementary motions. Thus, when only using, for example, the rotation component of the bone geometric transformation, a binary mask indicates that the other components are not involved. In order to deform a muscle by translating a control point, a binary mask has to specify that weight factors and basis functions are not used. Since the animation system does not use systematically all the elements of the transformations associated with bones and muscles, this approach produces a very compact representation of the animation stream. Moreover, the compactness of the animation stream can be still improved when dealing with rotations. During the animation, the rotation of a bone with respect to its parent is usually the main technique used. In the definition of the bone node, the rotation is represented as a quaternion. However, many motion editing systems use the rotation decomposition with respect to the Euler's angles. In practice, when less than three angles describe a joint transformation, an Euler's angle-based representation is more appropriate. Thus, to get a more compact animation stream, a rotation is represented, in the animation resource, as Euler's angles-based decomposition.

In [Craig89], it is shown that there are 24 different ways to specify a rotation by using a triplet of angles. By introducing a parameter characterizing the 24 possible combinations of the Euler's angles, Shoemake [Shoemake94] demonstrates that there is a one to one mapping between the quaternion (or rotation matrix) representation and the pair given by the Euler's angles and the introduced parameter. In order to take into account this remark we have introduced into the bone node a parameter called *rotationOrder*.

A triplet of Euler's angles $(\theta_1, \theta_2, \theta_3)$ describes the rotation of a local coordinate system r with respect to a static coordinate system s, *i.e.* how the bone coordinate system changes with respect to its parent coordinate system. This triplet parameterises the rotation as:

$$\rho = \rho_{\theta_3, A_3} \circ \rho_{\theta_2, A_2} \circ \rho_{\theta_1, A_1}, \tag{3.7}$$

where ρ_{θ_i, A_i} represents a rotation of angle θ_i with respect to the axis A_i, $i = 1,2,3$ with $A_2 \neq A_1$ and $A_2 \neq A_3$. When dealing with rotations with respect to the axes *X*, *Y* and *Z* of the system coordinate, 12 combinations are possible [Shoemake94]. Expressing these rotations both in the bone coordinate system r and in its parent coordinate system s, there are 24 possible values for *rotationOrder*. Shoemake demonstrates that by permuting, negating and swapping the columns of the rotation matrix, the number of independent configurations decreases from 24 to 2 cases.

For the rest of the bone transformation components (translation, scale, etc.) the representation in the animation resource is identical to the representation in the nodes.

3.4.3 Temporal frame interpolation

The issue of temporal frame interpolation has been often addressed in the computer animation literature [Foley92, ORourke]. From simple linear interpolation, appropriate for translations, to more complex schemes based on high degree polynomials, or on quaternions, in order to take into account the orientation, a large number of techniques are available. The advantages and the disadvantages of each one are well-known. Many of these techniques are supported by most of current animation software packages. Temporal frame interpolation is intensively used to perform animation from textual description or from interactive authoring. Relative to our framework, since the animation parameters can also be specified to the key-frames and not only frame by frame, real-time temporal interpolation methods are required to generate the animation data for all the frames by using the information specified in the key-frames.

For real-time purposes, we use a linear interpolation of the *translation* and *scale* components:

$$l(x) = (1 - x)x_0 + xx_1, \tag{3.8}$$

where x_0 is the translation/scale in the current key-frame, x_1 is the translation/scale in the next key-frame, and $x \in [0,1]$.

On the other hand, spherical linear quaternion interpolation is used for the *rotation* and *scaleOrientation* components:

$$Slerp(x) = \frac{x_0 \sin[\alpha(1-x)] + x_1 \sin(\alpha x)}{\sin(\alpha)}, \tag{3.9}$$

where x_0 (respectively x_1) is the rotation/scale orientation quaternion in the current (respectively next) key-frames, and $\alpha = \arccos(x_0 x_1)$ and $x \in [0,1]$.

3.4.4 Animation frame

In a SMS file or stream, for each key frame, we specify two types of information:

- a vector corresponding to the animation mask, called *animationMaskVector* which indicates the components of the geometrical transformation to be updated in the current frame;
- a vector corresponding to the animation values called *animationValueVector* which specifies the new values of the components to be updated,

detailed below.

animationMaskVector

In the animation mask of a key-frame, a positive integer *KeyFrameIndex* indicates to the decoder the number of frames which have to be obtained by temporal interpolation. If this number is zero, the decoder sends the frame, directly to the animation engine. Otherwise, the decoder computes *n* intermediate frames (*n=KeyFrameIndex*) and sends them to the animation engine, together with the content of the received key-frame.

Some bones or muscles of the SMS virtual character may not be animated in all frames. The *boneIDs* and *muscleIDs* of the updated bones and muscles, respectively, are parts of the *animationMaskVector*. In addition, *animationMaskVector* contains the animation mask of each bone, *boneAnimationMaskVector*, and the animation mask of each muscle, *muscleAnimationMaskVector*. These vectors are detailed below.

animationValueVector

animationValueVector contains the new values of each bone and muscle geometric transformation that have to be transmitted and it is obtained by concatenation of all the *boneAnimationValueVector* and *muscleAnimationValueVector* fields, detailed below.

For compression efficiency, SMS stream specifications limits the maximum number of bones and respectively, of muscle nodes to 1024. These bone and muscle nodes can belong to one or more skinned models and are grouped in a SBVCAnimation node. Thus, the fields *boneID* and *muscleID* must be unique in the scene graph and theirs values lie in the interval [0, …,1023].

boneAnimationMaskVector

To address high compression efficiency, we use a hierarchical processing of the bone motion as follows. At the first level, the bone motion is decomposed into translation, rotation, scale, scale orientation and center transformation. At the second level, all of these components that are set to 1 in the bone mask, are individually decomposed in elementary motions (*e.g.* translation along the *X* axis, rotation with respect to *Y* axis, etc.). This hierarchical processing makes it possible to obtain short mask vectors.

The size of the *boneAnimationMaskVector* can vary from 2 bits (corresponding to a single elementary motion) to 21 bits (all the components of the local transformation of the bone change with respect to the previous key-frame).

boneAnimationValueVector

boneAnimationValueVector contains the values to be updated corresponding to all elementary motions with a mask value of 1. The order of the elements in *boneAnimationValueVector* is obtained by analyzing *boneAnimationMaskVector*.

muscleAnimationMaskVector

The muscle animation parameters in the SMS stream are coordinates of the control points of the NURBS curve, weights of the control points or/and knot values.

The number of control points and the number of elements of the knot sequence are integers between 0 and 63 and they are encoded in the *muscleAnimationMaskVector*, after the *muscleID* field. As in the case of bone, a hierarchical processing is used to represent the mask.

muscleAnimationValueVector

muscleAnimationValueVector contains the new values of the muscle animation parameters. As in the case of bone, this vector is ordered according to the *muscleAnimationMaskVector.*

In order to allow easy data exchange and editing, we specify XML-compliant representations for *boneAnimationMaskVector*, *boneAnimationValueVector, muscleAnimationMaskVector, muscleAnimationValueVector* and also for the animation mask of the entire frame, *animationMaskVector* and the associated animation values, *animationValueVector*.

The XML compliant representation of the SMS file is presented in Appendix B.

Since the compression schemes developed in the MPEG-4 FBA framework offer good performances, we have adapted the two promoted algorithms (predictive- and DCT-based) for compressing the SMS animation file.

3.5 Experimental results

In order to evaluate the performance of the proposed compression schemes, we have converted the test data generated for the evaluation of the FBA framework, as described in Section 2.4.1, into a SMS compliant format. The specificity of this data set consists in the lack of muscle deformation; moreover dealing with a human-like model, only rotations are allowed as bone transformations. By encoding the data set after the conversion step we have obtained the results presented in Table 3.1.

	Q=1		Q=2		Q=4		Q=8		Q=16		Q=31	
	P	DCT	P	DCT	P	DCT	P	DCT	P	DCT	P	DCT
A	1.45	3.01	1.46	2.66	1.38	1.94	1.30	1.62	1.24	1.22	1.21	0.94
B	1.51	2.87	1.41	2.41	1.42	1.73	1.42	1.45	1.45	1.19	1.34	0.85
C	1.93	3.92	1.98	3.47	1.84	2.45	1.83	2.04	1.77	1.53	1.73	1.15
D	1.55	2.89	1.52	2.42	1.44	1.83	1.42	1.53	1.42	1.05	1.45	0.82
E	1.86	3.42	1.80	3.15	1.78	2.31	1.74	1.89	1.73	1.44	1.67	1.09
F	1.42	2.24	1.41	2.12	1.43	1.43	1.36	1.24	1.32	0.93	1.33	0.74
G	1.97	3.09	1.86	2.65	1.85	1.94	1.82	1.57	1.72	1.24	1.72	0.94
H	1.90	3.01	1.82	2.52	1.88	1.82	1.77	1.52	1.71	1.14	1.75	0.85
I	1.62	2.99	1.68	2.56	1.53	1.83	1.53	1.56	1.54	1.17	1.45	0.82
J	1.43	2.34	1.42	2.07	1.45	1.42	1.38	1.23	1.37	0.82	1.32	0.66

Table 3.1. Bit-rates [kbps] corresponding to frame predictive-based and DCT-based encoding schemes applied to "A" to "J" signs data.
Q denotes the global quantization value. The frame-rate is set to 10 fps.

A comparative analysis of the obtained results shows that the use of a SMS representation leads to a bit-rate slightly greater (max 10%) than the bit-rate obtained by using the FBA. This effect is mainly due to the differences between the different representations used for the mask. While the FBA uses a standardized order to associate the mask elements with the avatar joints, in the SMS an identifier must be transmitted in order to accomplish this task. This overload is the cost for allowing the definition and the animation of a generic skeleton structure (other skeleton than humans can be animated using SMS). Moreover, the SMS bone/muscle addressing mechanism allows also to carry into the same stream, the animation information referring to bones and muscles belonging to different 2D/3D synthetic objects.

Note that the SMS stream can carry the information containing the ranges of the animation parameters. For this particular experiment, when performing the frame predictive-based algorithm, we have used the converted value ranges of the angle value ranges standardized in the FBA. In the general case, the compression ratio obtained by both SMS encoding schemes strongly depends on the skeleton complexity, the number of muscles used and the motion complexity. We have tested the two compression schemes (DCT- and frame-based) and we have obtained, when considering a human-like virtual character and natural motion (generated from a motion capture system), a bit rate of 25 kbps while keeping a good motion quality [ViSiCAST]. Moreover, specific encoding techniques allow to achieve better results. Let us give two examples.

Since the SMS model does not impose the structure of the skeleton to be animated, there is no possible to fix, independently of the model to be animated, the ranges for animation parameters. However, an efficient use of the compression methods based on quantization of an interval requires an adaptation of this interval with respect to the values to be encoded. For this reason, into the SMS

stream, it is possible to transmit, at the beginning of the animation, and from time to time (in the case of a bradcast scenario), this set of ranges.

On the other hand, the use of the temporal interpolation described in Section 3.4.3, allows to reduce the bit-rate. In order to evaluate the performances of the interpolation-based encoding we have subsampled the data set from the previous experiment by keeping only key-frames. The key-frames are obtained by an analysor, which takes into account the variations of the motion parameters. If all the rotations and translations corresponding to a frame can be obtained by temporal interpolation from the previous and the next one, the frame is skipped and a flag is transmitted instead. The principle of the analysor in the case of one parameter is illustrated in Figure 3.9. The first picture contains all the values for a parameter for all the frames; since a value at certain moment in time can be obtained by linear interpolation from the values just before and after this moment of time, this value is not transmitted anymore. By analysing the entire sequence for all the parameters, only the useful frames are keeped, compressed and transmitted.

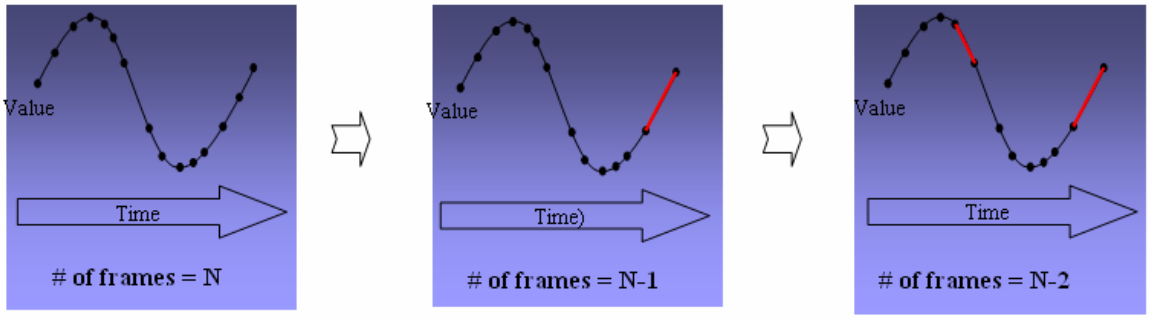


Figure 3.9. Principle of the analysor (illustration for one parameter).

Depending on the data set, this procedure can reduce the data size up to 40%. Table 3.2 shows the compression results, applied on the subsampled data set.

Sign	Q=1		Q=2		Q=4		Q=8		Q=16		Q=31	
	P	DCT	P	DCT	P	DCT	P	DCT	P	DCT	P	DCT
A	1,04	2,12	1,03	1,85	0,95	1,34	0,92	1,15	0,88	0,86	0,83	0,65
B	1,02	1,91	0,93	1,60	0,94	1,15	0,94	0,95	0,98	0,81	0,92	0,56
C	1,15	2,33	1,16	2,07	1,12	1,48	1,11	1,23	1,05	0,93	1,03	0,68
D	1,03	1,92	1,01	1,61	0,93	1,23	0,92	1,04	0,96	0,66	0,94	0,53
E	1,12	2,14	1,15	1,94	1,13	1,45	1,07	1,16	1,08	0,86	1,05	0,67
F	0,93	1,44	0,93	1,35	0,95	0,96	0,87	0,83	0,84	0,60	0,86	0,47
G	1,44	2,33	1,42	1,96	1,38	1,49	1,38	1,17	1,26	0,97	1,27	0,73
H	1,34	2,13	1,28	1,77	1,34	1,26	1,27	1,07	1,20	0,82	1,22	0,61
I	1,02	1,91	1,07	1,66	0,99	1,17	0,97	1,00	0,98	0,77	0,96	0,57
J	0,91	1,46	0,87	1,32	0,93	0,87	0,88	0,77	0,86	0,53	0,84	0,44

Table 3.2. Bit-rates [kbps] corresponding to frame predictive-based and DCT-based encoding schemes coupled with temporal interpolation, applied to "A" to "J" signs data. Q denotes the global quantization value. The frame-rate is set to 10 fps.

In order to guarantee the adaptability of the SMS stream to the transmission bandwidth and to the terminal performances, we have introduced in the animation data the following streaming mechanisms.

- At the beginning of each frame we introduce a specialized start code. When dealing with a server pushing animation data over a network, if the available bandwidth is smaller than the animation bitrate, only the beginning of the frame is transmitted. The presence of this start code allows the server to synchronize the transmission without decoding the information. The same mechanism allows to maintain the syncronization on the end-user terminal.
- Since muscle animation requires at the terminal level more complex computations than bone transformation, we have introduced another start code indicating the presence of muscle-related data in the stream. In the case of a low computation performance terminal the decoder will skip this latter data.

The geometric transformation representation together with the compression schemes proposed in the SMS framework allow to perform and transmit realistic animation onlow-bit-rate networks (less than 64 kbps). Thus we have provided here a new framework that overcomes the main FBA limitation mentioned in Chapter 2, specifically, the realistic animation performance and exploits and improve the compression methods defined for FBA. The next section details the main differences between these two frameworks.

3.6 SMS versus FBA

A comparative analysis of the FBA and SMS frameworks is synthesized in Table 3.3 below.

Criteria	FBA	SMS
Model type	Virtual human character	Generic virtual character
Geometry definition	Segmented character	Seamless character
Hierarchy	Standardized Hierarchy	Hierarchy build on a generic skeleton
Local deformation	Cluster based for face, deformation tables for body	Curve-based deformation
Scene graph nodes	Define a Face and Body Node and use H-Anim PROTOs for specifying the model geometry	Define a own set of 6 nodes
Animation parameters	296 for body, 68 for face	Undefined number of parameters, arbitrary number of bones and muscles are supported
Animation editing support	Forward kinematics	Forward kinematics, inverse kinematics, temporal frame interpolation
Compression	Frame predictive-based, DCT based	Frame predictive-based, DCT based

Table 3.3. Main FBA and SMS features.

While the FBA is founded on the representation of avatars as segmented characters, that makes it an appropriate framework for cartoon-like applications, SMS offers a higher degree of realistic representation, dealing with the concept of skeleton-driven animation.

When dealing with the avatar body, the FBA standardizes a fixed number of animation parameters (296) by attaching to each anatomical segment up to three rotation angles. The SMS framework does not limit the number of animation parameters (bones and muscles). Moreover, the animation parameters refer to an extended set of geometrical transformations (rotations, translations, scaling factors).

Shape deformations are present in both frameworks. For example, the FBA standardizes a number of control points in order to perform facial deformations, while the SMS allows to add curve-based deformers at any level of the skin. In FBA the deformation tools are cluster-based and in SMS curve-based. The FBA standardizes the number and location of the control points, while SMS gives this freedom to the designer, being thus possible to achieve muscle-like deformations on any part of the virtual character skin in order to get a realistic animation.

Both frameworks address streaming animation and provide low-bit-rate compression schemes. Both FBA and SMS allow the above-mentioned compression methods, frame-based and DCT-based. Moreover, to improve the compression ratio, SMS supports advanced animation techniques such as temporal frame interpolation and inverse kinematics. For both FBA and SMS, the bit-rate of the compressed stream depends on the movement complexity (number of segments/joints involved in motion) and generally lies in the range of 5-40 kbps for a frame rate of 25fps.

In the FBA framework, the animation stream contains information relative to the animation of a single human virtual character, while, in the SMS framework, it is possible to animate several characters by using a unique stream. Moreover, the SMS allows to define and animate generic 2D/3D objects. This property is very useful when dealing with a scene where a large number of avatars/generic objects are present.

In the SMS animation, more complex computations are required than in the case of the FBA one. Thus, concerning the terminal capabilities, dedicated 3D hardware or software optimization is well-suited for implementing SMS animation. However, the SMS deformation mechanism is in line with the development of graphics APIs and graphics hardware.

3.7 The SMS framework in the MPEG-4 Context

Since the first version of the MPEG-4 Requirements document, issued in 1996, the need for animation tools of synthetic objects has been formulated. In October 2000, MPEG-4 has published the version 15 of the “MPEG-4 Requirements” document [Pereira00]. In the section dedicated to 3D graphics tools, the document refines the requirement that the MPEG-4 standard shall support animation (Figure 3.10). More precisely, the document formulates the requirement to support “efficient representation of animation” through “2D and 3D Shape deformation” and through “dynamically configurable animation streams”.

……

4.9.2 Animation support

Requirement

The standard shall provide efficient representations of animation.

Specification

The standard shall support:

Curve based animation

2D and 3D Shape deformation

Dynamically configurable animation streams

Physically based animation

……

Figure 3.10. Extract from the "MPEG-4 Requirements, version 15 (La Baule revision)".

In January 2001, a "Call for Proposal" for technologies supporting these requirements has been issued. All the technologies developed in response to this "Call for Proposal" have been developed under the name "Animation Framework eXtension" (AFX).

As a response to this "Call for Proposal", we have submitted in March 2001 to the MPEG-4 SNHC[7] group the animation of a generic articulated model [Preda01-a]. More precisely, we have proposed the bone-based deformation controller applied to a seamless representation of a generic 3D object. At the same time, we also have formulated the need of an enriched animation support. Specifically, we have shown the advantages of using temporal interpolation and IK in order to facilitate the animation editing and to enable a more compact representation of the animation parameters. The quality of the preliminary results in terms of realism of representation, decided the SNHC group to open a Core Experiment (CE) called "Bone-based animation for generic articulated models" [Sevenier01-a]. The purposes of the CE were:

- to specify a node syntax supporting the definition of a generic articulated model;
- to investigate the advantages of using temporal interpolation and IK, with respect to existing MPEG-4 tools, *e.g.* BIFS-Anim;
- to investigate a possible update of the FBA stream in order to take into account the animation of a generic synthetic object.

The participants to the CE were INT[8] and Superscape[9]. During the standardization process, different techniques have been evaluated by cross-checking between INT and Superscape.

[7] SNHC - Synthetical and Natural Hybrid Coding, a MPEG-4 working group which deals with providing specifications for 3D graphics
[8] INT – Institut National des Télécommunications, Evry, France

In the next step, achieved in July 2001, we have added IK support in the bone node [Preda01-b]. Due to the fast evolution of the hardware performances, it was not recommended that the standard imposes a specific IK solver. Supporting IK consists in proposing a specific representation of the inverse kinematics constraints.

In mid-2001, the H-Anim group published an update of the specifications in order to take into account a seamless representation of a human avatar. Consequently, we have proposed in July 2001 [Preda01-c] to adapt the interfaces of the already-defined nodes to the notations proposed by H-Anim. This adaptation consists in using the same names for the node fields which are common in both frameworks. The major differences between H-Anim specifications and SMS framework are the following:

- H-Anim allows the animation and definition of the human avatars, while SMS deals with generic models,
- H-Anim supports only the bone controller, while SMS supports bone and muscle controllers,
- Animation of H-Anim avatars is only possible by using VRML interpolators and can not be streamed, while SMS animation can be streamed and in addition, efficient compression techniques can be used.

At this date, the integration of the SMS nodes in the MPEG-4 reference software was formulated as a new goal within the CE.

The results of the CE with respect to the efficiency of using BIFS-Anim or an extended form of the FBA stream, led us to propose in October 2001, at the *ad-hoc* meeting hold in Rennes (France), a dedicated streaming syntax for bones-based animation [Preda01-d]. At that time, we have presented the animation results produced by the implementation of the SMS concepts [Preda01-e]. Our implementation is part of the MPEG-4 reference software. The discussions carried out in the SNHC group revealed the need of a local deformation tool, which can be applied to both 2D and 3D synthetic objects. A new CE was started with the goal to investigate the efficiency of using the bone controllers for such effects. The participants to this CE were INT, Superscape and Vimatix[10].

As response to this CE, in December 2001, we have demonstrated that efficient local deformation can be efficiently address with a new deformation controller, *i.e.* the generic curve-based controller [Preda01-f]. The evaluation of this controller was carried out by INT and Superscape (for 3D objects) and Vimatix (for 2D objects), and the positive results then obtained decided the SNHC group to accept it as part of the MPEG-4 specifications.

Since one of the goals of the CE was to investigate on compressed animation, we have conducted experiments on using BIFS-Anim to carry SMS animation. The poor results obtained by using the BIFS-Anim compression scheme, led us to propose a new stream syntax [Preda01-g]. Mainly, this stream is an adaptation of the predictive- and DCT-based compression methods.

In May 2002, we have proposed [Preda02-a] a mechanism to integrate the SMS animation stream into an MPEG-4 stream. Two scenarious have been addressed: in-band and out-band.

[9] SUPERSCAPE Inc., England
[10] Vimatix Inc., Israel

The in-band solution consists in using the generic BIFS-Anim stream and encapsulating data from the SMS stream, namely the animation mask and the animation values as parts of the BIFS-Anim stream. A BIFS-Anim stream is designed to animate objects in an MPEG-4 scene. Two approaches are supported in BIFS in order to perform the animation [ISOIEC01]: (1) to directly encode the updated fields values and (2) to include dedicated animation streams. In the SMS framework, good compression results are obtained by using the second approach. Thus, for compression performance reasons, we have recommended to use this method.

Within the out-band solutions, a dedicated animation stream is always associated with a unique dedicated node. The fact that each synthetic object animated according to the SMS framework must to have attached a dedicated stream becomes a substantial limitation in terms of stream management (distribution, synchronization), when dealing with a large number of synthetic objects. In order to overcome this limitation, we have implemented the so-called SBVCAnimation node, which is linked to the animation stream and groups a set of virtual characters. It has the following interface:

```
SBVCAnimation {
    exposedField   MFNode      virtualCharacters      [ ]
    exposedField   MFURL       url                    [ ]
}
```

where *virtualCharacters* is the list of all the SMS virtual characters animated from the same resource and *url* allows to localize this resource.

The last effort for promoting the SMS framework in MPEG-4 was to specify a non-compressed (human-readable) format to specify the animation parameters. It allows to simple edit and interchange the animation data. A textual format was first proposed [Preda02-b] and, at the recommendation of the SNHC group, a XMT-compliant format was issued [Preda02-c].

Between March 2001 and December 2002, the SMS framework has been evaluated within the SNHC group. The implementation of the related scene graph nodes and animation stream have been carried out in the MPEG-4 reference software. As a result, the framework is part of the Amendment 4 of the MPEG-4 Systems standard. In the MPEG-4 standard specifications, the SMS framework is usually refered as BBA (Bone-Based Animation).

3.8 Summary

The purpose of this chapter was to introduce the main concepts of the Skeleton, Muscle and Skin framework. By introducing a generic mathematical model for mesh deformations, we have proposed two controllers (bone and muscle) as the core of this framework. Generally, a controller is defined as a triplet composed of a geometric support, an influence volume around this support and a measure of affectedness within the influence volume. We have shown how the proposed controller can be used for defining and animating generic virtual characters in terms of *bones* and *muscles*. In order to embed generic virtual characters into a 3D scene we provide a SMS architecture with scene graph nodes. We addressed then the issue of an efficient and compact representation of the animation parameters by (1) enriching the animation capabilities with temporal interpolation and inverse kinematics support and (2) adapting two data compression techniques to the SMS animation stream, namely predictive- and DCT-based. An experimental evaluation on synthetic data sets shows that the SMS framework offers a

good support for performing the transmission of a realistic virtual character animation over low-bit-rate networks. A comparative evaluation between SMS and FBA frameworks shows the advantages of the first one. Finally, we have described how we have promoted the SMS framework in the MPEG-4 standard.

Chapter Four

MPEG-4 at Work: Sign Language Transmission over Digital Television and Web-based Networks

The main application that drives our work is the sign language transmission over various networks and different types of terminals. The first section of this chapter gives a brief overview of the sign language characteristics and presents the social context which motivates the research interest on this topic. The next section is a detailed presentation of several proposed solutions for sign language transmission over several kinds of networks that we analyse with respect to some key criteria such as the required equipment, real-time capabilities, reusability of the content or necessary bandwidth.

4.1 Introducing sign language communication

The number of people suffering from some degree of hearing loss is estimated at 8.7 million in Britain and 4 million in France, including, 600 000 and 800 000 deafs from birth, respectively [RNID]. In order to communicate, this community uses the sign language which is specific to each country. While face-to-face communication between two hearing impaired people is highly efficient, distant communication systems require special features that cannot be found on mainstream devices such as telephones, for example. Aware of the difficulties to build the infrastructures needed to allow equal access to multimedia contents for this community, legislative bodies across Europe [EUCouncil02] promote technological developments in this field.

Web-based communication became one of the preferred systems, but the main limitation concerns the available bandwidth.

Other than peer-to-peer communication, a system allowing the access of hearing impaired people to the huge multimedia contents should be available. In order to support this feature, such a system has to take into account the visual information to be delivered. Another requirement is that this additional information could be switched on and off by the user at the terminal level.

There are two main approaches for a sign language communication system which are analyzed in the following sections. On the one hand, video-based representation of the signer can be differentially compressed and transmitted. While not very expensive in terms of production costs, this approach does not allow low bit-rate transmissions, as required for phone line-based communication systems (max 64 kbps). A second approach is based on the synthetic representation of the signer, transmission of the motion parameters and reconstruction of the gestures at the end-user. This approach allows very low bit-rate communication and fulfils advanced requirements such as reusability (for the producer) and interactivity (for the user), but the cost of the production tools remains high.

4.2 Analysed solutions for sign language communication

According to the data representation (natural/synthetic) along the transmission chain (producer/network/user), we propose four classes of solutions to transmit sign language content over digital television, wireless and web-based networks:

- Video object-based encoding of a natural signer,
- Video object-based encoding of a virtual signer,
- FBA compliant human avatar,
- SMS compliant human avatar.

We present in the following the major aspects of the first two solutions (the third and the fourth ones have been addressed in detail in Chapter 2 and Chapter 3, respectively). We also perform a comparative study on their advantages and limitations. As, in our opinion, the compliance with an open standard is one of the main requirements of a communication system, we show how the proposed solutions can be implemented with the MPEG-4 standard. In this study, the main criterion is the

required bit-rate for sign language transmission. However, we also study the implications with respect to producing and consuming the content.

4.2.1 MPEG-4 video object-based encoding of a natural signer

Video transmission of sign language content is widely reported in the literature and is a natural extension of the well-known application of visioconference. For our test purposes, we have chosen two sign language sequences: “Florea” and “Carlos” [CEPSAS]. Both are recorded in studio conditions, that is, with a near-uniform background as well as good lighting conditions.

There are three versions of this video-based solution, according to the implementation of a preliminary segmentation step. The simplest one considers the whole image and encodes it as a single object. We call this solution NSV (Natural Signing Video). While very easy to implement, this solution completely ignores the specific nature of a signing sequence. Usually, in a signing sequence there is a single signer and the background does not change over the time; the shot is fixed and comprises both hands, the head and the upper body of the signer as illustrated in Figure 4.1 and Figure 4.3. Such specific characteristics of a signing sequence, imposing only some illumination restrictions with respect to the background colour, easily allow performing a segmentation of the image into two objects, *i.e.* the signer and the background. This kind of colour-based segmentation, often used in television production, makes it possible to set in real-time the action in a customized background. In a sign language sequence, the useful information, *i.e.* the signer’s shape, occupies less than half of the image bounding box, as the signing space is larger than the body shape. By encoding only the information corresponding to the signer’s texture and shape and ignoring the rest of the image, the required bit rate decreases, while keeping the same visual quality. Please note that when the background of the captured sequence is uniform, such a selective coding brings only a gain up to 10% in bandwidth. In this context, the segmentation presents an interest mainly for applications dealing with 2D scene composition. Figure 4.2 and Figure 4.4 illustrate the segmentation masks (including two objects, the signer and the background) of the sequences presented in Figure 4.1 and Figure 4.3, respectivelly. We call this second solution NSV2O (Natural Signing Video with 2 Objects).

Figure 4.1. Natural signer: the “Florea” sequence.

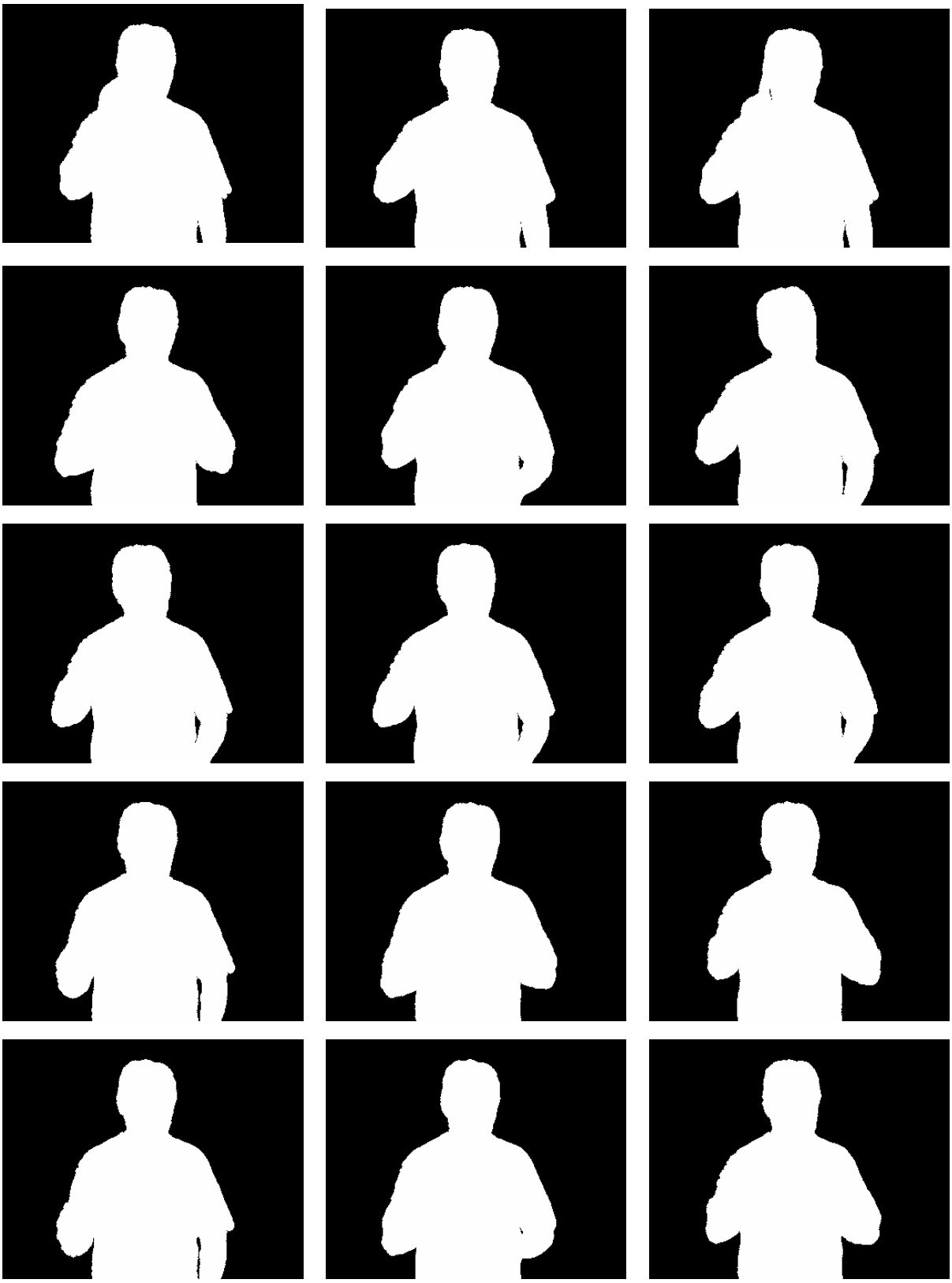


Figure 4.2. Binary segmentation mask (the signer - white - and the background - black) for the "Florea" sequence.

Figure 4.3. Natural signer: the “Carlos” sequence.

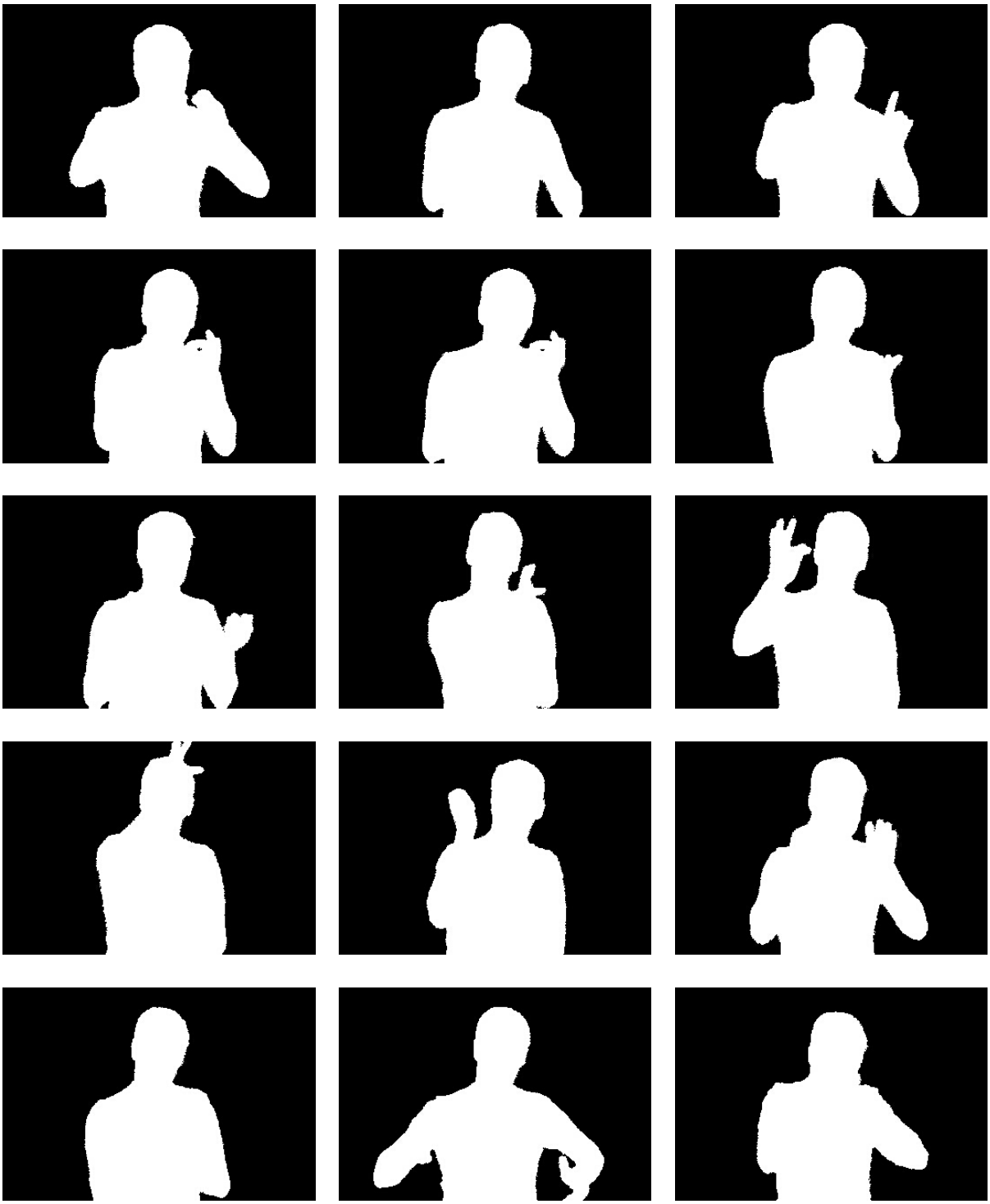
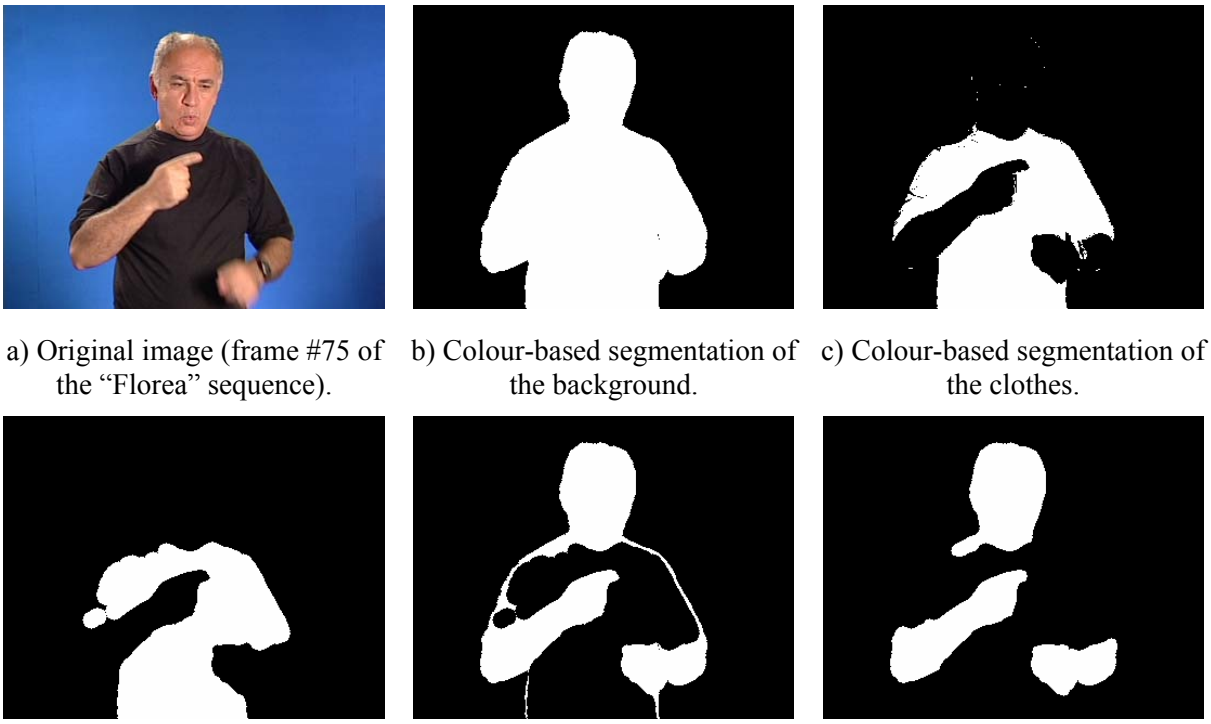


Figure 4.4. Binary segmentation mask (the signer - white - and the background - black) for the "Carlos" sequence.

The third version of the segmentation-based encoding method consists in attaching to each component of the signer's shape different importance. The visual quality of the face and hands should be kept high in order to provide an intelligible message, while the texture mapped on the signer's body has lesser importance. In this case, the segmentation into several objects, *i.e.* the head, the left hand, the right hand, the body and the background, is required. We call this solution NSV3O – Natural Signing Video with 3 Objects (the considered video objects are: background, clothes and skin). While such segmentation of single-performer sequences has already been addressed in the literature by means of colour-based [Mozelle98b] or hybrid colour-motion-based [Wang94] techniques, the robustness of such approaches was not demonstrated to justify its usefulness in the case of an unconstrained real-time communication system. However, in order to test the NSV3O solution, we applied a colour-based segmentation technique to our video sequence, 'Florea" and "Carlos", acquired in studio conditions. The main idea that we use in our approach is to segment the mask corresponding to the background and clothes by using the color information, and to complement it in order to obtain the mask of the head and the hands. Indeed, directly segmenting the head and the hands raises several difficulties due to the high motion and different light reflectance on the skin which result in a spread colour distribution of these elements in the colour space. Conversely, the background and clothes colours can be easily controlled and properly set during the acquisition in order to avoid interferences with the colour subspace corresponding to the head and the hands.

Applying basic mathematical morphological operators, the masks of the head and the hands of the signer have been generated. The step-by-step procedure is illustrated in Figure 4.5 and Figure 4.6, for the above-mentioned video sequences.



a) Original image (frame #75 of the "Florea" sequence).

b) Colour-based segmentation of the background.

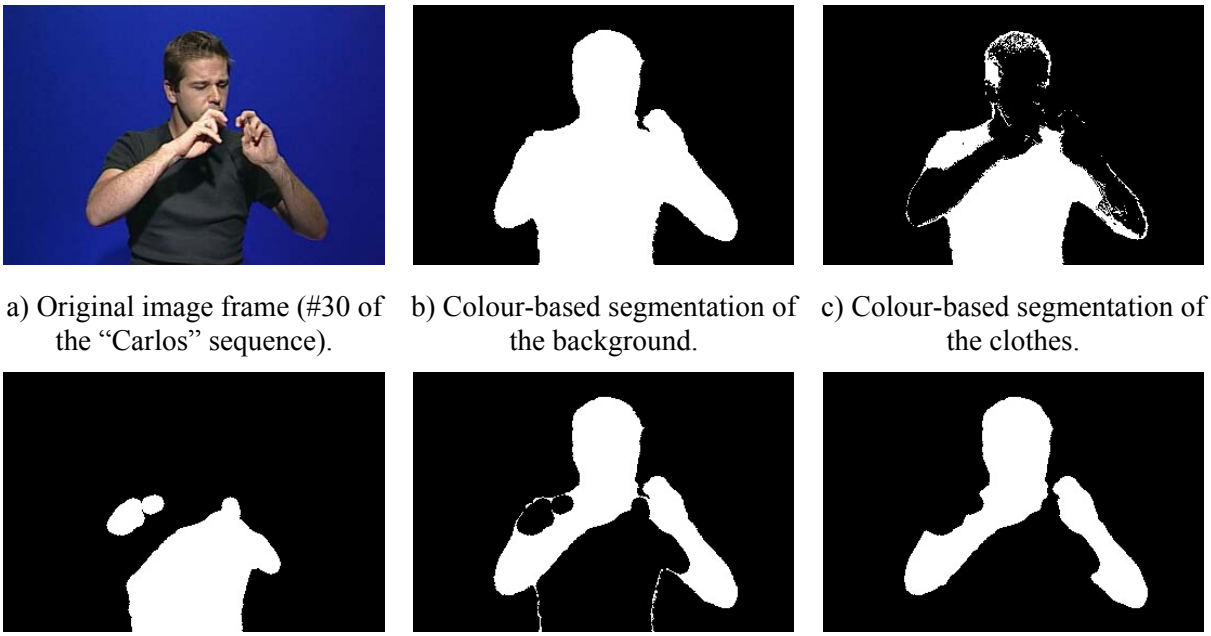
c) Colour-based segmentation of the clothes.

d) Opening of image c).

e) "XOR" between image b) and image d).

f) Opening of image e).

Figure 4.5. The main steps of the head-and-hands segmentation procedure ("Florea" signer).



a) Original image frame (#30 of the “Carlos” sequence).

b) Colour-based segmentation of the background.

c) Colour-based segmentation of the clothes.

d) Opening of image c).

e) “XOR” between image b) and image d).

f) Opening of image e).

Figure 4.6. The main steps of the head-and-hands segmentation procedure (“Carlos” signer).

Figure 4.7 shows the NSV3O segmentation masks obtained for several frames from the “Carlos” sequence.

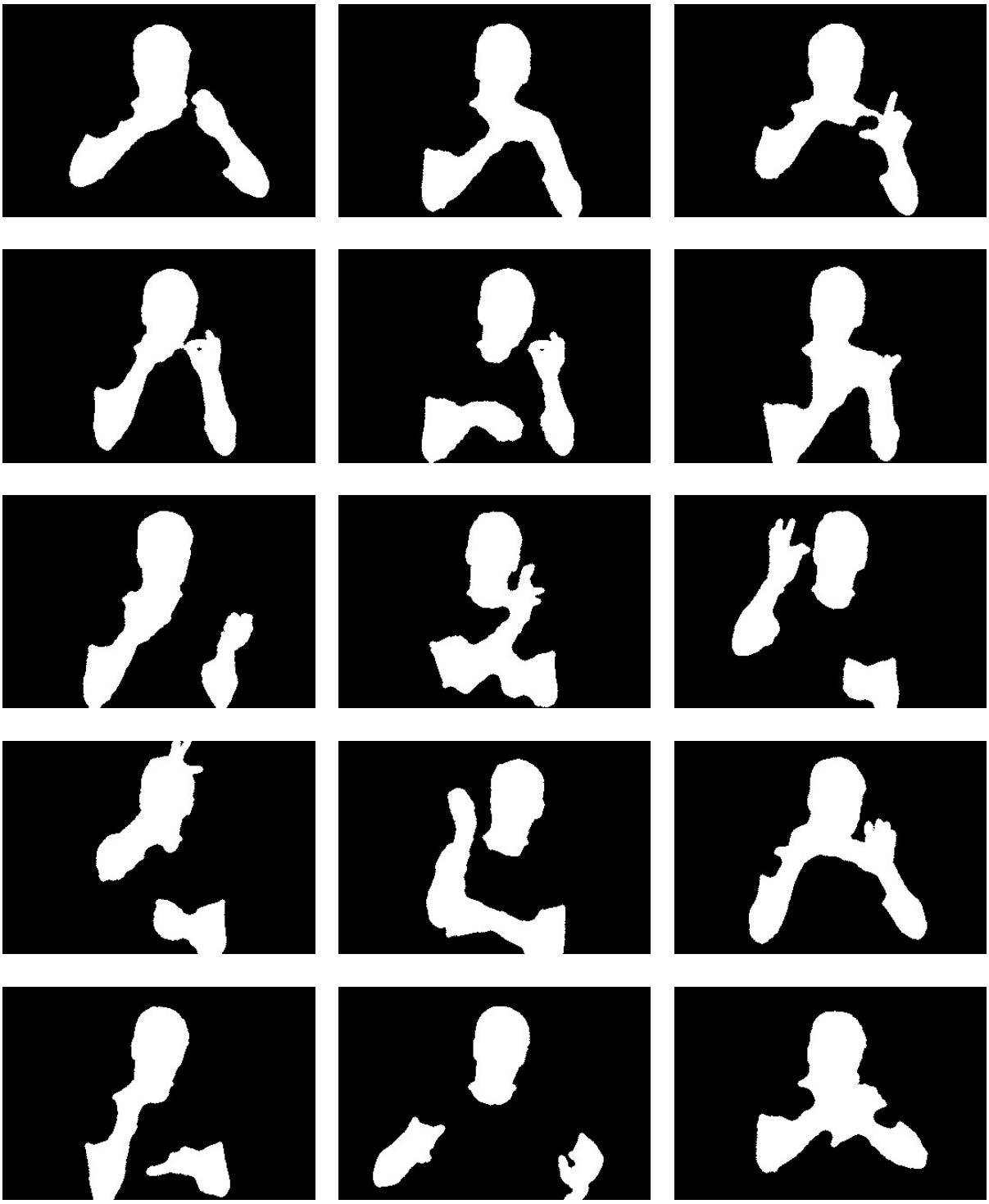


Figure 4.7. Binary mask for the NSV3O method for the “Carlos” sequence.

All three techniques referred to in the previous sections can easily be implemented for real-time performances in a production studio environment, as they only deal with colour information. However, note that the robustness of such approaches is hardly demonstrated.

Once the masks for the NSV2O and NSV3O are obtained, the initial sequence and the masks are the input of the MPEG-4 video encoder. For further details on MPEG-4 video encoding one can refer to [ISOIEC01] or to Chapter 8 in [Pereira02]. The results of the encoding process, in terms of bit-rate, are discussed in Section 4.2.4.6.

4.2.2 MPEG-4 Video object-based encoding of a virtual signer

To overcome the well-known difficulties related to the video object segmentation step, we propose a hybrid approach: virtual signer and video encoding. The key idea is to substitute the natural signer with a virtual (synthetic) one, to animate it in order to compose the sign gestures, and to deliver the content as a video sequence. With this method, any kind of avatar animation, proprietary or standardized, can be used to produce the video sequences. The representation format during the transmission and the playback is MPEG-4 compliant. The present method, called SSV (Synthetic Signer as Video) is based on the following procedure:

1. represent the signing content using a virtual signer in a 3D scene,
2. export, at each frame, the image of the virtual signer,
3. export, at each frame the video objects mask, extracted as described below.

As the 3D avatar has assigned labels on each geometric component (body, hands, hair, eyes …), when projecting it onto a 2D plane, the correspondence between the 2D image pixels and the 3D labels is conserved. Thus, the 2D mask is easily obtained, by checking this correspondence.

In the following, we propose two methods for virtual signer mask extraction. Both use the characteristics of graphic APIs like OpenGL or Direct3D. The first one uses colour information of the virtual signer. A typical 3D application allows to attribute colour information per vertex. In a virtual character application, a common scenario consists in creating textures for each part of the body and mapping the texture on the model geometry. We propose here to modify the appearance of the virtual character by specifying a uniform color to each segment. Moreover, the lights in the 3D scene will be set to zero. Figure 4.8 a) shows the normally textured model and Figure 4.8 b) the modified appearance. By exporting the image corresponding to the 2D scene which contains the modified appearance with uniform colours, a simple labelling allows to identify each object.

a) b)

Figure 4.8. Virtual character rendering: a) normal and b) modified (no lights and no textures) appearance model.

The second method to extract the mask associated with the virtual signer image is specific to the rendering engine. Each graphic primitive (vertex, line, triangle, object) can be labelled during the rendering step and, in this way, each pixel in the image is assigned to the corresponding label. In the case of the virtual character, the labelling is possible if the geometry of the model is obtained as a concatenation of labelled 3D objects.

Once the segmentation step is performed, MPEG-4 video-based encoding is used.

4.2.3 MPEG-4 Virtual Character Animation (FBA and BBA)

To minimize the transmission costs and to ensure the compatibility at the end-user often means to choose standardized solutions. The solutions previously described use a standardized representation for both transmission and client terminal (MPEG-4 Video), but the transmission bandwidth remains significant (more than 64kbps are required). By introducing the FBA and the BBA frameworks, MPEG-4 allows a complete production-transmission-playback chain based on the synthetic representation of the virtual character. In the previous chapters, we have presented a detailed analysis of the MPEG-4 FBA and SMS (which, as stated at the end of the Chapter 3, becomes MPEG-4 BBA).

4.2.4 Comparative evaluation

We have established some comparative criteria according to the major components of digital content chaine: production, transmission and consumntion. The goal of our analysis is to give recommendations allowing to choose one of other solution, by empasizing the advantages and limiatations of each one on the three levels. Let us analyse the proposed methods with respect to the following criteria:

- equipment required at the producer's side,
- real-time capabilities to produce the content,

- reusability of the content, *i.e.* using the captured content for producing a new one by synthetizing data,
- minimum bandwidth required for the transmission of the content,
- terminal capabilities,
- user interaction with the content.

Since we have declined a total number of 6 possible solutions for transmitting singn language content, we will score each solution with respect to each key-criterion from 1 to 6 (6 being the best score).

4.2.4.1 Equipment (hardware and software) required at the producer's side

Let us first consider the NSV-based solutions. As the encoded data is a video sequence of the natural signer, only an ordinary camera is needed to capture the face and the upper body of the signer. In addition, in the case of NSV2O and NSV3O only an unexpensive piece of software is required for the segmentation. While this segmentation step can be performed in real-time when using a colour-based solution, the segmentation robustness for the NSV2O and especially for the NSV3O is not easily guaranteed. In the case of the NSV2O and NSV3O, an object-based video encoder is required. Table 4.1 shows the scores of the NSV, NSV2O and NSV3O with respect to the equipment required.

In the case of the synthetic signer as video solution, sign language communication is based on the visual representation. Thus, when using a virtual character to perform the gesture, a high-quality character representation (geometry and texture) as well as smooth and accurate motions (animation) are required. Nowadays, high-quality virtual characters are all the more common, either created by using powerful modelling tools as 3D Studio Max or Maya, or obtained by using 3D scanners as those builded by Cyberware. Furthermore, motion capture systems as MotionStar®, are getting numerous and motion libraries become available. There are a lot of production and post-production studios all around the world able to provide high-quality virtual character animation. However, the cost of such facilities remains high and a lot of research work, especially in the field of vision-based motion capture and motion retargeting, is still to be developed and integrated into a practical production chain. Table 4.1 shows our estimation of the SSV method.

In the case of the FBA/BBA solution, high-quality virtual characters as well as smooth and accurate motions are also required. To obtain a BBA compliant model is in line with current character designer software. For an FBA model, segmentation in anatomical segments, as described in Section 2.2 is to be considered. The animation method provided by the BBA and based on mesh deformation allows for seamless animation. To obtain similar effects with the FBA, special procedures to get the local deformation tables are to be considered. Motion capture and motion editing are possible in both frameworks, with one difference as the FBA constrains the possible movements for some anatomical segments and does not offer a unique solution during the conversion step of the rotation matrices into the FBA parameters. While the cost of the hardware needed to capture the motions remains important, the existence of a standardized format encourages the production of content in that format. More and more data, which can be reused, are thus becoming available. Table 4.1 shows our estimation of the FBA and BBA methods.

NSV	NSV2O	NSV3O	SSV	FBA	BBA
6	5	4	2	3	3

Table 4.1. Scores with respect to the "Equipment required at the content producer's side" criterion.

4.2.4.2 Real-time capabilities of producing the content

While video acquisition is a real-time operation and colour-based segmentation can be implemented in real-time, advanced and robust segmentation based on motion tracking as required in the case of NSV3O, needs software optimization or hardware support to achieve real-time operations. Table 4.2 shows the scores of the NSV, NSV2O and NSV3O with respect to the real-time capabilities.

Depending on the technique used for obtaining the motion, *i.e.* animation editing or motion capture, the SSV solution can be used respectively, for off-line and on-line applications. Current motion capture systems allow real-time operations, but a time-consuming procedure is usually required to calibrate the system. For the segmentation and video-based encoding, see the methods presented in Section 4.2.2. Table 4.2 shows our estimation of the SSV method.

In the case of FBA/BBA solutions, motion capture and motion editing can be used to produce the content. Furthermore, special procedures to export the data in FBA and respectively BBA compliant formats are needed, but both conversions can be implemented for real-time performances. Table 4.2 reports the scores of FBA and BBA methods.

NSV	NSV2O	NSV3O	SSV	FBA	BBA
6	6	5	4	4	4

Table 4.2. Scores with respect to the "real-time capabilities of producing the content" criterion.

4.2.4.3 Reusability of the content

In the *natural signer video* solutions, the 2D nature of the recorded information and the 3D nature of a generic gesture expressed in sign language makes it practically impossible to reuse the recorded content by editing the video. Even recording simple gestures such as letters, and composing the words letter-by-letter or recording words and composing sentences word-by-word can hardly be achieved, as special constraints have to be taken into consideration when blending the components. Table 4.3 shows the scores of the NSV, NSV2O and NSV3O with respect to the reusability of the content.

One of the major advantages of using a virtual character is the possibility to interact with, to change parameters, to obtain other data and to use it again and again. When dealing with synthetic sign language, the content reusability can be addressed either for the model definition or for the motion data. Once a model is built, changing its geometry or appearance allows to easily generate other models. This technique to get new models from a generic one, usually using pictures of a real person and some minimal manual calibration, is widely used for multimedia exhibitions [AvatarMe]. Furthermore, within some limits, the same animation parameters can be applied to all the characters and thus obtain consistent motions. For a more realistic animation, retargeting procedures adapt the motion to the virtual character geometry.

In a sign language communication system, the reusability consists in first capturing the relevant gestures and then blending the animation to obtain smooth motions. Motion libraries can be built with respect to specific communication domains, *e.g.* weather forecast. In this case, associating motion to standard sentences, a simple application can generate the daily weather forecast report in sign language. The SSV method is evaluated in Table 4.3.

The FBA and BBA frameworks allow high reusability of both the virtual character models and the animation data. Furthermore, the reusability can also be offered to the end-users. Using an FBA or a BBA compliant software, one can change or create its own avatar and use the same animation data (Table 4.3).

NSV	NSV2O	NSV3O	SSV	FBA	BBA
1	1	1	5	6	6

Table 4.3. Scores with respect to the "reusability of the content" criterion.

4.2.4.4 Terminal capabilities

For all the *natural signer video* solutions, the user's terminal must run a MPEG-4 video decoder and a 2D renderer. While the use of any video decoder on powerful devices such as PCs can be software-optimized for real-time decoding capabilities, the support on less powerful devices such as mobile phones must be hard wired. In order to minimize the cost of hardware developments, standardized solutions must be favoured. This justifies, once again, our interest in considering MPEG-4 compliant solutions. Table 4.4 shows the scores of the NSV, NSV2O and NSV3O with respect to terminal capabilities.

In the SSV solution, the representation of the avatar is video-based. Therefore, an MPEG-4 video decoder and a 2D renderer are required. The SSV method has the same requirements as the NSV, NSV2O and NSV3O methods regarding the user's terminal capabilities (Table 4.4).

With only the transmission of model representation and animation parameters, required in the case of FBA/BBA solutions, the virtual character is generated by the end-user's terminal. An animation parameters decoder and a 3D rendering engine is required at the end-user. The low complexity of both FBA and BBA decoders enables real-time decoding on low cost terminals.

In the case of an FBA compliant avatar, when no local deformations are addressed, only rigid transformations are required. This makes it possible to use the FBA on less performant platforms, without dedicated hardware or software. We have demonstrated during the IBC2002 exhibition [Ibc02], the real-time performances of FBA for the Siemens-Fujitsu "Activy" set-top box, a device that does not dispose of 3D capability.

To animate a BBA-compliant character, software and/or hardware optimization is required in order to achieve real-time animation. Recent rendering engines such as those provided by Superscape [SScape] or HI [HI], allow real-time rendering of complex virtual character animation on mobile phones. Table 4.4 reports the scores assigned to the FBA and BBA methods.

NSV	NSV2O	NSV3O	SSV	FBA	BBA
6	6	6	6	5	4

Table 4.4. Scores with respect to the "terminal capabilities" criterion.

4.2.4.5 User interaction with the content

Video-based solutions, NSV*i*O do not allow the user to navigate into the image, to change the viewpoint or to interact with the natural signer. Only for pre-recorded content, simple operations on the time-line such as pause, fast forward or rewind commands can be performed. Table 4.5 shows the scores of the NSV, NSV2O and NSV3O with respect to this criterion. The same comments hold for the SSV solution.

An FBA avatar, as well as a BBA generic model, is defined in a MPEG-4 3D scene. Using the generic interaction mechanisms provided by MPEG-4 BIFS, the user can navigate into the 3D scene, change the viewpoint, zoom on the hands or face (in 2D only translations and zoom can be possible)Elaborate MPEG-4 3D players allow the user to change the avatar or some of its attributes (appearance or geometry). Therefore FBA/BBA-based solutions provides the users with the highest interactivity (Table 4.5).

NSV	NSV2O	NSV3O	SSV	FBA	BBA
2	2	2	2	6	6

Table 4.5. Scores with respect to the "user interaction with the content" criterion.

4.2.4.6 Minimum bandwidth required for the transmission of the content

Let us now compare the different technical solutions for signing transmission with respect to the bandwidth criterion. Note that reducing the bandwidth can be achieved by acting on:

- the image resolution,
- the quantization step for the DCT coefficients (intra-frame), or for prediction errors (predictive frame),
- the use of the segmentation of the signer's shape; only the signer shape or parts of it are transmitted,
- the time sampling.

In our experiments, the "Florea" and "Carlos" sequences have a resolution of 352 x 280 and 352 x 240, respectively. A low resolution (176 x 140) version of the "Florea" sequence is also used in order to obtain compression results for near QCIF format. The goal of the tests is to identify the minimum bit-rate needed to encode the video sequence, while keeping the sign language message comprehensible, with some visual comfort.

For all the tests, the structure of the group of video planes is as illustrated in Figure 4.9: 3 planes P (Predictive) between two planes I (Intra) and 2 planes B (Bidirectional) between two planes P. Note that, with this scheme, encoding I frames require the largest part of the bandwidth as they are standalone encoded. Each P plane encodes the difference between the corresponding frame and the closest I (or P) frame preceding it. Each B plane encodes the difference between the corresponding frame and one or the average of two P or I (or one of each) frames, one preceding and another

succeeding it [ISOIEC01]. The MPEG-4 Video Encoder used in our tests is compliant with the "Core Visual Profile"[Pereira02].

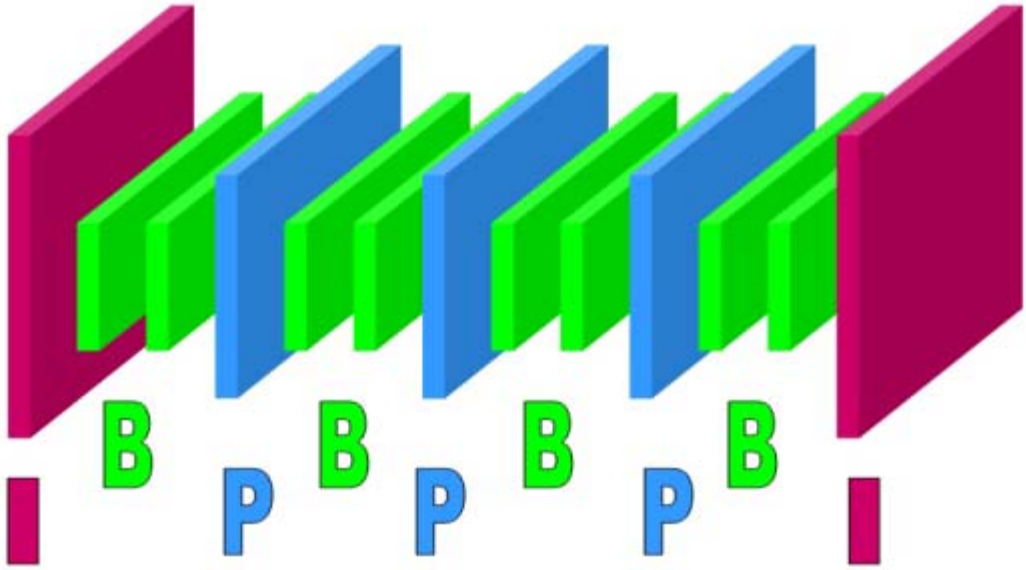
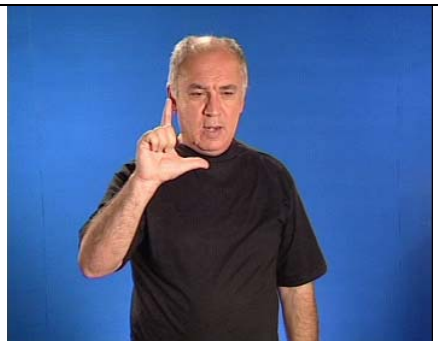
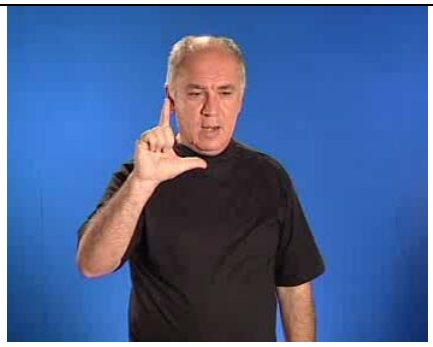


Figure 4.9. I, P, B structure of the Group of Planes (GOP).

a) The first series of tests consists in encoding the entire image as a single object (NSV). Table 4.6, Table 4.7 and Table 4.8 show the compression results with respect to the quantization step, time sampling and image resolution, respectively.

Sequence	Resolution	Frame-rate	Quantization step	Image #43	Average bit-rate [kbps]
"Florea"	352 x 280	25	2		1578
"Florea"	352 x 280	25	6		488

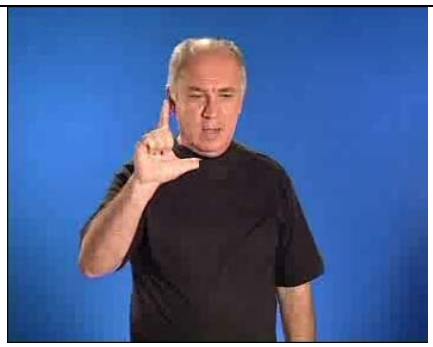
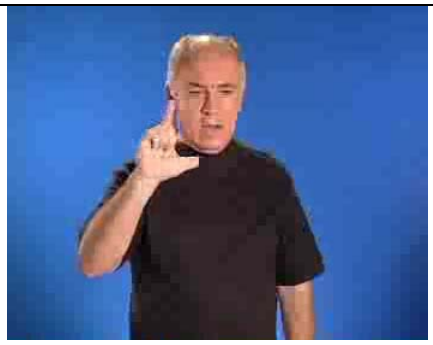
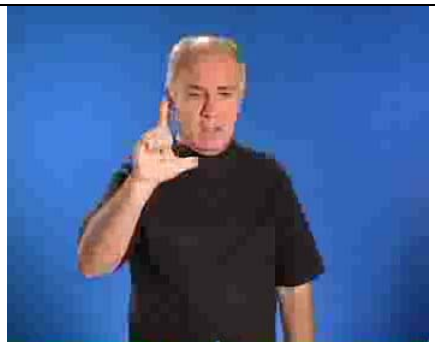
“Florea”	352 x 280	25	12		294
“Florea”	352 x 280	25	24		176
“Florea”	352 x 280	25	31		161

Table 4.6. NSV: compression results with respect to the quantization step (example for “Florea” sequence”).

The dependence of the obtained bitrate with respect the the quantization step is illustrated in Figure 4.10.

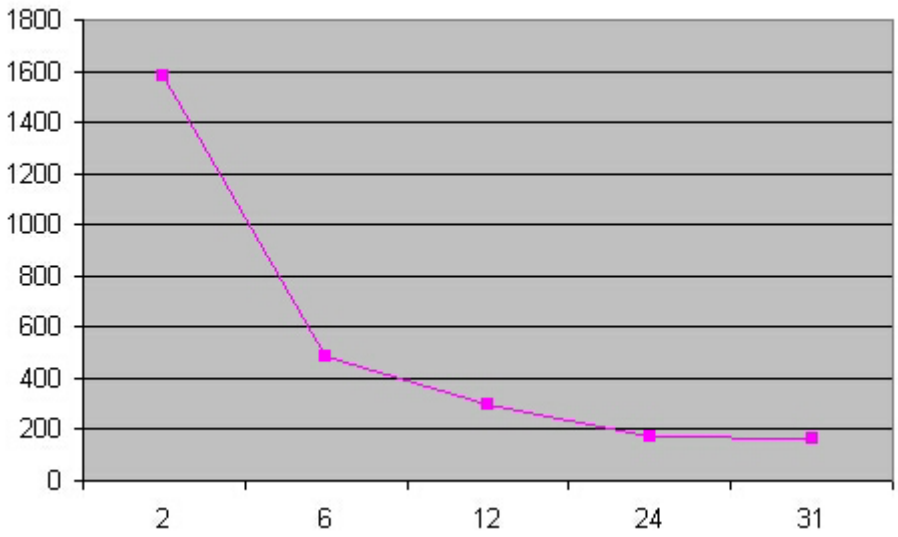


Figure 4.10. Bitrate [kbps] versus Q.

When increasing the Q over 12, the poor quality of the image makes the sign language message incomprehensible [ViSiCAST]. Thus, in our experiments we will keep Q=12.

The next step of our experiments consists in analysing the dependency between the bitrate and the frame-rate. By successively eliminating intermediate frames from the video sequence, while keeping the same duration in time, we have applied the encoding procedure for 20, 15 and 10 fps.

Sequence	Resolution	Fram-rate	Quantization step	Average bit-rate [kbps]
“Florea”	352 x 280	25	12	294
“Florea”	352 x 280	20	12	246
“Florea”	352 x 280	15	12	198
“Florea”	352 x 280	10	12	137

Table 4.7. NSV: compression results with respect to the time sampling (example for “Florea” sequence”).

The dependency between the bitrate and the frame-rate is illustrated in Figure 4.11. One can remark the linear behaviour of this dependency.

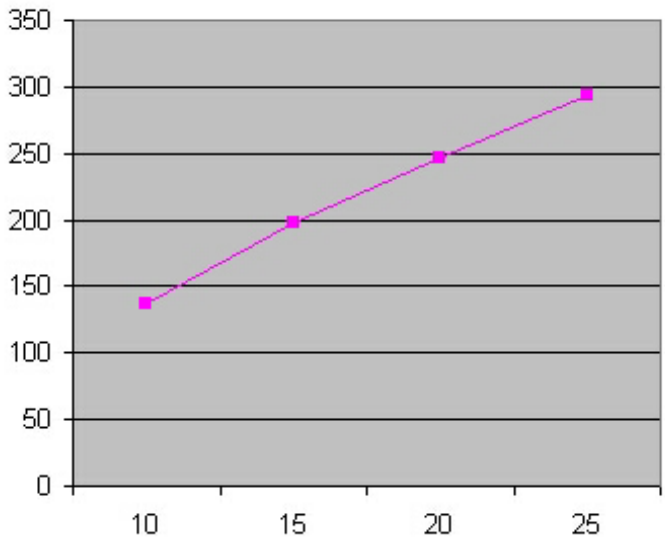


Figure 4.11. Bitrate [kbps] versus frame-rate.

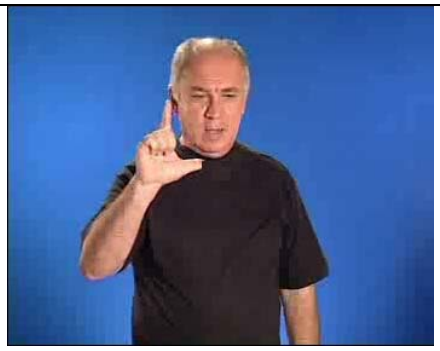
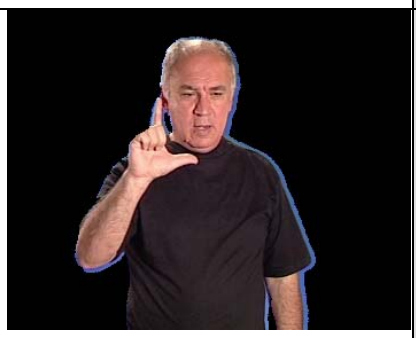
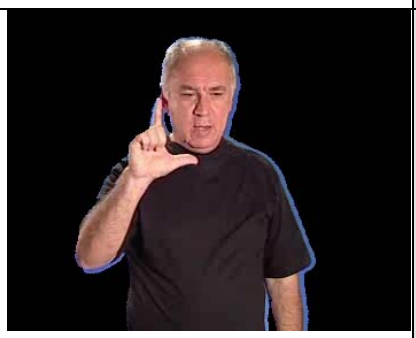
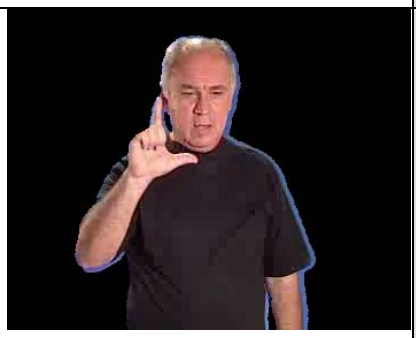
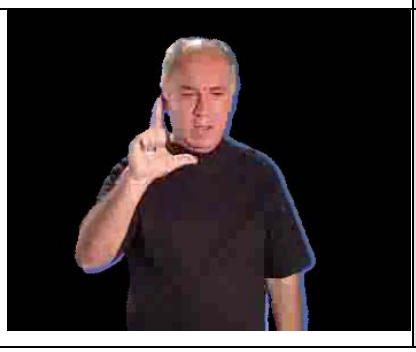
Se-quence	Resolution	Frame-rate	Quanti-zation step	Image #43	Average bit-rate [kbps]
"Florea"	352 x 280	25	12		294
"Florea LR"	176 x 140	25	12		105
"Carlos"	352 x 240	25	12		234

Table 4.8. NSV: compression results with respect to the image resolution.

b) The second series of tests is made with the NSV2O method. Because of the specificity of the sign language sequence with respect to the uniformity of the background colour, the results in terms of bit-rate are very similar with those obtained in the first series of tests. Table 4.9, Table 4.10 and Table

4.11 show the compression results with respect to the quantization step, time sampling and image resolution, respectively.

Se-quence	Resolu-tion	Signer Shape (%)	Frame-rate	Quanti-zation step	Image #43	Average bit-rate [kbps]
"Florea"	352 x 280	32%	25	2		1134
"Florea"	352 x 280	32%	25	6		336
"Florea"	352 x 280	32%	25	12		186
"Florea"	352 x 280	32%	25	24		138

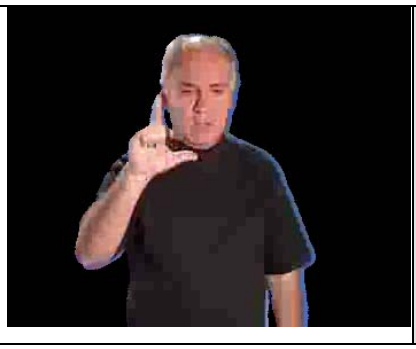
"Florea"	352 x 280	32%	25	31		131

Table 4.9. NSV2O: compression results with respect to the quantization step.

The dependence of the obtained bitrate with respect the the quantization step is illustrated in Figure 4.12.

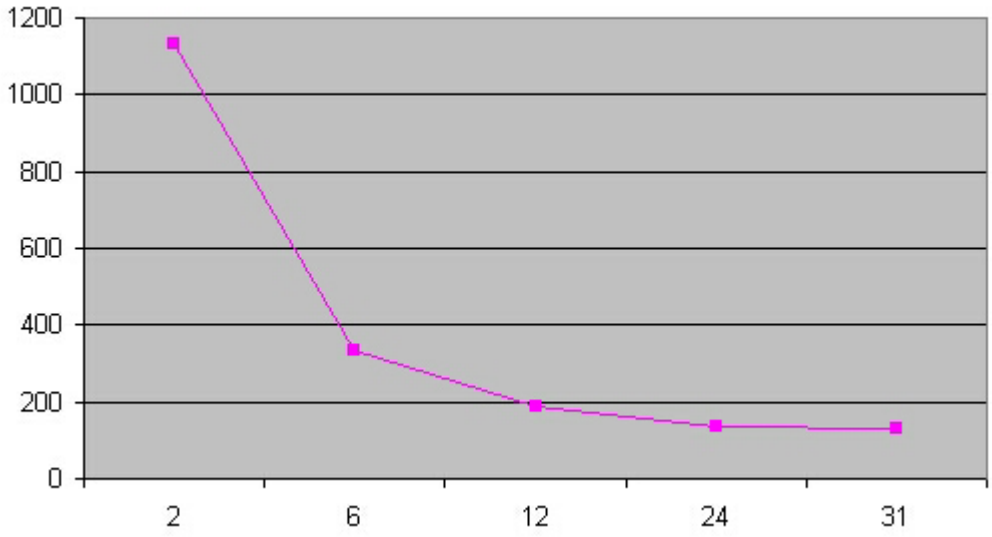


Figure 4.12. Bitrate [kbps] versus Q.

Se-quence	Resolution	Signer Shape (%)	Frame rate	Quantization step	Average bit-rate [kbps]
"Florea"	352 x 280	32%	25	12	186
"Florea"	352 x 280	32%	20	12	154
"Florea"	352 x 280	32%	15	12	131
"Florea"	352 x 280	32%	10	12	87

Table 4.10. NSV2O: compression results with respect to the time sampling.

The dependency between the bitrate and the frame-rate is illustrated in Figure 4.13. One can remark the linear behaviour of this dependency.

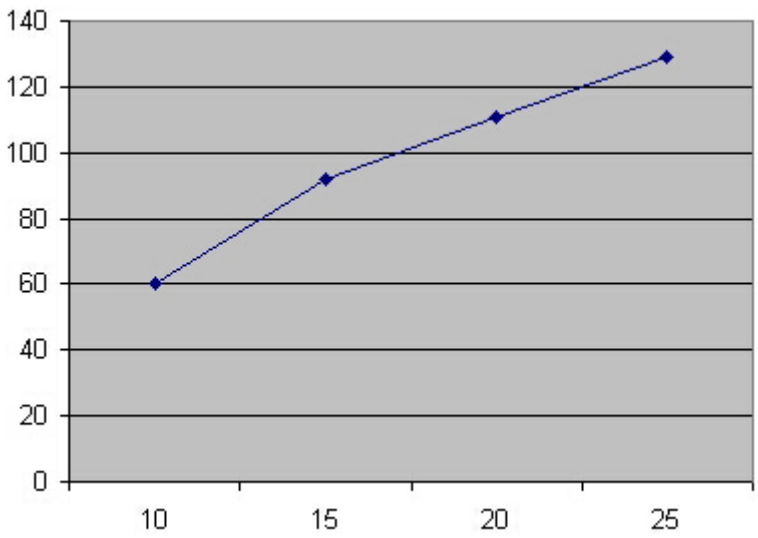


Figure 4.13. Bitrate [kbps] versus frame-rate.

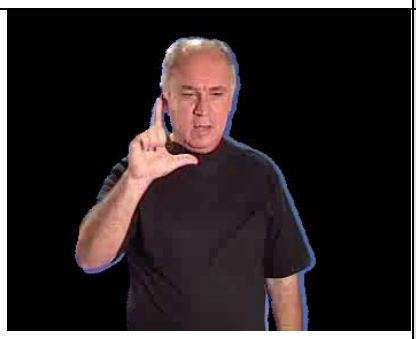
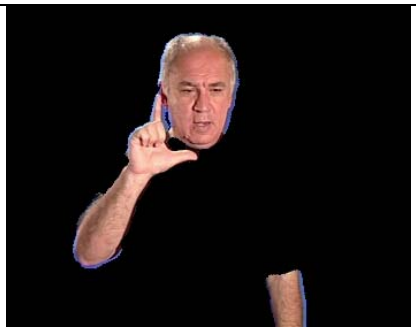
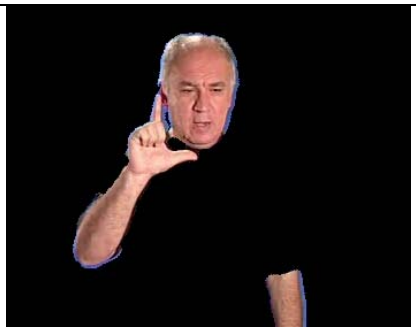
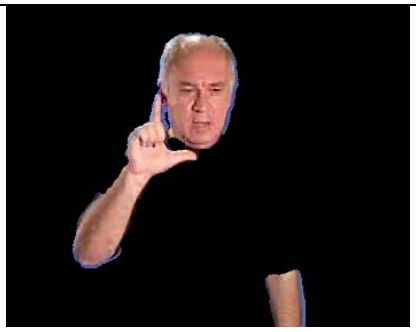
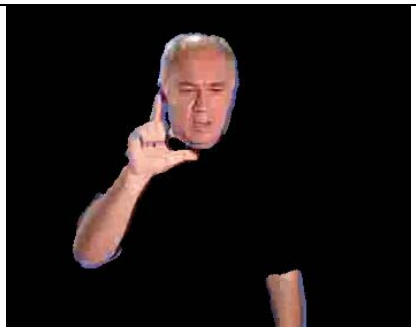
Se-quence	Re-solution	Signer Shape (%)	Frame -rate	Quanti-zation step	Image #43	Average bit-rate [kbps]
"Florea"	352 x 280	32%	25	12		186
"Florea LR"	176 x 140	32%	25	12		59
"Carlos"	352 x 240	29%	25	12		230

Table 4.11. NSV2O: compression results with respect to the image resolution.

c) The third series of tests is made with the NSV3O method. Table 4.12, Table 4.13 and Table 4.14 show the compression results with respect to the quantization step, time sampling and image resolution, respectively.

Se-quence	Re-solution	Hands and Head Shape (%)	Frame -rate	Quanti -zation step	Image #43	Average bit-rate [kbps]
"Florea"	352 x 280	14%	25	2		792
"Florea"	352 x 280	14%	25	6		247
"Florea"	352 x 280	14%	25	12		129
"Florea"	352 x 280	14%	25	24		90

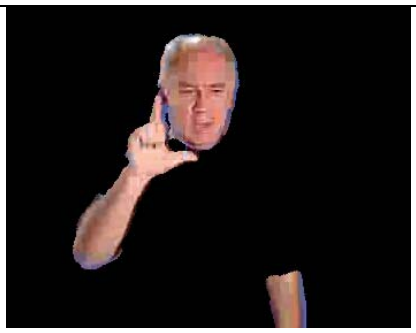
"Florea"	352 x 280	14%	25	31		86

Table 4.12. NSV3O: compression results with respect to the quantization step.

The dependence of the obtained bitrate with respect the the quantization step is illustrated in Figure 4.14.

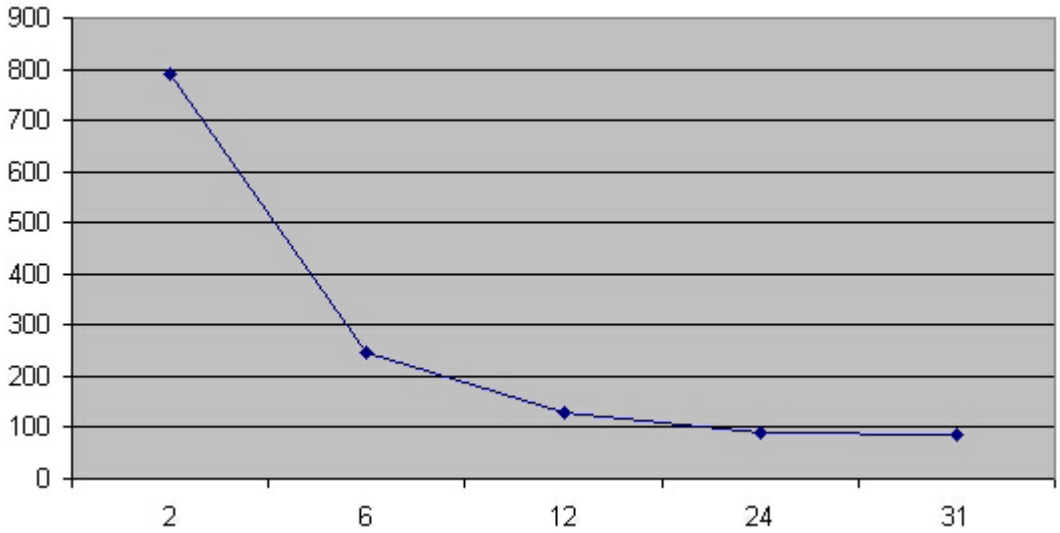


Figure 4.14. Bitrate [kbps] versus Q.

Se-quence	Resolution	Hands and Head Shape (%)	Frame-rate	Quantization step	Average bit-rate [kbps]
"Florea"	352 x 280	14%	25	12	129
"Florea"	352 x 280	14%	20	12	111
"Florea"	352 x 280	14%	15	12	92
"Florea"	352 x 280	14%	10	12	60

Table 4.13. NSV3O: compression results with respect to the time sampling.

The dependency between the bitrate and the frame-rate is illustrated in Figure 4.15. One can remark the linear behaviour of this dependency.

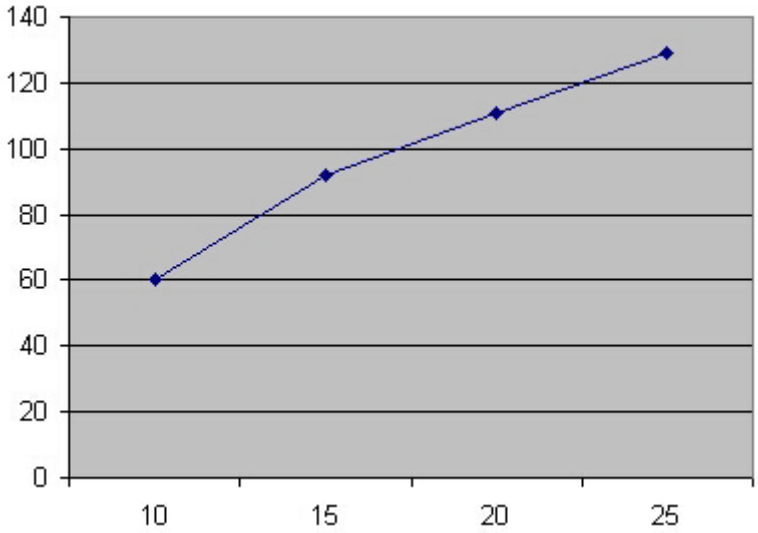


Figure 4.15. Bitrate [kbps] versus frame-rate.

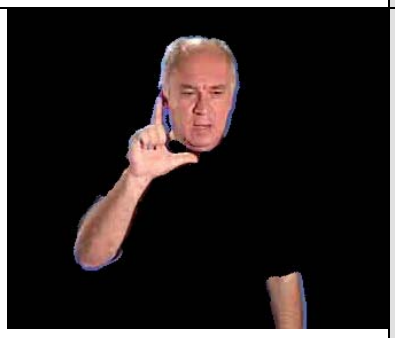
Se-quence	Re-solution	Hands and Head Shape (%)	Frame-rate	Quanti zation step	Image #43	Average bit-rate [kbps]
"Florea"	352 x 280	14%	25	12		129
"Florea LR"	176 x 140	14%	25	12		51
"Carlos"	352 x 240	15%	25	12		214

Table 4.14. NSV3O: compression results with respect to the image resolution.

d) While the results in terms of bandwidth obtained with the NSV3O method makes it possible for the sign language transmission on low bit-rate networks, the comfort of the communication is greatly

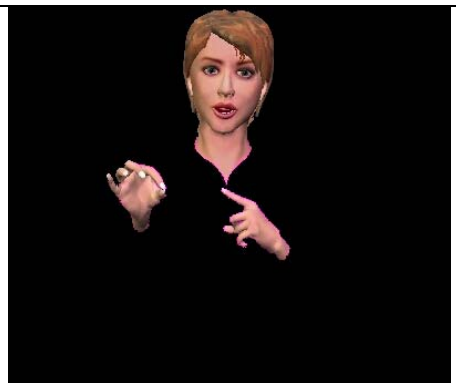
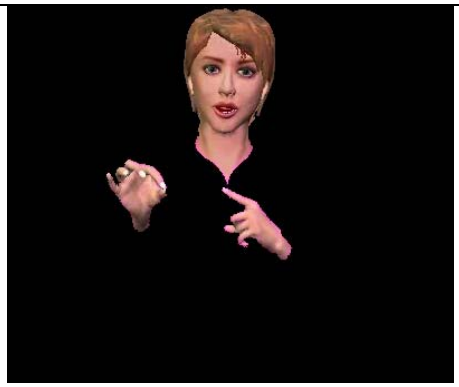
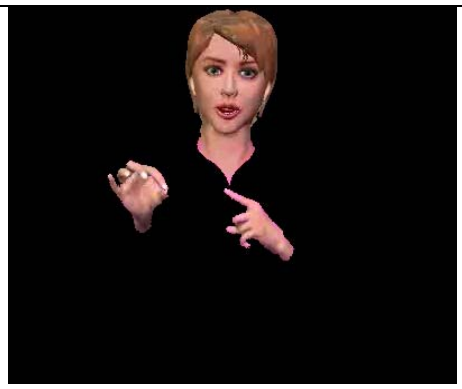
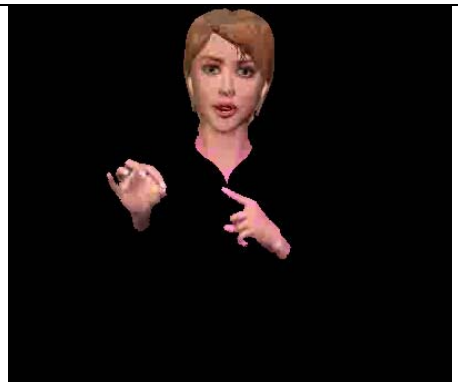
reduced as only a representation of the hands and the head of the signer are transmitted. In order to improve on comfort, we propose to transmit also, at very low bit-rate (poor quality), the shape representing the clothes (NSV3O+C). The overload of encoding this information is presented in Table 4.15.

Se-quence	Re-solution	Cloths Shape (%)	Frame-rate	Quanti zation step	Image #43 after composition	Overload bit-rate [kbps]
"Florea"	352 x 280	18%	25	31		94
"Florea LR"	176 x 140	18%	25	31		31
"Carlos"	352 x 240	14%	25	31		70

Table 4.15. Encoding overload for NSV3O+C.

The scores of the NSV, NSV2O and NSV3O with respect to the required bandwidth criterion are synthesized in Table 4.17.

In the SSV approach, the robust segmentation allows to individually encode each shape corresponding to hands, head, body or background. The results in terms of compression are very similar with those obtained in the case of the NSV3O and the same mechanisms to reduce the bit-rate can be used: image resolution, time sampling and DCT coefficients quantization step. We have used for our tests a proprietary avatar animation solution (provided by Televirtual [Tvirt]) and the MPEG-4 shape-based encoder previously introduced in Section 4.2.1. The size of the exported image is 400 by 340 and the frame-rate of the sequence is 25. Table 4.16 shows the compression results with respect to the quantization step for the DCT coefficients corresponding to the hands and the head.

Se-quence	Re-solution	Head and hands shape (%)	Fra me rate	Quantiz ation step	Image #115	Average bit-rate [kbps]
"Visia"	400 x 340	9%	25	2		703
"Visia"	400 x 340	9%	25	6		212
"Visia"	400 x 340	9%	25	12		102
"Visia"	400 x 340	9%	25	24		70

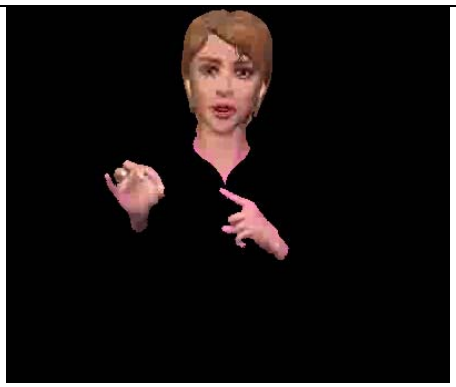
"Visia"	400 x 340	9%	25	31		66

Table 4.16. SSV: compression results with respect to the quantization step.

The scores of SSV with respect to the required bandwidth criterion are to be found in Table 4.17.

For both FBA and BBA, the definition of the virtual character is independent of the animation. This characteristic allows to transmit the virtual character in a static posture once per session and then, for each frame, to transmit the minimal set of parameters which affect the static posture. This scenario makes possible a very low bit-rate communication. With respect to the static posture of the virtual character, both FBA- and BBA-complaint models allow mesh compression, as promoted in the Amendment 1 of the MPEG-4 Systems standard [ISO99]. The size of an FBA-compliant avatar without deformation tables is less than the size of a BBA model. Nevertheless, when the deformation tables are to be considered for the FBA model, the size of the model increases by up to 50% [Guttierez02] from the original model.

Both FBA and BBA frameworks propose compression schemes for the animation parameters as described in Section 2.1 and Section 3.4. The encoders control the motion quality using the quantization steps. As in both FBA and BBA streams, the rotations (the main component when dealing with virtual character animation) are based on angles composition, the performances in terms of bit-rate are quite similar when animating a virtual human-like object. Depending on the motion complexity (the number of animated joints), the required bit-rate can vary for the same motion quality.

In our tests related to sign language transmission, we consider that the avatar animation affects the upper body, the head, the arms, the wrists and the fingers. The test data consists in two sequences obtained with a motion capture system. First, two conversions from generic rotation matrices, resulted from the capture system, to both FBA and BBA formats are performed. Then, by applying the compression algorithms (predictive, and DCT-based) we tune the encoding parameters (mainly the quantization step) and visually check the motion quality.

With both approaches (FBA and BBA), we have obtained for a frame-rate of 25 fps, bit-rates from 5 kbps to 40 kbps, depending on the value of the quantization step. For the same quantization step, the overload when using BBA is less than 10%.

We pursued the experiment, for BBA by selecting key-frames. The condition that two frames become a set of key-frames is to be able to generate from them, with an acceptable error, the intermediate frames by using linear interpolation for translations, and quaternion spherical linear interpolation for rotations. Depending on the accepted threshold of error and the motion speed, the number of key-

frames can be up to 40% from the number of total frames. By encoding only the key-frames, a gain of up to 40% can be obtained.

The second approach to reduce the required bit-rate is to build inverse kinematics chains, and to transmit only the animation parameters related to the end-effectors. In this respect, we have build for each arm a kinematic linkage, with the end-effectors at the wrists level. A set of angular constraints have been imposed for the arms and forearms. The animation parameters for the arms and forearms are not transmitted. This approach, even if decreases the bit-rate with up to 10%, has two main limitations. The first one is related to the computing time needed for solving the IK problem. The second one is related to the fact that, from the point of view of sign language, the semantic of the message to be transmitted can be lost when the arms and forearms are not exactly in the right position. In conclusion, IK-based approach is a powerful tool for helping in authoring sign language content, but cannot be used successfully during the real-time animation.

NSV	NSV2O	NSV3O	SSV	FBA	BBA
1	2	3	4	5	6

Table 4.17. Scores with respect to the “required bandwidth” criterion.

4.2.4.7 Comparison summary

With the first class of solutions – *MPEG-4 Video-compliant object based encoding of a natural signer* – producing and delivering content is widely supported. However, content reusability and scene interactivity are not possible. Please note that the content delivery performances are dependent on the segmentation step. More specifically, when a three-object-based segmentation is performed, low bit-rate transmission (less than 64 kbps) can be achieved. The constraint here is to provide with real-time and robust segmentation.

With the second class of solutions – *MPEG-4 Video-compliant object-based encoding of a synthetic signer* – producing content is one of the main difficulties, but the content reusability is possible. The user’s terminal only requires relatively small performances. However, transmission bandwidth remains an important limitation for low bit-rate networks.

In order to decrease the required bandwidth we have to investigate a completely new type of solutions which avoids the video-based representation. Within the FBA/BBA class, the signer is represented as a synthetic object for the entire chain: production, transmission and playback in a standardized framework. Here, only the parameters needed for virtual character definition and animation are transmitted and the 3D reconstruction is performed at the terminal level. The results in terms of compression satisfy sign language transmission on low bit-rate networks.

Figure 4.16 shows a multimedia player containing video, audio, 2D and 3D graphics. Separate MPEG-4 streams refer to each of the above-analysed solutions.

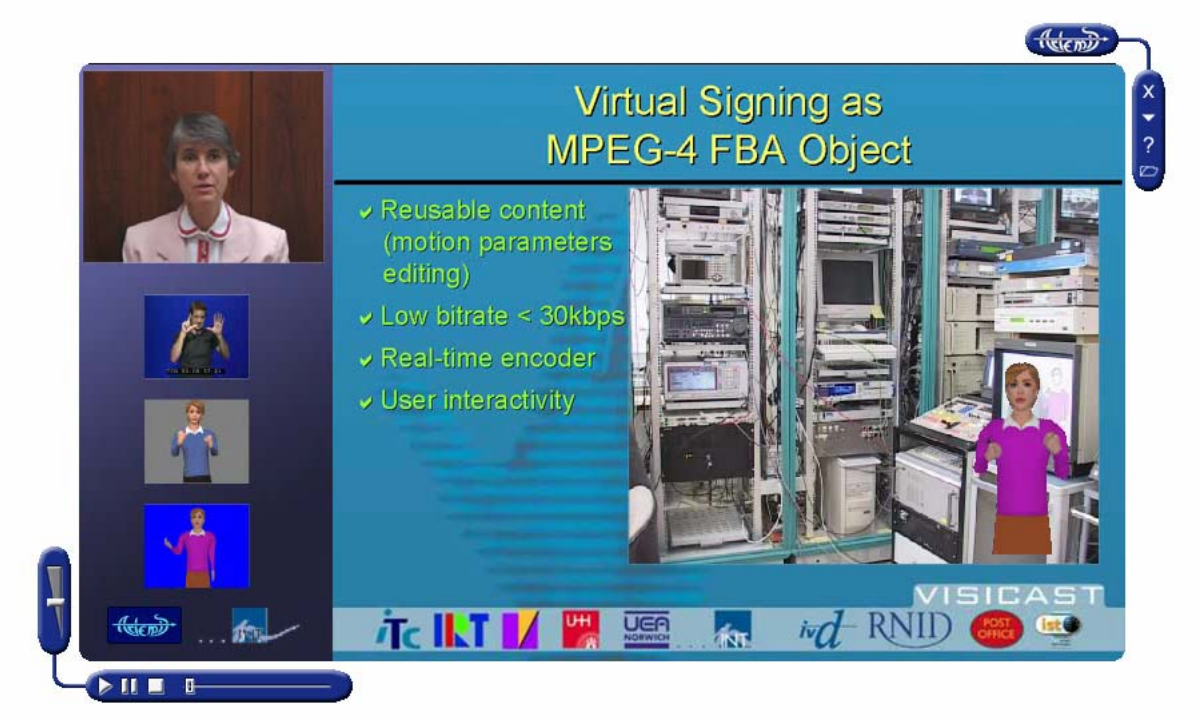


Figure 4.16. Multimedia Player.

As announced at the beginning of this manuscript, our goal is to propose a complete sign language transmission system over MPEG-2 TS and IP. Figure 4.17 shows the synopsis of this system, by integrating the solutions analysed in Section 4.2.

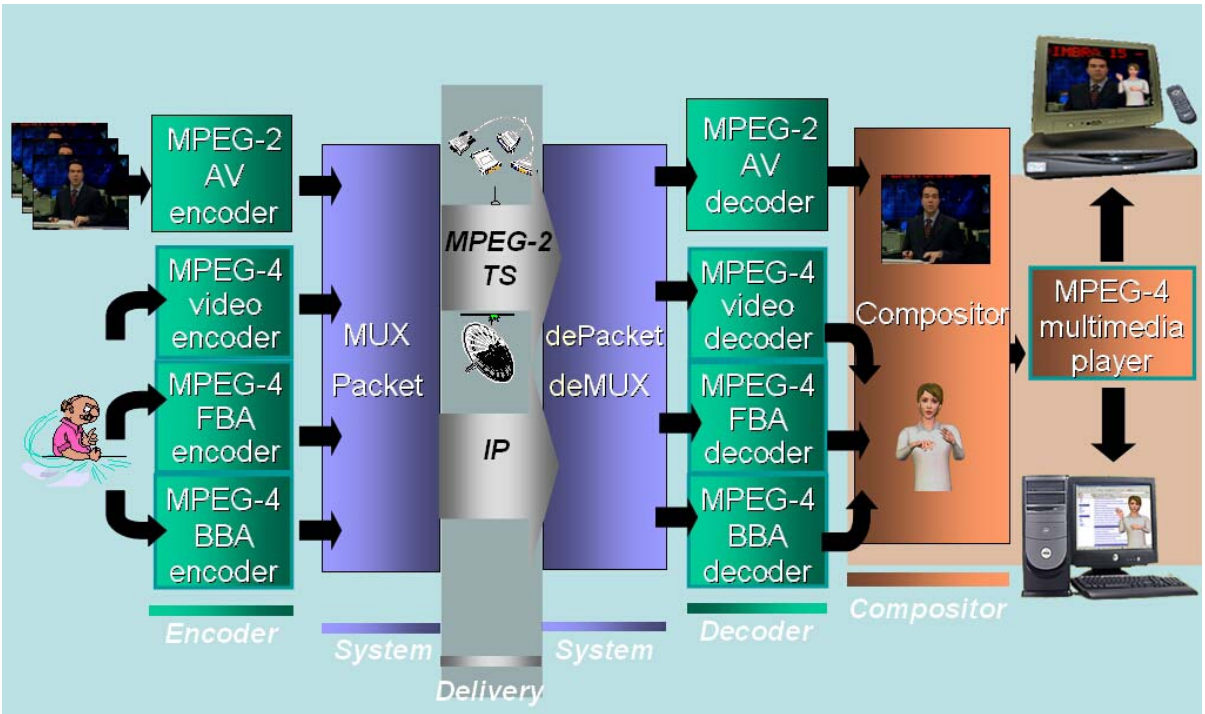


Figure 4.17. Synoptic description of a broadcast architecture ensuring sign language transmission into a standardized framework.

4.2.5 Implementation in a real system: the ViSiCAST project

A specific application related to a sign language communication system using virtual characters, developed in the ViSiCAST project [Visicast], allows us to set up some of the previously-described solutions in a real transmission environment.

ViSiCAST develops, evaluates and applies realistic virtual humans (avatars) technologies for generating European sign languages. By building applications for the signing system in television, multimedia, web and face-to-face transactions, ViSiCAST aims to improve the life of Europe's hearing impaired population, by giving them access to public services or entertainment and by offering them the possibility to develop and enjoy their own multimedia content for communication, leisure or learning. The ViSiCAST framework aims to provide the following:

1. systems for the generation, storage and transmission of virtual signing,
2. user-friendly methods to capture signing gestures (where appropriate),
3. a machine readable system to describe sign language gestures (hand, face and body) which can be used to retrieve stored gestures or to build them from low-level motion components. It will use this descriptive language to develop translation tools from speech and text to sign.

Our work in the ViSiCAST project addressed the feasibility the sign language transmission over digital television. Figure 4.18 shows the ViSiCAST equipment for encoding and transmission of virtual character animation data in a MPEG-2 transport layer as well as a PC-based terminal, able to receive, decode and render video and virtual character animation.

a) Server and transmission. b) PC-based end-user's terminal.

Figure 4.18. Snapshots of ViSiCAST broadcast system.

The transmission of the virtual character animation over IP and MPEG-2 Transport Stream has been successfully demonstrated at the "International Broadcasting Convention" in September 2002 [Ibc02].

4.3 Summary

In this chapter, we have described six possible solutions for sign language transmission over networks. These solutions are classified with respect to the data format used to represent the content at different levels in the chain: production, transmission and consuming. Thus, we decline video-based solutions, 3D-solutions and a hybrid (video and 3D) solution. The goal of our study was to analyse under which conditions and with respect to which component of the multimedia chain (production, transmission, consumption), a solution is more appropriate that the others.

Therefore, we have analysed each solution with respect to some key-criteria and presented the specific advantages and limitations.

Then, we have performed compression tests with each method and we have presented and discussed the results with respect to the quality/bit-rate.

Our specific application of a communication system for the deaf community based on virtual characters allow us demonstrate that the proposed SMS framework (MPEG-4 BBA) makes it possible to represent a realistic virtual character animation, at a very low bit-rate transmission.

Conclusions and Future Work

Human, and in general virtual character, representation is a practice that goes deeply in humanity's history. For all the visual arts, as painting or sculpture, representing humans reveals the interest. Highly realistic or metaphoric representations of humans are now famous pieces of art.

In the modern era, with the evolution of new technologies, the physical support for such representations smoothly changes. The first step in this evolution was first achieved by the cinema as it introduces the concept of time. The representations are not static any more, but they can evolve in time and move in the scene. The first 2D cartoons are then produced, bringing up a new and revolutionary concept: the animation. While, in a static representation, to depict life requires artistic skills, when performing animation, only simulating motion makes it possible. The problems are here related to the need to provide with a new static representation at very dense time samples. The evolution in computer science solves some of these problems. Indeed, nowadays digital synthetic content becomes a new form of art.

Without claiming to address any artistic aspects related to virtual character animation, the work presented in this thesis reports on technical problems raised by the representation, animation and transmission of virtual characters. Specifically, social reasons motivate us to address these issues with a sign language communication system. The complexity of the problems that the implementation of

such system addresses let us think that such application should be considered in an integrated framework, where production, transmission and consumption of multimedia content are to be analysed. We have started our presentation by introducing this framework. Starting with a survey of the tools from two existent multimedia standards, MPEG-4 and VRML, we sketch out a first look of how our framework should be build.

An overview of the 3D virtual character-related techniques, in terms of model definition and animation is presented in Chapter 1. Addressing the problems that are relevant in a production environment, special attention is paid to automatic methods to build (3D scanners) and animate (motion capture) a virtual character.

In the second chapter, an extended evaluation of the MPEG-4 FBA tools has been made. As the FBA specifications do not recommend any specific methods to obtain a real description of a 3D human body and its associated animation parameters, we have addressed both issues. Related to the first one we have proposed a semi-automatic approach to get a MPEG-4 compliant body representation from a 3D mesh model of a human. This semi-automatic approach based on a segmentation algorithm was implemented as a part of the Virtual Human Modelling (VHM) authoring tool. Related to the second issue, we have presented a dedicated authoring tool named ARTEMIS Animation Avatar Interface (3AI), implemented to facilitate the editing and extraction of animation parameters. To help with these tasks, this authoring tool includes various interpolation schemes and one inverse kinematics method. A practical evaluation of both the compression and the animation performances of the FBA framework, allows us to analyse objectively (in terms of transmission bit rate) and subjectively (in terms of realistic representation) the performances and limitations of the FBA framework.

In order to overcome the FBA limitations related to the realistic animation performances, we have introduced a new framework, called SMS, for virtual character animation, in the third chapter. The SMS framework was built on a generic deformation model consisting in a controller set up as a triple composed by a geometric support, a volume embedding the geometric support and a measure of the affectedness of the controller in the volume. Then, we have proposed two declinations of the model (bone and muscle controllers) and consider them as the core of our framework. We then have shown how they can be used for defining and animating generic virtual characters. In order to represent generic virtual characters into a 3D scene, we have provided the SMS architecture with the scene graph nodes. In order to enable low-bit-rate transmission we have provided the SMS architecture with an efficient representation of animation parameters. A set of tests allows to state that the SMS enables animation on low-bit-rate networked environment. A comparative study between the SMS and FBA frameworks, shows the advantages of the SMS one. Finally, we describe how we have promoted the SMS framework to the MPEG-4 standard.

In the last chapter, we give a detailed report on experiments that we have conducted in terms of data compression, in order to ensure sign language transmission over IP and MPEG-2. We have proposed six possible solutions for sign language transmission over networks. The six solutions have been classified with respect to the data format used to represent the content at different link of the chain: production, transmission and consuming. We described video-based solutions, 3D-solutions and a hybrid (video and 3D) solution. With respect to some key-criteria, we analysed each solution presenting their specific advantages and limitations. By performing compression tests for each method, we have presented and discussed the capabilities in terms of the visual quality/bit-rate.

In conclusion, the SMS framework allows to define and to animate by means of deformations a generic synthetic object. By including this framework into the MPEG-4 multimedia standard that allows to integrate natural and synthetic objects representation, we can state that the requirements announced at the beginning of this manuscript have been fulfil.

While our work was conducted with the strong motivation to offer a sign language communication system based on virtual characters, other applications can be addressed by the SMS framework. As, in this framework, we can model and animate any kind of generic virtual characters, an immediate extension of the applications field is related to 3D cartoons. The results in terms of animation compression while keeping the possibility of transmitting rich content made us to start investigations into another kind of networked environments, namely, distributed ones. Here, we will address 3D on-line communities, with accent on on-line multimedia gaming.

Related to the MPEG-4 standardization process our perspectives include the identification of potential industrial applications that can use SMS tools and the integration into a MPEG-4 3D profile of these tools with others AFX technologies.

Appendix A

Related Publications

A.1.1 Book chapters

M. Preda, F. Prêteux, "MPEG-4 Human Virtual Body Animation" in *M. Bourges-Sévenier (Ed.), MPEG-4 Jump-Start (Chapter 9)*, Prentice Hall, Upper Saddle River, NJ, 2001.

A.1.2 Journal papers

M. Preda, F. Prêteux, "Insights into low-level animation and MPEG-4 standardization", *Signal Processing: Image Communication*, Volume/Issue 17/9 pp. 717-741, November 2002.

M. Preda, F. Prêteux, "Virtual Character within MPEG-4 Animation Framework eXtension", to appear in *IEEE Transactions on Circuits and Systems for Video Technology*, 2003.

A.1.3 Conference papers

M. Preda, F. Prêteux, "Advanced animation framework for virtual character within the MPEG-4 standard" *Proceedings IEEE International Conference on Image Processing (ICIP'2002)*, Rochester, NY, 22-25 September 2002.

M. Preda, F. Prêteux, "Critic review on MPEF-4 Face and Body Animation", *Proceedings IEEE International Conference on Image Processing (ICIP'2002)*, Rochester, NY, 22-25 September 2002.

A.Salomie, A.Gavrilescu, R.Deklerck, I.Ravyse, J.Cornelis, M. Preda, "Efficient animation of meshgrid models using a skin and bone system", *Proceedings 3rd IEEE Benelux Signal Processing Symposium (SPS'2002)*, Leuven, Belgium, March 21-22 2002.

M. Preda, F. Prêteux, "Advanced virtual humanoid animation framework based on the MPEG-4 SNHC standard", *Proceedings EUROIMAGE International Conference on Augmented, Virtual Enviroments and Three-Dimensional Imaging (ICAV3D'01)*, Mykonos, Greece, May 2001, pp. 311-314.

F. Prêteux, C.Fetita, M.Malciu, M. Preda, "Advanced methods for 3D object representation and animation. Application to medical imaging and virtual humanoids" *Proceedings International Conference Communications 2000*, Bucarest, Romania, 7-9 December 2000, pp. 38-49.

M. Preda, T. Zaharia, F. Prêteux, "3D body animation and coding within a MPEG-4 compliant framework", *Proceedings International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging (IWSNHC3DI'99)*, Santorini, Greece, 15-17 September 1999, pp. 74-78.

T.Zaharia, M. Preda, F.Prêteux, "Sign language indexation within the MPEG-7 framework", *Proceedings SPIE Conference 3816 on Mathematical Modeling, Bayesian Estimation and Inverse Problems*, Denver, CO, July 1999, pp. 214-228.

A.1.4 Patents

J-C. Dufourd, C. Concolato, F. Prêteux, M. Preda, "Procédé et équipement pour la gestion des interactions multimédias mono- ou multi-utilisateurs entre des périphériques de commande et des applications multimédias exploitant la norme MPEG-4", Brevet n°0101648, février 2001.

A.1.5 Tutorials

M. Preda, "Virtual Character Animation within MPEG-4", part of MPEG-4 Animation Framework eXtension tutorial, *Proceedings of Third Workshop and Exhibition on MPEG-4 (WEMP4)*, 25-27 June 2002, San Jose, USA.

M. Preda, "MPEG-4 FBA and BBA", invited speaker at *IST concertration meeting*, 25-27 June 2002, Bologna, Italy.

A.1.6 Technical reports ISO

M. Preda, C. Concolato, F. Prêteux, "XML representation for BBA", ISO/IEC JTC1/SC29/WG11, MPEG01/M9015 Shanghai, October 2002.

M. Gutierrez, F. Vexo, M. Preda, "Higher order interpolation of key postures: an improved MPEG-4 deformation algorithm", ISO/IEC JTC1/SC29/WG11, MPEG01/M9030 Shanghai, October 2002.

M. Preda, F. Prêteux, "Uncompressed and human-readable animation file format for Virtual Characters", ISO/IEC JTC1/SC29/WG11, MPEG02/M8503 Klagenfurt, July 2002.

M. Preda, F. Prêteux, "Designing muscle-like deformation", ISO/IEC JTC1/SC29/WG11, MPEG02/M8504 Klagenfurt, July 2002.

M. Preda, F. Prêteux, "Comments on AFX Virtual Character specifications", ISO/IEC JTC1/SC29/WG11, MPEG02/M8514 Klagenfurt, July 2002.

M. Preda, Gauthier Lafruit, F. Prêteux, Call for Requirements for On-Line Gaming (OLGA): Draft Version, ISO/IEC JTC1/SC29/WG11, MPEG02/M8664 Klagenfurt, July 2002.

E. Petajan, M. Preda, "Proposed text for DCOR related to FBA", ISO/IEC JTC1/SC29/WG11, MPEG01/ M8458 Fairfax, May 2002.

M. Preda, F. Prêteux, JC Dufourd, "Specifications for an XMT compliant FBA file format", ISO/IEC JTC1/SC29/WG11, MPEG01/ M8424 Fairfax, May 2002.

M. Preda, F. Prêteux, "BIFS-Anim extension for encapsulating BBA stream", ISO/IEC JTC1/SC29/WG11, MPEG01/ M8307 Fairfax, May 2002.

M. Preda, F. Prêteux, "BBA within an AFX profile", ISO/IEC JTC1/SC29/WG11, MPEG01/M8151, Jeju, South Korea, March 2002.

M. Preda, F. Prêteux, "Streamed animation within Animation Framework eXtension (AFX)", ISO/IEC JTC1/SC29/WG11, MPEG01/M7588, Pattaya, Thailand, December 2001.

M. Preda, F. Prêteux, "Skin and Bone implementation in reference software", ISO/IEC JTC1/SC29/WG11, MPEG01/M7587, Pattaya, Thailand, December 2001.

M. Preda, F. Prêteux, "Skin&Bones Coding as an FBA Coding Extension", ISO/IEC JTC1/SC29/WG11, MPEG01/M7756, Pattaya, Thailand, December 2001.

M. Preda, F. Prêteux, "Deformable Bones or Muscle-based Deformation", ISO/IEC JTC1/SC29/WG11, MPEG01/M7757, Pattaya, Thailand, December 2001.

M. Preda, F. Prêteux, ”FNB Comments on the FBA Stream Syntax”, ISO/IEC JTC1/SC29/WG11, MPEG01/M7758, Pattaya, Thailand, December 2001.

M. Preda, F. Prêteux, ”Node syntax harmonization between bone-based animation and H-Anim 2001”, ISO/IEC JTC1/SC29/WG11, MPEG01/M7486 Sydney, July 2001.

M. Preda, F. Prêteux, ”New Node Syntax Supporting Inverse Kinematics in Bone-Based Animation”, ISO/IEC JTC1/SC29/WG11, MPEG01/M7487 Sydney, July 2001.

E. Petajan, M. Preda, Report of FBA Conformance ad-hoc, ISO/IEC JTC1/SC29/WG11,MPEG01/M7103,Singapore, March 2001.

M. Preda, F. Prêteux, M. Bourges-Sevenier, ”Generic articulated model definition and animation”, ISO/IEC JTC1/SC29/WG11, MPEG01/M7116, Singapore, March 2001.

JC. Dufourd, C. Concolato, F. Prêteux, M. Preda, ”An extensible, generic, low-cost architecture for user interaction”, ISO/IEC JTC1/SC29/WG11, MPEG01/M6811, January 2001

T. Zaharia, M. Preda, F. Prêteux, ”Similarity measures for motion-based retrieval”, ISO/IEC JTC1/SC29/WG11, MPEG99/M5595, Maui, Hawaii, USA, December 1999.

T. Zaharia, M. Preda, F. Prêteux, ”Motion content set: New contributions”, ISO/IEC JTC1/SC29/WG11, MPEG99/M5594, Maui, Hawaii, USA, December 1999.

T. Zaharia, M. Preda, F. Prêteux, ”3D shape descriptors: Results and performance evaluation”, ISO/IEC JTC1/SC29/WG11, MPEG99/M5592, Maui, Hawaii, USA, December 1999.

F. Prêteux, T. Zaharia, M. Preda, ”Core Experiment on motion trajectories: New data set and preliminary results”, ISO/IEC JTC1/SC29/WG11, MPEG97/M5253, Melbourne, Australia, October 1999.

T. Zaharia, F. Prêteux, M. Preda, ”3D Shape spectrum descriptor”, ISO/IEC JTC1/SC29/WG11, MPEG97/M5242, Melbourne, Australia, October 1999.

F. Prêteux, T. Zaharia, M. Preda, ”Results of Core Experiment on 3D shape: The cord histogram descriptor related performances”, ISO/IEC JTC1/SC29/WG11, MPEG97/M5240, Melbourne, Australia, October 1999.

F. Prêteux, T. Zaharia, M. Preda, ”Preliminary results of CE on motion trajectory”, ISO/IEC JTC1/SC29/WG11, MPEG97/M4871, Vancouver, Canada, July 1999.

F. Prêteux, T. Zaharia, M. Preda, ”Parametric object motion descriptor”, ISO/IEC JTC1/SC29/WG11, MPEG97/M4870, Vancouver, Canada, July 1999.

F. Prêteux, M. Preda, T. Zaharia, ”Predictive- versus DCT-based BAP coding”, ISO/IEC JTC1/SC29/WG11, MPEG97/M4554, Seoul, Korea, March 1999.

F. Prêteux, M. Preda, T. Zaharia, ”Results of Core Experiment on BAP coding (DCT Coding) ”, ISO/IEC JTC1/SC29/WG11, MPEG97/M4283, Roma, Italy, December 1998.

F. Prêteux, M. Preda, T. Zaharia, ”Preliminary results on hand BAT interpolation”, ISO/IEC JTC1/SC29/WG11, MPEG97/M4278, Roma, Italy, December 1998.

F. Prêteux, M. Preda, G.Mozelle, ”Body animation, BAPs coding and bitstream exchanges”, ISO/IEC JTC1/SC29/WG11, MPEG97/M4147, Atlantic City, NJ, October 1998.

F. Prêteux, M. Preda, G.Mozelle, ”Hand animation and BAPs extraction : Reports on Core Experiment 3”, ISO/IEC JTC1/SC29/WG11, MPEG97/M3592, Dublin, Ireland, July 1998.

F. Prêteux, M. Preda, G.Mozelle, ”Donation to ISO of hand animation software”, ISO/IEC JTC1/SC29/WG11, MPEG97/M3590, Dublin, Ireland, July 1998.

F. Prêteux, P.Horain, H.Ouhaddi, M. Preda, ”Report on Core Experiment 3 on Hand BAPs interpretation”, ISO/IEC JTC1/SC29/WG11, MPEG97/M3332, Tokyo, Japan, March 1998.

Appendix B

XML compliant format for BBA

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:bba="urn:mpeg:mpeg4:bba:schema:2002"
   targetNamespace="urn:mpeg:mpeg4:bba:schema:2002" elementFormDefault="qualified">
      <annotation>
            <documentation>
                  Bone-Based Animation Parameters Textual format Schema
            </documentation>
      </annotation>
      <annotation>
            <documentation> basic types </documentation>
      </annotation>
      <simpleType name="ZeroOneType">
            <restriction base="integer">
                  <minInclusive value="0"/>
                  <maxInclusive value="1"/>
            </restriction>
      </simpleType>
      <simpleType name="ListZeroOneType">
            <list itemType="bba:ZeroOneType">
      </simpleType>
      <simpleType name="BBAParameterType">
            <restriction base="integer">
                  <minInclusive value="-314159"/>
                  <maxInclusive value="314159"/>
            </restriction>
      </simpleType>
      <annotation>
            <documentation>
            ****************************************************************
            *Declaration of the top-level containers (BBA, Header, …) *
            ****************************************************************
            </documentation>
      </annotation>
      <element name="BBA">
            <complexType>
             <sequence>
              <element ref="bba:header" use="optional"/>
              <element name="body" minOccurs="0">
                <complexType>
                  <element ref="bba:Frame" minOccurs="0" maxOccurs="unbounded"/>
                </complexType>
              </element>
             </sequence>
            </complexType>
      </element>
      <element name="header">
            <complexType>
         <attribute name="version" use="required" type="float" default="3.0"/>
         <attribute name="name" use="required" type="string"/>
         <attribute name="frameRate" use="required" type="unsignedInt" default="10"/>
         <attribute name="encodingParametersFile" use="optional" type="string"/>
            </complexType>
      </element>
      <annotation>
            <documentation>
            ********************************************************
            * Declaration of the dynamic containers (BodyFrameData)        *
            ********************************************************
```

```
                </documentation>
            </annotation>
            <element name= »Frame »>
             <complexType>
        <element ref= »bba :FrameMask » minOccurs= »1 » maxOccurs= »1 »/>
        <element ref= »bba :FrameValues » minOccurs= »1 » maxOccurs= »1 »/>
             </complexType>
            </element>
            <element name= »FrameMask »>
                <complexType>
                    <attribute name= »KFI »>
                        <simpleType>
                            <restriction base= »integer »>
                                <minInclusive value= »0 »/>
                                <maxInclusive value= »31 »/>
                            </restriction>
                        </simpleType>
                    </attribute>
                    <attribute name= »NSBB »>
                        <simpleType>
                            <restriction base= »integer »>
                                <minInclusive value= »0 »/>
                                <maxInclusive value= »1023 »/>
                            </restriction>
                        </simpleType>
                    </attribute>
                    <attribute name= »NSBM »>
                        <simpleType>
                            <restriction base= »integer »>
                                <minInclusive value= »0 »/>
                                <maxInclusive value= »1023 »/>
                            </restriction>
                        </simpleType>
                    </attribute>
                    <element ref= »bba :BoneIDMask » minOccurs= »0 »
maxOccurs= »unbounded »/>
                    <element ref= »bba :MuscleIDMask » minOccurs= »0 »
maxOccurs= »unbounded »/>
                </complexType>
            </element>

            <element name="BoneIDMask">
                <complexType>
                    <attribute name="IDB" use="required">
                        <simpleType>
                            <restriction base="integer">
                                <minInclusive value="0"/>
                                <maxInclusive value="1023"/>
                            </restriction>
                        </simpleType>
                    </attribute>
                    <element ref="bba:BoneAnimationMask" minOccurs="1" maxOccurs="1"/>
                </complexType>
            </element>
            <element name="BoneAnimationMask">
                <complexType>
                  <attribute name="isTranslation" use="optional" type="bba:ZeroOne"/>
                  <attribute name="isTranslationOnX" use="optional" type="bba:ZeroOne"/>
                  <attribute name="isTranslationOnY" use="optional" type="bba:ZeroOne"/>
```

```
            <attribute name="isTranslationOnZ" use="optional" type="bba:ZeroOne"/>
            <attribute name="isRotation" use="optional" type="bba:ZeroOne"/>
            <attribute name="isRotationOnAxis1" use="optional" type="bba:ZeroOne"/>
            <attribute name="isRotationOnAxis2" use="optional" type="bba:ZeroOne"/>
            <attribute name="isRotationOnAxis3" use="optional" type="bba:ZeroOne"/>
            <attribute name="isScale" use="optional" type="bba:ZeroOne"/>
            <attribute name="isScaleOnX" use="optional" type="bba:ZeroOne"/>
            <attribute name="isScaleOnY" use="optional" type="bba:ZeroOne"/>
            <attribute name="isScaleOnZ" use="optional" type="bba:ZeroOne"/>
            <attribute name="isScaleOrientation" use="optional" type="bba:ZeroOne"/>
            <attribute name="isScaleOrientationOnAxeX" use="optional" type="bba:ZeroOne"/>
            <attribute name="isScaleOrientationOnAxeY" use="optional" type="bba:ZeroOne"/>
            <attribute name="isScaleOrientationOnAxeZ" use="optional" type="bba:ZeroOne"/>
            <attribute name="isScaleOrientationValue" use="optional" type="bba:ZeroOne"/>
            <attribute name="isCenter" use="optional" type="bba:ZeroOne"/>
            <attribute name="isCenterOnX" use="optional" type="bba:ZeroOne"/>
            <attribute name="isCenterOnY" use="optional" type="bba:ZeroOne"/>
            <attribute name="isCenterOnZ" use="optional" type="bba:ZeroOne"/>
        </complexType>
</element>

<element name="MuscleIDMask">
        <complexType>
                <attribute name="IDM" use="required">
                        <simpleType>
                                <restriction base="integer">
                                        <minInclusive value="0"/>
                                        <maxInclusive value="1023"/>
                                </restriction>
                        </simpleType>
                </attribute>
                <attribute name="NCP" use="required">
                        <simpleType>
                                <restriction base="integer">
                                        <minInclusive value="0"/>
                                        <maxInclusive value="63"/>
                                </restriction>
                        </simpleType>
                </attribute>
                <attribute name="NK" use="required">
                        <simpleType>
                                <restriction base="integer">
                                        <minInclusive value="0"/>
                                        <maxInclusive value="63"/>
                                </restriction>
                        </simpleType>
                </attribute>
                <attribute name="NCP" use="required">
                        <simpleType>
                                <restriction base="integer">
                                        <minInclusive value="0"/>
                                        <maxInclusive value="63"/>
                                </restriction>
                        </simpleType>
                </attribute>
                <attribute name="MM" use="required">
                        <simpleType>
                                <restriction base="bba:ListZeroOneType">
                                        <minLength value="0"/>
```

```
                                        </restriction>
                                    </simpleType>
                            </attribute>
                    </complexType>
            </element>

            <element name="FrameValues">
                    <complexType>
                            <element ref="bba:BoneAnimationValue" minOccurs="0"
maxOccurs="unbounded"/>
                            <element ref="bba:MuscleAnimationValue" minOccurs="0"
maxOccurs="unbounded"/>
                    </complexType>
            </element>

            <element name="BoneAnimationValue">
                    <complexType>
                      <attribute name="TranslationOnX" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="TranslationOnY" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="TranslationOnZ" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="RotationOnAxis1" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="RotationOnAxis2" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="RotationOnAxis3" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="ScaleOnX" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="ScaleOnY" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="ScaleOnZ" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="ScaleOrientationOnAxeX" use="optional"
type="bba:BBAParameterType"/>
                      <attribute name="ScaleOrientationOnAxeY" use="optional"
type="bba:BBAParameterType"/>
                      <attribute name="ScaleOrientationOnAxeZ" use="optional"
type="bba:BBAParameterType"/>
                      <attribute name="ScaleOrientationValue" use="optional"
type="bba:BBAParameterType"/>
                      <attribute name="CenterOnX" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="CenterOnY" use="optional" type="bba:BBAParameterType"/>
                      <attribute name="CenterOnZ" use="optional" type="bba:BBAParameterType"/>
                    </complexType>
            </element>

            <element name="MuscleAnimationValue">
                    <complexType>
                      <list type="bba:BBAParameterType"/>
                    </complexType>
            </element>
    </schema>
```

References

[3DSM] 3D Studio Max, Version 3.1. Autodesk. 1999. San Francisco, CA.

[Activision] www.activision.com

[Animian98] Aminian K *et al.*, "Motion Analysis in Clinical Practice Using Ambulatory Accelerometry", Lecture Notes in *Artificial Intelligence 1537. Modeling and Motion Capture Techniques for Virtual Environments*, 1998.

[Armstrong85] W. W. Armstrong and M. Green, "The dynamics of articulated rigid bodies for purposes of animation", Proceedings of *Graphics Interface '85*, pages 407–415, 1985.

[AT] Ascension Technology MotionStar® www.ascension-tech.com/products/motionstar/.

[Badler93] N. Badler, C. Phillips, and B.Webber. "Simulating Humans: Computer Graphics, Animation, and Control" Oxford University Press, 1993.

[Badler93] N.I. Badler, M.J. Hollick, John P. Granieri, "Real-Time Control of a Virtual Human Using Minimal Sensors", *Presence*, Vol. 2, No. 1, pp. 82 – 86, 1993.

[Badler95] N. Badler, D. Metaxas, B. Webber and M. Steedman, "The Center for Human Modeling and Simulation", *Presence* 4(1), pp. 81-96, 1995.

[Bartels87] R. H. Bartels, J. C. Beatty, An introduction to splines for use in computer graphics and geometric modeling, Morgan Kaufman, Los Altos, CA, 1987.

[BDI] Boston Dynamics Inc, The digital biomechanics laboratory, www.bdi.com, 1998.

[Birn] Jeremy Birn, "TUTORIAL: NURBS Head Modeling", www.3drender.com.

[Blanz99] Blanz, V., Vetter, T., 'A Morphable Model For The Synthesis Of 3D Faces', Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH 1999, pp187-194.

[BlaxxunCom] blaxxun Community, VRML – 3D – Avatars – Multi-User Interaction, http ://www.blaxxun.com/vrml/home/ccpro.htm.

[BlaxxunNR] blaxxun interactive, NURBS Extension for VRML97, 2/2001.

[Blinn82] J. Blinn, "A Generalization of Algebraic Surface Drawing", *ACM Transactions on Graphics*, Vol. 1, No. 3, pp. 235-256, July, 1982.

[Bricken94] Bricken, W. and Coco, G. The VEOS project Presence. Vol. 3 No. 2. Spring 1994. pp. 111-129. MIT Press.

[Brotman88] L. S. Brotman and A. N. Netravali, "Motion interpolation by optimal control" *Proceedings of SIGGRAPH 88*, 22(4):309–315, August 1988.

[Campus] Roberto Campus Portofolio, www.robertocampus.com.

[Capin97] Capin T, Pandzic I, Magnenat Thalmann N, Thalmann D, "Virtual Human Representation and Communication in the VLNET Networked Virtual

Environments", *IEEE Computer Graphics and Applications*, Vol.17, No2, 1997, pp.42-53.

[Carlsson93] C. Carlsson and O. Hagsand. "DIVE – a multi-user virtual reality system", *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 1993.

[Carlsson93] C. Carlsson and O. Hagsand. "DIVE – a multi-user virtual reality system", *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 1993.

[Catmull78] Catmull E. and Clark J. "Recursively generated B-spline surfaces on arbitrary topological meshes" *Computer-Aided Design*, 10:350-355, September 1978.

[CEPSAS] Centre Européen de Promotion Sociale des Adultes Sourds, Poitiers, France.

[Cohen92] M. F. Cohen. "Interactive spacetime control for animation", *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):293–302, July 1992.

[EUCouncil02] Council of the European Union, "Resolution regarding accessibility to all public web site services and installations and their content", March 25, 2002.

[Craig89] Craig. J. Jr., "Introduction to Robotics: Mechanics and Control, 2nd edition", Addison-Wesley, Reading, MA, 1989.

[Dekker98] Dekker L, Khan S, West E, Buxton B, Treleaven P, "Models for Understanding the 3D Human Body Form", *Proceedings of IEEE Workshop on Model-Based 3D Image Analysis*, 65-74, Bombay, India, 1998.

[Dickinson98] S. Dickinson, A. Pentland, S. Stevenson, "Viewpoint-invariant indexing for content-based image retrieval", *Proceedings 1998 IEEE International Workshop on Content-Based Access of Image and Video Database*, pp. 20-30, Janvier 1998.

[Disney97] Disney Animation Studios, Flubber (1997) http://disney.go.com/disneyvideos/liveaction/flubber/.

[Division93] Division Ltd. dVS Technical Overview, Version 2.0.4, first edition, 1993.

[Doenges97] P. Doenges, T. Capin, F. Lavagetto, J. Ostermann, I. Pandzic, and E. Petajan, "Mpeg-4: Audio/video and synthetic graphics/audio for real-time, interactive media delivery" *Image Communications Journal*, 5(4):433–463, 1997.

[Douros99] Douros I, Dekker L, Buxton B F, "Reconstruction of the surface of the human body from 3D scanner data using B-spline" *Proceedings of SPIE on Three-Dimensional Image Capture and Applications*, San Jose,California, January, 1999.

[Earts] Electronic Arts, www.ea.com.

[Eberly01] D. H. Eberly, "3D game engine design" ISBN 1-55860-593-2, 2001.

[EC02-a] EC/21/2002, OJ L 108, 24.04.2002, pp. 7-77.

[EC02-b] Barcelona European Council Conclusions, paragraph 41.

[EC02-c] eEurope 2005: An information society for all, COM (2002), 263, pg 7.

[Escher98] M. Escher, I. Pandzic, and N. Magnenat-Thalmann. "Facial Animation and Deformation for MPEG-4", *Proceedings of Computer Animation'98*, 1998.

[EveSolal] Attitude Studio, Eve Solal, www.evesolal.com

[Falouutsos01] P. Falouutsos, M. van de Panne, D. Terzopoulos, "Composable Controllers for Physics-based Character Animation", in *Computer Graphics Proceedings, Annual Conference Series,* 2001, ACM SIGGRAPH.

[Farin] G. Farin, J. Hoschek, M.S. Kim (editors), "Handbook of Computer Aided Geometric Design", North-Holland; ISBN: 0444511040, June 1, 2002.

[Faro] Faro Technologies, www.farotechnologies.com.

[Foley92] J. D. Foley, A. van Damm, S.K. Feiner, and J.F. Hughes. "Computer Graphics – Principles and Practice", Addison Wesley, second edition, 1992.

[Funge99] Funge. J.D, "AI for Computer Games and Animation: A Cognitive Modeling Approach". A K Peters Ltd, August 1999.

[Gleicher97] M. Gleicher, "Motion editing with spacetime constraints", In *Proceedings of Symposium on Interactive 3D Graphics*, pages 139–148, 1997.

[Gleicher98] M. Gleicher, "Retargetting motion to new characters", *Computer Graphics Proceedings of SIGGRAPH 98*, 32:33–42, July 1998.

[Goldfarb69] D. Goldfarb, "Extension of Davidon's Variable Metric Method to Maximisation under Linear Inequality and Equality Constraints", *SIAM Journal Appl. Math.*, Vol. 17, pp. 739 – 764, 1969.

[Golub65] Golub, G. and Kahan, W. (1965). "Calculating the singular values and pseudo-inverse of a matrix", *Journal SIAM Numerical Analysis*, 2(2), 205-223.

[Greenhalgh95] C. Greenhalgh and S. Benford, "MASSIVE: a Distributed Virtual Reality System Incorporating Spatial Trading", in *15th International Conference on Distributed Computing Systems (DCS'95)*, Vancouver, Canada, May 30-June 2 1995. IEEE Computer Society Press.

[Guo96] S. Guo and J. Robergé, "A high-level control mechanism for human locomotion based on parametric frame space interpolation", in *Proceedings of Computer Animation and Simulation '96*, Eurographics Animation Workshop, pages 95–107. Springer-Verlag, 1996.

[Gurzki01] T. Gurzki, H. Hinderer, U. Rotter, "A Platform for Fashion Shopping with Individualized Avatars and Personalized Customer Consulting", in: *Proceedings of the World Conference on Mass Customisation*, 2001, Hong Kong, China, 2001.

[Guttierez02] M. Gutierrez, F. Vexo, M. Preda, "Higher order interpolation of key postures: an improved MPEG-4 deformation algorithm", ISO/IEC JTC1/SC29/WG11, MPEG01/M9030 Shanghai, October 2002.

[Hanim] Humanoid Animation Specification, www.h-anim.org.

[HI] HI Corporation, www.hicorp.co.jp.

[Hirose96] M. Hirose, G. Deffaux, Y. Nakagaki, "Development of an effective motion capture system based on data fusion and minimal use of sensors", *VRST 96*, July 1996.

[Ibc02] International Broadcasting Convention, www.ibc.org, Amsterdam, 12-16 September 2002.

[ISOIEC01] ISO/IEC 14496-1:2001 Information technology -- Coding of audio-visual objects -- Part 1: Systems, International Organization for Standardization, Swiss, 2001.

[Jones95] Jones P R M, Li P, Brook-Wavell K, West G M, "Format of human body modelling from 3-D body scanning", *International Journal of Clothing Science*, 7(1), 7-16, 1995.

[Ju00] Ju X., Werghi N and Siebert J P, "Automatic Segmentation of 3D Human Body Scans", *Proceedings IASTED Int. Conf. on Computer Graphics and Imaging 2000 (CGIM 2000)*, Las Vegas, USA, 2000.

[Kalra98] P. Kalra, N. Magnenat-Thalmann, L. Moccozet, G. Sannier, A. Aubel and D. Thalmann, "Real-Time Animation of Realistic Virtual Humans", *IEEE Computer Graphics and Applications* volume 18, number 5, pp 42-57, 1998.

[Khwaja98] A.A. Khwaja, M.O. Rahman, M.G. Wagner, "Inverse Kinematics of Arbitrary Robotic Manipulators Using Genetic Algorithms", in *Advances in Robot Kinematics: Analysis and Control*, Lenarcic and Husty (eds.), Kluwer Academic Publishers, pp. 375 – 382, 1998.

[Kim00] M. Kim, S. Wood, L.T. Cheok, "Extensible MPEG-4 Textual Format (XMT)", *ACM Multimedia 2000*, October 30 - November 4, 2000, Los Angeles, California.

[Kovach] P.J. Kovach, Inside Direct3D, Microsoft Press, 2000.

[Lander99] J. Lander, "Over my dead, polygonal body", Game Developer Magazine, Obtober 1999.

[Lasseter87] Lasseter, J. "Principles of Traditional Animation applied to 3D computer animation," pp. 35-44, SIGGRAPH 87.

[LAT] Living actor technology, www.living-actor.com.

[Lavagetto99] F. Lavagetto and R. Pockaj, "The Facial Animation Engine: Toward a High-level Interface for the Design of Mpeg-4 Compliant Animated Faces", *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 9, No. 2, pp. 277–289, March 1999.

[Li97] Li P, Jones P. R. M., "Automatic Editing and Curvefitting of 3-D Surface Scan Data of the Human Body", *Proceedings of International Conference on Recent*

Advances in 3-D Digital Imaging and Modelling, pp. 296-301, Ottawa, Canada, 1997.

[Liu94] Z. Liu, S. G. Gortler, and M. F. Cohen, "Hierarchical spacetime control", *Computer Graphics (Proceedings of SIGGRAPH 94)*, pp. 35–42, July 1994.

[Loop87] C. Loop, "Smooth subdivision surfaces based on triangles", Master's thesis, Department of Mathematics, University of Utah, August 1987.

[Lutz20] Lutz, E. G., "Animated Cartoons: How They are Made, Their Origin, and Their Development", Charles Scribner's Sons, 1920. Reprinted by Applewood Books, 1998.

[Mari] Mari JL, Sequeira J., "Using Implicit Surfaces to characterize shapes within Digital Volumes", *Congrès RECPAD,* 2000.

[Macedonia94] Macedonia, Michael R. Zyda, Michael J., Pratt, David R., Barham, Paul T. and Zeswitz, Steven, "NPSNET: A Network Software Architecture for Large Scale Virtual Environments", *Presence*, Vol. 3, No. 4. Fall 1994.

[Maestri99] Maestri G., "Digital Character Animation 2: Essential Techniques", New Riders, Paperback, July 1999.

[Maya99] Maya Infinity, Alias/Wavefront Inc., 1999.

[McKerrow91] P. J. McKerrow, "Introduction to Robotics", Addison-Wesley, Electronic Systems Engineering Series, Wokingham, 1991.

[Moeslund01] T.B. Moeslund and E. Granum, "A Survey of Computer Vision-Based Human Motion Capture", *Computer Vision and Image Understanding*, 2001.

[Molet99] T. Molet, R. Boulic, D. Thalmann, "Human Motion Capture Driven by Orientation Measurements", *Presence*, MIT, Vol. 8, No. 2, pp.187 – 203, 1999.

[Mozelle98] G. Mozelle, F. Prêteux, J.E. Viallet, "Tele-sign : A Compression Framework for Sign Language Distant Communication", *Proceedings SPIE Conference on Mathematical Modeling and Estimation Techniques in Computer Vision*, San Diego, CA, Vol. 3457, July 1998, pp. 94-110.

[Mozelle98b] G. Mozelle, "Compression de données multiples et dynamiques", Ph.D Thesis, Paris, July 1998.

[Nakamura86] Nakamura, Y. and Hanafusa, H., "Inverse kinematic solutions with singularity robustness for robot manipulator control", *Journal of Dynamic Systems, Measurement and Control*, 108(3), pp. 163-171, 1986.

[Nurre97] Nurre J H, "Locating Landmarks on Human Body Scan Data", *International Conference of Recent Advances 3D Digital Imaging and Modeling*, pp. 289-295, IEEE NJ USA, 1997.

[OpenGL] www.opengl.org.

[Orin84] Orin, D.E. and Schrader, W. W., "Efficient computation of the jacobian for robot manipulators", *The International Journal of Robotic Research*, 3(4), pp. 66-75, 1984.

[Orourke] M. O'Rourke, "Principles of Three-Dimensional Computer Animation: Modeling, Rendering, and Animating With 3d Computer Graphics", Hardcover Edition, May 1998.

[Pandy90] M. G. Pandy, F. E. Zajac, E. Sim, and W. S. Levine, "An optimal control model for maximum-height human jumping", *Journal of Biomechanics*, 23(12):1185–1198, 1990.

[Pandy99] M. G. Pandy and F. C. Anderson, "Three-dimensional computer simulation of jumping and walking using the same model", in *Proceedings of the VIIth International Symposium on Computer Simulation in Biomechanics*, August 1999.

[Pandzic97] Pandzic I, Magnenat Thalmann N, Capin T, Thalmann, "Virtual LifeNetwork: A Body-Centered Networked Virtual Environment", *Presence*, MIT, Vol.6, No 6, 1997, pp. 676-686.

[Pargas97] Pargas R P, Staples N J, Davis J S, "Automatic Measurement Extraction for Apparel from a Three-Dimensional Body Scan", *Optics and Lasers in Engineering*, 28(2), 157-172, 1997.

[Paul 81] R. Paul, "Robot Manipulators: Mathematics, Programming and Control", MIT Press, 1981.

[Pereira00] F. Pereira (Editor), "MPEG-4 Requirements, version 15 (La Baule revision)", ISO/IEC JTC1/SC29/WG11, M3746 October 2000, La Baule.

[Pereira02] F. Pereira, T. Ebrahimi (Editors), "The MPEG-4 Book", Prentice Hall PTR, ISBN 0-13-061621-4, USA, 2002.

[Phillips90] C.B. Phillips, J. Zhao, N. Badler, "Interactive Real-Time Articulated Figure Manipulation using Multiple Kinematic Constraints", *SIGGRAPH 90 Course notes (Human figure animation : approaches and applications)*, 1990.

[Piegl97] L. Piegl, W. Tiller, "The NURBS Book, Second edition", Springer, 1997.

[Polhemus] Polhemus STAR*TRACK motion capture system, http://www.polhemus.com.

[Preda99] M. Preda, T. Zaharia, F. Prêteux, "3D body animation and coding within a MPEG-4 compliant framework", in *proceedings International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging (IWSNHC3DI'99)*, Santorini, Greece, 15-17 September 1999, pp. 74-78.

[Preda01] M. Preda, F. Prêteux, "Advanced virtual humanoid animation framework based on the MPEG-4 SNHC standard" in *proceedings EUROIMAGE International Conference on Augmented, Virtual Enviroments and Three-Dimensional Imaging (ICAV3D'01)*, Mykonos, Greece, May 2001, pp. 311-314.

[Preda01-a] M. Preda, F. Prêteux, and M. Bourges-Sevenier, "Generic articulated model definition and animation", ISO/IEC JTC1/SC29/WG11, MPEG01/M7116, Singapore, March 2001.

[Preda01-b] M. Preda, F. Prêteux, "New node syntax supporting inverse kinematics in Bone-Based Animation", ISO/IEC JTC1/SC29/WG11, MPEG01/M7487, Sydney, July 2001.

[Preda01-c] M Preda and F Prêteux, "Node syntax harmonization between bone-based animation and H-Anim 2001", ISO/IEC JTC1/SC29/WG11, MPEG01/M7486 Sydney, July 2001.

[Preda01-d] M. Preda, F. Prêteux, "Streamed animation within Animation Framework eXtension (AFX)", ISO/IEC JTC1/SC29/WG11, MPEG01/M7588, Pattaya, Thailand, December 2001. (Submitted to SNHC group in October 2001).

[Preda01-e] M. Preda, F. Prêteux, "Skin and Bone implementation in the reference software", ISO/IEC JTC1/SC29/WG11, MPEG01/M7587, Pattaya, Thailand, December 2001.(Submitted to SNHC group in October 2001).

[Preda01-f] M. Preda, F. Prêteux, "Deformable Bones or Muscle-based Deformation?", ISO/IEC JTC1/SC29/WG11, MPEG01/M7757, Pattaya, Thailand, December 2001.

[Preda01-g] M Preda and F Prêteux, "Skin&Bones Coding as an FBA Coding Extension", ISO/IEC JTC1/SC29/WG11, MPEG01/M7756 Pattaya, Thailand, December 2001.

[Preda02-a] M Preda and F Prêteux, "BIFS-Anim extension for encapsulating BBA stream", ISO/IEC JTC1/SC29/WG11, MPEG02/M8307 Fairfax, May 2002.

[Preda02-b] M. Preda, F. Prêteux, "Uncompressed and human-readable animation file format for Virtual Characters", ISO/IEC JTC1/SC29/WG11, MPEG02/M8503 Klagenfurt, July 2002.

[Preda02-c] M. Preda, C. Concolato, F. Prêteux, "XML representation for BBA", ISO/IEC JTC1/SC29/WG11, MPEG01/M9015 Shanghai, October 2002.

[Prêteux98] F. Prêteux, M. Preda and G. Mozelle, "Donation to ISO of Hand Animation Software", ISO/IEC JTC1/SC29/WG11, M3590, July 1998, Dublin.

[Rao71] Rao, C. R. and Mitra, S. K., "Generalized Inverse of Matrices and its Applications", John Wiley and Sons, Inc., 1971.

[RNID] The Royal National Institute for the Deaf (RNID), www.rnid.org.uk.

[Ronfard96] R. Ronfard and J. Rossignac, "Full-range approximation of triangulated polyhedra", in *EUROGRAPHICS 96*, volume 15 of Computer Graphics Forum, Conference Issue, pages 67-76. Eurographics Association, 1996. Also available as IBM Research Report 20423, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, 1996.

[Rose96] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen, "Efficient generation of motion transitions using spacetime constraints", *Computer Graphics (Proceedings of SIGGRAPH 96)*, 30:147–154, August 1996.

[Rose98] C. Rose, M. F. Cohen, and B. Bodenheimer, "Verbs and Adverbs: Multidimensional motion interpolation", IEEE CG&A, 18(5):32–40, October 1998.

[S20] S20 Sonic Digitizers, Science Accessories Corporation.

[Schroeder92] W. Schroeder, J. Zarge, and W. Lorensen, "Decimation of triangle meshes", in *SIGGRAPH 92*, Computer Graphics, Annual Conference Series, pages 65-70. ACM SIGGRAPH, July 1992.

[Seamless] Seamless Solutions Inc. demos, http://www.seamless-solutions.com.

[Seo00] H. Seo, F. Cordier, L. Philippon, N. Magnenat-Thalmann, "Interactive Modelling of MPEG-4 Deformable Human Body Models", DEFORM'2000 Workshop, pp 120-131, November 29-30, 2000, Geneva.

[Seonaid] Introducing Seonaid, our online news presenter, Scotland government web page, www.scotland.gov.uk.

[Sevenier01-a] M. B. Sévenier *et all.* (editors), "Animation Framework eXtension Core Experiments Description", ISO/IEC JTC1/SC29/WG11, N4019 Singapore, 2001.

[Sevenier01-b] M. B. Sévenier *et all.* (editors), "PDAM of ISO/IEC 14496-1 / AMD4", ISO/IEC JTC1/SC29/WG11, N4415 December 2001, Pattaya.

[Shoemake94] K. Shoemake, "Euler Angle Conversion", Gramphics Gems IV, Academic Press Professional, Toronto, 1994.

[Siciliano 91] B. Siciliano, J-J. Slotine, "A General Framework for Managing Multiple Tasks in Highly Redundant Robotic Systems", *ICAR '91*, pp. 1211 - 1216, 1991.

[SScape] Superscape Inc., www.superscape.com

[TVirt] Televirtual, Inc., UK, www.televirtual.com

[Thalmann00] N. Magnenat-Thalmann, D. Thalmann, "Virtual Reality Software and Technology", Encyclopedia of Computer Science and Technology, Marcel Dekker, Vol. 41.

[Thomas81] Thomas, F. and Ollie J. "Disney Animation: The Illusion of Life", Abbeville Press, 1981.

[Tolani00] D. Tolani, A. Goswanmi, and N. Badler, "Real-time inverse kinematics techniques for anthropomorphic limbs", in *proceedings of Graphical Models* 62, pp. 353–358, 2000.

[Unuma95] M. Unuma, K. Anjyo, and R. Takeuchi, "Fourier principles for emotion-based human figure animation", *Computer Graphics (Proceedings of SIGGRAPH 95)*,

29:91–96, August 1995.

[Vandrea] Vandrea news presenter, Channel 5, British Broadcasting Television.

[ViSiCAST] ViSiCAST, Virtual Signer Communication, Animation, Storage and Transmission, IST, European Project 1999-2002, www.visicast.org.

[VPL89] VPL Research Inc. Dataglove Model 2 Operation Manual, January 1989.

[VRML] The Virtual Reality Modeling Language, International Standard ISO/IEC 14772-1:1997, www.vrml.org.

[Walsh75] G.R. Walsh, "Methods of Optimization", John Wiley and Sons, 1975.

[Wang94] J. Wang and E. Adelson, "Spatio-temporal segmentation of video data", in *SPIE Proc. Image and Video Processing II*, volume 2182, San Jose, CA, February 1994.

[Watt92] A. Watt, M. Watt, "Advanced Animation and Rendering Techniques", Addison-Wesley, ACM Press, 1992.

[WdPixar] Geri's game, Toy Story (1995), A Bug's Life (1998), Toy Story 2 (1999) and Monsters, Inc (2001). Walt Disney Pictures & Pixar.

[Welman93] C. Welman, "Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation", Master Thesis, Simon Fraser University, 1993.

[Wiley97] D. J. Wiley and J. K. Hahn, "Interpolation synthesis for articulated figure motion", in *Proceedings of IEEE Virtual Reality Annual International Symposium '97*, pp. 157–160. IEEE Computer Society Press, 1997.

[Wilhelms85] J. Wilhelms and B. A. Barsky, "Using dynamic analysis to animate articulated bodies such as humans and robots", *in Graphics Interface '85*, pages 97–104, May 1985.

[Witkin88] A. Witkin and M. Kass, "Spacetime constraints", *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):159–168, August 1988.

[Witkin95] A. Witkin and Z. Popovic, "Motion warping", *Computer Graphics (Proceedings of SIGGRAPH 95)*, pages 105--108, August 1995.

[Wolovich 84] W.A. Wolovich, H. Elliot, "A computational technique for inverse kinematics", in *proceedings of 32nd Conference on Decision and Control*, pp. 1359 - 1362, Dec.1984.

[Wooten98] W. Wooten, "Simulation of Leaping, Tumbling, Landing, and Balancing Humans", PhD thesis, Georgia Institute of Technology, March 1998.

[Wyvill86] G. Wyvill, C. McPhetters, B. Wyvill, "Data Structure for Soft Objects", *The Visual Computer*, Vol. 2, pp. 227-234, 1986.

[Zorin00] Zorin D., Schröder P. et al. "Subdivision for Modeling and Animation", *SIGGRAPH'00 Conference Course Notes*, July 2000.

List of indices

animation
bone-based deformation, 60
deformation, 52
forward kinematics, 18, 20
inverse kinematics, 18, 60
seamless, 16
transformation matrix, 62
virtual body avatar, 29

compression
DCT-based method, 32, 33, 50, 51, 52, 78, 79, 80, 81, 101, 109, 123
Minimum bandwidth, 101
predictive-based method, 32, 50, 51, 78, 80

digital television, 114

geometry
Metaballs, 13
NURBS, 11, 14, 36, 64, 65, 77, 133
NURBS, example, 11
parametric, 10, 11, 133
patch, 11
polygonal surface, 10

mobile, 100
modelling
3D scanner, 133
articulated synthetic object, 59
influence region, 61, 63
mesh propagation algorithm, 35
seamless, 10
virtual body, 28

RESUME

Dans le cadre de la nouvelle société de l'information multimédia et communicante, cette thèse propose des contributions méthodologiques et techniques relatives à la représentation, l'animation et la transmission des objets virtuels.
Les méthodes existantes sont analysées de façon comparée et les performances des standards multimédias actuels évaluées en termes de réalisme d'animation et de débit de transmission. Pour surmonter les limitations mises en évidence, un nouveau cadre de modélisation et d'animation de personnages virtuels est proposé. Le modèle SMS (*Skeleton, Muscle and Skin*), fondé sur le concept de contrôleur de déformation d'un maillage, est introduit et sa formulation mathématique développée. Le graphe de scène 3D et le flux de compression associés à SMS sont décrits. L'approche SMS est évaluée dans le cadre d'un nouveau service de transmission télévisuelle d'un signeur virtuel destinés aux déficients auditifs. Le modèle SMS a été promu dans le standard MPEG-4 version 5.

Mots-clès : Modélisation et animation de personnage virtuel, contrôleur de déformation, compression des paramètres d'animation, transmission bas débit, langue des signes, standard multimédia MPEG-4.

ABSTRACT

Within the framework of the emerging networked multimedia information society, this dissertation brings forward methodological and technical contributions for representing, animating and transmitting 3D virtual objects.
The existing methods are first analyzed and compared, and the performances of current multimedia standards are evaluated in terms of animation realism and transmission bitrate. To overcome their limitations, a novel framework for modeling and animating virtual characters is then proposed. The SMS (*Skeleton, Muscle and Skin*) model, based on the concept of mesh deformation controller, is introduced and mathematically elaborated. The 3D scene graph and compression stream associated to the SMS model are also described. The SMS approach is finally demonstrated and evaluated in the framework of a novel digital broadcast service dedicated to virtual signing for the hearing-impaired people community. The SMS model has been promoted as part of Version 5 of the MPEG-4 ISO standard.

Keywords: Virtual character definition and animation, deformation controller, animation parameters compression, low bit-rate transmission, sign language, MPEG-4 standard.